INSTITUT FÜR INFORMATIK
Lehr- und Forschungseinheit für
Programmier- und Modellierungssprachen
Oettingenstraße 67, D–80538 München

Ludwig–
Maximilians–
Universität–
München–

Einführung in die Logik für Informatiker

François Bry, Norbert Eisinger

Skriptum/Lecture Notes, Oktober 2005
http://www.pms.ifi.lmu.de/publikationen

Vorwort

Die wachsende Bedeutung der Logik in der Informatik ist nicht zu übersehen. Methoden und Ansätze, die auf der Logik beruhen, sind nicht nur in der theoretischen Informatik, sondern auch in der praktischen Informatik allgegenwärtig, wie die zunehmende Verbreitung von Begriffen wie etwa „Programmverifikation“, „Wissensrepräsentation“, „Model Checking“, „Field Programmable Logic“ bezeugt. Damit stellt sich die Frage, wie Informatikstudenten in die Logik eingeführt werden sollten.

Dieses Skriptum bietet eine mögliche Antwort zu dieser Frage. Es entstand aus einer Vorlesung „Logik für Informatiker“, die in den Wintersemestern 1997/98 und 1998/99 den Studenten am Ende des Grundstudiums oder Anfang des Hauptstudiums angeboten wurde. Um ein möglichst breites Publikum – und nicht nur die Aficionados von Mathematik und Theorie – zu erreichen, wurde die Vorlesung auf zwei Wochenstunden (mit weiteren zwei Wochenstunden für Übungen) beschränkt[1]. Neben einem „Leitfaden“, der in die Syntax, Semantik und Beweistheorie der Aussagen- und Prädikatenlogik erster Stufe einführt, enthält diese Vorlesung „Exkurse“, die Anwendungen der Logik in der Informatik – z.B. in relationalen Datenbanken – oder Vertiefungen – z.B. Prädikatenlogik höherer Stufe – gewidmet sind. Zudem wird wichtigen Standardtechniken wie etwa Induktionsbeweisen und rekursiven Definitionen auf den Grund gegangen.

Der „Leitfaden“ ist in seiner Zusammensetzung ziemlich herkömmlich. Die gewählte Darstellung, die Informatiker ansprechen sollte, mag es aber weniger sein. Wenn möglich werden z.B. Beweise bevorzugt, die Algorithmen andeuten. Einige Themen werden nicht nur gründlich, sondern auch „anwendungsnah“ behandelt. So wird z.B. der guten Informatikpraxis gemäß die Skolemisierung für Formeln erläutert, die nicht notwendigwerweise in Pränexform sind. Die behandelten Beweismethoden sind sowohl anschaulich als auch zur Automatisierung geeignet.

Eine überschaubare und leichtverständliche Darstellung aller Anwendungen der Logik in der Informatik ist unmöglich. Deswegen wurden für diese Vorlesung Anwendungen ausgewählt auf Grund sowohl ihrer Bedeutung in der Informatik als auch ihrer Illustrationskraft zur Veranschaulichung logischer Begriffe. Zweifelsohne kann die hier getroffene Auswahl in Frage gestellt werden.

Welche Themen der Logik eine solche Vorlesung behandeln soll, ist eine weitere Frage, zu der es unterschiedliche Meinungen geben kann. Wenn in den Anfangsjahren der Informatik die Berechenbarkeitstheorie den Ehrenplatz unter den Anwendungen der Logik einnahm, später die Beweistheorie die Aufmerksamkeit der Informatiker erhielt, erscheint es heute angemessen, in einer Einführung in die Logik für Informatiker die bisher in der Informatik oft zu kurz gekommene elementare Modelltheorie zu betonen. Dafür gibt es zwei komplementäre Gründe. Zum einen werden die anderen Themen ohnehin in verschiedenen Vorlesungen behandelt: Einführungen in die Berechenbarkeitstheorie in der Einführungsvorlesung in die Theoretischen Informatik, die in jedem Informatikgrundstudium angeboten wird; Logikkalküle in Standardvorlesungen des Hauptstudiums z.B. über Deduktionssysteme oder über Grundlagen der logischen oder funktionalen Programmierung. Zum anderen findet die Modelltheorie vielfach Anwendung in der Informatik – u.a. in der Theorie der

[1] Wobei einige der „Exkurse“ nicht in der Vorlesung behandelt, sondern den Studenten zum eigenständigen Lernen angeboten wurden.

relationalen Datenbanken, zur Formalisierung von Prozessen und Programmabläufen und in der Diskurs- und Wissensrepräsentation. In einigen Fällen – z.B. mit den Modal- und Nichtmonotonen Logiken – waren sogar Informatik- oder Computerlinguistik-Anwendungen Anlass zu modelltheoretischen Entwicklungen.

Sei letztlich erwähnt, dass dieses Skriptum ein Entwurf ist, der in Inhalt sowie Gestalt von den Erfahrungen der kommenden Semester beeinflusst werden kann.

München, 15. März 1999
F.B. N.E.

Vorwort 2003

Gegenüber der Fassung von 1999 wurden die verschiedenen Logiken einheitlicher formalisiert sowie viele Unstimmigkeiten bereinigt. Für wertvolle Hinweise zur Verbesserung danken wir Frau Dr. Sunna Torge und Herrn Dr. Slim Abdennadher sowie ganz besonders Herrn Prof. Dr. Wilfried Buchholz, dessen Kommentare beträchtliche Vereinfachungen einiger Beweise ermöglichten.

München, 10. Oktober 2003
F.B. N.E.

Vorwort 2005

Die Exkurse über Prädikatenlogik höherer Stufen und über UML und OCL, die in den früheren Fassungen noch nicht enthalten waren, wurden hinzugefügt. Der Abschnitt über die Semantik von Modal- und Temporallogiken wurde umgestaltet und erweitert. Außerdem haben wir zahlreiche Hinweise von Hörern der Vorlesung eingearbeitet, für die wir uns herzlich bedanken.

München, 14. Oktober 2005
F.B. N.E.

Inhaltsverzeichnis

1 Einleitung

Die Logik als Formalisierung des rationalen Denkens. Die Logik ist entstanden zur Formalisierung des rationalen Denkens unabhängig von einem Bereich des Wissens. Sie beruht auf der Annahme, dass es universelle Gesetze des rationalen Denkens gibt, die für sämtliche Wissensbereiche gleichermaßen gelten. Die Logik setzt sich zum Ziel, diese Gesetze zu untersuchen und zu formalisieren.

Die Wichtigkeit der Logik wurde bereits in der Antike – vor allem bei den Griechen – erkannt. Ihre Hauptanwendung war ursprünglich die Rhetorik, für die sie formale Grundlagen liefern sollte. Später haben mathematische Fragestellungen die Entwicklung der Logik geprägt, weil die Mathematik wenig Möglichkeiten zum Experimentieren bietet und deshalb das rationale Denken in dieser Wissenschaft eine besonders hervorgehobene Stellung einnimmt. Diese Entwicklung hat sich um die Wende zum zwanzigsten Jahrhundert beschleunigt. Mit der Informatik hat die Logik in der zweiten Hälfte des zwanzigsten Jahrhunderts einen neuen, wichtigen Anwendungsbereich bekommen.

Sprachgebrauch von „Logik“ und „klassische Logik“. Unter einer „Logik“ versteht man im engeren Sinn eine formale Sprache zur Repräsentation von Aussagen. Eine Aussage ist nach Aristoteles eine sprachliche Äußerung, von der es sinnvoll ist, zu sagen, sie sei wahr oder falsch. Diese Charakterisierung trifft zwar heute nicht mehr für sämtliche Logiken zu (es gibt zum Beispiel Logiken zur Repräsentation von Fragen und Antworten), aber für die meisten. Die bekannteste Logik ist die Prädikatenlogik erster Stufe, die die Aussagenlogik als Spezialfall umfasst. Sie wird oft auch die „klassische Logik“ genannt.

In einem weiteren Sinn wird das Wort „Logik“ aber auch als Bezeichnung der Wissenschaft verwendet, die sich mit der Untersuchung von verschiedenen „Logiken“ (im engeren Sinn) beschäftigt. In diesem Sinn bezeichnet „klassische Logik“ auch das Gebiet, dessen Untersuchungsgegenstand die Prädikatenlogik erster Stufe ist.

Die zentrale Stellung der klassischen Logik. Neben der klassischen Prädikatenlogik erster Stufe gibt es viele nützliche sogenannte nichtklassische Logiken, die die Repräsentation mancher Konzepte erleichtern, z.B.

- modale Logiken, womit Aussagen mit Modalitäten wie Notwendigkeit, Möglichkeit, zeitliche Abhängigkeiten, usw. versehen werden können,
- temporale Logiken zur Repräsentation zeitlicher Abhängigkeiten (davor, danach, nie, immer, usw.),
- die intuitionistische Logik, die Existenzaussagen konstruktiv deutet.

Unter den vielen Logiken, die es gibt, nimmt die klassische Logik eine zentrale Stellung ein: Sie ist die meistbenutzte Logik; sie dient als Grundlage der Mathematik; andere Logiken werden meist in Bezug auf sie definiert. Warum ist das so? Zunächst aus historischen Gründen, aber auch weil die klassische Logik sich immer wieder als passendes Werkzeug erwiesen hat – auch in der Informatik. Die meisten Informatikforscher pflegen eine besondere Beziehung zur klassischen Logik: Sie halten sie für unzureichend, lösen sich aber nicht von ihr.

Notwendigkeit einer künstlichen Sprache. Die klassische Logik beruht wie jede Logik auf einer formalen Sprache, die viele Ähnlichkeiten mit Programmiersprachen aufweist. Aber warum kann man keine natürliche Sprache als Grundlage verwenden? Weil solche Sprachen oft mehrdeutig sind. Schon ein so einfaches Wort wie „und“ kann verschieden ausgelegt werden: In den Sätzen „er wurde krank und nahm eine Medizin“ und „er nahm eine Medizin und wurde krank“ kann die Konjunktion als Ausdruck von (verschiedenen!) Kausalitäten verstanden werden, wogegen im Satz „es regnet und der Wind weht“ die Konjunktion vermutlich keine Kausalität ausdrückt. Solche Beispiele sind zahlreich.

Die (formale) Sprache der Prädikatenlogik hat ihre Wurzel in der Antike. Sie wurde aber vorwiegend im 19. Jahrhundert entwickelt von Gottlob Frege (Wismar 1848 – Bad Kleinen 1925), Giuseppe Peano (Cuneo 1858 – Turin 1932) und Charles Sanders Pierce (Cambridge, USA 1839 – Milford, USA 1914), die unabhängig voneinander die Quantoren vorschlugen. Von einem Autor zum anderen weicht die Sprache der Prädikatenlogik erster Stufe geringfügig ab. Diese Abweichungen sind unwesentlich, können jedoch verwirren.

Schließen als Rechnen. Die zentrale Idee der Logik ist, dass Schlussfolgerungen berechnet werden können, ähnlich wie Größen aus arithmetischen Gleichungen. Insbesondere Gottfried Wilhelm Leibniz (Leipzig 1646 – Hannover 1716) hat diese Idee vermittelt. Diese Vorstellung mag dem modernen Mensch natürlich erscheinen, ist aber eine der größten Entdeckungen der Geschichte. Damit wird die enge Beziehung zwischen Logik und Mathematik verstärkt. Was Schlussfolgerungen sind und wie sie berechnet werden können, ist das zentrale Thema dieser Vorlesung.

Anwendungen von Logiken. Sowohl die Prädikatenlogik als auch nichtklassische Logiken finden Anwendung in vielen Bereichen. In der Philosophie werden Logiken – meist die Prädikatenlogik – wie in der Mathematik zum einen verwendet, wenn formale, eindeutige Beschreibungen nötig sind, zum anderen wenn Begriffe wie Bedeutung, Wahrheit, Modalitäten, usw. untersucht werden. In der Linguistik dienen die Prädikatenlogik sowie andere Logiken zur Formalisierung von natürlichen Sprachen. In der Informatik werden Logiken als Modellierungsprachen eingesetzt (künstliche Intelligenz, Entwurf von Schaltkreisen), zur Formalisierung der Semantik von Programmiersprachen, als Teile von Programmiersprachen (Boole'sche Ausdrücke) und sogar als ganze Programmiersprachen (Logikprogrammierung).

Literatur

H.-D. Ebbinghaus, J. Flum, W. Thomas: *Einführung in die mathematische Logik*, 3. Auflage, Wissenschaftsverlag, 1992

M. Fitting: *First-Order Logic and Automated Theorem Proving*, Springer-Verlag, 2nd Ed., 1996

U. Schöning: *Logik für Informatiker*, Spektrum akademischer Verlag, 1995

H. Schwichtenberg: *Mathematische Logik*, Vorlesungsskriptum, 1995
http://www.mathematik.uni-muenchen.de/personen/schwichtenberg.html

H. Schwichtenberg: *Mathematische Logik II*, Vorlesungsskriptum, 1995
http://www.mathematik.uni-muenchen.de/personen/schwichtenberg.html

2 Syntax

Dieses Kapitel führt formale Sprachen zur Repräsentation von „Aussagen“ ein. Den Schwerpunkt bilden dabei die Sprachen der Aussagenlogik und der Prädikatenlogik erster Stufe. Formale Darstellungen von „Aussagen“ in einigen anderen Logiken werden ebenfalls angesprochen, zum Beispiel in der mehrsortigen Prädikatenlogik erster Stufe, in der Prädikatenlogik zweiter Stufe und in der Modallogik.

Abzählbare und aufzählbare Mengen. Zwei grundlegende mathematische Begriffe kommen im folgenden häufig vor und sollen deshalb kurz rekapituliert werden.

Eine Menge S heißt *abzählbar*, wenn es eine Surjektion $\mathbb{N} \to S$ gibt. Die Surjektion $\mathbb{N} \to S$ muss keine Bijektion sein. Damit sind insbesondere auch endliche Mengen abzählbar (manche Autoren definieren „abzählbar“ so, dass der endliche Fall nicht eingeschlossen ist). Wir verwenden die Bezeichnung *abzählbar unendlich* für abzählbare Mengen, die nicht endlich sind.

Eine Menge S heißt *rekursiv aufzählbar* oder kurz *aufzählbar*,wenn es einen Algorithmus $\mathit{aufzählung}_S$ mit den folgenden Eigenschaften gibt:

1. $\mathit{aufzählung}_S$ hat eine natürliche Zahl als Argument.
2. $\mathit{aufzählung}_S(i)$ terminiert für jedes $i \in \mathbb{N}$.
3. $\mathit{aufzählung}_S(i)$ liefert als Ergebnis ein Element aus S.
4. Für jedes $s \in S$ gibt es mindestens ein $i \in \mathbb{N}$, so dass $\mathit{aufzählung}_S(i)$ das Element s als Ergebnis liefert.

Man sagt, dass ein solcher Algorithmus die Menge S aufzählt. Die Bedingungen 1, 3 und 4 bedeuten, dass der Algorithmus eine Surjektion $\mathbb{N} \to S$ berechnet. Jede aufzählbare Menge ist also abzählbar. Bei einer aufzählbaren Menge muss die Surjektion durch einen Algorithmus berechenbar sein, bei einer abzählbaren dagegen nicht notwendigerweise.

Zum Beispiel ist für eine „vernünftige“ Programmiersprache die Menge aller syntaktisch korrekten Programme dieser Sprache abzählbar und aufzählbar. Die Teilmenge aller syntaktisch korrekten Programme, die für manche Eingaben nicht terminieren, ist zwar ebenfalls abzählbar, aber nicht aufzählbar.

Wie der Begriff „Algorithmus“ formalisiert werden kann, wird in dieser Vorlesung nicht näher präzisiert.

2.1 Aussagenlogik

Mit der Aussagenlogik werden hauptsächlich logische Verknüpfungen wie „nicht“, „und“, „oder“ untersucht. Zur syntaktischen Repräsentation dieser Verknüpfungen von Aussagen dienen sogenannte Junktoren: $\neg$ für die einstellige Verknüpfung „nicht“, $\wedge$ für die zweistellige Verknüpfung „und“, $\vee$ für „oder“, $\Rightarrow$ für „wenn dann“, $\Leftrightarrow$ für „genau dann wenn“.

Die einfachsten Aussagen, die damit verknüpft werden können, sind die konstant wahre Aussage, repräsentiert durch $\top$ (gesprochen „top“ oder auch „verum“) und die konstant falsche Aussage, repräsentiert durch $\bot$ (gesprochen „bottom“ oder auch „falsum“). Man nennt $\top$ und $\bot$ die nullstelligen Junktoren.

Welche weiteren Aussagen in der formalen Sprache repräsentiert werden müssen, hängt von der Anwendung ab, für die die Sprache verwendet wird. Es gibt deshalb nicht „die“

Sprache der Aussagenlogik, sondern verschiedene Sprachen der Aussagenlogik, die sich in ihrem anwendungsspezifischen Teil unterscheiden. Den anwendungsspezifischen Teil einer Sprache einer Logik nennt man üblicherweise die *Signatur* der Sprache.

Die Signatur einer Sprache der Aussagenlogik definiert sogenannte „Aussagensymbole". Jedes Aussagensymbol dient zur syntaktischen Repräsentation irgend einer Aussage, die wahr oder falsch sein kann, wie etwa „es regnet" oder „Franz ist krank". Die durch Aussagensymbole repräsentierten Aussagen werden als elementar betrachtet, das heißt, dass die Aussagenlogik keine sprachlichen Mittel zur Darstellung der inneren Struktur solcher Aussagen bietet. Beispielsweise ist einer aussagenlogischen Repräsentation nicht zu entnehmen, dass die Aussage „Franz ist krank" mit der Aussage „Hans ist krank" mehr gemeinsam hat als mit der Aussage „es regnet".

Definition 2.1.1 (Sprache [Aussagenlogik]). Eine Sprache $\mathcal{L}$ der Aussagenlogik ist gegeben durch ihre Signatur. Zusätzlich gehören zu $\mathcal{L}$ die logischen Symbole der Aussagenlogik.

- Die *logischen Symbole* der Aussagenlogik sind:
  1. Die Menge der *Hilfszeichen*) und (
  2. Die Menge der nullstelligen *Junktoren*: $\{\top, \bot\}$.
     Die Menge der einstelligen Junktoren: $\{\neg\}$.
     Die Menge der zweistelligen Junktoren: $\{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$.
- Die *Signatur* einer Sprache $\mathcal{L}$ der Aussagenlogik besteht aus einer einzigen Symbolmenge $Rel^0_{\mathcal{L}}$: einer Menge von *Aussagensymbolen*.
  Diese heißen auch *nullstellige Relationssymbole* oder *nullstellige Prädikatssymbole*.

Die Menge der Aussagensymbole ist abzählbar (üblicherweise sogar aufzählbar) und mit jeder der anderen erwähnten Mengen disjunkt, und kein Aussagensymbol besteht aus einer Sequenz von Symbolen, in der ein Symbol aus einer der anderen Mengen vorkommt. ■

Die alternativen Bezeichnungen „nullstellige Relations-/Prädikatssymbole" für Aussagensymbole haben ihren Grund darin, dass sie ein Spezialfall der entsprechenden Symbole der Prädikatenlogik erster Stufe sind. Manche Autoren verwenden die Bezeichnung „Aussagenvariablen", die aber im Zusammenhang mit der Prädikatenlogik erster oder gar zweiter Stufe Missverständnisse provoziert: in diesen Logiken gehören „Variablen" zu einer völlig anderen syntaktischen Kategorie als Aussagensymbole (siehe Abschnitt 2.3 und Abschnitt 2.11).

In Kapitel 3 werden wir sehen, dass auch andere Junktoren möglich sind als die in Definition 2.1.1 genannten. Das ist der Grund, warum manche Autoren mehr oder weniger Junktoren einführen. Die Schreibweise der Junktoren variiert auch ein wenig von Autor zu Autor: zum Beispiel wird der Implikationsjunktor statt $\Rightarrow$ manchmal $\supset$ geschrieben.

Als Aussagensymbole dienen häufig Großbuchstaben mit oder ohne Indizes. In manchen Anwendungen benutzt man für die Aussagensymbole auch Bezeichner wie in Programmiersprachen. Wir werden in Beispielen meistens Kleinbuchstaben wie p, q, r als Aussagensymbole verwenden.

Definition 2.1.2 (Formel [Aussagenlogik]). Sei $\mathcal{L}$ eine Sprache der Aussagenlogik.

- Jedes Aussagensymbol von $\mathcal{L}$ ist eine *atomare $\mathcal{L}$-Formel*.
  Eine atomare $\mathcal{L}$-Formel nennt man auch kurz *atomare Formel* oder einfach *Atom*.

- Die Menge $\mathcal{F}_\mathcal{L}$ der $\mathcal{L}$-Formeln, kurz Formeln, ist die kleinste Menge, die die folgenden Bedingungen erfüllt:
  1. $\top$ und $\bot$ und alle atomaren $\mathcal{L}$-Formeln sind in $\mathcal{F}_\mathcal{L}$.
  2. Ist $F \in \mathcal{F}_\mathcal{L}$, so ist auch $\neg F \in \mathcal{F}_\mathcal{L}$.
  3. Ist $F \in \mathcal{F}_\mathcal{L}$ und $G \in \mathcal{F}_\mathcal{L}$ und θ ein zweistelliger Junktor, so ist auch $(F\ \theta\ G) \in \mathcal{F}_\mathcal{L}$.

■

Informell und ungenau, weil dabei die Klammerung implizit bleibt, kann die Definition auch so formuliert werden: Sind $F_1, \dots, F_n$ (für $n = 0, 1, 2$) $\mathcal{L}$-Formeln, so sind es auch ihre Verknüpfungen mit n-stelligen Junktoren.

Zum Beispiel ist $(p \vee \neg(q \wedge r))$ eine $\mathcal{L}$-Formel für jede Sprache $\mathcal{L}$ der Aussagenlogik, deren Signatur die Symbole p, q und r als Aussagensymbole definiert. $\neg(\bot \Rightarrow \top)$ ist eine $\mathcal{L}$-Formel jeder Sprache $\mathcal{L}$ der Aussagenlogik.

Bemerkungen:

1. In der Terminologie der Theoretischen Informatik würde $\mathcal{L}$ nicht als „Sprache" bezeichnet, sondern als „Alphabet", und die Menge $\mathcal{F}_\mathcal{L}$ der $\mathcal{L}$-Formeln als eine „(formale) Sprache" über $\mathcal{L}$.
2. $\top$ und $\bot$ sind besondere, nicht zusammengesetzte $\mathcal{L}$-Formeln. Wir haben sie nicht als atomare Formeln definiert (eine willkürliche Entscheidung, die aber später gewisse technische Sonderfälle vermeidet).
3. Die Menge $\mathcal{F}_\mathcal{L}$ aller $\mathcal{L}$-Formeln ist abzählbar, weil nach Definition die Menge der Aussagensymbole von $\mathcal{L}$ abzählbar ist. Ist diese Menge darüberhinaus aufzählbar, so ist auch $\mathcal{F}_\mathcal{L}$ aufzählbar. ■

Existiert die „kleinste Menge", von der in Definition 2.1.2 die Rede ist? Aus den folgenden Gründen lautet die Antwort „ja".

Zum einen gibt es überhaupt eine Menge, die die Bedingungen 1 bis 3 der Definition 2.1.2 erfüllt, nämlich die Menge aller Wörter, die aus den Symbolen unserer Sprache $\mathcal{L}$, also aus $\top$, $\bot$, Aussagensymbolen, Junktoren und Klammern, gebildet werden können (siehe Vorlesung „Informatik IV").

Zum anderen erfüllt der Durchschnitt jeder Menge von Mengen, die die Bedingungen 1 bis 3 von Definition 2.1.2 erfüllen, ebenfalls diese Bedingungen. Der Durchschnitt der Menge aller Mengen, die die Bedingungen 1 bis 3 erfüllen, ist also die kleinste Menge, von der Definition 2.1.2 spricht. Jede Menge, die die Bedingungen 1 bis 3 erfüllt, enthält unter anderem das Element $\top$, deshalb ist die kleinste Menge nicht leer.

Prinzip der induktiven Definition. Das Prinzip, nach dem die $\mathcal{L}$-Formeln definiert werden, heißt „induktive Definition". Die Definition von Formeln der Aussagenlogik und später von Termen und Formeln anderer Logiken ist nach folgendem Schema ausgedrückt:

> „die Menge der X ist die kleinste Menge, die die folgenden Bedingungen erfüllt: …".

Statt dieses Schemas wird oft auch wie folgt formuliert:

> „X werden induktiv definiert durch folgende Bedingungen: …"
> wobei manche (vorsichtigen) Autoren am Ende noch hinzufügen:
> „nichts anderes ist ein(e) X".

Beide Ausdrucksweisen bedeuten das Gleiche. Genauer gesagt: die zweite Ausdrucksweise ist mittels der ersten definiert, die auch in Definition 2.1.2 gewählt wurde. Eine Definition der Menge $\mathcal{F}_\mathcal{L}$ aller $\mathcal{L}$-Formeln durch eine kontextfreie Grammatik mit Alphabet $\mathcal{L}$ ist eine ähnliche Definition, da die Menge aller von der Grammatik erzeugten Wörter ebenfalls induktiv definiert ist.

Wir überzeugen uns nun, dass tatsächlich „nichts anderes" eine $\mathcal{L}$-Formel sein kann als das, was in den drei Fällen der induktiven Definition vorkommt.

Satz 2.1.3 (Formelgestalt). Sei $\mathcal{L}$ eine Sprache der Aussagenlogik. Jede $\mathcal{L}$-Formel hat eine der Gestalten $\top$ oder $\bot$ oder atomar oder $\neg G$ für eine $\mathcal{L}$-Formel G oder $(G\ \theta\ H)$ für $\mathcal{L}$-Formeln G und H und einen zweistelligen Junktor θ.

Beweis: Angenommen, das wäre nicht der Fall, und es gäbe eine $\mathcal{L}$-Formel $F \in \mathcal{F}_\mathcal{L}$, die weder die Gestalt $\top$ oder $\bot$ oder atomar besitzt, noch die Gestalt $\neg G$, noch die Gestalt $(G\ \theta\ H)$. Dann betrachten wir die Menge $\mathcal{F}'_\mathcal{L} = \mathcal{F}_\mathcal{L} \setminus \{F\}$. Sie hat folgende Eigenschaften:

(i) $\top$, $\bot$ sowie alle atomaren Formeln sind in $\mathcal{F}'_\mathcal{L}$, da F keine dieser Gestalten hat.

(ii) Sei $G \in \mathcal{F}'_\mathcal{L}$, dann ist auch $\neg G \in \mathcal{F}'_\mathcal{L}$, da F nicht diese Gestalt hat.

(iii) Seien $G \in \mathcal{F}'_\mathcal{L}$ und $H \in \mathcal{F}'_\mathcal{L}$ und θ ein zweistelliger Junktor, dann ist auch $(G\ \theta\ H) \in \mathcal{F}'_\mathcal{L}$, da F nicht diese Gestalt hat.

Damit erfüllt $\mathcal{F}'_\mathcal{L}$ die Eigenschaften 1 bis 3 von Definition 2.1.2 und ist andererseits eine echte Teilmenge von $\mathcal{F}_\mathcal{L}$, im Widerspruch dazu, dass $\mathcal{F}_\mathcal{L}$ die kleinste derartige Menge ist. Demnach ist die Annahme falsch. Also hat jede $\mathcal{L}$-Formel eine der genannten Gestalten. ∎

Mit diesem Ergebnis ist aber noch nicht ausgeschlossen, dass eine $\mathcal{L}$-Formel mehrere dieser Gestalten haben könnte, zum Beispiel durch verschiedene Klammerungen bei verschachtelten zweistelligen Junktoren. Um die Eindeutigkeit der Gestalt von Formeln nachweisen zu können, müssen wir erst eine spezielle Beweistechnik einführen, das sogenannte Prinzip der strukturellen Induktion, das eng mit induktiven Definitionen verwandt ist.

Satz 2.1.4 (strukturelle Induktion). Sei $\mathcal{L}$ eine Sprache der Aussagenlogik. Um zu zeigen, dass alle $\mathcal{L}$-Formeln eine Eigenschaft $\mathcal{E}$ haben, genügt es, zu zeigen:

1. **Basisfälle:** $\top$ und $\bot$ sowie alle atomaren $\mathcal{L}$-Formeln besitzen die Eigenschaft $\mathcal{E}$.
2. **Induktionsfälle:**
   a) Wenn eine $\mathcal{L}$-Formel F die Eigenschaft $\mathcal{E}$ besitzt, so besitzt auch die Formel $\neg F$ die Eigenschaft $\mathcal{E}$.
   b) Wenn zwei $\mathcal{L}$-Formeln F_1 und F_2 die Eigenschaft $\mathcal{E}$ besitzen, dann besitzt für jeden zweistelligen Junktor θ auch die Formel $(F_1\ \theta\ F_2)$ die Eigenschaft $\mathcal{E}$.

Beweis: Sei ${\mathcal{F}_\mathcal{L}}^\mathcal{E} \subseteq \mathcal{F}_\mathcal{L}$ die Menge aller $\mathcal{L}$-Formeln mit Eigenschaft $\mathcal{E}$. Die genannten Basisfälle und Induktionsfälle seien für $\mathcal{E}$ erfüllt. Dann ist ${\mathcal{F}_\mathcal{L}}^\mathcal{E}$ eine Menge, die die drei Eigenschaften von Definition 2.1.2 erfüllt. Da $\mathcal{F}_\mathcal{L}$ aber die kleinste derartige Menge ist, gilt $\mathcal{F}_\mathcal{L} \subseteq {\mathcal{F}_\mathcal{L}}^\mathcal{E}$, also ${\mathcal{F}_\mathcal{L}}^\mathcal{E} = \mathcal{F}_\mathcal{L}$. ∎

Im Fall 2(b) ist das Wort „zwei“ so gemeint, dass F_1 und F_2 nicht notwendigerweise verschieden sind. Dies ist ein typisches Beispiel für die Art von Mehrdeutigkeiten, die formale Sprachen vermeiden sollen.

Die Argumentation im obigen Beweis beruht, wie im Beweis des Satzes zuvor, wesentlich auf der in der induktiven Definition geforderten Minimalität der Menge. Bevor wir das neue Beweisprinzip seinerseits verwenden, (u.a. zum Beweis des Eindeutigkeitssatzes), gehen wir noch kurz auf eine Verallgemeinerung ein.

Definition 2.1.5. Sei S eine Menge und sei Rel eine Menge von Relationen über S. Eine Teilmenge $T \subseteq S$ heißt abgeschlossen bezüglich Rel, wenn für alle $R \in Rel$ und für alle $x_1, \ldots, x_{n-1}, y \in S$ gilt: wenn $x_1 \in T, \ldots, x_{n-1} \in T$ und $(x_1, \ldots, x_{n-1}, y) \in R$ ist, dann ist auch $y \in T$. ■

Zur Veranschaulichung sei S die Menge der natürlichen Zahlen und T die Menge der geraden Zahlen. Die Relation R sei die Menge aller Tripel (x_1, x_2, y) von natürlichen Zahlen, für die $x_1 + x_2 = y$ ist. Dann ist T abgeschlossen bezüglich $\{R\}$, weil die Summe von zwei geraden Zahlen nicht nur eine beliebige natürliche Zahl, sondern eine gerade Zahl ist. Dagegen ist die Menge der ungeraden Zahlen nicht abgeschlossen bezüglich $\{R\}$.

Satz 2.1.6 (strukturelle Induktion, Verallgemeinerung). Sei S eine Menge, B (für Basis) eine Teilmenge von S, und Rel eine Menge von Relationen über S. Sei T eine Menge mit $B \subseteq T \subseteq S$, sei T abgeschlossen bezüglich Rel, und es gelte ferner $T \subseteq \bigcup_{i \in \mathbb{N}} B_i$ wobei

$$\begin{array}{lll} B_0 & := & B \\ B_{i+1} & := & B_i \cup \bigcup_{R \in Rel} \{y \mid x_1 \in B_i, \ldots, x_{n-1} \in B_i,\ (x_1, \ldots, x_{n-1}, y) \in R\} \end{array}$$

Dann gilt: $T = \bigcup_{i \in \mathbb{N}} B_i$.

Beweis: Es reicht, nachzuweisen dass für jedes $i \in \mathbb{N}$ gilt $B_i \subseteq T$. Diesen Nachweis führen wir durch vollständige Induktion über i.

Induktionsbasis $i = 0$: Nach Voraussetzung gilt $B_0 = B$ und $B \subseteq T$. Also ist $B_0 \subseteq T$.

Induktionsschritt $i \to i + 1$: Sei $B_i \subseteq T$. Nach Definition besteht B_{i+1} aus B_i sowie zusätzlichen Elementen $y \in S$. Da T abgeschlossen bezüglich Rel ist, gilt für jedes dieser Elemente dass $y \in T$ ist. Also ist $B_{i+1} \subseteq T$. ■

Das verallgemeinerte Beweisprinzip der strukturellen Induktion wird also auf das herkömmliche Beweisprinzip der vollständigen Induktion über die natürlichen Zahlen zurückgeführt, das vorausgesetzt wird. Die herkömmliche vollständige Induktion ist Teil der Axiomatisierung der natürlichen Zahlen – siehe Abschnitt 3.10.

Auf die $\mathcal{L}$-Formeln wird dieses verallgemeinerte Prinzip anwendbar für den Sonderfall $S =$ Menge der $\mathcal{L}$-Formeln, $B =$ Menge der atomaren $\mathcal{L}$-Formeln zusammen mit $\top$ und $\bot$, $Rel = \{R_\neg, R_\wedge, R_\vee, R_\Rightarrow, R_\Leftrightarrow\}$ mit $R_\neg = \{(F, \neg F) \mid F \in \mathcal{F}_\mathcal{L}\}$ und $R_\wedge = \{(F, G, (F \wedge G)) \mid F \in \mathcal{F}_\mathcal{L}$ und $G \in \mathcal{F}_\mathcal{L}\}$ usw., und $T =$ Menge der $\mathcal{L}$-Formeln mit Eigenschaft $\mathcal{E}$.

Doch nun zu der Frage, ob die Gestalt von aussagenlogischen Formeln eindeutig bestimmt ist.

Satz 2.1.7 (Eindeutigkeitssatz [Formel, Aussagenlogik]). Sei $\mathcal{L}$ eine Sprache der Aussagenlogik. Für jede $\mathcal{L}$-Formel F gilt genau eine der folgenden Aussagen:

1. $F = \top$ oder $F = \bot$ oder F ist atomar.
2. Es gibt eine eindeutige $\mathcal{L}$-Formel G, so dass $F = \neg G$.
3. Es gibt ein eindeutiges Paar (G, H) von $\mathcal{L}$-Formeln und einen eindeutigen zweistelligen Junktor θ, so dass $F = (G\ \theta\ H)$.

Beweis: Da F in jedem der Fälle mit einem anderen Symbol beginnt, gilt trivialerweise für jede $\mathcal{L}$-Formel höchstens eine der drei Aussagen. Um zu zeigen, dass mindestens eine der drei Aussagen auf F zutrifft, verwenden wir Satz 2.1.3. Demnach hat F eine der Gestalten, die in den drei Aussagen vorkommen.

- Hat F die Gestalt $F = \top$ oder $F = \bot$ oder F ist atomar, dann gilt Aussage 1 für F.
- Hat F die Gestalt $F = \neg G$ für eine $\mathcal{L}$-Formel G, bleibt noch die Eindeutigkeit nachzuweisen. Sei also außerdem $F = \neg G'$ für eine $\mathcal{L}$-Formel G'. Wenn G und G' verschieden wären, entstünden durch Voranstellen des selben Symbols $\neg$ vor beide Wörter wieder verschiedene Wörter, d.h., $\neg G$ und $\neg G'$ wären verschieden im Widerspruch dazu, dass beide gleich F sind. Also ist $G = G'$. Damit gilt Aussage 2 für F.
- Hat F die Gestalt $F = (G\ \theta\ H)$ für $\mathcal{L}$-Formeln G und H und einen zweistelligen Junktor θ, so bleibt noch die Eindeutigkeit für diesen Fall nachzuweisen, und dann gilt Aussage 3 für F.

Wenn dieser letzte Nachweis erbracht ist, ist Satz 2.1.7 bewiesen.

Dieser Nachweis ist nicht schwierig, aber etwas mühsam. Wir verwenden folgende Hilfsmittel: für ein Wort H, also eine Sequenz von Symbolen, seien $\ell(H)$ bzw. $r(H)$ die Anzahl der in H vorkommenden linken (öffnenden) Klammern bzw. rechten (schließenden) Klammern. Unter einem Präfix einer Formel verstehen wir ein nichtleeres Wort, mit dem die Formel beginnt. Ein echter Präfix einer Formel ist ein Präfix, der nicht gleich der Formel ist.

(a) Jede $\mathcal{L}$-Formel hat die Eigenschaft, gleich viele linke wie rechte Klammern zu enthalten, das heißt, für alle $H \in \mathcal{F}_{\mathcal{L}}$ ist $\ell(H) = r(H)$.

Durch strukturelle Induktion.
Basisfälle: $\top$, $\bot$ und atomare $\mathcal{L}$-Formeln enthalten weder linke noch rechte Klammern, also gleich viele.
Induktionsfall Negation: Es gelte $\ell(F) = r(F)$. Da $\ell(\neg F) = \ell(F)$ und $r(\neg F) = r(F)$ ist, gilt dann auch $\ell(\neg F) = r(\neg F)$.
Induktionsfall zweistelliger Junktor θ: Es gelte $\ell(F) = r(F)$ sowie $\ell(G) = r(G)$. Wegen $\ell(\ (F\theta G)\ ) = 1 + \ell(F) + \ell(G)$ und $r(\ (F\theta G)\ ) = 1 + r(F) + r(G)$ gilt dann auch $\ell(\ (F\theta G)\ ) = r(\ (F\theta G)\ )$.

(b) Jede $\mathcal{L}$-Formel hat die Eigenschaft, ein vom Negationssymbol verschiedenes Symbol zu enthalten.

Durch strukturelle Induktion.
Basisfälle: $\top$, $\bot$ und atomare $\mathcal{L}$-Formeln bestehen jeweils aus einem vom Negationssymbol verschiedenen Symbol.

Induktionsfall Negation: F enthalte ein vom Negationssymbol verschiedenes Symbol. Dann enthält auch $\neg F$ dieses Symbol.
Induktionsfall zweistelliger Junktor θ: unabhängig von F und G enthält $(F\ \theta\ G)$ das vom Negationssymbol verschiedene Symbol (.

(c) Ein echter Präfix P einer $\mathcal{L}$-Formel ist entweder eine Sequenz von Negationssymbolen oder es gilt $\ell(P) > r(P)$.

Durch strukturelle Induktion.
Basisfälle: Die Behauptung gilt für $\top$, $\bot$ und atomare $\mathcal{L}$-Formeln, da diese gar keinen echten Präfix haben.
Induktionsfall Negation: Für F gelte, dass jeder echte Präfix Q eine Sequenz von Negationssymbolen ist oder $\ell(Q) > r(Q)$ ist. Sei P ein echter Präfix von $\neg F$. Ist $P = \neg$, so ist P eine Sequenz von Negationssymbolen und die Behauptung gilt. Ist $P = \neg Q$ für einen echten Präfix Q von F, so ist entweder P eine Sequenz von Negationssymbolen oder $\ell(P) = \ell(Q) > r(Q) = r(P)$ und die Behauptung gilt für $\neg F$.
Induktionsfall zweistelliger Junktor θ: Für F und G gelte, dass jeder echte Präfix Q eine Sequenz von Negationssymbolen ist oder $\ell(Q) > r(Q)$ ist. In jedem der beiden Fälle gilt also $\ell(Q) \geq r(Q)$
Sei P ein echter Präfix von $(F\ \theta\ G)$. Ist $P =$ (, so gilt $\ell(P) = 1 > 0 = r(P)$. Ist $P = (Q$ mit Q echter Präfix von F, so gilt $\ell(P) = 1+\ell(Q) \geq 1+r(Q) > r(Q) = r(P)$. Ist $P = (F$ oder $P = (F\theta$, so ist $\ell(P) = 1 + \ell(F) = 1 + r(F) > r(F) = r(P)$. Ist $P = (F\theta Q$ mit Q echter Präfix von G, so ist $\ell(P) = 1+\ell(F)+\ell(Q) = 1+r(F)+\ell(Q) \geq 1 + r(F) + r(Q) > r(F) + r(Q) = r(P)$. Stets gilt also $\ell(P) > r(P)$ und damit die Behauptung für $(F\ \theta\ G)$.

(d) Ein echter Präfix einer $\mathcal{L}$-Formel ist keine $\mathcal{L}$-Formel.

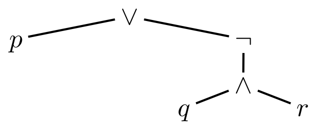
Dies ergibt sich unmittelbar aus (c), (b) und (a).

(e) Ist eine $\mathcal{L}$-Formel von der Gestalt $(G_1\ \theta_1\ H_1)$ sowie von der Gestalt $(G_2\ \theta_2\ H_2)$, so gilt $G_1 = G_2$ und $\theta_1 = \theta_2$ und $H_1 = H_2$.

Sonst wäre G_1 ein echter Präfix von G_2 oder G_2 ein echter Präfix von G_1, im Widerspruch zu (d).

Damit ist die gewünschte Eindeutigkeit für die dritte Aussage von Satz 2.1.7 nachgewiesen. ■

Der Satz 2.1.7 besagt, dass die in der Definition 2.1.2 implizite Grammatik eindeutig ist, also jeder $\mathcal{L}$-Formel genau einen Syntaxbaum zuordnet. Diese Eindeutigkeit berechtigt dazu, aussagenlogische Formeln in der vertrauten Weise durch Bäume darzustellen, wie z.B. die Formel $(p \vee \neg(q \wedge r))$ durch den Baum:



Die Eindeutigkeit wird wesentlich durch die Klammerung von Formeln der Gestalt $(F\ \theta\ G)$ erreicht. Ohne diese Klammern wäre zum Beispiel die syntaktische Struktur der Symbolfolge $p \wedge q \vee r$ nicht eindeutig.

Warum will man die Syntax so definieren, dass der Eindeutigkeitssatz gilt? Wenn die Syntax mehrdeutig ist, gilt dies auch für die Semantik. Ohne eindeutige Syntax hätte man also auch keine eindeutige Bedeutung. Mehrdeutigkeiten können Missverständnisse verursachen. Der Zweck einer formalen Sprache besteht aber gerade darin, Missverständnisse auszuschließen. Mehrdeutigkeiten erschweren überdies die Automatisierung oder machen sie sogar ganz unmöglich.

Für die natürliche Sprache gilt kein Eindeutigkeitssatz. Der englische Satz „The male guides fish.“ hat zum Beispiel zwei völlig verschiedene syntaktische Strukturen, bei denen die Wörter unterschiedlichen Wortarten zugeordnet werden: zum einen die Struktur Subjekt-Verb-Objekt mit dem Verb „to guide“ und der Bedeutung „Das Männchen führt Fische“, zum anderen die Struktur Subjekt-Verb mit dem Verb „to fish“ und der Bedeutung „Die männlichen Führer fischen“. Im Deutschen sind solche Beispiele wegen der Flexionsendungen und der Groß-/Kleinschreibung seltener, aber es gibt sie, zum Beispiel die Frage: „Fliegen sieben Bienen?“

Meistens bevorzugt man in solchen Beispielen zwar jeweils eine der Lesarten, aber nicht aus syntaktischen Gründen, sondern aus semantischen.

Definition 2.1.8 (Teilformel). Sei $\mathcal{L}$ eine Sprache der Aussagenlogik. Die *unmittelbaren Teilformeln* einer $\mathcal{L}$-Formel werden wie folgt definiert:

1. $\top$, $\bot$ und atomare $\mathcal{L}$-Formeln haben keine unmittelbare Teilformel.
2. F ist die einzige unmittelbare Teilformel von $\neg F$.
3. F und G sind die einzigen unmittelbaren Teilformeln von $(F \ \theta \ G)$, wobei θ ein zweistelliger Junktor ist.

Die Menge der *Teilformeln* einer $\mathcal{L}$-Formel F ist die kleinste Menge, die F enthält sowie alle unmittelbaren Teilformeln aller ihrer Elemente. ■

Die Teilformeln einer Formel F können auch als die Teilwörter von F definiert werden, die $\mathcal{L}$-Formeln sind. In der Baumdarstellung einer Formel entsprechen die Teilformeln den Teilbäumen und die unmittelbaren Teilformeln den unmittelbaren Teilbäumen (also denen, deren Wurzel ein Kindknoten der Gesamtwurzel ist).

Induktive Definitionen und Unendlichkeit. Für jede Sprache $\mathcal{L}$ der Aussagenlogik enthält die Menge $\mathcal{F}_{\mathcal{L}}$ unendlich viele Elemente. Tatsächlich ist die Spezifikation von unendlichen Mengen einer der Hauptzwecke von induktiven Definitionen: wenn man die durch die Basisfälle gegebenen Elemente in die Induktionsfälle einsetzt, erhält man weitere Elemente, die man wieder in die Induktionsfälle einsetzen kann, um weitere Elemente zu erhalten, und so weiter.

Aus dieser Unendlichkeit der Gesamtmenge folgt aber nicht, dass ihre Elemente unendlich groß wären. Im Gegenteil kann man ganz einfach durch strukturelle Induktion zeigen, dass jede $\mathcal{L}$-Formel aus endlich vielen Symbolen besteht. Eine unendlich große Kombination von Junktoren kann also keine $\mathcal{L}$-Formel sein, auch wenn sie Stelligkeiten, Klammerung usw. nicht verletzt. Solche unendlichen Gebilde können zwar durchaus in Mengen vorkommen, die die Bedingungen 1 bis 3 von Definition 2.1.2 erfüllen, aber eben nicht in der kleinsten derartigen Menge.

Dieser Unterschied ist typisch für Mengen, die mit dem hier betrachteten Prinzip der induktiven Definition gebildet werden können: die Menge aller Wörter über einem Alphabet enthält unendlich viele Wörter, aber jedes davon besteht nur aus endlich vielen Zeichen; die Menge aller Binärbäume enthält unendlich viele Bäume, aber jeder davon besteht nur aus endlich vielen Knoten und Kanten; die Menge aller natürlichen Zahlen enthält unendlich viele Zahlen, aber jede davon ist endlich.

Wenn man unendlich große Elemente zulassen will, benötigt man ein allgemeineres Prinzip namens *transfinite Induktion*, das neben Basisfällen und Induktionsfällen auch sogenannte Limesfälle berücksichtigt.

Meta- und Objektsprachen. Eine Aussage wie „wenn eine Formel die Gestalt $(F \Rightarrow G)$ hat, dann sind F und G eindeutig bestimmt“ enthält eine Implikation in der Objektsprache, in der wir Formeln bilden können, und eine andere Implikation in der Metasprache, in der wir über Formeln und andere Dinge der Objektsprache reden. Man darf die beiden Sprachebenen nicht durcheinanderbringen. Wir reservieren deshalb Symbole wie $\Rightarrow$ für die Objektsprache und benutzen in der Metasprache nie Symbole, sondern nur verbale Umschreibungen wie „wenn … dann …“.

Wenn eine Objektsprache z.B. Integrale oder Halbgruppen oder Wahrscheinlichkeitsverteilungen beschreibt, kommen Junktoren und Quantoren nur in der Metasprache vor, und man benutzt dann üblicherweise die Symbole auf der Ebene der Metasprache. In diesen Fällen kann kein Konflikt mit den Symbolen der Objektsprache entstehen. Die Logik ist das einzige wissenschaftliche Gebiet, in dem die Symbole für Junktoren und Quantoren in der Objektsprache vorkommen und deshalb nicht mehr für die Metasprache zur Verfügung stehen, also das einzige Gebiet, in dem das Problem eines Konflikts zwischen den Symbolen überhaupt auftritt.

2.2 Exkurs: Präfix- und Postfixnotation und Präzedenzen

In der eingeführten Syntax dienen die Klammern offensichtlich dem Zweck, dass der Eindeutigkeitssatz gilt. Es gibt Alternativen, die den selben Zweck ohne Klammern erreichen: die Präfixnotation, die Postfix- oder polnische Notation und Präzedenzen.

Die Präfixnotation. Statt $(F\theta G)$ wird geschrieben: θFG. Der einstellige Junktor wird wie bisher geschrieben, also $\neg F$.

Die Stelligkeit jedes Junktors lässt eindeutig erkennen, wieviele Teilformeln als Argumente auf den Junktor folgen. Diese Notation ist weniger gut lesbar als die Infixnotation.

Die Postfixnotation. Statt $(F\theta G)$ wird geschrieben: $FG\theta$. Statt $\neg F$ wird geschrieben: $F\neg$.

Auch hier ist aus der Stelligkeit der Junktoren eindeutig rekonstruierbar, was jeweils die Argumente sind, und auch hier ist die Lesbarkeit schlechter als mit der Infixnotation. Die Postfixnotation heißt auch polnische Notation, weil sie von dem polnischen Logiker Jan Łukasiewicz (Lvov 1878 – Dublin 1956) stammt.

Präzedenzen. Klammern können auch dadurch vermieden werden, dass Präzedenzen zwischen den Junktoren festgelegt werden. Dabei handelt es sich um eine Kurzschreibweise, deren präzise Bedeutung mit Hilfe der Schreibweise mit Klammern erklärt werden muss.

2.3 Prädikatenlogik erster Stufe

Die formale Sprache der Prädikatenlogik erster Stufe, abgekürzt PL1S, benutzt die selben Junktoren $\top$, $\bot$, $\neg$, $\wedge$, $\vee$, $\Rightarrow$ und $\Leftrightarrow$ wie die formale Sprache der Aussagenlogik, aber sie bietet sprachliche Mittel zur Darstellung der inneren Struktur von elementaren Aussagen.

Elementare Aussagen werden wie in der Aussagenlogik durch atomare Formeln repräsentiert, die aber aus mehreren Bestandteilen zusammengesetzt sein können. Die Prädikatenlogik erster Stufe benutzt sogenannte „Terme" zur syntaktischen Repräsentation von Objekten und „Relationssymbole" zur syntaktischen Repräsentation von Eigenschaften von und Beziehungen zwischen Objekten. Terme können ihrerseits aus mehreren Bestandteilen zusammengesetzt sein.

Zum Beispiel kann die Aussage „Franz ist krank" in der Prädikatenlogik erster Stufe repräsentiert werden durch eine atomare Formel, deren Relationssymbol für die Eigenschaft „ist krank" steht, und die als Argument einen Term hat, der Franz repräsentiert. Die Gemeinsamkeit dieser Aussage mit „Hans ist krank" äußert sich darin, dass die atomaren Formeln, mit denen die Aussagen repräsentiert werden, mit dem selben Relationssymbol gebildet sind, aber verschiedene Terme als Argument haben.

Eine andere Erweiterung bilden die Quantoren $\forall$ (für alle) und $\exists$ (es gibt), mit denen Aussagen über unbestimmte Objekte repräsentiert werden können. Derartige Objekte werden durch „Variablen" repräsentiert, die eine spezielle Art von Termen sind.

All diese Erweiterungen erfordern natürlich für eine Sprache der Prädikatenlogik erster Stufe mehr Klassen von Symbolen als für eine Sprache der Aussagenlogik, und zwar sowohl bei den logischen Symbolen als auch bei der Signatur, dem anwendungsspezifischen Teil einer Sprache.

Definition 2.3.1 (Sprache [PL1S]). Eine Sprache $\mathcal{L}$ der Prädikatenlogik erster Stufe ist gegeben durch ihre Signatur. Zusätzlich gehören zu $\mathcal{L}$ die logischen Symbole der Prädikatenlogik erster Stufe.

- Die *logischen Symbole* der Prädikatenlogik erster Stufe sind:
  1. Die Menge der *Hilfszeichen*) und , und (
  2. Die Menge der nullstelligen *Junktoren*: $\{\top, \bot\}$.
     Die Menge der einstelligen Junktoren: $\{\neg\}$.
     Die Menge der zweistelligen Junktoren: $\{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$.
  3. Die Menge der *Quantoren* $\{\forall, \exists\}$.
  4. Eine abzählbar unendliche Menge von *Variablen*,
     notiert $u, v, w, x, y, z, \ldots$ mit oder ohne Indizes.
  5. Das Gleichheitsrelationssymbol $\doteq$ falls vorhanden.

- Die *Signatur* einer Sprache $\mathcal{L}$ der Prädikatenlogik erster Stufe besteht aus folgenden Symbolmengen:
  1. Für jede natürliche Zahl $n \geq 0$ eine (eventuell leere) Menge $Rel^n_{\mathcal{L}}$ von n-stelligen *Relationssymbolen*. Diese werden auch *Prädikatssymbole* genannt.
     0-stellige Relationssymbole heißen auch *Aussagensymbole*.
     Enthält $Rel^2_{\mathcal{L}}$ das besondere zweistellige Relationssymbol $\doteq$, genannt *Gleichheitsrelationssymbol*, so heißt $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe *mit Gleichheit*.

2. Für jede natürliche Zahl $n \geq 0$ eine (eventuell leere) Menge $Fun_{\mathcal{L}}^n$ von n-stelligen *Funktionssymbolen*.
   0-stellige Funktionssymbole heißen auch *Konstanten*.

Alle erwähnten Mengen sind abzählbar (üblicherweise sogar aufzählbar) und paarweise disjunkt, und kein Symbol einer dieser Mengen besteht aus einer Sequenz von Symbolen, in der ein Symbol aus einer der anderen Mengen vorkommt.
Es wird vorausgesetzt, dass $\mathcal{L}$ mindestens eine Konstante enthält. ■

Bemerkungen:

1. Die Variablen gehören nicht zur Signatur, obwohl sie syntaktisch in Formeln an Stellen vorkommen dürfen, an denen auch Signaturelemente erlaubt sind.
2. Es werden unendlich viele Variablen vorausgesetzt, damit Aussagen über beliebig viele Objekte möglich sind und die Länge von Formeln (und Beweisen) nicht dadurch beschränkt ist, dass irgendwann die verfügbaren Variablen ausgehen. Diese Annahme ist unverzichtbar, aber unproblematisch.
3. Die Voraussetzung, dass es mindestens eine Konstante gibt, hat technische Gründe, die im Abschnitt über Herbrand-Interpretationen klar werden, und wird sonst nirgends gebraucht. ■

In *Anwendungen* der Prädikatenlogik erster Stufe ist es meistens sinnvoll, als Relationssymbole und Funktionssymbole Bezeichner wie in Programmiersprachen zu verwenden. In Texten *über* die Prädikatenlogik bevorzugt man dagegen kleine, kompakte Beispiele und verwendet oft Signaturen, für die gilt:
$p, q, r, \ldots$ sind Relationssymbole der gerade benötigten Stelligkeiten.
$f, g, h, \ldots$ sind Funktionssymbole der gerade benötigten Stelligkeiten $\neq 0$.
$a, b, c, \ldots$ sind Konstanten, also Funktionssymbole der Stelligkeit 0.
Dazu kommt bei den logischen Symbolen noch:
$u, v, w, x, y, z, \ldots$ sind Variablen.
Wenn die Sprache $\mathcal{L}$ gar nicht explizit angegeben ist, wird normalerweise angenommen, dass sie nach diesen Konventionen gebildet ist, mit Kleinbuchstaben vom Ende des Alphabets als Variablen und vom Anfang des Alphabets als Konstanten.

Die Stelligkeit ist Teil eines Funktionssymbols. Ist es zulässig, in einer Sprache das selbe Funktionssymbol, zum Beispiel f, mit unterschiedlichen Stelligkeiten n_1 und n_2 zu verwenden? Die Bemerkung „Alle erwähnten Mengen sind … paarweise disjunkt“ in Definition 2.3.1 verbietet das.

Solange aus dem Kontext eindeutig erkennbar ist, welche Stelligkeit gemeint ist, könnte man ein solches „Überladen“ von Funktionssymbolen aber zulassen. Die Programmiersprache Prolog erlaubt es zum Beispiel und bietet die spezielle Notation f/n_1 bzw. f/n_2 für die Fälle an, in denen man explizit zwischen dem n_1-stelligen und dem n_2-stelligen Funktionssymbol f unterscheiden will. Formal ist es aber nicht ganz einfach, Bedingungen dafür anzugeben, dass die Stelligkeit aus dem Kontext eindeutig erkennbar ist. Deshalb verbieten viele Autoren das „Überladen“, so wie es auch durch Definition 2.3.1 ausgeschlossen wird.

Definition 2.3.2 (Term [PL1S]). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe. Die Menge $\mathcal{T}_{\mathcal{L}}$ der $\mathcal{L}$-Terme, kurz Terme, ist die kleinste Menge, die die folgenden Bedingungen erfüllt:

1. Jede Variable von $\mathcal{L}$ ist ein $\mathcal{L}$-Term.
2. Jede Konstante von $\mathcal{L}$ ist ein $\mathcal{L}$-Term.
3. Sind $t_1, \ldots, t_n$ $\mathcal{L}$-Terme, $n \geq 1$, und ist f ein n-stelliges Funktionssymbol von $\mathcal{L}$, so ist auch $f(t_1, \ldots, t_n)$ ein $\mathcal{L}$-Term. ∎

Satz 2.3.3 (Eindeutigkeitssatz [Term, PL1S]). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe. Für jeden $\mathcal{L}$-Term t gilt genau eine der folgenden Aussagen:

1. Es gibt eine eindeutige Variable x von $\mathcal{L}$ mit $t = x$.
2. Es gibt eine eindeutige Konstante c von $\mathcal{L}$ mit $t = c$.
3. Es gibt ein eindeutiges Funktionssymbol f mit Stelligkeit $n \geq 1$ und ein eindeutiges n-Tupel $(t_1, \ldots, t_n)$ von $\mathcal{L}$-Termen, so dass $t = f(t_1, \ldots, t_n)$.

Beweis: Ähnlich wie der Beweis von Satz 2.1.7. ∎

Definition 2.3.4 (Formel [PL1S]). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe.

- Jedes nullstellige Relationssymbol von $\mathcal{L}$ ist eine atomare $\mathcal{L}$-Formel.
  Sind $t_1, \ldots, t_n$ $\mathcal{L}$-Terme, $n \geq 1$, und ist p ein n-stelliges Relationssymbol von $\mathcal{L}$, so ist $p(t_1, \ldots, t_n)$ eine atomare $\mathcal{L}$-Formel.

  Eine atomare $\mathcal{L}$-Formel nennt man auch kurz *atomare Formel* oder einfach *Atom*.

- Die Menge $\mathcal{F}_\mathcal{L}$ der $\mathcal{L}$-Formeln, kurz Formeln, ist die kleinste Menge, die die folgenden Bedingungen erfüllt:

  1. $\top$ und $\bot$ und alle atomaren $\mathcal{L}$-Formeln sind in $\mathcal{F}_\mathcal{L}$.
  2. Ist $F \in \mathcal{F}_\mathcal{L}$, so ist auch $\neg F \in \mathcal{F}_\mathcal{L}$.
  3. Ist $F \in \mathcal{F}_\mathcal{L}$ und $G \in \mathcal{F}_\mathcal{L}$ und θ ein zweistelliger Junktor, so ist auch $(F \; \theta \; G) \in \mathcal{F}_\mathcal{L}$.
  4. Ist $F \in \mathcal{F}_\mathcal{L}$, ist x eine Variable, und ist Q ein Quantor, so ist auch $QxF \in \mathcal{F}_\mathcal{L}$. ∎

Satz 2.3.5 (Eindeutigkeitssatz [Formel, PL1S]). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe. Für jede $\mathcal{L}$-Formel F gilt genau eine der folgenden Aussagen:

1. $F = \top$ oder $F = \bot$
   oder es gibt ein eindeutiges nullstelliges Relationssymbol p mit $F = p$,
   oder es gibt ein eindeutiges n-stelliges ($n \geq 1$) Relationssymbol p und ein eindeutiges n-Tupel $(t_1, \ldots, t_n)$ von $\mathcal{L}$-Termen mit $F = p(t_1, \ldots, t_n)$.
2. Es gibt eine eindeutige $\mathcal{L}$-Formel G, so dass $F = \neg G$.
3. Es gibt ein eindeutiges Paar (G, H) von $\mathcal{L}$-Formeln und einen eindeutigen zweistelligen Junktor θ, so dass $F = (G \; \theta \; H)$.
4. Es gibt einen eindeutigen Quantor $Q \in \{\forall, \exists\}$, eine eindeutige Variable x, und eine eindeutige $\mathcal{L}$-Formel G, so dass $F = QxG$.

Beweis: Ähnlich wie der Beweis von Satz 2.1.7. ∎

Im Fall des Gleichheitsrelationssymbols werden für eine atomare Formel $\doteq(t_1, t_2)$ auch die Notationen $(t_1 \doteq t_2)$ und $t_1 \doteq t_2$ benutzt. Einige andere zweistellige Relationssymbole und einige zweistellige Funktionssymbole können ebenfalls in Infixform geschrieben werden.

Der Lesbarkeit halber werden oft anstelle der vorgeschriebenen Klammern (und) andere Klammern wie etwa [und] oder { und } verwendet.

Bemerkungen:

1. $\top$ und $\bot$ sind wie im Fall der Aussagenlogik nicht als atomare Formeln definiert. (Geschmackssache)
2. Teilformeln können ähnlich wie für die Aussagenlogik definiert werden.
3. Die Menge $\mathcal{T}_{\mathcal{L}}$ aller $\mathcal{L}$-Terme ist abzählbar, weil nach Definition alle Symbolmengen von $\mathcal{L}$ abzählbar sind. Sind diese Mengen darüberhinaus aufzählbar, so ist auch $\mathcal{T}_{\mathcal{L}}$ aufzählbar.

   Entsprechendes gilt für die Menge $\mathcal{F}_{\mathcal{L}}$ aller $\mathcal{L}$-Formeln. ∎

Terme vs. Formeln. Mit Definition 2.3.2 und Definition 2.3.4 sind zwei verschiedene syntaktische Gebilde eingeführt, zum einen Terme und zum anderen Formeln. Sie müssen trotz ihrer syntaktischen Ähnlichkeiten strikt auseinandergehalten werden. Definiert eine Sprache $\mathcal{L}$ zum Beispiel a und b als Konstanten, f als zweistelliges Funktionssymbol und p als zweistelliges Relationssymbol, so ist $f(a, f(a, b))$ ein Term dieser Sprache und $p(a, f(a, b))$ eine atomare Formel dieser Sprache.

Nach Definition können Terme sowohl in atomaren Formeln als auch in Termen vorkommen (Funktionssymbole dürfen also geschachtelt werden). Atomare Formeln können dagegen weder in Termen noch in atomaren Formeln vorkommen (Relationssymbole dürfen also nicht geschachtelt werden). Junktoren können Formeln verknüpfen, insbesondere auch atomare Formeln, aber keine Terme. Die atomaren Formeln bilden sozusagen die Schnittstelle zwischen den beiden Arten von syntaktischen Gebilden.

Die strikte Unterscheidung zwischen Termen und Formeln mag rein syntaktisch gesehen willkürlich erscheinen. Sie ist aber wesentlich für die Definitionen zur Semantik (Kapitel 3). Die Grundidee ist, dass Formeln zur Repräsentation von Aussagen dienen, die wahr oder falsch sein können, während Terme zur Repräsentation von Objekten dienen, über die wahre oder falsche Aussagen gemacht werden können, die aber nicht selbst wahr oder falsch sind.

Im obigen Beispiel könnten a und b Zahlen repräsentieren, f eine arithmetische Verknüpfung und p eine arithmetische Vergleichsrelation. Dann repräsentiert der Term $f(a, f(a, b))$ wieder eine Zahl, die Formel $p(a, f(a, b))$ aber eine Aussage über zwei Zahlen.

Freie und gebundene Variablen und Variablenvorkommen. In Termen und Formeln können Variablen mehrfach vorkommen. Sind zum Beispiel p, q, r einstellige Relationssymbole, so ist $((p(x) \land q(x)) \land r(x))$ eine Formel mit drei Vorkommen der Variablen x. Für Formeln mit Quantoren bleibt in einigen Fällen noch zu klären, welche Quantoren sich auf welche Variablenvorkommen beziehen. Das geschieht nach dem selben Prinzip wie in Programmiersprachen mit Blockschachtelung, wenn man die Quantoren analog zu Deklarationen sieht.

Sei zum Beispiel F die Formel $(\ \forall x(p(x) \land \exists x\, q(x)) \quad \land \quad r(x)\ )$. In F bezieht sich der Allquantor auf die Teilformel $(p(x) \land \exists x\, q(x))$, in deren rechter Teilformel er aber durch den Existenzquantor überschattet wird. Man sagt, dass das Vorkommen von x in $p(x)$ durch den Allquantor gebunden ist und das Vorkommen von x in $q(x)$ durch den Existenzquantor. Das Vorkommen von x in $r(x)$ ist dagegen durch gar keinen Quantor gebunden und wird ein freies Vorkommen von x in der Formel F genannt. Für die Formel $\exists y F$ gilt genau das gleiche, weil y in F nicht vorkommt. In der Formel $\exists x F$ ist das Vorkommen von x in $r(x)$ durch den äußersten Existenzquantor gebunden, der seinerseits durch den Allquantor in der linken Teilformel von F überschattet wird.

Definition 2.3.6 (Bereich eines Quantors). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe und F eine $\mathcal{L}$-Formel. In Teilformeln von F der Gestalt QxG mit einem Quantor Q heißt Qx ein *Quantor für x*. Sein *Bereich* (englisch: scope) ist die Teilformel G mit Ausnahme aller Teilformeln von G, die selbst die Gestalt $Q'xH$ haben, die also mit einem Quantor für die selbe Variable x gebildet sind.

Jedes Vorkommen von x, das nicht im Bereich eines Quantors für x steht, nennt man ein *freies Vorkommen von x in F*.

Jedes Vorkommen von x im Bereich eines Quantors für x nennt man durch diesen Quantor *gebunden in F*. ∎

Frei oder gebunden zu sein ist nicht eine Eigenschaft eines Variablenvorkommens für sich allein betrachtet, sondern eines Variablenvorkommens in einer betrachteten Formel. Beispielsweise ist x gebunden in der Formel $\forall x\, p(x)$, aber frei in ihrer Teilformel $p(x)$.

Formeln mit freien Variablenvorkommen sind normalerweise für die Wissensmodellierung uninteressant. Sie sind aber aus technischen Gründen notwendig, um induktive Definitionen auf Formeln angeben zu können.

Wir führen noch eine handliche Notation für die Menge aller Variablen ein, die in einem Term oder einer Formel vorkommen bzw. frei vorkommen. Die Notation kann für Terme und Formeln gleich sein, weil der Kontext Missverständnisse ausschließt.

Definition 2.3.7 (Variablen in Term/Formel).

- Die Menge $var(t)$ aller Variablen, von denen es in einem $\mathcal{L}$-Term t mindestens ein Vorkommen gibt, wird wie folgt definiert:
  1. Ist t eine Variable, so ist $var(t) = \{t\}$.
  2. Ist t eine Konstante, so ist $var(t) = \emptyset$.
  3. Ist t von der Gestalt $f(t_1, \ldots, t_n)$ für ein n-stelliges Funktionssymbol f und $\mathcal{L}$-Terme $t_1, \ldots, t_n$, so ist $var(t) = \bigcup_{i=1}^{n} var(t_i)$.

- Die Menge $var(F)$ aller Variablen, von denen es in einer $\mathcal{L}$-Formel F mindestens ein Vorkommen gibt, wird wie folgt definiert:
  1. $var(\top) = var(\bot) = \emptyset$.
  2. Ist F eine atomare Formel der Gestalt p, so ist $var(F) = \emptyset$.
     Ist F eine atomare Formel der Gestalt $p(t_1, \ldots, t_n)$, so ist $var(F) = \bigcup_{i=1}^{n} var(t_i)$.
  3. $var(\neg G) = var(G)$
  4. $var(\ (G\theta H)\ ) = var(G) \cup var(H)$ für einen zweistelligen Junktor θ.
  5. $var(QxG) = var(G)$ für einen Quantor Q.

- Die Menge $fvar(F)$ aller Variablen, von denen es in einer $\mathcal{L}$-Formel F mindestens ein freies Vorkommen gibt, wird wie folgt definiert:
  1. $fvar(\top) = fvar(\bot) = \emptyset$.
  2. Ist F eine atomare Formel der Gestalt p, so ist $fvar(F) = \emptyset$.
     Ist F eine atomare Formel der Gestalt $p(t_1, \ldots, t_n)$, so ist $fvar(F) = \bigcup_{i=1}^{n} var(t_i)$.
  3. $fvar(\neg G) = fvar(G)$
  4. $fvar(\ (G\theta H)\ ) = fvar(G) \cup fvar(H)$ für einen zweistelligen Junktor θ.
  5. $fvar(QxG) = fvar(G) \setminus \{x\}$ für einen Quantor Q. ∎

Die beiden Quantorfälle sind nicht so offensichtlich wie der Rest. Seien x eine Variable, a eine Konstante, p ein einstelliges Relationssymbol. In einer Formel F der Gestalt QxG wird das x unmittelbar hinter dem Quantor nicht mitgezählt, aber es entscheidet darüber, ob etwaige Vorkommen von x in G in der Gesamtformel F frei sind oder nicht:

$$var(\ \forall x\, p(a)\ ) = var(p(a)) = var(a) = \emptyset \qquad var(\ \forall x\, p(x)\ ) = var(p(x)) = var(x) = \{x\}$$

$$\begin{aligned} fvar(\ \forall x\, p(a)\ ) &= fvar(p(a)) \setminus \{x\} \\ &= fvar(a) \setminus \{x\} = \emptyset \setminus \{x\} = \emptyset \end{aligned} \qquad \begin{aligned} fvar(\ \forall x\, p(x)\ ) &= fvar(p(x)) \setminus \{x\} \\ &= fvar(x) \setminus \{x\} = \{x\} \setminus \{x\} = \emptyset \end{aligned}$$

Definition 2.3.8 (Grundterm, Grundformel, geschlossene Formel).

- Ein Term t mit $var(t) = \emptyset$ heißt *Grundterm* oder *geschlossener Term*.
- Eine Formel F mit $var(F) = \emptyset$ heißt *Grundformel*.
  Atomare Grundformeln nennt man auch *Grundatome*.
- Eine Formel F mit $fvar(F) = \emptyset$, heißt *geschlossene Formel* oder *Satz*. ■

„Grundterm“ und „geschlossener Term“ sind also Synonyme, „Grundformel“ und „geschlossene Formel“ sind keine. Jede Grundformel ist geschlossen, aber nicht umgekehrt:
$p(a)$ ist eine Grundformel, also auch geschlossen.
$\forall x\, p(x)$ ist keine Grundformel, aber geschlossen.
$p(x)$ ist keine Grundformel und nicht geschlossen.

2.4 Exkurs: Termdarstellungen in Programmier- und Modellierungssprachen

Programmier- und Modellierungssprachen, die unter anderem in Datenbanksystemen, in der Computer-Linguistik und in der Wissensrepräsentation eingesetzt werden, bieten Konstrukte an, die Termen oder atomaren Formeln einer Sprache der Prädikatenlogik erster Stufe ähneln. Oft weichen diese Konstrukte von den Definitionen der Logik in einigen Aspekten ab, die aber nicht prinzipieller Natur sind. Implementierungen solcher Konstrukte basieren meistens auf herkömmlichen Termen oder Formeln bzw. auf linearisierten Darstellungen davon.

Positionen vs. Rollen. Nach den Syntaxdefinitionen der Prädikatenlogik sind die Positionen von Untertermen in Atomen und in Termen wesentlich: die Atome $person(Anna, Maier)$ und $person(Maier, Anna)$ sind verschieden, weil sie an gleichen Positionen verschiedene Terme als Argumente haben.

Manche Sprachen sehen für jedes n-stellige Relationssymbol n Bezeichner vor, sogenannte „Rollen“. Zur Bildung eines Atoms bekommt das Relationssymbol nicht einfach n Unterterme als Argumente, sondern n Paare der Form *Rolle*:*Unterterm*. Die Reihenfolge dieser Paare ist dabei beliebig. Zum Beispiel ist es naheliegend, dem Relationssymbol *person* die Rollen *Vorname* und *Nachname* zuzuordnen, so dass das obige Atom in der Form $person(Vorname{:}Anna, Nachname{:}Maier)$ geschrieben werden kann, aber auch in der Form $person(Nachname{:}Maier, Vorname{:}Anna)$.

Die syntaktische Variante mit Rollen hat praktische Vorteile: sie ist flexibler, weil sie die Reihenfolge der Argumente nicht fest vorgibt, und sie ist lesbarer, insbesondere bei Relationssymbolen mit hoher Stelligkeit. Deshalb wird sie für manche Anwendungen, wie etwa Datenbanken, Wissensverwaltungssysteme, und auch in einigen Programmiersprachen, bevorzugt. Sie kann aber (leicht) auf die herkömmliche Syntax zurückgeführt werden und stellt deshalb keine Veränderung der Ausdrucksstärke gegenüber der Prädikatenlogik dar.

Meist können Rollen auch typisiert werden. Dies entspricht einer mehrsortigen Logik (Abschnitt 2.10) und verändert die Ausdrucksstärke ebenfalls nicht.

Tupel vs. Atome. Eine relationale Datenbank besteht aus einer (endlichen) Menge von (endlichen) Relationen. Jede Relation besitzt eine Stelligkeit und enthält Tupel der selben Stelligkeit. Ist R eine n-stellige Relation und $(t_1, \ldots, t_n)$ ein Tupel von R, so schreibt man $(t_1, \ldots, t_n) \in R$. Eine Alternative dazu ist die Schreibweise als Atom einer Sprache der Prädikatenlogik: $R(t_1, \ldots, t_n)$.

Der Informationsgehalt ist bei beiden Schreibweisen der selbe. Die „relationale Schreibweise“ ist üblich in Arbeiten über relationale Datenbanksysteme. Die „logische Schreibweise“ hat sich mit den deduktiven Datenbanksystemen, einer Erweiterung der relationalen Datenbanksysteme, etabliert.

Wenn auch die Alternative zwischen Positionen- und Rollenschreibweise berücksichtigt wird, ergeben sich insgesamt vier Darstellungsmöglichkeiten für die Tupel einer relationalen Datenbank.

Verbundartige Darstellungen. Oft werden Sprachkonstrukte angeboten, die an Verbunde wie in imperativen Programmiersprachen erinnern. Sie sollen insbesondere objektorientierte Modellierungen unterstützen, bei denen alle Daten über das selbe reale Objekt in einer einzigen syntaktischen Einheit repräsentiert werden.

```
Person
   Vorname:       Anna
   Nachname:      Maier
   Geburtsdatum:  27.1.1984
```

Ob man dieses Gebilde so schreibt wie hier oder zum Beispiel in der Syntax von XML-Elementen, es kann in die Prädikatenlogik übersetzt werden. Liest man die erste Zeile als dreistelliges Relationssymbol und jede folgende Zeile als Paar *Rolle*:*Unterterm*, handelt es sich einfach um ein Atom in Rollenschreibweise, das in der Standardsyntax der Prädikatenlogik mit Positionenschreibweise zum Beispiel $\mathit{person}(\mathit{Anna}, \mathit{Maier}, 27.1.1984)$ geschrieben würde.

Eine andere Übersetzung ergibt eine nicht-atomare Formel:
$\mathit{person}(c) \ \land \ \mathit{vorname}(c) \doteq \mathit{Anna} \ \land \ \mathit{nachname}(c) \doteq \mathit{Maier} \ \land \ \mathit{geburtsdatum}(c) \doteq 27.1.1984$
Was vorher als Rolle gelesen wurde, wird dabei zu einem einstelligen Funktionssymbol. Diese Übersetzung kann komplexere Strukturen beschreiben als die erste, zum Beispiel Verschachtelungen der verbundartigen Gebilde.

Wenn objektorientiert modelliert werden soll, können verbundartige Konstrukte geeignete syntaktische Einheiten zur Repräsentation realer Objekte sein. Durch die Übersetzung in der zweiten Weise würde die zum selben Objekt gehörige Information auf mehrere Atome verteilt, was ja gerade unerwünscht ist. Die Übersetzung ist also offensichtlich nicht deshalb interessant, weil die sparsame Syntax der Prädikatenlogik besser wäre.

Aber für die Prädikatenlogik steht ein ausgereiftes Instrumentarium für Fragen der Semantik zur Verfügung, das durch die Übersetzung anwendbar wird. Man übersetzt also in die Prädikatenlogik, um angeben zu können, was Konstrukte wie das obige bedeuten sollen.

Zyklische Strukturen. Manche objektorientierten Programmier- und Modellierungssprachen erlauben Selbstverweise wie z.B.:

```
Angestellter
   Vorname:       Anna
   Nachname:      Maier
   Geburtsdatum:  27.1.1984
   Vorgesetzter:  self
```

Das Schlüsselwort „`self`“ verweist auf die syntaktische Einheit, in der es vorkommt. Ob die gesamte Struktur nun einem Term oder einer Formel entsprechen soll, die Prädikatenlogik lässt weder „zyklische Terme“ noch „zyklische Formeln“ zu. Zur Vereinfachung der Sprechweise sei angenommen, dass das Gebilde einem Term entsprechen soll.

In Implementierungen von objektorientierten Sprachen wird solch ein „zyklischer Term“ in naheliegender Weise unter Verwendung der Speicheradresse der Struktur oder durch Hinzunahme eines eindeutigen Termbezeichners als zusätzliches Argument dargestellt. Mit anderen Worten, intern wird der zyklische Term durch einen gewöhnlichen, azyklischen Term dargestellt.

Was macht also zyklische Terme aus, wenn sie als azyklische Terme implementiert werden?

Die Antwort ergibt sich aus der Unterscheidung zwischen der Modellierungssprache und ihrer Implementierungssprache. Die Speicheradressen oder die Bezeichner zur Auflösung der Zyklen sind für einen Benutzer der Modellierungssprache unsichtbar. Ferner stellt die Modellierungssprache keine Operation zur Verfügung, womit diese Speicheradressen oder Bezeichner als solche direkt geändert oder sonstwie bearbeitet werden könnten. So wird eine Abstraktionsbarriere gebildet, die dem Benutzer der Modellierungssprache zyklische Terme zur Verfügung stellt und deren Verwendung gemäß der Semantik der Modellierungssprache einschränkt. In der Implementierungssprache ist im Gegensatz dazu die azyklische Termdarstellung sichtbar. Die Speicheradressen oder Bezeichner, wodurch Zyklen in Termen repräsentiert werden, können in der Implementierungssprache direkt bearbeitet werden.

Objektidentität. Die „Objektidentität“ ist ein weiteres Merkmal von objektorientierten Programmier- und Modellierungssprachen, das in der Prädikatenlogik nicht vorhanden ist.

Die Objektidentität ermöglicht, über mehrere Objekte zu verfügen, die sich syntaktisch nicht unterscheiden. Mit anderen Worten ermöglicht die Objektidentität die Umgehung des Leibniz'schen Prinzips „identitas indiscernibilium“, d.h. der Identität von ununterscheidbaren Objekten (siehe Abschnitt 3.6).

So können z.B. zwei verschiedene Personen mit gleichem Vor- und Nachnamen und Geburtsdatum durch zwei Objekte der selben Gestalt repräsentiert werden:

```
Person                             Person
   Vorname:       Anna                Vorname:       Anna
   Nachname:      Maier               Nachname:      Maier
   Geburtsdatum:  27.1.1984           Geburtsdatum:  27.1.1984
```

Eine solche Duplizierung und Unterscheidung zweier identischer Terme oder Atome ist in der Prädikatenlogik unmöglich, weil ihr der Mengenbegriff zu Grunde liegt.

Termen oder Atomen der Prädikatenlogik kann dadurch eine Identität gegeben werden, dass ihnen ein eindeutiger Bezeichner als zusätzliches Argument hinzugefügt wird. Der Vorteil der Programmier- und Modellierungssprachen mit Objektidentität liegt darin, dass der

Benutzer dieser Sprachen keine solchen Bezeichner zur Kenntnis nimmt, sondern so viele Objekte anlegt, die eventuell ununterscheidbar sind, wie die Anwendung es erfordert.

Die Implementierung der Objektidentität beruht selbstverständlich auf eindeutigen Bezeichnern oder Speicheradressen, so dass die Implementierung von Termen mit Identität durch gewöhnliche Terme erfolgt. Ähnlich wie bei den zyklischen Termen sichert in der Programmier- oder Modellierungssprache eine Abstraktionsbarriere die Semantik der Objektidentität.

Präfix- vs. Postfixlinearisierung. Implementierungen von Programmiersprachen und von Beweismethoden übersetzen Terme in Symbolsequenzen, die möglichst benachbarte Speicherzellen belegen. Diese Übersetzung wird Linearisierung genannt. Sie beruht meist auf der Präfix- oder Postfixdarstellung (siehe Abschnitt 2.2).

Der Term $f(a, g(b, c), d)$ wird in Implementierungen von logischen Programmiersprachen oder Beweismethoden wie folgt präfixlinearisiert:

$$f/3 \quad a/0 \quad g/2 \quad b/0 \quad c/0 \quad d/0$$

wobei die Notation s/n zur Angabe der Stelligkeit n eines Symbols s dient.

Eine der zentralen Operationen in logischen Programmiersprachen und Beweismethoden ist die sogenannte „Unifikation". Sie dient dazu, Paare von Termen zu vergleichen und anzugleichen. Zum Beispiel führt die Unifikation der Terme $f(a, g(b, c), d)$ und $f(a, x, d)$ dazu, dass die Variable x an den Term $g(b, c)$ gebunden wird.

Betrachtet man die Baumstruktur der Terme, wird deutlich, dass der Vergleich zweier Terme von der Wurzel her erfolgen sollte:

Aus diesem Grund ist die Präfixlinearisierung zur Implementierung von logischen Programmiersprachen und von Beweismethoden besonders geeignet. Dann entspricht die Unifikation einem parallelen Durchgang durch die beiden Symbolsequenzen.

Dies kann am Beispiel der Terme $f(a, g(b, c), d)$ und $f(a, x, d)$ veranschaulicht werden, deren Präfixlinearisierungen wie folgt sind:

$$\begin{array}{llllll} f/3 & a/0 & g/2 & b/0 & c/0 & d/0 \\ f/3 & a/0 & x & d/0 & & \end{array}$$

Beide Sequenzen werden Symbol nach Symbol von links nach rechts gelesen. Solange die beiden gelesenen Symbole identisch sind, wird das Verfahren fortgesetzt. Wird in einer Sequenz eine ungebundene Variable wie x gelesen, dann wird sie an den Teilterm gebunden, der in der anderen Sequenz erreicht wurde, hier $g(b, c)$, und dieser Term wird in der anderen Sequenz übersprungen. Wieviele Symbole in der Sequenz zum Überspringen dieses Terms überlesen werden müssen, kann aus den angegebenen Stelligkeiten berechnet werden. Werden zwei Symbole s_1/n_1 und s_2/n_2 gelesen mit $s_1 \neq s_2$ oder $n_1 \neq n_2$, dann scheitert die Unifikation.

Zur Implementierung funktionaler Programmiersprachen werden Terme dagegen postfixlinearisiert, weil solche Sprachen die Terme auswerten. Die Postfixdarstellung ist günstig für die übliche kellerbasierte Auswertung. Betrachten wir den Term $f(a, g(b, c))$ und seine Postfixlinearisierung:

$a/0 \;\; b/0 \;\; c/0 \;\; g/2 \;\; f/2$

Sei angenommen, dass die Symbole die folgenden Werte haben:

Symbol	$a/0$	$b/0$	$c/0$	$g/2$	$f/2$
Wert	1	2	3	$\times$	$+$

Die Symbolsequenz der Postfixlinearisierung wird ebenfalls Symbol nach Symbol von links nach rechts gelesen. Wird ein nullstelliges Symbol gelesen, so wird sein Wert, hier eine Zahl, auf den Keller gelegt. Wird ein zweistelliges Symbol gelesen, so wird sein Wert, hier eine arithmetische Operation, auf die zwei oberen Kellerzellen angewandt, diese zwei oberen Kellerzellen werden geräumt, und das Berechnungsergebnis wird auf den Keller gelegt. Während der Auswertung des obigen Beispiels verändert sich also der Keller wie folgt:

			3		
		2	2	6	
	1	1	1	1	7
Schritt	1.	2.	3.	4.	5.

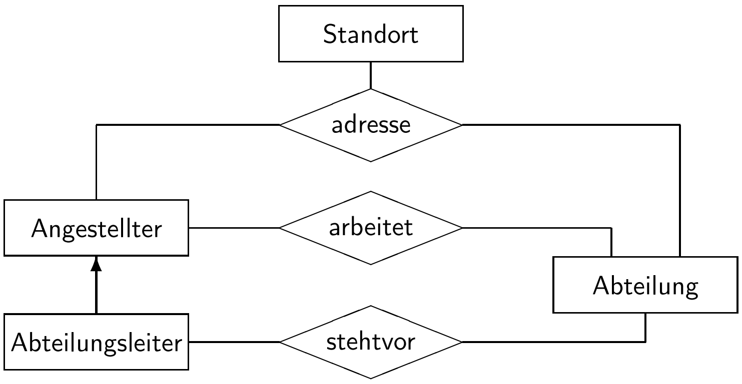
2.5 Exkurs: Das Entitäts-Relations-Modell und der Tupel-Kalkül

Im Datenbankbereich sind Formalismen eingeführt worden, die einerseits syntaktisch wesentlich von der Prädikatenlogik erster Stufe abweichen, andererseits aber leicht darin übertragen werden können.

Entitäts-Relations-Modell.

Das Entitäts-Relations-Modell (*entity relationship model*, ER-Modell) dient dazu, die Struktur eines zu modellierenden Ausschnitts der Realität zu beschreiben und damit ein Datenbankschema zu spezifizieren.

Wir führen das Entitäts-Relations-Modell informell anhand eines Beispiels ein. In dem zu modellierenden Ausschnitt der Realität gibt es Angestellte, Abteilungen, Abteilungsleiter und Standorte. Jeder Angestellte arbeitet in einer Abteilung. Jeder Abteilungsleiter steht einer Abteilung vor. Alle Abteilungsleiter sind Angestellte. Jeder Angestellte hat einen Standort, der auch der Standort seiner Abteilung ist.



Die wichtigsten Bestandteile des Entitäts-Relations-Modells sind Entitäten (durch Rechtecke dargestellt), Relationen (durch Rauten dargestellt) und Hierarchiebeziehung (durch

Pfeile dargestellt). Letztere ist eigentlich nur ein häufig vorkommender Spezialfall einer zweistelligen Relation mit vordefinierter Semantik. Im Englischen wird sie auch die „*is-a*“-Relation genannt

Die Anzahl der Kanten, die mit einer Raute verbunden sind, ist die Stelligkeit der Relation, die die Raute definiert. Diese Kanten werden mit „Rollen“ beschriftet. Zum Beispiel könnte die linke Kante der Relation arbeitet mit wer und die rechte mit wo beschriftet sein. Weitere Beschriftungen der Kanten erlauben die Angabe von „Kardinalitäten“ wie z.B.:

In diesem Beispiel wäre die Bedeutung der Kardinalitätsbeschriftung, dass jeder Angestellte in bis zu 3 Abteilungen arbeiten kann. Beschriftet man beiden Kante der Relation stehtvor mit der Kardinalität 1, bedeutet dies, dass jeder Abteilungsleiter genau einer Abteilung vorsteht und jede Abteilung genau einen Abteilungsleiter hat.

Es gibt noch weitere Bestandteile des Modells, zum Beispiel Attribute, auf die hier nicht näher eingegangen werden soll.

Die Übersetzung aus dem Entitäts-Relations-Modell in eine Sprache der PL1S geschieht nach folgendem Schema:

- Jede Entität wird durch ein einstelliges Relationssymbol dargestellt.
- Jede Relation mit n Rollen wird durch ein n-stelliges Relationssymbol dargestellt.
- Eine Hierarchiebeziehung, graphisch dargestellt durch einen Pfeil von einer Entität a zu einer Entität b, wird dargestellt durch die Formel $\forall x\ (a(x) \Rightarrow b(x))$.
- Eine Kardinalitätsbedingung wird durch eine allquantifizierte Formel dargestellt.

Aufgrund einer solchen Übersetzung kann die Bedeutung der Bestandteile des Entitäts-Relations-Modells mit Hilfe der – in Kapitel 3 zu besprechenden – wohlverstandenen Semantik der Prädikatenlogik erster Stufe präzise definiert werden.

Tupel-Kalkül.

In der PL1S stehen die Variablen für Objekte, wie etwa Angestellte, Abteilungen, usw. Im Tupel-Kalkül gibt es dagegen Variablen für Tupel solcher Objekte.
Die Aussage

> „arbeitet“ ist eine Beziehung zwischen Angestellten und Abteilungen

kann z.B. in einem Tupel-Kalkül wie folgt dargestellt werden:

$$\forall(x \in arbeitet) \Rightarrow (x.wer \in Angestellter \wedge x.wo \in Abteilung)$$

wenn die Rollen so benannt sind wie im obigen Beispiel. Statt $x \in arbeitet$ könnte man auch $arbeitet(x)$ schreiben. Im Gegensatz zur Prädikatenlogik steht das x dann aber nicht für ein einziges Argument, sondern für das ganze Argumenttupel, in diesem Fall also für das Paar $(x.wer, x.wo)$.

Der Grund für die im Tupel-Kalkül hervorgehobene Stellung der Tupel gegenüber den einzelnen Argumenten ist, dass die Tupel die Modellierungs- und Speicherungseinheiten des relationalen Datenmodells sind.

2.6 Exkurs: UML und OCL

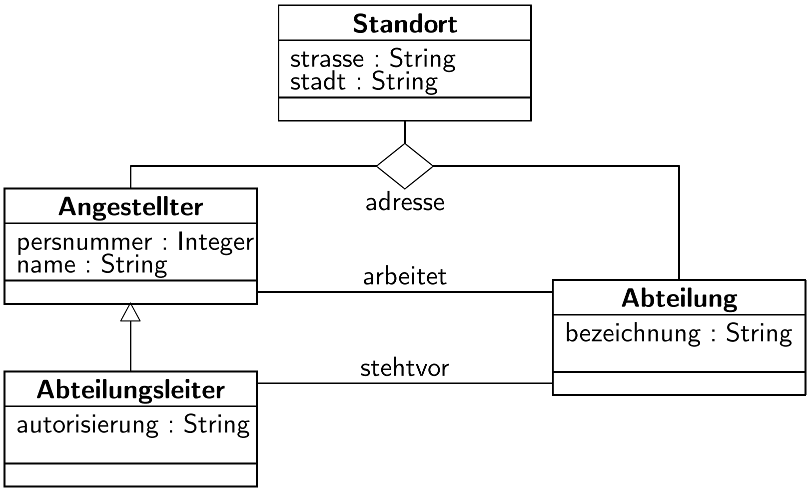
UML steht für „Unified Modeling Language“, OCL für „Object Constraint Language“. UML ist ein Standard der Object Management Group (OMG) zur Beschreibung der Strukturen und Abläufe von klassenbasierten (im Gegensatz zu prototypbasierten) objektorientierten Softwaresystemen.

Ein solches System besteht aus *Objekten*, die Nachrichten austauschen können. Jedes Objekt besitzt eine eindeutige *Identität*, ist aufgebaut aus *Attributen* und verfügt über *Methoden*, das heißt Operationen, mit denen es auf Nachrichten reagieren kann. Die Menge aller Attributwerte bildet den *Objektzustand*, der durch Methoden des Objekts verändert werden kann. Jedes Objekt ist *Instanz* einer *Klasse*. Eine Klasse modelliert die gemeinsamen Merkmale der Menge ihrer Objekte. Die Klasse eines Objekts bestimmt, welche Attribute und Methoden es hat. Zwischen Klassen können Beziehungen bestehen, insbesondere *Unterklassen-Oberklassen-Beziehungen* und *Assoziationen*.

UML umfasst circa 15 Teilsprachen für unterschiedliche Aspekte der objektorientierten Modellierung. Die meisten dieser Teilsprachen sind *Diagrammsprachen*, haben also eine graphische Syntax. Die Teilsprachen lassen sich grob einteilen in Verhaltensdiagramme und Strukturdiagramme. Verhaltensdiagramme dienen zur dynamischen Modellierung, z.B. Zustandsdiagramme zur Beschreibung von Übergängen zwischen Objektzuständen, Sequenzdiagramme zur Beschreibung der zeitlichen Anordnung des Nachrichtenaustauschs zwischen Objekten. Strukturdiagramme dienen zur statischen Modellierung. Zu ihnen gehören insbesondere die Klassendiagramme, die eine Weiterentwicklung der Diagramme des Entitäts-Relations-Modells sind.

UML-Klassendiagramme.

Das folgende Klassendiagramm modelliert den gleichen Ausschnitt der Realität wie das obige Entitäts-Relations-Diagramm, aber mit zusätzlichen Attributen.



Eine Klasse entspricht einer Entität und wird ebenso durch ein Rechteck dargestellt. Die waagrechten Striche darin trennen Klassennamen, Attribute und Methoden. Wenn, wie im Beispiel, keine Methoden betrachtet werden, bleibt der entsprechende Teil im Rechteck leer.

Die Unterklassen-Oberklassen-Beziehung wird durch einen Pfeil dargestellt, dessen Spitze ein nicht ausgefülltes Dreieck ist. In diesem Beispiel wird damit modelliert, dass alle Abteilungsleiter spezielle Angestellte sind. Die Klasse **Abteilungsleiter** erbt deshalb alle Attribute der Klasse **Angestellter**. Objekte der Klasse **Abteilungsleiter** haben aber, im Gegensatz zu sonstigen Objekten von **Angestellter**, ein zusätzliches Attribut autorisierung, das angeben soll, ob die jeweiligen Abteilungsleiter den Angestellten ihrer Abteilungen nur Urlaub oder auch Gehaltserhöhungen bewilligen dürfen.

Die sonstigen Beziehungen zwischen Klassen werden Assoziationen genannt. Die Darstellung von mehrstelligen Assoziationen ist fast gleich zu der Darstellung von Relationen im ER-Modell. Bei zweistelligen Assoziationen ist die graphische Darstellung gegenüber dem ER-Modell einerseits etwas vereinfacht, andererseits aber differenzierter. In UML gibt es zusätzlich gerichtete Assoziationen, qualifizierte Assoziationen, Aggregationen (Teil-Ganzes-Beziehungen), Kompositionen (Teil-Ganzes-Beziehungen mit Existenzabhängigkeit) und einige andere. Assoziationen können außerdem mit Rollennamen und Multiplizitäten versehen werden, analog zu Rollennamen und Kardinalitäten im ER-Modell. Einige dieser Sprachmittel können auch im obigen Beispiel eingesetzt werden, um die Modellierung zu verbessern.

Einige weitere graphische Sprachmittel dienen zur Darstellung von Konzepten, die in objektorientierten Programmiersprachen verbreitet sind, zum Beispiel Zugriffsrechte für Attribute, abstrakte Klassen und Schnittstellen. Es gibt Werkzeuge, die UML-Klassendiagramme in Programmcode geeigneter Programmiersprachen übersetzen können und umgekehrt.

Wie im Fall des ER-Modells kann man eine Übersetzung der Diagramme in Formeln einer Sprache der Prädikatenlogik erster Stufe angeben, um auf diese Weise die Bedeutung der Diagramme zu spezifizieren. Das obige Diagramm wird in folgende Formeln übersetzt:

$$\forall x \forall y\, ((\mathit{Standort}(x) \wedge \mathit{strasse}(x,y)) \Rightarrow \mathit{String}(y))$$
$$\forall x \forall y\, ((\mathit{Standort}(x) \wedge \mathit{stadt}(x,y)) \Rightarrow \mathit{String}(y))$$
$$\forall x \forall y\, ((\mathit{Angestellter}(x) \wedge \mathit{persnummer}(x,y)) \Rightarrow \mathit{Integer}(y))$$
$$\forall x \forall y\, ((\mathit{Angestellter}(x) \wedge \mathit{name}(x,y)) \Rightarrow \mathit{String}(y))$$
$$\forall x \forall y\, ((\mathit{Abteilung}(x) \wedge \mathit{bezeichnung}(x,y)) \Rightarrow \mathit{String}(y)$$
$$\forall x \forall y\, ((\mathit{Abteilungsleiter}(x) \wedge \mathit{autorisierung}(x,y)) \Rightarrow \mathit{String}(y)$$

$$\forall x\, (\mathit{Abteilungsleiter}(x) \Rightarrow \mathit{Angestellter}(x))$$

$$\forall x \forall y \forall z\, (\mathit{adresse}(x,y,z) \Rightarrow (\mathit{Angestellter}(x) \wedge \mathit{Standort}(y) \wedge \mathit{Abteilung}(z)))$$
$$\forall x \forall y\, (\mathit{arbeitet}(x,y) \Rightarrow (\mathit{Angestellter}(x) \wedge \mathit{Abteilung}(y)))$$
$$\forall x \forall y\, (\mathit{stehtvor}(x,y) \Rightarrow (\mathit{Abteilungsleiter}(x) \wedge \mathit{Abteilung}(y)))$$

Object Constraint Language (OCL).

OCL ist eine Teilsprache von UML, die eine textuelle Syntax hat. OCL dient dazu, Bedingungen zu formulieren, die in einem objektorientierten Softwaresystem gelten. Man kann zum Beispiel die Bedingungen, die durch ein UML-Klassendiagramm repräsentiert sind, auch in OCL ausdrücken.

Interessanter ist aber, dass man in OCL zusätzliche Bedingungen formulieren kann, die mit den Sprachmitteln der UML-Klassendiagramme nicht ausdrückbar sind. Im oben modellierten Ausschnitt der Realität gelte zum Beispiel zusätzlich, dass alle Abteilungsleiter auch in den Abteilungen arbeiten, denen sie vorstehen. Diese Bedingung ist im Klassendiagramm nicht enthalten und lässt sich auch nicht damit formulieren. In OCL kann man sie wie folgt formulieren:

```
context Abteilungsleiter
inv: self.stehtvor -> forAll(y | self.arbeitet->includes(y))
```

Die Semantik von derartigen OCL-Ausdrücken wird üblicherweise mengentheoretisch definiert. Aber man kann sie auch durch eine Übersetzung in die Prädikatenlogik spezifizieren, in diesem Fall in die Formel $\forall x\,(\mathit{Abteilungsleiter}(x) \Rightarrow \forall y\,(\mathit{stehtvor}(x,y) \Rightarrow \mathit{arbeitet}(x,y)))$.

2.7 Beschränkte Quantifikation

Definition 2.7.1 (BQ-Formel). Die Menge der beschränkt quantifizierten $\mathcal{L}$-Formeln, kurz BQ-$\mathcal{L}$-Formeln oder BQ-Formeln, ist die kleinste Menge, die die folgenden Bedingungen erfüllt:

1. $\top$ und $\bot$ und alle atomaren $\mathcal{L}$-Formeln sind BQ-$\mathcal{L}$-Formeln.
2. Ist F eine BQ-$\mathcal{L}$-Formel, so ist $\neg F$ eine BQ-$\mathcal{L}$-Formel.
3. Sind F und G BQ-$\mathcal{L}$-Formeln und ist θ ein zweistelliger Junktor, so ist auch $(F\ \theta\ G)$ eine BQ-$\mathcal{L}$-Formel.
4. Ist F eine BQ-$\mathcal{L}$-Formel und A ein $\mathcal{L}$-Atom mit $\{x_1, \ldots, x_n\} \subseteq \mathit{var}(A)$, so sind $\forall x_1 \ldots \forall x_n\, \neg A$ und $\forall x_1 \ldots \forall x_n\, (A \Rightarrow F)$ und $\exists x_1 \ldots \exists x_n\, A$ und $\exists x_1 \ldots \exists x_n\, (A \wedge F)$ ebenfalls BQ-$\mathcal{L}$-Formeln. ■

Beispiel 2.7.2.

$\forall x\,(p(x) \Rightarrow \exists y\,(q(x,y) \wedge r(x,y)))$	ist eine geschlossene BQ-Formel.
$\forall x\, p(x)$	ist keine BQ-Formel.
$\exists x\, \neg p(x)$	ist keine BQ-Formel ■

BQ-Formeln sind in Anwendungen wichtig, weil sie ein Charakteristikum natürlicher Sprachen widerspiegeln: Wenn natürlichsprachliche Gegenstücke zu Quantoren verwendet werden, wird immer die Variable durch eine positive Aussage qualifiziert:

„alle Menschen sind sterblich“	entspricht	$\forall x\,(m(x) \Rightarrow s(x))$.
„es gibt gute Romane“	entspricht	$\exists x\,(r(x) \wedge g(x))$.

Lange bevor die PL1S formalisiert wurde, untersuchte Aristoteles, welche Argumentationsmuster mit Sätzen möglich sind, deren Subjekt „alle Menschen“, „keine Menschen“, „manche Menschen“ lautet. Dabei behandelte er die folgenden vier Formen:

alle A sind B
manche A sind B
keine A sind B
manche A sind keine B

die alle BQ-Formeln entsprechen, nämlich den BQ-Formeln:

$\forall x\,(a(x) \Rightarrow b(x))$
$\exists x\,(a(x) \wedge b(x))$
$\forall x\,(a(x) \Rightarrow \neg b(x))$ oder $\neg \exists x\,(a(x) \wedge b(x))$
$\exists x\,(a(x) \wedge \neg b(x))$

Dabei handelt es sich um ein allgemeines Prinzip: Natürlichsprachliche Quantifizierungen lassen sich bestens mittels BQ-Formeln in die PL1S übersetzen.

2.8 Exkurs: Regelbasierte Formalismen

Ein in der Informatik weitverbreiteter Formalismus basiert auf „Regeln“ der Form

$$A_1, \ldots, A_n \to C$$

wo alle A_i und C Atome sind.

Meistens ist für Regeln eine operationale Semantik vorgegeben, die eine „Datenbasis“ von Atomen als Grundlage hat. Wenn Instanzen $A'_1, \ldots, A'_n$ in der Datenbasis vorhanden sind und die zugehörige Instanz C' nicht, dann kann die Regel „feuern“, das heißt, C' zu der Datenbasis hinzufügen. Die in der Regel vorkommenden Variablen werden dabei aufgrund der Datenbasisinhalte gebunden (siehe auch Abschnitt 4.7 und Abschnitt 4.8, zum Begriff „Instanz“ auch Definition 3.8.15).

Regelsysteme werden unter anderem in Deduktions- und Produktionssystemen eingesetzt um z.B. Expertensysteme zu spezifizieren, in der Logikprogrammierung, oder in deduktiven und aktiven Datenbanken um Datenbanksichten (views) oder Aktionen (triggers) zu definieren.

Aus der Sicht der Logik ist eine derartige Regel eine Abkürzung für den Satz

$$\forall x_1 \ldots \forall x_m \; (\; (A_1 \wedge (\ldots \wedge A_n) \ldots) \Rightarrow C \;)$$

wobei die x_j die Variablen sind, die in den A_i oder in C vorkommen. Mit dieser Sichtweise kann man Regeln auch eine deklarative Semantik zuordnen, nämlich die logische Semantik der Sätze, für die sie stehen, und zwar so, dass die operationale und die deklarative Semantik übereinstimmen. Dies ermöglicht zum Beispiel bessere Analysen der Eigenschaften solcher Regelsysteme.

Die Regel heißt „bereichsbeschränkt“, wenn jede in C vorkommende Variable ebenfalls in mindestens einem A_i vorkommt. Bereichsbeschränkte Regelsysteme haben besonders günstige Eigenschaften. Die Bereichsbeschränkung ist eine Erweiterung der in Abschnitt 2.7 eingeführten beschränkten Quantifizierung. Der Satz

$$\forall x \forall y \forall z \; ((p(x, y) \wedge p(y, z)) \Rightarrow q(x, z))$$

der einer bereichsbeschränkten Regel entspricht, ist zwar keine BQ-Formel gemäß Definition 2.7.1, aber diese Definition kann entsprechend verallgemeinert werden.

2.9 Prädikatenlogik erster Stufe und natürliche Sprache

Wie schon mehrfach angesprochen, gibt es viele Phänomene der natürlichen Sprache, die sich sehr gut mit PL1S darstellen lassen. Andererseits gibt es aber auch viele Phänomene der natürlichen Sprache, für die die PL1S keine geeigneten Hilfsmittel zur Verfügung stellt. Unter anderem fehlen der Prädikatenlogik Konstrukte zur Darstellung von

- zeitlichen Zusammenhängen,
- Modalitäten (es ist möglich, dass ..., es ist sicher, dass ..., usw.),
- unscharfen Quantifizierungen (die meisten, wenige, ziemlich viele usw.),
- Zustandsänderungen (z.B. durch Bewegungen eines Roboters),
- Intensionalität (d.h. Aussagen über Aussagen machen),

- Typen (im Sinne der Typisierung in modernen Programmiersprachen).

Eine weitere Einschränkung wird in Abschnitt 2.11 angesprochen: gewisse einfache Sachverhalte, die in der Mathematik völlig geläufig sind, können nicht in der PL1S ausgedrückt werden.

Ist die PL1S gut oder schlecht? Die Frage ergibt nur einen Sinn in Bezug auf eine Anwendung. Zur Formalisierung des mathematischen Begriffs „Beweis", hat sich die PL1S als hervorragend geeignet erwiesen. Zur Wissensmodellierung ist sie in manchen Fälle gut, in anderen zu eingeschränkt.

2.10 Exkurs: Mehrsortige Prädikatenlogik erster Stufe

Intuitiv steht eine quantifizierte Formel $\forall x\, F$ für eine Aussage der Art „für alle Objekte gilt die von F repräsentierte Aussage" (eine genaue Definition findet sich in Kapitel 3). Es ist aber im Alltag und in der Informatik- und sogar Mathematikpraxis fast immer so, dass man nur über ganz bestimmte Objekte quantifizieren will. Wenn F eine Aussage über die Professoren des Instituts für Informatik ist, zum Beispiel, dass ihre Promotionsbetreuer auch Professoren sind, ergibt es keinen Sinn, die Aussage für Java-Programme oder für Primzahlen zu betrachten. Für diese Objekte wäre die Aussage weder wahr noch falsch, sondern unsinnig.

Man würde normalerweise formulieren „für alle Professoren..." oder „für alle Java-Programme..." oder „für alle Primzahlen..." und damit die in Frage kommenden Objekte eingrenzen.

Eine mehrsortige Logik bietet zu diesem Zweck spezielle zusätzliche Sprachkonstrukte an, sogenannte Sorten, die den Variablen und Termen syntaktisch zugeordnet werden und damit eine „Typisierung" ganz ähnlich zu höheren Programmiersprachen erlauben.

Definition 2.10.1 (Sprache [mehrsortige PL1S]). Eine Sprache $\mathcal{L}$ der mehrsortigen Prädikatenlogik erster Stufe ist gegeben durch ihre Signatur. Zusätzlich gehören zu $\mathcal{L}$ die logischen Symbole der mehrsortigen Prädikatenlogik erster Stufe.

- Die *logischen Symbole* der mehrsortigen Prädikatenlogik erster Stufe sind:
  1. Die Menge der *Hilfszeichen*) und , und (
  2. Die Menge der nullstelligen *Junktoren*: $\{\top, \bot\}$.
     Die Menge der einstelligen Junktoren: $\{\neg\}$.
     Die Menge der zweistelligen Junktoren: $\{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$.
  3. Die Menge der *Quantoren* $\{\forall, \exists\}$.
  4. Eine nichtleere (oft endliche) Menge $\mathcal{S}$ von *Sorten*.
  5. Für jede Sorte $s \in \mathcal{S}$ eine abzählbar unendliche Menge von Variablen, notiert $u^s, v^s, w^s, x^s, y^s, z^s, \ldots$ mit oder ohne Indizes.
  6. Für einige $s \in \mathcal{S}$ kann es ein Gleichheitsrelationssymbol $\doteq^s$ geben.

- Die *Signatur* einer Sprache $\mathcal{L}$ der mehrsortigen Prädikatenlogik erster Stufe besteht aus folgenden Symbolmengen:
  1. Für jede natürliche Zahl $n \geq 0$ und jedes n-Tupel $(s_1, \ldots, s_n)$ von Sorten eine (eventuell leere) Menge $Rel_{\mathcal{L}}^{(s_1,\ldots,s_n)}$ von n-stelligen *Relationssymbolen* der Sorten $(s_1, \ldots, s_n)$. Sie werden auch *Prädikatssymbole* der Sorten $(s_1, \ldots, s_n)$ genannt.
     0-stellige Relationssymbole heißen auch *Aussagensymbole*.

2. Für jede natürliche Zahl $n \geq 0$ und jedes $(n+1)$-Tupel $(s_1, \ldots s_n, s_{n+1})$ von Sorten eine (eventuell leere) Menge $Fun_{\mathcal{L}}^{(s_1,\ldots s_n,s_{n+1})}$ von n-stelligen Funktionssymbolen der Argumentsorten $(s_1, \ldots, s_n)$ und der Ergebnissorte s_{n+1}.
   0-stellige Funktionssymbole heißen auch *Konstanten*.

Alle erwähnten Mengen sind abzählbar (üblicherweise sogar aufzählbar) und paarweise disjunkt, und kein Symbol einer dieser Mengen besteht aus einer Sequenz von Symbolen, in der ein Symbol aus einer der anderen Mengen vorkommt.
Es wird vorausgesetzt, dass $\mathcal{L}$ mindestens eine Konstante jeder Sorte enthält. ∎

Bemerkung: Die Voraussetzung, dass es mindestens eine Konstante jeder Sorte gibt, hat nur technische Gründe und könnte aufgegeben werden. ∎

Terme und Formeln einer Sprache der mehrsortigen Prädikatenlogik erster Stufe werden jetzt ganz analog wie für die PL1S definiert. Der entscheidende Unterschied ist jeweils, dass die Sorten in naheliegender Weise zusammenpassen müssen. Ist zum Beispiel c_1 eine Konstante der Sorte s_1 und c_2 eine Konstante der Sorte s_2 und f ein zweistelliges Funktionssymbol der Argumentsorten (s_1, s_2) und der Ergebnissorte s_3, dann ist $f(c_1, c_2)$ ein Term der Sorte s_3, aber $f(c_2, c_1)$ ist syntaktisch nicht zulässig. Damit können unsinnige Gebilde wie „das kleinste gemeinsame Vielfache der *if*-Anweisung und des Präsidenten“ nach rein syntaktischen Kriterien ausgeschlossen werden.

Die Definitionen von Teilformeln, freien und gebundenen Variablen usw. werden ebenfalls mit geringfügigen Anpassungen übertragen. Die Eindeutigkeitssätze für Terme und Formeln gelten natürlich ebenfalls ganz analog.

Untersorten. Es gibt Varianten der mehrsortigen Prädikatenlogik erster Stufe, die zusätzlich erlauben, „Untersortenbeziehungen“ zwischen Sorten festzulegen. Ist s_1 eine Untersorte von s_2, dann ist jeder Term der Sorte s_1 automatisch auch Term der Sorte s_2.

Auch diese Erweiterung wurde durch praktische Anforderungen der Wissensrepräsentation motiviert, da viele Bereiche in natürlicher Weise hierarchisch strukturiert werden können. In einer solchen Logik könnte man zum Beispiel eine Sorte für Studenten und eine Sorte für Professoren verwenden, die beide in der Untersortenbeziehung zu einer dritten Sorte für Menschen stehen. Andere geläufige Beispiele ergeben sich aus den verschiedenen aufeinander aufbauenden Arten von Zahlen.

Vergleich mit PL1S. Obwohl die mehrsortige Prädikatenlogik erster Stufe aus praktischer Sicht viele Vorteile gegenüber der klassischen Prädikatenlogik erster Stufe aufweist, unterscheidet sie sich aus theoretischer Sicht nur unwesentlich davon. Man kann nämlich die Mehrsortigkeit in der klassischen PL1S simulieren.

Dazu codiert man jede Sorte s durch ein einstelliges Relationssymbol $\dot{s}$ und drückt die in der mehrsortigen Signatur enthaltene Information durch Formeln der PL1S aus. Ist zum Beispiel f ein zweistelliges Funktionssymbol der Argumentsorten (s_1, s_2) und der Ergebnissorte s_3 (das heißt, $f \in Fun_{\mathcal{L}}^{(s_1,s_2,s_3)}$), beschreibt man dies durch die Formel

$$\forall x \forall y\, (\dot{s_1}(x) \land \dot{s_2}(y) \Rightarrow \dot{s_3}(f(x, y))).$$

Quantifizierte Formeln übersetzt man nach den Mustern

$\forall x^s F$	wird dargestellt durch	$\forall x(\dot{s}(x) \Rightarrow F)$
$\exists x^s F$	wird dargestellt durch	$\exists x(\dot{s}(x) \wedge F)$

Man erkennt hier wieder BQ-Formeln.

Damit lässt sich alles, was in einer mehrsortigen Logik ausgedrückt werden kann, auch in der klassischen Prädikatenlogik erster Stufe ausdrücken. Man verliert nichts an Ausdrucksfähigkeit, wenn man die Sorten aufgibt.

Umgekehrt kann die klassische Prädikatenlogik erster Stufe als eine „mehrsortige“ Logik mit einer einzigen Sorte aufgefasst werden (man spricht tatsächlich oft von der „einsortigen“ Prädikatenlogik erster Stufe). Auch in der Gegenrichtung ändert sich somit nichts an der Ausdrucksfähigkeit.

2.11 Exkurs: Prädikatenlogik zweiter Stufe

In der Prädikatenlogik erster Stufe beziehen sich Quantoren auf Objekte. Manche Aussagen, zum Beispiel in der Mathematik, quantifizieren aber nicht nur über Objekte, sondern über Eigenschaften von Objekten. Ein Beispiel dafür stellt der Satz 2.1.4 (Prinzip der strukturellen Induktion für aussagenlogische Formeln) dar:

- Für jede Eigenschaft $\mathcal{E}$ von $\mathcal{L}$-Formeln gilt: wenn ...

Andere Beispiele dieser Art sind

- Zwei Objekte sind genau dann gleich, wenn sie sich durch keine Eigenschaft unterscheiden.
- Es gibt keine zweistellige Beziehung, die zwischen a und b besteht und symmetrisch und antisymmetrisch ist.

Derartige Aussagen lassen sich nicht in der Prädikatenlogik erster Stufe formulieren. Dort werden Eigenschaften von Objekten und Beziehungen zwischen Objekten durch Relationssymbole repräsentiert. Rein syntaktisch kann aber an der Stelle eines Relationssymbols in einer Formel nie eine Variable stehen, und Quantoren kommen nur in Verbindung mit Variablen vor.

Die Prädikatenlogik zweiter Stufe (PL2S) ist eine Erweiterung der PL1S um Relations- und Funktionsvariablen, mit denen Quantifizierungen der obigen Art möglich sind. Damit ist die Prädikatenlogik zweiter Stufe ausdrucksstärker als die Prädikatenlogik erster Stufe.

Gegeben seien Mengen von Funktions- und von Relationsvariablen wie folgt. Die bisherigen Variablen entsprechen den nullstelligen Funktionsvariablen.

Definition 2.11.1 (Sprache [PL2S]). Eine Sprache $\mathcal{L}$ der Prädikatenlogik zweiter Stufe ist gegeben durch ihre Signatur. Zusätzlich gehören zu $\mathcal{L}$ die logischen Symbole der Prädikatenlogik zweiter Stufe.

- Die *logischen Symbole* der Prädikatenlogik zweiter Stufe sind:
  1. Die Menge der *Hilfszeichen*) und , und (
  2. Die Menge der nullstelligen *Junktoren*: $\{\top, \bot\}$.
     Die Menge der einstelligen Junktoren: $\{\neg\}$.
     Die Menge der zweistelligen Junktoren: $\{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$.

3. Die Menge der *Quantoren* $\{\forall, \exists\}$.
4. Für jede natürliche Zahl $n \geq 0$ eine abzählbar unendliche Menge von n-stelligen *Relationsvariablen*, notiert $R_1^n, R_2^n, \dots$ mit oder ohne Indizes.
5. Für jede natürliche Zahl $n \geq 0$ eine abzählbar unendliche Menge von n-stelligen *Funktionsvariablen*, notiert $F_1^n, F_2^n, \dots$ (für $n = 0$ auch $u, v, w, x, y, z, \dots$) mit oder ohne Indizes.
6. Das Gleichheitsrelationssymbol $\doteq$ falls vorhanden.

- Die *Signatur* einer Sprache $\mathcal{L}$ der Prädikatenlogik zweiter Stufe besteht aus folgenden Symbolmengen:

  1. Für jede natürliche Zahl $n \geq 0$ eine (eventuell leere) Menge $Rel_{\mathcal{L}}^n$ von n-stelligen *Relationssymbolen*. Diese werden auch *Prädikatssymbole* genannt.
     0-stellige Relationssymbole heißen auch *Aussagensymbole*.
     Enthält $Rel_{\mathcal{L}}^2$ das besondere zweistellige Relationssymbol $\doteq$, genannt *Gleichheitsrelationssymbol*, so heißt $\mathcal{L}$ eine Sprache der Prädikatenlogik zweiter Stufe *mit Gleichheit*.
  2. Für jede natürliche Zahl $n \geq 0$ eine (eventuell leere) Menge $Fun_{\mathcal{L}}^n$ von n-stelligen *Funktionssymbolen*.
     0-stellige Funktionssymbole heißen auch *Konstanten*.

Alle erwähnten Mengen sind abzählbar (üblicherweise sogar aufzählbar) und paarweise disjunkt, und kein Symbol einer dieser Mengen besteht aus einer Sequenz von Symbolen, in der ein Symbol aus einer der anderen Mengen vorkommt.
Es wird vorausgesetzt, dass $\mathcal{L}$ mindestens eine Konstante enthält. ■

Terme werden ähnlich wie für eine Sprache der Prädikatenlogik erster Stufe definiert. Der einzige Unterschied ist, dass auch Funktionsvariablen anstelle von Funktionssymbolen zum Termaufbau verwendet werden können.

Definition 2.11.2 (Term [PL2S]). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik zweiter Stufe. Die Menge $\mathcal{T}_{\mathcal{L}}$ der $\mathcal{L}$-Terme, kurz Terme, ist die kleinste Menge, die die folgenden Bedingungen erfüllt:

1. Jede nullstellige Funktionsvariable von $\mathcal{L}$ ist ein $\mathcal{L}$-Term.
2. Jede Konstante von $\mathcal{L}$ ist ein $\mathcal{L}$-Term.
3. Sind $t_1, \dots, t_n$ $\mathcal{L}$-Terme, $n \geq 1$, und ist f ein n-stelliges Funktionssymbol von $\mathcal{L}$ oder eine n-stellige Funktionsvariable von $\mathcal{L}$, so ist auch $f(t_1, \dots, t_n)$ ein $\mathcal{L}$-Term. ■

Bei den Formeln ist der Aufbau ebenfalls analog zur PL1S. Die Unterschiede sind zum einen, dass auch Relationsvariablen anstelle von Relationssymbolen zum Aufbau von atomaren Formeln verwendet werden können, zum anderen, dass Quantoren auch mit Relations- und Funktionsvariablen erlaubt sind.

Definition 2.11.3 (Formel [PL2S]). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik zweiter Stufe.

- Jedes nullstellige Relationssymbol von $\mathcal{L}$ ist eine atomare $\mathcal{L}$-Formel.
  Jede nullstellige Relationsvariable von $\mathcal{L}$ ist eine atomare $\mathcal{L}$-Formel.

Sind $t_1, \ldots, t_n$ $\mathcal{L}$-Terme, $n \geq 1$, und ist p ein n-stelliges Relationssymbol von $\mathcal{L}$, oder eine n-stellige Relationsvariable von $\mathcal{L}$, so ist $p(t_1, \ldots, t_n)$ eine atomare $\mathcal{L}$-Formel.

Eine atomare $\mathcal{L}$-Formel nennt man auch kurz *atomare Formel* oder einfach *Atom*.

- Die Menge $\mathcal{F}_\mathcal{L}$ der $\mathcal{L}$-Formeln, kurz Formeln, ist die kleinste Menge, die die folgenden Bedingungen erfüllt:
  1. $\top$ und $\bot$ und alle atomaren $\mathcal{L}$-Formeln sind in $\mathcal{F}_\mathcal{L}$.
  2. Ist $F \in \mathcal{F}_\mathcal{L}$, so ist auch $\neg F \in \mathcal{F}_\mathcal{L}$.
  3. Ist $F \in \mathcal{F}_\mathcal{L}$ und $G \in \mathcal{F}_\mathcal{L}$ und θ ein zweistelliger Junktor, so ist auch $(F\ \theta\ G) \in \mathcal{F}_\mathcal{L}$.
  4. Ist $F \in \mathcal{F}_\mathcal{L}$, ist x eine Relationsvariable oder eine Funktionsvariable von $\mathcal{L}$, und ist Q ein Quantor, so ist auch $QxF \in \mathcal{F}_\mathcal{L}$. ∎

Die Eindeutigkeitssätze für Terme und Formeln gelten natürlich ebenfalls für Sprachen der Prädikatenlogik zweiter Stufe. Teilformeln und die Menge der in einem Term frei oder nicht frei vorkommenden Relationsvariablen und Funktionsvariablen werden ähnlich definiert wie für Sprachen der Aussagenlogik bzw. PL1S.

Das dritte der eingangs genannten Beispiele kann jetzt (mit vereinfachter Klammerung) folgendermaßen dargestellt werden.

$$\begin{array}{ll} \neg\exists R^2 & (\ R^2(a,b) \\ & \wedge\ \forall x \forall y\, [R^2(x,y) \Rightarrow R^2(y,x)] \\ & \wedge\ \forall x \forall y\, [R^2(x,y) \wedge R^2(y,x) \Rightarrow x \doteq y]\ ) \end{array}$$

Bemerkung: Es erscheint vielleicht naheliegend, anstelle der ausgeschriebenen Definitionen für symmetrisch und antisymmetrisch entsprechende Bezeichner einzuführen und etwa so zu definieren:

$$\forall R^2 \quad (\ \mathit{symmetrisch}(R^2) \Leftrightarrow \forall x \forall y\, [R^2(x,y) \Rightarrow R^2(y,x)]\ )$$

Das wäre aber keine Formel der PL2S, weil R^2 kein Term und folglich $\mathit{symmetrisch}(R^2)$ keine Formel ist.

Die Logik ist bezüglich der Syntax erheblich strikter als zum Beispiel die Programmiersprache Prolog, deren Formalisierung zwar eigentlich nur auf einem Fragment der Prädikatenlogik erster Stufe beruht, die aber andererseits ähnliche Ausdrücke wie den obigen erlaubt, die über die Prädikatenlogik erster (und sogar zweiter) Stufe hinausgehen. ∎

In dem unerlaubten Ausdruck $\mathit{symmetrisch}(R^2)$ steht *symmetrisch* für eine Relation über Relationen. Das ist erst in einer Sprache der dritten Stufe zugelassen. Sprachen höherer Stufen sind Sprachen, in denen Relationen über Relationen über Relationen usw. ausgedrückt werden können.

2.12 Exkurs: Prädikatenlogik höherer Stufen

Um eine Sprache der Prädikatenlogik 3. oder höherer Stufe definieren zu können, reicht es nicht mehr aus, den Variablen und Signaturelementen nur Stelligkeiten zuzuordnen. Man muss ihnen Typen des folgenden Typsystems zuordnen. Es besteht aus zwei Basistypen

ι (iota) für Objekte („Individuen“) $\qquad$ o (omikron) für Wahrheitswerte

sowie den Typkonstruktoren $\times$ für kartesische Produkte von Typen und $\rightarrow$ für funktionale

Typen. Mit diesem Typsystem ordnet man zum Beispiel in der Prädikatenlogik erster Stufe den Variablen und Signaturelementen in der folgenden Weise Typen zu:

Variable	$x : \iota$		
Konstante	$c : \iota$	0-stelliges Relationssymbol	$p : o$
1-stelliges Funktionssymbol	$f : \iota \to \iota$	1-stelliges Relationssymbol	$q : \iota \to o$
2-stelliges Funktionssymbol	$g : \iota \times \iota \to \iota$	2-stelliges Relationssymbol	$r : \iota \times \iota \to o$

Diese Typen sind lediglich ein technisches Hilfsmittel, um die Stufenzählung einführen zu können. Um das Prinzip zu erläutern, betrachten wir nur den vereinfachten Fall ohne Funktionssymbole und Funktionsvariablen. Dafür definieren wir induktiv sogenannte „einfache Typen", denen wir Stufen zuordnen:

- ι ist ein einfacher Typ der Stufe 0
- sind $\tau_1, \ldots, \tau_n$ einfache Typen, dann ist $\tau_1 \times \ldots \times \tau_n \to o$ ein einfacher Typ der Stufe $1 + \max\{\text{Stufe von } \tau_1, \ldots, \text{Stufe von } \tau_n\}$.

Die Stufe eines einfachen Typs ist also nur die Schachtelungstiefe des Typkonstruktors $\to$. Zum Beispiel ist $\iota \;\times\; ((\iota \to o) \to o) \;\times\; (\iota \times \iota \to o) \;\;\to\;\; o$ ein einfacher Typ der Stufe 3.

Mit Hilfe dieser Stufen von Typen werden die Stufen der Prädikatenlogik nach folgendem Schema definiert:

Eine Sprache der Prädikatenlogik der Stufe	enthält Symbole mit Typen der Stufen	enthält Variablen mit Typen der Stufen
$2k-1$	$\leq k$	$\leq k-1$
$2k$	$\leq k$	$\leq k$

Diese Zählweise ist etwas gewöhnungsbedürftig, weil für jede Stufe von Typen zwei Stufen der Prädikatenlogik unterschieden werden, je nachdem, ob nur Relationssymbole mit Typen dieser Stufe oder auch Variablen mit Typen dieser Stufe vorkommen. In der folgenden Illustration sind die Stufen der Typen (also die Werte k im obigen Schema) jeweils fettgedruckt:

Prädikatenlogik der Stufe	erlaubt	Beispiel
$1 = 2 \cdot \mathbf{1} - 1$	Variable mit Typ Stufe **0** $x : \iota$ Relationssymbol mit Typ Stufe **1** $r : \iota \times \iota \to o$	$\forall x\, r(x,x)$
$2 = 2 \cdot \mathbf{1}$	Relationsvariable mit Typ Stufe **1** $R : \iota \times \iota \to o$	$\exists R\, \forall x\, R(x,x)$
$3 = 2 \cdot \mathbf{2} - 1$	Relationssymbol mit Typ Stufe **2** $\mathfrak{reflexiv} : (\iota \times \iota \to o) \to o$	$\mathfrak{reflexiv}(r)$ $\forall R\, (\mathfrak{reflexiv}(R) \Leftrightarrow \forall x\, R(x,x))$
$4 = 2 \cdot \mathbf{2}$	Relationsvariable mit Typ Stufe **2** $\mathfrak{R} : (\iota \times \iota \to o) \to o$	$\exists \mathfrak{R}\, \forall R\, (\mathfrak{R}(R) \Leftrightarrow \forall x\, R(x,x))$
$\vdots$	$\vdots$	$\vdots$

Die Typen von Funktionssymbolen und -variablen sind keine „einfachen Typen" im obigen Sinn, für die wir eine Stufe definiert haben. Man müsste also noch die Stufen dieser Typen passend definieren, dann gilt das obige Schema der Stufen der Prädikatenlogik auch für Logiken mit Funktionssymbolen und -variablen.

Ein Typ der Stufe 1 hat die Form $\iota\times\ldots\times\iota\to\iota$ für ein Funktionssymbol oder $\iota\times\ldots\times\iota\to o$ für ein Relationssymbol. Diese Typen sind eindeutig bestimmt, sobald man die Art des Symbols und die Stelligkeit kennt. Deshalb kann man die Prädikatenlogik erster Stufe so aufbauen, dass die Signatur nur Symbolmengen verschiedener Stelligkeiten festlegt und die Typen gar nicht explizit erwähnt.

Die Prädikatenlogik zweiter Stufe benötigt ebenfalls nur Typen der Stufe 1. Deshalb kann auch sie noch ohne explizite Typen formalisiert werden.

Ab der Prädikatenlogik dritter Stufe reichen die Stelligkeiten allein nicht mehr, und man muss die Typen explizit in die Formalisierung einbeziehen.

2.13 Exkurs: Syntax von Modal- und Temporallogiken

Modallogik. Ziel der Modallogik ist es, die Aussagen- und Prädikatenlogik so zu erweitern, dass nicht nur wahr und falsch, sondern auch Modalitäten wie „notwendig“, „möglich“, „bekannt“, „irgendwann“ und ähnliches ausgedrückt werden können. Die Beziehung der Modallogik zur klassischen Logik ist ähnlich wie das Verhältnis des Konjunktiv zum Indikativ in der natürlichen Sprache: In der Modallogik wird wie im Konjunktiv ausgedrückt, was sein könnte oder sollte. Die Modallogik wird in der Philosophie, in der Linguistik und Computer-Linguistik, in der Künstlichen Intelligenz und zur Programmanalyse eingesetzt.

Die Syntax einer Sprache der modalen Aussagenlogik bzw. der Modallogik erster Stufe besteht in einer geringfügigen Erweiterung einer Sprache der Aussagenlogik bzw. der Prädikatenlogik erster Stufe. Es werden lediglich zwei einstellige *Modaljunktoren* (auch „Modaloperatoren“) hinzugefügt: $\Box$ (gelesen „Box“) und $\Diamond$ (gelesen „Raute“ oder „Diamant“).

Definition 2.13.1 (Sprache [Modallogik]).

- Eine Sprache $\mathcal{L}$ der modalen Aussagenlogik ist gegeben durch ihre Signatur. Zusätzlich gehören zu $\mathcal{L}$ die logischen Symbole der modalen Aussagenlogik.

  Die *logischen Symbole* der modalen Aussagenlogik sind die logischen Symbole der Aussagenlogik erweitert um die Menge der einstelligen Modaljunktoren: $\{\Box, \Diamond\}$.

  Die *Signatur* einer Sprache $\mathcal{L}$ der modalen Aussagenlogik ist genauso definiert wie für eine Sprache der Aussagenlogik.

- Eine Sprache $\mathcal{L}$ der Modallogik erster Stufe ist gegeben durch ihre Signatur. Zusätzlich gehören zu $\mathcal{L}$ die logischen Symbole der Modallogik erster Stufe.

  Die *logischen Symbole* der Modallogik erster Stufe sind die logischen Symbole der Prädikatenlogik erster Stufe erweitert um die Menge der einstelligen Modaljunktoren: $\{\Box, \Diamond\}$.

  Die *Signatur* einer Sprache $\mathcal{L}$ der Modallogik erster Stufe ist genauso definiert wie für eine Sprache der Prädikatenlogik erster Stufe. ■

Definition 2.13.2 (Formel [modale Aussagenlogik]). Sei $\mathcal{L}$ eine Sprache der modalen Aussagenlogik.

- Jedes Aussagensymbol von $\mathcal{L}$ ist eine *atomare $\mathcal{L}$-Formel.*

  Eine atomare $\mathcal{L}$-Formel nennt man auch kurz *atomare Formel* oder einfach *Atom.*

- Die Menge $\mathcal{F}_{\mathcal{L}}$ der $\mathcal{L}$-Formeln, kurz Formeln, ist die kleinste Menge, die die folgenden Bedingungen erfüllt:

  1. $\top$ und $\bot$ und alle atomaren $\mathcal{L}$-Formeln sind in $\mathcal{F}_{\mathcal{L}}$.
  2. Ist $F \in \mathcal{F}_{\mathcal{L}}$, so ist auch $\neg F \in \mathcal{F}_{\mathcal{L}}$.
  3. Ist $F \in \mathcal{F}_{\mathcal{L}}$ und $G \in \mathcal{F}_{\mathcal{L}}$ und θ ein zweistelliger Junktor, so ist auch $(F\ \theta\ G) \in \mathcal{F}_{\mathcal{L}}$.
  4. Ist $F \in \mathcal{F}_{\mathcal{L}}$, und ist μ ein einstelliger Modaljunktor, so ist auch $\mu F \in \mathcal{F}_{\mathcal{L}}$. ■

Definition 2.13.3 (Formel [Modallogik]). Sei $\mathcal{L}$ eine Sprache der Modallogik erster Stufe.

- Jedes nullstellige Relationssymbol von $\mathcal{L}$ ist eine atomare $\mathcal{L}$-Formel.
  Sind $t_1, \dots, t_n$ $\mathcal{L}$-Terme, $n \geq 1$, und ist p ein n-stelliges Relationssymbol von $\mathcal{L}$, so ist $p(t_1, \dots, t_n)$ eine atomare $\mathcal{L}$-Formel.

  Eine atomare $\mathcal{L}$-Formel nennt man auch kurz *atomare Formel* oder einfach *Atom.*

- Die Menge $\mathcal{F}_{\mathcal{L}}$ der $\mathcal{L}$-Formeln, kurz Formeln, ist die kleinste Menge, die die folgenden Bedingungen erfüllt:

  1. $\top$ und $\bot$ und alle atomaren $\mathcal{L}$-Formeln sind in $\mathcal{F}_{\mathcal{L}}$.
  2. Ist $F \in \mathcal{F}_{\mathcal{L}}$, so ist auch $\neg F \in \mathcal{F}_{\mathcal{L}}$..
  3. Ist $F \in \mathcal{F}_{\mathcal{L}}$ und $G \in \mathcal{F}_{\mathcal{L}}$ und θ ein zweistelliger Junktor, so ist auch $(F\ \theta\ G) \in \mathcal{F}_{\mathcal{L}}$.
  4. Ist $F \in \mathcal{F}_{\mathcal{L}}$, ist x eine Variable, und ist Q ein Quantor, so ist auch $QxF \in \mathcal{F}_{\mathcal{L}}$.
  5. Ist $F \in \mathcal{F}_{\mathcal{L}}$, und ist μ ein einstelliger Modaljunktor, so ist auch $\mu F \in \mathcal{F}_{\mathcal{L}}$. ■

Dabei werden $\mathcal{L}$-Terme wie in Definition 2.3.2 erklärt. Was die Syntax angeht, können die Modaljunktoren $\Box$ und $\Diamond$ also wie zusätzliche einstellige Junktoren analog zu $\neg$ benutzt werden. Die Definitionen entsprechen den Definitionen der $\mathcal{L}$-Formeln für die Aussagenlogik und für die Prädikatenlogik erster Stufe – es ist jeweils nur der letzte Fall für die Modaljunktoren hinzugekommen.

Zum Beispiel sind die folgenden Symbolfolgen Formeln, und zwar F_1, F_2 und F_3 Formeln einer Sprache der modalen Aussagenlogik und F_4 und F_5 Formeln von Sprachen der Modallogik erster Stufe (von verschiedenen Sprachen wegen der Stelligkeiten von q):

$$
\begin{aligned}
F_1 &= (\neg\Box\neg p \Rightarrow \Diamond p)\\
F_2 &= (\Diamond(p \wedge q) \Rightarrow \Diamond p)\\
F_3 &= (\Diamond\Box p \Leftrightarrow \Box\Diamond q)\\
F_4 &= \Diamond\forall x(p(x) \vee q(x))\\
F_5 &= \exists x\Box(\forall y(p(x) \wedge q(x,y)) \Rightarrow \Diamond r(x))
\end{aligned}
$$

Um das intuitive Verständnis zu erleichtern, kann man $\Box$ als „notwendig“ und $\Diamond$ als „möglich“ lesen. Dann bedeuten die Formeln F_1, F_2 und F_3:

F_1: Gilt nicht notwendigerweise $\neg p$, dann gilt möglicherweise p.
F_2: Ist $(p \wedge q)$ möglich, dann ist p möglich.
F_3: Es ist genau dann möglich, dass p notwendig ist,
wenn es notwendig ist, dass q möglich ist.

Die intuitive Dualität zwischen den Modalitäten „notwendig“ und „möglich“ gilt in den meisten Semantiken. Sie erinnert an die Dualität zwischen den Quantoren: Für jede Formel F gilt nicht notwendigerweise F, also $\neg\Box F$, genau dann wenn möglicherweise $\neg F$ gilt, also $\Diamond\neg F$.

Begriffe wie freie Vorkommen einer Variablen in einer Formel oder der Bereich eines Quantors werden ähnlich wie in der klassischen Logik definiert (Definition 2.3.6). Das Prinzip der strukturellen Induktion lässt sich für Formeln der modalen Aussagenlogik und der Modallogik erster Stufe ähnlich zum Satz 2.1.4 ausformulieren und beweisen. Die Eindeutigkeitssätze für Formeln der modalen Aussagenlogik und der Modallogik erster Stufe werden ähnlich wie Satz 2.1.7 und Satz 2.3.5 bewiesen.

Auslegungen der Modallogik. Die Modaljunktoren $\Box$ und $\Diamond$ können auf viele verschiedene Weisen ausgelegt werden. Sie können sowohl physikalische als auch logische Notwendigkeiten und Möglichkeiten ausdrücken. In beiden Fälle spricht man dann von einer *alethischen Logik*.

In einer anderen Auslegung steht $\Box$ für „geboten“, $\Diamond$ für „zulässig“ im juristischen oder moralischen Sinn. In dem Fall spricht man von einer *deontischen Logik*. In einer solchen Logik bedeutet $(\neg\Diamond p \Rightarrow \Box\neg p)$ dass, wenn p nicht zulässig ist, dann $\neg p$ geboten ist.

Der Modaljunktor $\Box$ wird auch ausgelegt als: „es ist bekannt, dass ...“ oder „es wird geglaubt, dass ...“. In solchen Fällen wird der Modaljunktor $\Diamond$ nicht immer verwendet. Zum Beispiel kann die Formel $(\Box p \Rightarrow \Box\Box p)$ bedeuten, dass, wenn p bekannt ist, dann auch bekannt ist, dass p bekannt ist. Manche Sprachen bieten mehrere Modaljunktoren $\Box_i$ (auch $\mathbf{K}_i$) zur Darstellung dessen, was verschiedene Akteure i wissen oder glauben. Weitere Sprachen drücken mit $\Box$ aus, was gewusst wird, mit $\Diamond$, was nicht ausgeschlossen wird. In all diesen Fällen spricht man von *epistemischen Logiken*.

Ferner können die Modaljunktoren $\Box$ und $\Diamond$ auch zeitlich ausgelegt werden, $\Box$ als „immer“ und $\Diamond$ entweder als „irgendwann“ oder als „manchmal“. Intuitiv bedeuten dann die Formeln F_1, F_2 und F_3:

F_1: Gilt nicht immer $\neg p$, dann gilt irgendwann p, bzw.
gilt nicht immer $\neg p$, dann gilt manchmal p.

F_2: Gilt irgendwann $(p \wedge q)$, dann gilt irgendwann p, bzw.
gilt manchmal $(p \wedge q)$, dann gilt manchmal p.

F_3: Es gibt genau dann einen Zeitpunkt, ab dem p immer gilt, wenn es immer einen Zeitpunkt gibt, zu dem q gilt, bzw.
falls manchmal immer p gilt, dann ist es immer der Fall, dass manchmal q gilt.

Die Auslegung „manchmal“ für $\Diamond$ wird in der sogenannten *Prozesslogik* verwendet. Die meisten *Temporallogiken* beruhen auf der Auslegung „irgendwann“.

Je nachdem, wie die Modaljunktoren ausgelegt werden, sollen verschiedene Sachverhalte gelten, die durch verschiedene Axiommengen dargestellt werden. Für eine alethische Logik können unter anderem die Axiome

$$A_1 = (\Box F \Rightarrow F)$$
$$A_2 = (F \Rightarrow \Box F)$$

für alle Formeln F der Logik angebracht sein. Sie bedeuten „alles, was sein muss, ist“ und „alles, was ist, muss sein“. Für eine deontische Logik dagegen sind diese Axiome unter Umständen unpassend. Es ist eben kein universelles Prinzip, dass „alles ist, was geboten ist“ (A_1) und „alles, was ist, geboten ist“ (A_2).

Bei allen Auslegungen ist die modale Aussagenlogik besonders interessant, weil ihre Ausdrucksstärke zwischen der klassischen Aussagenlogik und der klassischen Prädikatenlogik erster Stufe liegt und sie im Gegensatz zur klassischen Prädikatenlogik erster Stufe entscheidbar ist.

Lineare Temporallogik. In der Informatik spielen oft Zeitabläufe eine Rolle, z.B. zur Formalisierung der Transitionen von Schaltwerken, der Ausführungsschritte eines Programms oder der Änderungen einer Datenbank. Die meisten dieser Zeitabläufe sind diskret und linear, d.h., ihnen liegt eine endliche oder unendliche aufzählbare Sequenz $\langle z_0, z_1, z_2, \ldots\rangle$ von Zeitpunkten zu Grunde, die ein erstes Element z_0 besitzt. Wird eine diskrete und lineare Modellierung der Zeit angenommen und wird dem Modaljunktor $\Box$ die Bedeutung „immer“, dem Modaljunktor $\Diamond$ die Bedeutung „irgendwann“ gegeben, so spricht man von einer *linearen Temporallogik*.

Es ist sinnvoll und wünschenswert, dass eine Sprache der linearen Temporallogik nicht nur über Temporaljunktoren zum Ausdruck von „immer“ und „irgendwann“ verfügt, sondern auch über einen Junktor zur Bezeichnung des nächsten Zeitpunkts. Dieser Junktor ist $\circ$ (gelesen „nächst“).

Definition 2.13.4 (Sprache [Temporallogik]).

- Eine Sprache $\mathcal{L}$ der linearen temporalen Aussagenlogik ist gegeben durch ihre Signatur. Zusätzlich gehören zu $\mathcal{L}$ die logischen Symbole der linearen temporalen Aussagenlogik.

  Die *logischen Symbole* der linearen temporalen Aussagenlogik sind die logischen Symbole der Aussagenlogik erweitert um die Menge der einstelligen Temporaljunktoren: $\{\Box, \Diamond, \circ\}$.

  Die *Signatur* einer Sprache $\mathcal{L}$ der linearen temporalen Aussagenlogik ist genauso definiert wie für eine Sprache der Aussagenlogik.

- Eine Sprache $\mathcal{L}$ der linearen Temporallogik erster Stufe ist gegeben durch ihre Signatur. Zusätzlich gehören zu $\mathcal{L}$ die logischen Symbole der linearen Temporallogik erster Stufe.

  Die *logischen Symbole* der linearen Temporallogik erster Stufe sind die logischen Symbole der Prädikatenlogik erster Stufe erweitert um die Menge der einstelligen Temporaljunktoren: $\{\Box, \Diamond, \circ\}$.

  Die *Signatur* einer Sprache $\mathcal{L}$ der linearen Temporallogik erster Stufe ist genauso definiert wie für eine Sprache der Prädikatenlogik erster Stufe. ■

Die Definition der $\mathcal{L}$-Formeln für eine Sprache der linearen Temporallogik ist identisch zu den Definitionen für die Modallogik. Im jeweils letzten Fall kann μ diesmal $\Box$ oder $\Diamond$ oder $\circ$ sein.

Zum Beispiel sind F_6 und F_7 Formeln einer Sprache der linearen temporalen Aussagenlogik:

$$F_6 = (\Diamond\circ p \Leftrightarrow \circ\Diamond p)$$
$$F_7 = (\Diamond p \Rightarrow (p \vee \circ\Diamond p))$$

die wie folgt verstanden werden können:

F_6: Es gilt genau dann irgendwann, dass p unmittelbar danach zutrifft, wenn ab dem nächsten Zeitpunkt irgendwann p gilt.

F_7: Wenn p irgendwann gilt, dann gilt p jetzt, oder ab dem nächsten Zeitpunkt gilt irgendwann p.

Eine Sprache der linearen Temporallogik eignet sich zur Beschreibung von Eigenschaften von Programmabläufen, wie z.B.:

$$\forall x\, (\mathit{Prozess}(x) \Rightarrow \Box(\mathit{wartend}(x) \Rightarrow \Diamond \mathit{aktiv}(x)))$$

d.h., jeder Prozess wird immer, wenn er wartet, irgendwann danach aktiv, oder

$$\forall x\, (\mathit{Prozess}(x) \Rightarrow (\Box\Diamond \mathit{wartend}(x) \Rightarrow \Box\Diamond \mathit{aktiv}(x)))$$

d.h. jeder Prozess, der immer wieder wartet, ist auch immer wieder aktiv.

Ferner bedeutet die Korrektheit eines Programms P, dass nach endlicher Zeit eine Eigenschaft E_P erfüllt ist. Dies lässt sich in einer linearen Temporallogik einfach durch $\Diamond E_P$ ausdrücken.

Die Temporaljunktoren $\mathcal{W}$ und $\mathcal{U}$. Manche Temporallogiken bieten statt oder neben $\circ$, $\Box$ und $\Diamond$ die zweistelligen Junktoren „bis“ (notiert $\mathcal{W}$ für „waiting for“) und „es sei denn“ (notiert $\mathcal{U}$ für „unless“), womit sich manche Eigenschaften von Programmabläufen bequem formulieren lassen. Zum Beispiel drückt die folgende Formel F_8 aus, dass die anschließende Anweisung den Wert w der Variablen x nicht verändert, es sei denn, $p(x, y)$ gilt.

$$F_8 = (x \doteq w \Rightarrow \circ(x \doteq w))\ \mathcal{U}\ p(x, y)$$

```
if p(x,y) then x := x + 1 end
```

Die Junktoren $\Box$ und $\Diamond$ können mittels $\mathcal{W}$ und $\mathcal{U}$ sowie $\top$ und $\bot$ definiert werden. $\Box F$ („immer F“) entspricht beispielsweise $F\ \mathcal{W}\ \bot$ („F bis $\bot$ gilt“).

Verzweigte Temporallogik. Weitere Temporallogiken beruhen nicht auf der Annahme, dass die Zeit linear ist. Um Alternativen und den Nichtdeterminismus beschreiben zu können, wird angenommen, dass die Zeit „verzweigt“ ist, d.h., jeder Zeitpunkt kann mehrere direkte Nachfolgerzeitpunkte besitzen. Man spricht dann von *verzweigten Temporallogiken.* Für solche Logiken können sich die Temporaljunktoren $\circ$, $\Box$ und $\Diamond$ als unzureichend erweisen, weil ihnen die Intuition von einer einzigen Sequenz von Zeitpunkten zu Grunde liegt. Um dieses Problem zu beseitigen, verfügen manche Temporallogiken über die folgenden Temporaljunktoren mit den angegebenen intuitiven Auslegungen:

$\forall\Box$	für alle Zeitpunktsequenzen S und alle $z \in S$
$\exists\Box$	es gibt Zeitpunktsequenzen S, so dass für alle $z \in S$
$\forall\Diamond$	für alle Zeitpunktsequenzen S und manche $z \in S$
$\exists\Diamond$	es gibt Zeitpunktsequenzen S und $z \in S$
$\forall\circ$	für alle unmittelbaren Nachfolgerzeitpunkte
$\exists\circ$	es gibt unmittelbare Nachfolgerzeitpunkte

Dabei ist $\forall\Box$, $\exists\Box$, $\forall\Diamond$, $\exists\Diamond$, $\forall\circ$, $\exists\circ$ jeweils *ein* Temporaljunktor, der mit zwei Symbolen geschrieben wird.

3 Semantik

In diesem Kapitel wird präzisiert, wie Formeln der Aussagenlogik und der Prädikatenlogik erster Stufe Wahrheitswerte zugeordnet werden können. Darauf bauen dann formale Definitionen von semantischen Begriffen wie der „Folgerung“ auf.

Die klassische Logik – Aussagenlogik sowie Prädikatenlogik – hat zwei Wahrheitswerte: wahr, notiert w, und falsch, notiert f. Dabei handelt es sich um eine Festlegung, die keineswegs selbstverständlich ist. Mehrwertige Logiken wären in der Informatik oft vorteilhafter, etwa zur Wissensrepräsentation (drei sinnvolle Wahrheitswerte sind: bekanntlich wahr, bekanntlich falsch, Wahrheitswert unbekannt) oder zur Beschreibung von Programmabläufen (sinnvolle Wahrheitswerte sind: terminiert erfolgreich, terminiert erfolglos, terminiert nicht).

Wie die zwei Wahrheitswerte notiert werden, ist unwesentlich: oft werden 0 statt f und 1 statt w verwendet. Man könnte z.B. auch $*$ und $\odot$ benutzen. Was unter Wahrheit und Falschheit zu verstehen ist, soll außerdem nicht von nichtmathematischen Überlegungen belastet werden: hier handelt es sich im Grunde nur um Namen für die Elemente einer zweielementigen Menge $\{f, w\}$. Die Bedeutung dieser Elemente wird nicht weiter festgelegt, sondern nur Regeln für ihre Verwendung.

Die Semantik der Aussagenlogik wird dadurch definiert, dass jedem n-stelligen Junktor eine n-stellige *Boole'sche Funktion*, d.h., eine (totale) Funktion $\{f, w\}^n \to \{f, w\}$, zugeordnet wird. Wir werden sehen, dass sich der Wahrheitswert jeder zusammengesetzten Formel F aus den Wahrheitswerten der in F vorkommenden atomaren Formeln ergibt. Zunächst untersuchen wir aber, wie Junktoren mittels Boole'scher Funktionen interpretiert werden können.

Funktion. Von nun an wird das Wort „Funktion“ immer für „totale Funktion“ stehen. Unter einer *Funktion* $F : A \to B$ versteht man eine *Relation* über $A \times B$ (also eine Teilmenge von $A \times B$) mit der zusätzlichen Eigenschaft, dass es für jedes $a \in A$ ein eindeutiges $b \in B$ gibt mit $(a, b) \in F$. Eine Funktion $F : A \to B$ kann also auch als „linkstotale“ und „rechtseindeutige“ Relation über $A \times B$ definiert werden.

3.1 Boole'sche Funktionen

Die Bezeichnung geht zurück auf George Boole (Lincoln 1815 – Cork 1864).

Nullstellige Boole'sche Funktionen

Für die nullstelligen Junktoren ist die zuzuordnende Boole'sche Funktion ebenfalls nullstellig, also einfach ein Wahrheitswert, und natürlich w für $\top$ und f für $\bot$.

Bemerkung: Die Formeln $\top$ und $\bot$ sollten nicht mit den Wahrheitswerten w und f verwechselt werden, obwohl manchmal das gleiche Symbol für einen nullstelligen Junktor und seinen Wahrheitswert verwendet wird. ■

Einstellige Boole'sche Funktionen

Die einfachste einstellige Boole'sche Funktion $\{f, w\} \to \{f, w\}$ ist die Identität Id, aber sie entspricht keinem einstelligen Junktor. Welche einstellige Boole'sche Funktion zum einstelligen Junktor $\neg$ gehört, ergibt sich unmittelbar, weil nur zwei Wahrheitswerte vorausgesetzt wurden (wie die Negation in mehrwertigen Logiken zu interpretieren wäre, ist dagegen im allgemeinen nicht so offensichtlich):

$$\begin{array}{ccc} \{f,w\} & \to & \{f,w\} \\ f & \mapsto & w \\ w & \mapsto & f \end{array}$$

Üblicherweise hat diese Boole'sche Funktion den Namen NOT und wird durch eine sogenannte „Wahrheitstafel" angegeben:

	NOT
f	w
w	f

oder

X	NOT X
f	w
w	f

Bemerkung: In der Variante auf der rechten Seite ist das X nicht etwa ein Aussagensymbol und auch keine Variable der Prädikatenlogik, denn die Definition einer Boole'schen Funktion enthält keine syntaktischen Bestandteile einer Sprache $\mathcal{L}$ irgend einer Logik. X ist eine Variable im üblichen mathematischen Sinn, die für einen der Werte w bzw. f steht. Im Kontext von Boole'schen Funktionen spricht man von Boole'schen Variablen. ■

Zweistellige Boole'sche Funktionen

Zweistellige Boole'sche Funktionen $\{f,w\} \times \{f,w\} \to \{f,w\}$ können in gleicher Weise durch Wahrheitstafeln angegeben werden wie einstellige. Zum Beispiel ist durch folgende Wahrheitstafel eine zweistellige Boole'sche Funktion namens OR definiert:

		OR
f	f	f
f	w	w
w	f	w
w	w	w

oder

X	Y	X OR Y
f	f	f
f	w	w
w	f	w
w	w	w

Welche zweistelligen Boole'schen Funktionen den zweistelligen Junktoren zuzuordnen sind, ist weniger unmittelbar als bei den null- und einstelligen. Kombinatorisch gibt es $2^4 = 16$ zweistellige Boole'sche Funktionen (liest man jede Spalte als 4-stellige Binärzahl mit 0 statt f und 1 statt w, erhält man die Nummer der Spalte):

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
f	f	f	f	f	f	f	f	f	f	w	w	w	w	w	w	w	w
f	w	f	f	f	f	w	w	w	w	f	f	f	f	w	w	w	w
w	f	f	f	w	w	f	f	w	w	f	f	w	w	f	f	w	w
w	w	f	w	f	w	f	w	f	w	f	w	f	w	f	w	f	w

Davon sind aber einige uninteressant, da sie gar nicht von allen ihren Argumenten abhängen. Diese uninteressanten Boole'schen Funktionen sind:

0: die konstante zweistellige Funktion $\mathsf{FALSE}^2(X,Y) = f$
3: die zweistellige Projektion auf das 1. Argument $\pi_1^2(X,Y) = X$ (auch Id_1^2 genannt)
5: die zweistellige Projektion auf das 2. Argument $\pi_2^2(X,Y) = Y$ (auch Id_2^2 genannt)
10: die zweistellige Negation des 2. Arguments $\mathsf{NOT}_2^2(X,Y) = \mathsf{NOT}\ Y$
12: die zweistellige Negation des 1. Arguments $\mathsf{NOT}_1^2(X,Y) = \mathsf{NOT}\ X$
15: die konstante zweistellige Funktion $\mathsf{TRUE}^2(X,Y) = w$

Es verbleiben also noch zehn interessante zweistellige Boole'sche Funktionen. Sie haben folgende Namen:

1	2	4	6	7	8	9	11	13	14
AND	NIMP	NRIMP	NEQU XOR	OR	NOR	EQU	RIMP	IMP	NAND

Die Namensgebung suggeriert natürlich – zumindest in einigen Fällen – welchen zweistelligen Junktoren diese zweistelligen Boole'schen Funktionen wohl zugeordnet werden. Allerdings stehen den zehn zweistelligen Boole'schen Funktionen nur vier zweistellige Junktoren gegenüber, die in Kapitel 2 für die Aussagenlogik oder Prädikatenlogik eingeführt wurden. Tatsächlich hätte man aber auch genauso viele zweistellige Junktoren einführen können wie es Boole'sche Funktionen gibt.

Die Schreibweisen der zusätzlichen Junktoren und die Zuordnung zwischen Junktoren und Boole'schen Funktionen ergibt sich aus folgender Tabelle:

		$\wedge$	$\vee$	$\Rightarrow$	$\Leftarrow$	$\not\wedge$	$\not\vee$	$\not\Rightarrow$	$\not\Leftarrow$	$\Leftrightarrow$	$\not\Leftrightarrow$
		AND	OR	IMP	RIMP	NAND	NOR	NIMP	NRIMP	EQU	NEQU XOR
f	f	f	f	w	w	w	w	f	f	w	f
f	w	f	w	w	f	w	f	f	w	f	w
w	f	f	w	f	w	w	f	w	f	f	w
w	w	w	w	w	w	f	f	f	f	w	f

Aus der Systematik dieser Tabelle erkennt man auch, warum meistens nicht alle diese Junktoren eingeführt werden: In vielen Fällen lassen sich ihre Boole'schen Funktionen auf die Boole'schen Funktionen anderer Junktoren zurückführen und damit als überflüssig ansehen. Zum Beispiel erhält man RIMP („reverse implication") einfach durch Vertauschen der Argumente von IMP. Die Boole'sche Funktion NIMP entsteht durch Negation von IMP, d.h., X NIMP Y = NOT(X IMP Y) oder NIMP = NOT $\circ$ IMP.

Mehrstellige Boole'sche Funktionen

Junktoren einer Stelligkeit größer als zwei werden selten eingeführt. Der Grund ist, dass ihre Boole'schen Funktionen $\{f, w\}^n \to \{f, w\}$ im gleichen Sinne überflüssig sind wie einige der zweistelligen: Man kann sie stets auf Boole'sche Funktionen zurückführen, deren Stelligkeit höchsten zwei ist.

Um etwas systematischer klären zu können, wie sich Boole'sche Funktionen aufeinander zurückführen lassen, benötigen wir zunächst einige Begriffe.

Definition 3.1.1.
AND, OR, IMP, RIMP, NAND, NOR, NIMP, NRIMP heißen *primäre* Boole'sche Funktionen. EQU, NEQU heißen *sekundäre* Boole'sche Funktionen.

Eine Menge B von Boole'schen Funktionen heißt eine *vollständige Basis* (auch: *funktional vollständig*), falls jede Boole'sche Funktion als Komposition von Funktionen in B definiert werden kann.

Eine vollständige Basis B heißt *minimal*, falls keine echte Teilmenge von B eine vollständige Basis ist. ∎

Bemerkungen:

1. Primär sind die zweistelligen Boole'schen Funktionen, deren Spalten in den obigen Wahrheitstafeln entweder ein einziges w oder ein einziges f enthalten. Sekundär sind die zweistelligen Boole'schen Funktionen, deren Spalten in den obigen Wahrheitstafeln zwei w und zwei f enthalten.
2. Die obigen Wahrheitstafeln definieren, wie gesagt, zweistellige Boole'sche Funktionen. Offensichtlich kann jede n-stellige Boole'sche Funktion durch eine Wahrheitstafel mit $n+1$ Spalten und 2^n Zeilen definiert werden.

   Beispiel: die dreistellige Boole'sche Funktion, die w liefert genau dann wenn alle ihre Argumente gleich sind, wird durch folgende Wahrheitstafel mit 8 Zeilen angegeben:

f	f	f	w
f	f	w	f
f	w	f	f
f	w	w	f
w	f	f	f
w	f	w	f
w	w	f	f
w	w	w	w

3. Die Komposition bedeutet wie üblich die Hintereinanderausführung. Zum Beispiel kann die Boole'sche Funktion OR wie folgt als Komposition von NOT und AND definiert werden: $X \ \mathsf{OR}\ Y = \mathsf{NOT}(\mathsf{NOT} X \ \mathsf{AND}\ \mathsf{NOT}\, Y)$. ∎

Satz 3.1.2.

1. Für jede primäre Boole'sche Funktion F ist $\{\mathsf{F}, \mathsf{NOT}\}$ eine vollständige Basis.
2. $\{\mathsf{NAND}\}$ und $\{\mathsf{NOR}\}$ sind minimale vollständige Basen.

Beweis: (skizziert)

1. a) Jede zweistellige Boole'sche Funktion kann als Komposition von AND und NOT definiert werden.

   Die Boole'schen Funktionen $\mathsf{F}_1, \mathsf{F}_2, \mathsf{F}_3, \mathsf{F}_4$, seien wie folgt definiert:

		F_1	F_2	F_3	F_4
f	f	f	w	w	w
f	w	w	f	w	w
w	f	w	w	f	w
w	w	w	w	w	f

   Sei F eine zweistellige Boole'sche Funktion mit Wahrheitstafel:

X	Y	$X \ \mathsf{F}\ Y$
f	f	z_1
f	w	z_2
w	f	z_3
w	w	z_4

Sei I_{F} die Teilmenge von $\{1,2,3,4\}$ mit $z_i = f$ für $i \in I_{\mathsf{F}}$. Weiter sei die Schreibweise $\mathsf{AND}^{\star}\{\mathsf{F}_{i_1},\ldots,\mathsf{F}_{i_n}\}(X,Y)$ definiert als Abkürzung für die rechtsassoziativ geklammerte Form von $[X\ \mathsf{F}_{i_1}\ Y]\ \mathsf{AND}\ \ldots\ \mathsf{AND}\ [X\ \mathsf{F}_{i_n}\ Y]$. Dann gilt:

$$\mathsf{F} = \mathsf{AND}^{\star}\{\mathsf{F}_i \mid i \in I_{\mathsf{F}}\}$$

weil die Wahrheitstafel der zweistelligen Boole'schen Funktion $\mathsf{AND}^{\star}\{\mathsf{F}_i \mid i \in I_{\mathsf{F}}\}$ genau in den selben Zeilen den Wert f hat wie die Wahrheitstafel von F.

Wir haben also das beliebige zweistellige F als Komposition von $\{\mathsf{AND},\mathsf{F}_1,\mathsf{F}_2,\mathsf{F}_3,\mathsf{F}_4\}$ dargestellt. Jetzt stellen wir nur noch die F_i als Komposition von $\{\mathsf{AND},\mathsf{NOT}\}$ dar:

$$\begin{array}{rcl} X\ \mathsf{F}_1\ Y & = & \mathsf{NOT}(\mathsf{NOT}\ X\ \mathsf{AND}\ \mathsf{NOT}\ Y) \\ X\ \mathsf{F}_2\ Y & = & \mathsf{NOT}(\mathsf{NOT}\ X\ \mathsf{AND}\ \phantom{\mathsf{NOT}}\ Y) \\ X\ \mathsf{F}_3\ Y & = & \mathsf{NOT}(\phantom{\mathsf{NOT}}\ X\ \mathsf{AND}\ \mathsf{NOT}\ Y) \\ X\ \mathsf{F}_4\ Y & = & \mathsf{NOT}(\phantom{\mathsf{NOT}}\ X\ \mathsf{AND}\ \phantom{\mathsf{NOT}}\ Y) \end{array}$$

Man beachte das systematische Muster: Die Funktion F_i hat genau in der i-ten Zeile den Wert f. Seien (a,b) die (X,Y)-Werte dieser i-ten Zeile. Dann hat die Definition von F_i die Form

$$X\ \mathsf{F}_i\ Y \quad = \quad \mathsf{NOT}(\mathsf{A}(X)\ \mathsf{AND}\ \mathsf{B}(Y))$$

$$\mathsf{A} = \begin{cases} Id & \text{falls } a = w \\ \mathsf{NOT} & \text{falls } a = f \end{cases} \qquad \mathsf{B} = \begin{cases} Id & \text{falls } b = w \\ \mathsf{NOT} & \text{falls } b = f \end{cases}$$

Bemerkung: Die Notation 0 statt f und 1 statt w und $\times$ statt AND macht das Obige anschaulicher.

b) Jede n-stellige Boole'sche Funktion mit $n \geq 3$ kann als Komposition von zweistelligen Boole'schen Funktionen und NOT definiert werden.

   Das unter 1 (a) beschriebene Muster kann auch auf drei oder mehr Argumente übertragen werden.

c) Aus 1 (a) und 1 (b) folgt, dass $\{\mathsf{AND},\mathsf{NOT}\}$ eine vollständige Basis ist.

d) Ist F eine primäre Boole'sche Funktion, so kann AND als Komposition von F und NOT definiert werden.

   Das zeigt man für jede primäre Boole'sche Funktion F durch eine geeignete Definition, zum Beispiel für OR durch $X\ \mathsf{AND}\ Y = \mathsf{NOT}(\mathsf{NOT} X\ \mathsf{OR}\ \mathsf{NOT}\ Y)$.

e) AND lässt sich nicht als Komposition von einer sekundären Boole'schen Funktion und NOT definieren.

2. Anhand der Wahrheitstafel überprüft man leicht, dass $\mathsf{NOT} X = X\ \mathsf{NAND}\ X = X\ \mathsf{NOR}\ X$ gilt. Das Ergebnis folgt dann aus 1 (c), 1 (d) und der Tatsache, dass $\emptyset$ keine vollständige Basis ist. ∎

Es ist üblich, für Boole'sche Funktionen die gleiche Notation zu verwenden wie für die Junktoren, denen sie zugeordnet sind (also zum Beispiel $\wedge$ auch für AND). Wir werden uns von nun an diesem Brauch anschließen. Dabei darf aber der Unterschied zwischen Junktor und Boole'scher Funktion nicht übersehen werden!

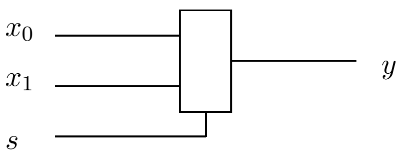
3.2 Exkurs: Schaltkreise und Boole'sche Algebren

Die Steuerungseinheit eines Computers kann als elektronischer Schaltkreis angesehen werden, der eine Menge von Boole'schen Funktionen implementiert. Damit sie nicht eingeschränkt ist, muss diese Menge von Boole'schen Funktionen eine vollständige Basis bilden.

Minimale vollständige Basen haben gegenüber nichtminimalen Basen den Vorteil, keine redundanten Elemente zu enthalten, was kostengünstig ist. Die minimale vollständige Basis {NAND} hat den zusätzlichen Vorteil, dass die Funktion NAND mit Transistoren einfach zu implementieren ist. Das ist der Grund, warum die meisten Schaltkreise viele sogenannte „NAND-Gatter" enthalten.

Bei der Entwicklung von Schaltkreisen werden zunächst die benötigten Boole'schen Funktionen bestimmt. Betrachten wir als Beispiel einen Multiplexer. Das ist ein Schaltkreis, der eingesetzt wird, wenn eine einzige Leitung für unterschiedliche Zwecke verwendet wird oder wenn eine Komponente Eingabewerte aus verschiedenen anderen Komponenten erhalten soll. Er entspricht einer mehrstelligen Boole'schen Funktion, die einfach eines ihrer Argumente als Ergebnis liefert, aber welches der Argumente, hängt von Steuergrößen ab.

Der einfache Multiplexer hat drei Eingänge s, x_0, x_1 und einen Ausgang y. Der Wert des Steuereingangs s legt dabei fest, ob der Eingang x_0 oder der Eingang x_1 auf den Ausgang y geschaltet wird (s steht für „switch" oder Schalter): falls $s = w$ dann $y = x_1$, falls $s = f$ dann $y = x_0$. Dieses Verhalten kann durch eine Wahrheitstafel beschrieben werden.



s	x_0	x_1	y
f	f	f	f
f	f	w	f
f	w	f	w
f	w	w	w
w	f	f	f
w	f	w	w
w	w	f	f
w	w	w	w

Dieser Multiplexer entspricht einer dreistelligen Boole'schen Funktion $y = \mathsf{MUX}(s, x_0, x_1)$ mit der gegebenen Wahrheitstafel. Mit der Beweistechnik von Satz 3.1.2, Punkt 1 (a), genauer gesagt, mit einer dazu dualen Variante, kann man aus der Wahrheitstafel unmittelbar ablesen, wie diese Funktion auf die Basis $\{\mathsf{NOT}, \mathsf{AND}, \mathsf{OR}\}$ zurückgeführt werden kann:

$$\begin{aligned} \mathsf{MUX}(s, x_0, x_1) = \quad & (\mathsf{NOT}\ s\ \mathsf{AND}\ (\ x_0\ \mathsf{AND}\ \mathsf{NOT}\ x_1)) \\ \mathsf{OR}\quad & (\mathsf{NOT}\ s\ \mathsf{AND}\ (\ x_0\ \mathsf{AND}\ x_1)) \\ \mathsf{OR}\quad & (\ s\ \mathsf{AND}\ (\mathsf{NOT}\ x_0\ \mathsf{AND}\ x_1)) \\ \mathsf{OR}\quad & (\ s\ \mathsf{AND}\ (\ x_0\ \mathsf{AND}\ x_1)) \end{aligned} \qquad (\star)$$

Diese Darstellung der Boole'schen Funktion wird dann weiter umgewandelt, zum Beispiel auf eine andere – für die technische Realisierung günstigere – Basis zurückgeführt, oder vereinfacht, wobei es unterschiedliche Kriterien für „einfach" gibt. Eine vereinfachte Darstellung in unserem Beispiel wäre $\mathsf{MUX}(s, x_0, x_1) = (\mathsf{NOT} s\ \mathsf{AND}\ x_0)\ \mathsf{OR}\ (s\ \mathsf{AND}\ x_1)$. Diese Darstellung der Funktion sollte keine Überraschung sein, wenn man sich an das beschriebene Verhalten des Multiplexers erinnert!

Was bei diesen Umwandlungen erlaubt ist, damit die umgewandelte Darstellung auch in jedem Fall die selbe Boole'sche Funktion beschreibt wie die Ausgangsdarstellung, ergibt sich aus den Rechenregeln der Boole'schen Algebra.

Im Formalismus der Boole'schen Algebra würde $(\star)$ so dargestellt:

$$\begin{aligned} \mathsf{MUX}(s, x_0, x_1) = \quad & (\overline{s} \times x_0 \times \overline{x_1}) \\ + \quad & (\overline{s} \times x_0 \times x_1) \\ + \quad & (s \times \overline{x_0} \times x_1) \\ + \quad & (s \times x_0 \times x_1) \end{aligned} \qquad (\star\star)$$

Eine Boole'sche Algebra kann entweder als „komplementärer distributiver Verband" (siehe Algebra-Vorlesung) definiert werden, oder wie folgt mit einem Axiomensystem.

Definition 3.2.1 (Boole'sche Algebra). Eine Boole'sche Algebra ist ein Tripel $(B, +, \times)$, so dass B eine Menge ist und $+$ und $\times$ zwei Operationen über B (d.h. Funktionen $B^2 \to B$) mit:

1. Für jede Operation gibt es ein neutrales Element, notiert 0 für $+$ und 1 für $\times$: für alle $x \in B$ gilt $0 + x = x + 0 = x$ und $1 \times x = x \times 1 = x$.
2. Beide Operationen sind kommutativ: für alle $x, y \in B$ gilt $x + y = y + x$ und $x \times y = y \times x$.
3. Jede Operation ist distributiv bezüglich der anderen: für alle $x, y, z \in B$ gilt $(x+y)\times z = (x \times z) + (y \times z)$ und $(x \times y) + z = (x+z) \times (y+z)$.
4. Zu jedem Element $x \in B$ gibt es ein komplementäres Element $\overline{x} \in B$ mit $x + \overline{x} = 1$ und $x \times \overline{x} = 0$. ■

Bemerkung: Die Assoziativität von $+$ und $\times$ folgt aus diesen Axiomen. ■

Diese Axiome gelten für das Tripel $(\{f, w\}, \mathsf{OR}, \mathsf{AND})$, wobei f das neutrale Element für OR und w das neutrale Element für AND ist. Die Boole'sche Funktion NOT bildet jedes Element von $\{f, w\}$ auf sein komplementäres Element ab. Die obige Vorgehensweise, die zur Darstellung $(\star)$ geführt hat, ergibt also immer eine Darstellung, für die die Rechenregeln der Boole'schen Algebra gelten.

3.3 Interpretationen und Modelle aussagenlogischer Formeln

In diesem Abschnitt werden wir zunächst festlegen, wie einer aussagenlogischen Formel ein Wahrheitswert zugeordnet wird, und darauf aufbauend semantische Begriffe wie die Folgerungsbeziehung formalisieren.

Betrachten wir zur Motivation zwei Aussagen:

Aussagensymbol p repräsentiert die Aussage: „ich arbeite"
Aussagensymbol q repräsentiert die Aussage: „ich spiele"

Intuitiv folgt aus der von p repräsentierten Aussage die von $(p \vee q)$ repräsentierte Aussage: wenn ich arbeite, dann gilt dass ich arbeite oder spiele. Interessanterweise folgt sie auch dann, wenn die beiden Aussagensymbole ganz anders interpretiert werden, zum Beispiel:

Aussagensymbol p repräsentiert die Aussage: „es regnet"
Aussagensymbol q repräsentiert die Aussage: „es schneit"

Die Folgerungsbeziehung ist also unabhängig von der eigentlichen (anwendungsspezifischen) Interpretation der Aussagensymbole. Sie ergibt sich bereits völlig aus den Zusammenhängen der Wahrheitswerte von p und q mit den Wahrheitswerten von $(p \vee q)$.

Diese Zusammenhänge sind hier gerade durch die Wahrheitstafel der Boole'schen Funktion OR definiert, die dem Disjunktionsjunktor zugeordnet ist. Jede Zeile dieser Wahrheitstafel legt den Wahrheitswert von $(p \vee q)$ für eine der möglichen Kombinationen von Wahrheitswerten von p, q fest. Jede Zeile stellt damit eine mögliche Interpretation der Aussagensymbole dar, die von sämtlichen anwendungsspezifischen Aspekten bis auf die Wahrheitswerte abstrahiert. In allen Interpretationen, in denen p wahr ist, ist nach der Wahrheitstafel auch $(p \vee q)$ wahr. Diese Tatsache ist die formale Begründung für die Folgerungsbeziehung zwischen den beiden Aussagen.

Definition 3.3.1 (Interpretation [Aussagenlogik]). Sei $\mathcal{L}$ eine Sprache der Aussagenlogik. Eine $\mathcal{L}$-*Interpretation* oder Wahrheitsbelegung ist eine Funktion M von der Menge der Aussagensymbole von $\mathcal{L}$ in die Menge $\{f, w\}$ der Wahrheitswerte.

Jede $\mathcal{L}$-Interpretation M wird zu einer Funktion $M^* : \mathcal{F}_{\mathcal{L}} \to \{f, w\}$ auf der Menge aller $\mathcal{L}$-Formeln fortgesetzt wie folgt (F, F_1, F_2 bezeichnen $\mathcal{L}$-Formeln):

$$\begin{array}{rcll}
M^*(\top) & := & w & \\
M^*(\bot) & := & f & \\
M^*(F) & := & M(F) & \text{falls } F \text{ Aussagensymbol von } \mathcal{L} \text{ ist} \\
M^*(\neg F) & := & \textsf{NOT}\ M^*(F) & \\
M^*(\ (F_1 \wedge F_2)\ ) & := & M^*(F_1)\ \textsf{AND}\ M^*(F_2) & \\
M^*(\ (F_1 \vee F_2)\ ) & := & M^*(F_1)\ \textsf{OR}\ M^*(F_2) & \\
M^*(\ (F_1 \Rightarrow F_2)\ ) & := & M^*(F_1)\ \textsf{IMP}\ M^*(F_2) & \\
M^*(\ (F_1 \Leftrightarrow F_2)\ ) & := & M^*(F_1)\ \textsf{EQU}\ M^*(F_2) &
\end{array}$$

■

Bemerkungen:

1. Die obige Form der Definition von M^* macht die Zuordnung zwischen Junktoren und Boole'schen Funktionen ganz explizit. Diese Definition liefert die Begründung dafür, warum man die Wahrheitswerte von Formeln auch über Wahrheitstafeln bestimmen kann, deren Überschriften $\mathcal{L}$-Formeln sind (im Gegensatz zu den Wahrheitstafeln in Abschnitt 3.1, in denen keine $\mathcal{L}$-Formeln vorkamen).
2. Man überzeuge sich, dass die Definition von M^* völlig gleichwertig zu der folgenden ist, die deutlicher betont, dass die Junktoren die gleiche Bedeutung haben sollen wie die entsprechenden metasprachlichen Junktoren. Diese zweite Form der Definition hat den Vorteil, dass ihr Stil sich auch auf die über die Aussagenlogik hinausgehenden Bestandteile der Prädikatenlogik übertragen lässt.

$$\begin{array}{rcl}
M^*(\top) & := & w \\
M^*(\bot) & := & f \\
M^*(F) & := & M(F) \quad \text{falls } F \text{ Aussagensymbol von } \mathcal{L} \text{ ist} \\
M^*(\neg F) & := & \begin{cases} w & \text{falls nicht gilt: } M^*(F) = w \\ f & \text{andernfalls} \end{cases} \\
M^*(\ (F_1 \wedge F_2)\ ) & := & \begin{cases} w & \text{falls } M^*(F_1) = w \text{ und } M^*(F_2) = w \\ f & \text{andernfalls} \end{cases}
\end{array}$$

$$M^*(\ (F_1 \vee F_2)\ ) \ := \ \begin{cases} w & \text{falls } M^*(F_1) = w \text{ oder } M^*(F_2) = w \\ f & \text{andernfalls} \end{cases}$$

$$M^*(\ (F_1 \Rightarrow F_2)\ ) \ := \ \begin{cases} w & \text{falls gilt: wenn } M^*(F_1) = w \text{ so } M^*(F_2) = w \\ f & \text{andernfalls} \end{cases}$$

$$M^*(\ (F_1 \Leftrightarrow F_2)\ ) \ := \ \begin{cases} w & \text{falls entweder } M^*(F_1) = w \text{ und } M^*(F_2) = w \\ & \qquad \text{oder weder } M^*(F_1) = w \text{ noch } M^*(F_2) = w \\ f & \text{andernfalls} \end{cases}$$

■

Ist aber durch die Definition, egal in welcher der beiden Formen, überhaupt eine Funktion M^* festgelegt? Diese muss ja total und wohldefiniert sein, das heißt, für jede $\mathcal{L}$-Formel einen und nur einen Wert liefern. Dass dies in der Tat so ist, ergibt sich aus einem allgemeinen Ergebnis über das verwendete Definitionsprinzip.

Satz 3.3.2 (strukturelle Rekursion [Formel, Aussagenlogik]). Sei $\mathcal{L}$ eine Sprache der Aussagenlogik. Um eine Funktion $\Phi : \mathcal{F}_\mathcal{L} \to B$ von der Menge der $\mathcal{L}$-Formeln in einen Wertebereich B eindeutig zu definieren, genügen folgende Angaben für Φ:

1. Basisfälle:
   - 1.1 Die Werte $b_\top$ und $b_\bot$ von Φ auf $\top$ und $\bot$ werden explizit angegeben.
   - 1.2 Der Wert b_A von Φ auf jeder atomaren $\mathcal{L}$-Formel A wird explizit angegeben.
2. Rekursionsfälle:
   - 2.1 Der Wert von Φ auf zusammengesetzten Formeln der Gestalt $\neg F$ wird in Abhängigkeit von dem Wert von Φ auf der Teilformel F definiert.
   - 2.2 Der Wert von Φ auf zusammengesetzten Formeln der Gestalt $(F\ \theta\ G)$ wird für jeden zweistelligen Junktor θ in Abhängigkeit von den Werten von Φ auf den Teilformeln F und G definiert.

Beweis: Seien Definitionen der Form 1.1, 1.2, 2.1, 2.2 gegeben. Die Rekursionsfälle seien mittels einer einstelligen Funktion $\varphi_\neg : B \to B$ und zweistelligen Funktionen $\varphi_\theta : B \times B \to B$ für zweistellige Junktoren θ definiert.

Sei $\mathcal{R}$ die Menge aller Relationen über $\mathcal{F}_\mathcal{L} \times B$, die alle Bedingungen dieser Definitionen erfüllen, also die Menge aller $R \subseteq \mathcal{F}_\mathcal{L} \times B$, für die gilt:

1.1 $(\top, b_\top) \in R$ und $(\bot, b_\bot) \in R$.
1.2 $(A, b_A) \in R$ für jedes Aussagensymbol A von $\mathcal{L}$.
2.1 Wenn $(F, b) \in R$, dann auch $(\neg F, \varphi_\neg(b)) \in R$.
2.2 Wenn $(F_1, b_1) \in R$ und $(F_2, b_2) \in R$, dann auch $((F_1 \theta F_2), \varphi_\theta(b_1, b_2)) \in R$ für jeden zweistelligen Junktor θ.

Offensichtlich erfüllt $\mathcal{F}_\mathcal{L} \times B$ diese Bedingungen, also ist $(\mathcal{F}_\mathcal{L} \times B) \in \mathcal{R} \neq \emptyset$. Ferner garantiert 1.1, dass $R \neq \emptyset$ ist für jedes $R \in \mathcal{R}$.

Wir konstruieren eine zweistellige Relation Φ über $\mathcal{F}_\mathcal{L} \times B$:

$(F, b) \in \Phi$ gdw. für alle $R \in \mathcal{R}$ gilt $(F, b) \in R$. (†)

Anders ausgedrückt ist $\Phi = \bigcap_{R \in \mathcal{R}} R$. Da die Bedingungen 1.1, 1.2, 2.1, 2.2 für alle $R \in \mathcal{R}$ gelten, gelten sie auch für Φ, das heißt, $\Phi \in \mathcal{R}$. Wir zeigen, dass Φ eine Funktion ist.

1. Φ ist linkstotal, d.h., für jede $\mathcal{L}$-Formel F gilt

   es gibt $b \in B$ mit $(F, b) \in \Phi$ $\qquad (\star)$

   Beweis durch strukturelle Induktion nach Satz 2.1.4, wobei $(\star)$ die nachzuweisende Eigenschaft $\mathcal{E}$ ist. Sämtliche Teilbeweise ergeben sich unmittelbar aus der Tatsache, dass Φ die Eigenschaften 1.1, 1.2, 2.1, 2.2 erfüllt.

2. Φ ist rechtseindeutig, d.h., für jede $\mathcal{L}$-Formel F gilt

   wenn $(F, b) \in \Phi$ und $(F, b') \in \Phi$, dann $b = b'$ $\qquad (\star\star)$

   Beweis durch strukturelle Induktion nach Satz 2.1.4, wobei $(\star\star)$ die nachzuweisende Eigenschaft $\mathcal{E}$ ist.

   Basisfall $\top$*:* Wegen 1.1 gilt $(\top, b_\top) \in \Phi$. Angenommen, es gäbe ein Tupel $(\top, b') \in \Phi$ mit $b' \neq b_\top$. Dann würde auch $R = \Phi \setminus \{(\top, b')\}$ die Bedingungen 1.1, 1.2, 2.1, 2.2 erfüllen (dass keines der in diesen Bedingungen geforderten Tupel gleich dem weggenommenen ist, ergibt sich aus dem Eindeutigkeitssatz 2.1.7). Also wäre $R = \Phi \setminus \{(\top, b')\} \in \mathcal{R}$, und damit wäre $(\top, b') \notin \Phi$. Widerspruch.
   Also besitzt $\top$ die Eigenschaft $(\star\star)$.

   Die übrigen Basisfälle werden analog gezeigt.

   Induktionsfall Negation: F besitze die Eigenschaft $(\star\star)$. Wegen der Linkstotalität von Φ gibt es $b \in B$ mit $(F, b) \in \Phi$. Wegen 2.1 gilt $(\neg F, \varphi_\neg(b)) \in \Phi$. Angenommen, es gäbe ein Tupel $(\neg F, b') \in \Phi$ mit $b' \neq \varphi_\neg(b)$. Dann würde auch $R = \Phi \setminus \{(\neg F, b')\}$ die Bedingungen 1.1, 1.2, 2.1, 2.2 erfüllen (nach dem Eindeutigkeitssatz 2.1.7 kann durch das Wegnehmen dieses Tupels höchstens Bedingung 2.1 verletzt sein, wenn es nämlich ein $b_1 \in B$ gibt mit $\varphi_\neg(b_1) = b'$ und $(F, b_1) \in \Phi$; dies ist aber ausgeschlossen, da nach Induktionsannahme F die Eigenschaft $(\star\star)$ besitzt, somit $b_1 = b$ und $\varphi_\neg(b) = \varphi_\neg(b_1) = b'$ ist, wo ja gerade $b' \neq \varphi_\neg(b)$ sein soll). Also wäre $R = \Phi \setminus \{(\neg F, b')\} \in \mathcal{R}$, und damit wäre $(\neg F, b') \notin \Phi$. Widerspruch.
   Also besitzt $\neg F$ die Eigenschaft $(\star\star)$.

   Die übrigen Induktionsfälle werden analog gezeigt.

Also ist das so konstruierte Φ eine Funktion, die die Bedingungen 1.1, 1.2, 2.1, 2.2 erfüllt. Es bleibt noch der Nachweis zu führen, dass es die einzige derartige Funktion ist.

Angenommen, es gibt noch eine weitere Funktion Φ', die die Bedingungen 1.1, 1.2, 2.1, 2.2 erfüllt. Nach Definition von $\mathcal{R}$ gilt $\Phi' \in \mathcal{R}$. Nach Definition von Φ gilt $\Phi \subseteq \Phi'$. Für jedes $F \in \mathcal{F}_\mathcal{L}$ ergibt sich damit: wenn $\Phi(F) = b$, also $(F, b) \in \Phi$, dann $(F, b) \in \Phi'$. Da Φ' nach Annahme eine Funktion ist, gibt es kein anderes b' mit $(F, b') \in \Phi'$. Das heißt $\Phi'(F) = b = \Phi(F)$ für jedes F, also $\Phi' = \Phi$. ∎

Folgesatz 3.3.3. Die Fortsetzung M^* einer $\mathcal{L}$-Interpretation M gemäß Definition 3.3.1 ist eine Funktion $\mathcal{F}_\mathcal{L} \to \{f, w\}$, liefert also für jede $\mathcal{L}$-Formel einen eindeutigen Wahrheitswert. ∎

Definition 3.3.4 (semantische Begriffe [Aussagenlogik]). Sei $\mathcal{L}$ eine Sprache der Aussagenlogik. M bezeichne eine $\mathcal{L}$-Interpretation (oder Wahrheitsbelegung), F und G bezeichne $\mathcal{L}$-Formeln, S bezeichne eine Menge von $\mathcal{L}$-Formeln.

- *M erfüllt F* (oder M ist ein *Modell* von F), notiert $M \models F$, gdw. $M^*(F) = w$. Andernfalls: *M falsifiziert F*, notiert $M \not\models F$.

  M erfüllt S (oder M ist *Modell* von S), notiert $M \models S$, gdw. $M \models F$ für jedes $F \in S$. Andernfalls: *M falsifiziert S*, notiert $M \not\models S$.

- Eine $\mathcal{L}$-Formel (oder Menge von $\mathcal{L}$-Formeln) heißt

allgemeingültig	gdw. sie von jeder $\mathcal{L}$-Interpretation erfüllt wird
erfüllbar	gdw. sie von einer $\mathcal{L}$-Interpretation erfüllt wird
falsifizierbar	gdw. sie von einer $\mathcal{L}$-Interpretation falsifiziert wird
unerfüllbar	gdw. sie von jeder $\mathcal{L}$-Interpretation falsifiziert wird

  Eine allgemeingültige Formel nennt man auch Tautologie, eine unerfüllbare Formel(menge) auch widersprüchlich, eine erfüllbare Formel(menge) auch widerspruchsfrei.

- Aus *F folgt G*, notiert $F \models G$, gdw. für jede $\mathcal{L}$-Interpretation M gilt: wenn $M \models F$ dann $M \models G$ (jedes Modell von F ist auch Modell von G).

  F, G sind *logisch äquivalent*, notiert $F \models\!\mid G$, gdw. $F \models G$ und $G \models F$ (F und G haben die selben Modelle).

- Die Folgerungsbeziehung $F \models G$ und die logische Äquivalenz $F \models\!\mid G$ sind genauso definiert, wenn F oder G oder sowohl F als auch G eine Formelmenge ist. ■

Bemerkungen:

1. Das Symbol $\models$ ist stark überladen. Es wird für die Modellbeziehung und für die Folgerungsbeziehung verwendet und zwar sowohl für Formeln als auch für Mengen von Formeln. Steht auf der linken Seite von $\models$ eine Interpretation (Wahrheitsbelegung), handelt es sich um die Modellbeziehung. Steht links eine Formel oder eine Formelmenge, dann handelt es sich um die Folgerungsbeziehung.
2. Im Fall der Folgerungsbeziehung zwischen einer Menge S und einer Formel G ist man versucht, $S \models G$ durch $F_S \models G$ zu ersetzen, wobei F_S die Konjunktion aller Elemente von S ist. Da Formeln aber endliche Gebilde sind, ist dies nur dann zulässig, wenn S eine endliche Menge ist, was die Definition 3.3.4 aber keineswegs voraussetzt.
3. Um auszudrücken, dass M ein Modell von F ist, ist für die Aussagenlogik die Notation $M^*(F) = w$ etwas gebräuchlicher, für die Prädikatenlogik die Notation $M \models F$. Wir benutzen ab sofort durchgängig die Notation $M \models F$, auch für die Aussagenlogik. ■

Satz 3.3.5. Sei $\mathcal{L}$ eine Sprache der Aussagenlogik, seien F, G $\mathcal{L}$-Formeln.

1. Ist F allgemeingültig, so ist $\neg F$ unerfüllbar.
2. Ist F unerfüllbar, so ist $\neg F$ allgemeingültig.
3. Ist F erfüllbar und falsifizierbar, so ist $\neg F$ erfüllbar und falsifizierbar.
4. $F \models G$ gdw. $\neg G \models \neg F$
   gdw. $(F \Rightarrow G)$ allgemeingültig ist
   gdw. $(F \wedge \neg G)$ unerfüllbar ist.
5. $F \models\!\mid G$ gdw. $\neg F \models\!\mid \neg G$
   gdw. $(F \Leftrightarrow G)$ allgemeingültig ist
   gdw. für jede $\mathcal{L}$-Interpretation M gilt: $M^*(F) = M^*(G)$

Beweis: Anwendung der Definitionen. ■

Der Satz erklärt, warum man sich für die Allgemeingültigkeit oder Unerfüllbarkeit interessiert: Die Folgerungsbeziehung lässt sich darauf zurückführen. Die Zurückführung auf die Unerfüllbarkeit wird mit der Technik des Widerspruchsbeweises häufig ausgenutzt. In Kapitel 4 werden wir Berechnungsverfahren – Beweismethoden genannt – kennenlernen, mit denen man ermitteln kann, ob eine Formel allgemeingültig bzw. unerfüllbar ist.

Mit der letzten Aussage des Satzes zeigt man leicht, dass die logische Äquivalenz $\models\!\mid$ eine Äquivalenzrelation auf der Menge $\mathcal{F}_\mathcal{L}$ aller $\mathcal{L}$-Formeln ist. Sie ist darüberhinaus mit der Struktur der Formeln verträglich im folgenden Sinn (eine Äquivalenzrelation, die diese Art von Strukturverträglichkeit besitzt, nennt man eine Kongruenzrelation).

Satz 3.3.6 (Ersetzungssatz). Seien F, G_1, G_2 $\mathcal{L}$-Formeln und V_{G_1} ein Vorkommen von G_1 als Teilformel in F. Es bezeichne $F\langle G_2/V_{G_1}\rangle$ die $\mathcal{L}$-Formel, die aus F entsteht, wenn das Vorkommen V_{G_1} von G_1 in F durch G_2 ersetzt wird. Dann gilt:
wenn $G_1 \models\!\mid G_2$ dann $F \models\!\mid F\langle G_2/V_{G_1}\rangle$.

Beweis: Strukturelle Induktion. ∎

Bemerkung: Oft wird $F\langle G_2/G_1\rangle$ statt $F\langle G_2/V_{G_1}\rangle$ geschrieben. Diese Notation ist nicht eindeutig, weil mehrere verschiedene Vorkommen von G_1 Teilformeln in F sein können. ∎

Satz 3.3.7. Für alle $\mathcal{L}$-Formeln F, G, H gilt:

1. Regeln der neutralen Elemente:
   $(\bot \vee F) \models\!\mid (F \vee \bot) \models\!\mid F$ $\qquad$ $(\top \wedge F) \models\!\mid (F \wedge \top) \models\!\mid F$
2. Kommutativitätsregeln:
   $(F \vee G) \models\!\mid (G \vee F)$ $\qquad$ $(F \wedge G) \models\!\mid (G \wedge F)$
3. Distributivitätsregeln:
   $((F \vee G) \wedge H) \models\!\mid ((F \wedge H) \vee (G \wedge H))$ $\qquad$ $((F \wedge G) \vee H) \models\!\mid ((F \vee H) \wedge (G \vee H))$
4. Komplementaritätsregeln:
   $(F \vee \neg F) \models\!\mid \top$ $\qquad$ $(F \wedge \neg F) \models\!\mid \bot$
5. Assoziativitätsregeln:
   $((F \vee G) \vee H) \models\!\mid (F \vee (G \vee H))$ $\qquad$ $((F \wedge G) \wedge H) \models\!\mid ((F \wedge (G \wedge H))$
6. Idempotenzregeln:
   $(F \vee F) \models\!\mid F$ $\qquad$ $(F \wedge F) \models\!\mid F$
7. Absorptionsregeln:
   $(F \vee (F \wedge G)) \models\!\mid F$ $\qquad$ $(F \wedge (F \vee G)) \models\!\mid F$
8. Regel der doppelten Negation:
   $\neg\neg F \models\!\mid F$
9. De Morgan'sche Regeln:
   $\neg(F \vee G) \models\!\mid (\neg F \wedge \neg G)$ $\qquad$ $\neg(F \wedge G) \models\!\mid (\neg F \vee \neg G)$
10. Regeln zur Auflösung des Äquivalenzjunktors:
    $(F \Leftrightarrow G) \models\!\mid ((\neg F \vee G) \wedge (F \vee \neg G)) \models\!\mid ((F \wedge G) \vee (\neg F \wedge \neg G))$
    $\neg(F \Leftrightarrow G) \models\!\mid ((F \wedge \neg G) \vee (\neg F \wedge G)) \models\!\mid ((\neg F \vee \neg G)) \wedge (F \vee G)$
11. Regeln zur Auflösung des Implikationsjunktors:
    $(F \Rightarrow G) \models\!\mid (\neg F \vee G)$ $\qquad$ $\neg(F \Rightarrow G) \models\!\mid (F \wedge \neg G)$

12. Regeln der Implikation mit neutralen Elementen:
 $(F \Rightarrow \bot) \models\!\!\!\models \neg F$ $\qquad (\top \Rightarrow F) \models\!\!\!\models F$
13. Kontrapositionsregel:
 $(F \Rightarrow G) \models\!\!\!\models (\neg G \Rightarrow \neg F)$
14. Exportregel:
 $((F \wedge G) \Rightarrow H) \models\!\!\!\models (F \Rightarrow (G \Rightarrow H))$

Beweis: Anwendung der Definitionen. ∎

Warum diese Beziehungen zwischen aussagenlogischen Formeln „Regeln" genannt werden, ergibt sich zum Teil aus Satz 3.3.5 und Satz 3.3.6. Man kann die Beziehungen dazu verwenden, eine Ausgangsformel in eine andere Formel umzuwandeln, die eine spezielle syntaktische Gestalt hat, ohne dass sich dadurch der Wahrheitswert unter irgend einer Interpretation ändert – die beiden Formeln werden völlig gleich interpretiert im Sinn der Einleitung zu Abschnitt 3.3 (mehr zu diesem Thema in Kapitel 4).

Rein äußerlich haben die Regeln 1 bis 4 eine starke Ähnlichkeit zu den in Definition 3.2.1 geforderten Eigenschaften einer Boole'schen Algebra. Der wesentliche Unterschied ist, dass in den obigen Regeln das Zeichen $\models\!\!\!\models$ steht, wo in Definition 3.2.1 ein Gleichheitszeichen steht. Mit dieser Sichtweise entsprechen die Regeln 5 bis 9 Rechenregeln, die in jeder Boole'schen Algebra gelten. Der nächste Satz zeigt, dass diese Sichtweise in der Tat gerechtfertigt ist. Wir erinnern daran, dass die logische Äquivalenz $\models\!\!\!\models$ eine Äquivalenzrelation auf der Menge $\mathcal{F}_\mathcal{L}$ aller $\mathcal{L}$-Formeln ist.

Satz 3.3.8. Sei $\mathcal{L}$ eine Sprache der Aussagenlogik. Für eine $\mathcal{L}$-Formel F bezeichne $[F]$ die Äquivalenzklasse von F bezüglich $\models\!\!\!\models$, also die Menge $\{G \in \mathcal{F}_\mathcal{L} \mid F \models\!\!\!\models G\}$. Sei $\mathcal{F}_\mathcal{L}/_{\models\!\!\!\models}$ die Menge aller Äquivalenzklassen von $\mathcal{L}$-Formeln. Wir definieren drei Funktionen auf $\mathcal{F}_\mathcal{L}/_{\models\!\!\!\models}$:

$$\begin{array}{rcl} \overline{[F]} & := & [\neg F] \\ [F] \sqcap [G] & := & [(F \wedge G)] \\ [F] \sqcup [G] & := & [(F \vee G)] \end{array}$$

Dann ist $(\mathcal{F}_\mathcal{L}/_{\models\!\!\!\models}, \sqcup, \sqcap)$ eine Boole'sche Algebra mit Nullelement $[\bot]$ und Einselement $[\top]$ und Komplementfunktion $\overline{}$.

Beweis: (skizziert)
Zunächst müssen wir uns vergewissern, dass $\overline{}$, $\sqcap$, $\sqcup$ mit den obigen Definitionen wirklich Funktionen sind. Die Linkstotalität ist offensichtlich, weil Äquivalenzklassen nie leer sind. Für die Rechtseindeutigkeit ist zu zeigen, dass die auf den Klassen definierten Werte nicht von den gewählten Repräsentanten abhängen. Wir zeigen dies exemplarisch für $\sqcap$.

Sei also $[F] = [F']$ und $[G] = [G']$, das heißt, $F \models\!\!\!\models F'$ und $G \models\!\!\!\models G'$. Durch zweimaliges Anwenden des Ersetzungssatzes 3.3.6 auf $(F \wedge G)$ erhalten wir $(F \wedge G) \models\!\!\!\models (F' \wedge G) \models\!\!\!\models (F' \wedge G')$, das heißt, $[(F \wedge G)] = [(F' \wedge G')]$ und damit $[F] \sqcap [G] = [F'] \sqcap [G']$, was zu zeigen war.

Von den in Definition 3.2.1 geforderten Eigenschaften zeigen wir exemplarisch die Kommutativität von $\sqcap$.

$$\begin{array}{rcll} [F] \sqcap [G] & = & [(F \wedge G)] & \text{nach Definition} \\ & = & [(G \wedge F)] & \text{weil nach Satz 3.3.7 gilt } (F \wedge G) \models\!\!\!\models (G \wedge F) \\ & = & [G] \sqcap [F] & \text{nach Definition} \end{array}$$

Die übrigen Teilbeweise folgen dem gleichen kanonischen Schema. ∎

Bemerkung: Das obige Argument $(F \wedge G) \models\!\!\mid (F' \wedge G)$ macht wesentlich davon Gebrauch, dass der Ersetzungssatz 3.3.6 die logische Äquivalenz sichert, sobald ein einziges Vorkommen einer Teilformel durch eine dazu äquivalente ersetzt wird, aber nicht notwendigerweise alle Vorkommen.

Wäre der Ersetzungssatz so formuliert, dass alle Vorkommen ersetzt werden müssen und käme F in G vor, wäre das obige Argument nicht begründet. ∎

Das Nullelement $[\bot]$ dieser Boole'schen Algebra besteht aus allen unerfüllbaren $\mathcal{L}$-Formeln, das Einselement $[\top]$ aus allen allgemeingültigen.

Es reicht also, anhand der Wahrheitstafeln die Regeln 1 bis 4 in Satz 3.3.7 nachzuweisen, weil diese für den Beweis von Satz 3.3.8 benötigt werden. Die Regeln 5 bis 9 in Satz 3.3.7 ergeben sich dann ebenso wie alle anderen Rechenregeln für Boole'sche Algebren nach einem einheitlichen Argumentationsmuster, das hier am Beispiel der de Morgan'schen Regel dargestellt ist:

$$\begin{array}{rcl} \overline{[F] \sqcup [G]} & = & \overline{[F]} \sqcap \overline{[G]} \\ {[\neg(F \vee G)]} & = & [(\neg F \wedge \neg G)] \\ \neg(F \vee G) & \models\!\!\mid & (\neg F \wedge \neg G) \end{array}$$

Die erste Zeile gilt, weil $(\mathcal{F}_\mathcal{L}/_{\models\!\mid}, \sqcup, \sqcap)$ nach Satz 3.3.8 eine Boole'sche Algebra ist. Die zweite Zeile ergibt sich daraus durch Einsetzen der Definitionen von $\overline{}$, $\sqcap$, $\sqcup$. Die dritte Zeile entsteht aus der zweiten mit der Definition der Äquivalenzklassen.

Die Rechenregeln aus der ersten Zeile lassen sich also rein schematisch auf aussagenlogische Formeln übertragen, indem man $=$ durch $\models\!\!\mid$ ersetzt.

Wie oben erwähnt, erlaubt Satz 3.3.5, die Folgerungsbeziehung und die logische Äquivalenz auf die Allgemeingültigkeit oder Unerfüllbarkeit zurückzuführen. Der folgende Satz ist ein wichtiges Hilfsmittel für den Nachweis der Unerfüllbarkeit.

Satz 3.3.9 (Endlichkeits- oder Kompaktheitssatz [Aussagenlogik]). Sei S eine unendliche Menge von $\mathcal{L}$-Formeln. Ist jede endliche Teilmenge von S erfüllbar, so ist S erfüllbar.

Beweis: Wenn alle endlichen Teilmengen einer Menge S von $\mathcal{L}$-Formeln erfüllbar sind, sagen wir, dass S die Eigenschaft E besitzt.

Sei S eine unendliche Menge, die die Eigenschaft E besitzt. Wir zeigen nun, dass S erfüllbar ist.

Da die Menge $\mathcal{F}_\mathcal{L}$ aller $\mathcal{L}$-Formeln abzählbar ist, gibt es eine Surjektion $\mathbb{N}\setminus\{0\} \to \mathcal{F}_\mathcal{L}$. Sei F_n jeweils das Bild von $n \in \mathbb{N} \setminus \{0\}$. Dann kommt jede $\mathcal{L}$-Formel in der Folge $F_1, F_2, \ldots, F_n, \ldots$ vor. Wir definieren rekursiv:

$$\begin{array}{rcl} S_0 & := & S \\ S_{n+1} & := & \begin{cases} S_n \cup \{F_{n+1}\} & \text{falls diese Menge die Eigenschaft } E \text{ besitzt,} \\ S_n \cup \{\neg F_{n+1}\} & \text{sonst.} \end{cases} \end{array}$$

1. Für alle $n \geq 0$ besitzt S_n die Eigenschaft E.
   Dies wird durch vollständige Induktion über n bewiesen.

   Induktionsbasis $n = 0$: Nach Annahme über S besitzt $S_0 = S$ die Eigenschaft E.

Induktionsschritt $n \to n+1$: Sei angenommen, dass S_n die Eigenschaft E besitzt. Wir zeigen, dass es keine zwei endlichen Teilmengen T_1 und T_2 von S_n gibt, so dass sowohl $T_1 \cup \{F_{n+1}\}$ als auch $T_2 \cup \{\neg F_{n+1}\}$ unerfüllbar ist.

Seien also $T_1 \subseteq S_n$ und $T_2 \subseteq S_n$ endlich. Da $T_1 \cup T_2$ endlich ist und S_n die Eigenschaft E besitzt, ist $T_1 \cup T_2$ erfüllbar. Sei M ein Modell von $T_1 \cup T_2$. Dieses Modell erfüllt sowohl T_1 als auch T_2 und natürlich eine der beiden Formeln F_{n+1} und $\neg F_{n+1}$. Also erfüllt M eine der beiden Mengen $T_1 \cup \{F_{n+1}\}$ und $T_2 \cup \{\neg F_{n+1}\}$.

Falls $S_n \cup \{F_{n+1}\}$ die Eigenschaft E besitzt, dann ist $S_{n+1} = S_n \cup \{F_{n+1}\}$ nach Definition und besitzt also die Eigenschaft E. Wenn aber $S_n \cup \{F_{n+1}\}$ die Eigenschaft E nicht besitzt, folgt aus dem obigen, dass $S_n \cup \{\neg F_{n+1}\}$ sie besitzt und nach Definition ist $S_{n+1} = S_n \cup \{\neg F_{n+1}\}$. In allen Fällen besitzt S_{n+1} die Eigenschaft E.

2. $S^* := \bigcup_{n \in \mathbb{N}} S_n$ besitzt die Eigenschaft E

   Sei T eine endliche Teilmenge von S^*. Da T endlich ist, gibt es ein $i \in \mathbb{N}$ so dass $T \subseteq S_i$. Da jedes S_i die Eigenschaft E besitzt, ist T erfüllbar.

3. Offensichtlich $S \subseteq S^*$ und für jede $\mathcal{L}$-Formel F_i ist entweder $F_i \in S^*$ oder $\neg F_i \in S^*$.

4. Sei M die folgende Interpretation: für jedes Aussagensymbol A sei $M(A) := w$ gdw. $A \in S^*$. Dann gilt für jede $\mathcal{L}$-Formel F, dass $M \models F$ gdw. $F \in S^*$. Beweis durch strukturelle Induktion.

   Basisfälle: Nach Definition von M.

   Induktionsfall zweistelliger Junktor $\wedge$: Sei F eine Formel der Gestalt $(F_1 \wedge F_2)$ und es gelte $M \models F_1$ gdw. $F_1 \in S^*$ und $M \models F_2$ gdw. $F_2 \in S^*$.
   Angenommen $M \models F$, aber $F \notin S^*$. Das heißt, $M \models F_1$ und $M \models F_2$ und $F_1 \in S^*$ und $F_2 \in S^*$, aber $(F_1 \wedge F_2) \notin S^*$. Wegen 3. ist dann $\neg(F_1 \wedge F_2) \in S^*$. Also enthält S^* die unerfüllbare endliche Teilmenge $\{F_1, F_2, \neg(F_1 \wedge F_2)\}$, im Widerspruch zu 2.
   Angenommen $M \not\models F$, aber $F \in S^*$. Dann gilt für wenigstens eine der Teilformeln $M \not\models F_i$, also $F_i \notin S^*$. Wegen 3. ist dann $\neg F_i \in S^*$. Also enthält S^* die unerfüllbare endliche Teilmenge $\{\neg F_1, (F_1 \wedge F_2)\}$ bzw. $\{\neg F_2, (F_1 \wedge F_2)\}$, im Widerspruch zu 2.
   Also gilt $M \models F$ gdw. $F \in S^*$.

   Die übrigen Induktionsfälle werden in ähnlicher Weise bewiesen.

5. Da $S \subseteq S^*$, erfüllt M auch die Menge S. ∎

Der Name „Kompaktheitssatz“ bezieht sich auf die Topologie: Wenn ein passender topologischer Raum definiert wird, kann der Satz kurz und elegant unter Anwendung eines nach Tychonov genannten Satzes bewiesen werden, mit dem die Kompaktheit eines Produktraumes festgestellt wird.

Die Wichtigkeit des Endlichkeitssatzes ergibt sich aus seiner Kontraposition:

Folgesatz 3.3.10. Eine beliebige Menge von $\mathcal{L}$-Formeln ist unerfüllbar genau dann wenn sie eine endliche Teilmenge hat, die unerfüllbar ist. ∎

3.4 Exkurs: Natürlichsprachliche Interpretationen der Junktoren

Mit den bisher vorgestellten Definitionen sind die Bedeutungen der Junktoren in der klassischen Logik eindeutig festgelegt. Diese Bedeutung entspricht in vielen Fällen der natürlichsprachlichen, aber gelegentlich weicht sie auch davon ab.

Beispiel 3.4.1 (Doppelte Negation).

> „Das Thema ist nicht uninteressant.“
> „I hob koan Alkohol net trunken!“

In der klassischen Logik hat $\neg\neg F$ stets den selben Wahrheitswert wie F. Der obere Satz bedeutet aber nicht ganz genau das selbe wie „Das Thema ist interessant“. Hier wirkt die doppelte Verneinung als Abschwächung. Im zweiten Satz dagegen wirkt die doppelte Verneinung als Verstärkung gegenüber „I hob koan Alkohol trunken“ (jedenfalls im Bayrischen).

Abschwächungen und Verstärkungen würden mehr als zwei Wahrheitswerte erfordern. ∎

Beispiel 3.4.2 (Konjunktion).

> „Er wurde krank und nahm Tabletten.“
> „Er nahm Tabletten und wurde krank.“
>
> „Es regnet und ich arbeite, die Sonne scheint und ich faulenze.“

Die Konjunktion der klassischen Logik ist kommutativ, aber die oberen beiden Sätze bedeuten nicht das selbe. Hier versteht man das Wort „und“ im Sinne einer zeitlichen Reihenfolge „und anschließend“, die eine Kausalität suggeriert. Im unteren Satz werden die Konjunktionen direkt als Kausalität verstanden.

Die klassische Logik unterstützt weder die Beschreibung von zeitlichen Zusammenhängen noch von Kausalität. ∎

Beispiel 3.4.3 (Implikation).

> „Wenn das Halteproblem nicht entscheidbar ist, hat der Mond keine Atmosphäre.“

Beide Teilsätze dieses Satzes sind wahr, also ist nach der Definition der Implikation in der klassischen Logik der gesamte Satz wahr. In der Umgangssprache würde man ihn aber zurückweisen, weil man einen völlig abwegigen kausalen Zusammenhang darunter versteht. ∎

Beispiel 3.4.4 (Exklusives Oder).

> „Entweder du fährst links, oder du fährst rechts.“
> „Entweder du bezahlst den Schaden, oder ich hole die Polizei.“

Mit „entweder ... oder“ wird normalerweise ein exklusives Oder bezeichnet. Im oberen Satz ist dies wohl auch tatsächlich gemeint, da die beiden Möglichkeiten einander ausschließen.

Im unteren Satz dagegen meint der Sprecher trotz des „Entweder ... oder“ vermutlich nicht, dass die beiden Möglichkeiten einander ausschließen, sondern dass sie ganz im Gegenteil sogar auf das Gleiche hinauslaufen. Hier klingt der unausgesprochene Zusatz mit, dass das Holen der Polizei bewirkt, dass der Angesprochene dann auch bezahlt. ∎

Wenn, wie in diesen Beispielen, die natürlichsprachliche Interpretation anders ist als die der klassischen Logik, welche Interpretation ist dann richtig? Weder die eine, noch die andere, die Antwort hängt vom Kontext ab.

Die klassische Logik gibt eine präzise Definition für die gängigste Interpretation der Junktoren. Gleichzeitig bietet sie ein Instrumentarium, das als Grundlage für die Definition anderer Interpretationen dienen kann. Tatsächlich sind aus Bemühungen, natürlichsprachliche Phänomene zu modellieren und automatisch zu bearbeiten (z.B. für die Übersetzung) neue Logiken entstanden, die von der klassischen Logik abweichen, aber doch darauf aufbauen.

3.5 Interpretationen und Modelle von Formeln der PL1S

Die Semantik der PL1S wurde von Alfred Tarski (Warschau 1902 – Berkeley 1983) formalisiert und wird häufig auch die „Tarski-Semantik“ genannt. Sie ist wesentlich komplizierter als die Semantik für die Aussagenlogik. Zunächst muss sie festlegen, für welche Objekte die Konstanten und andere Terme stehen können. Die Menge aller betrachteten Objekte wird im sogenannten Universum zusammengefasst. Darauf aufbauend ist zu definieren, wie der Wahrheitswert von atomaren Formeln zustande kommt. Ferner muss die Deutung der Quantoren festgelegt werden. Das einzige, was einfach aus der Semantik der Aussagenlogik übernommen werden kann, ist die Deutung der Junktoren. Trotzdem ist die Semantik der PL1S nur eine Erweiterung der Semantik für die Aussagenlogik.

Die folgenden beiden Sätze bilden die formale Grundlage für die erforderlichen Definitionen, anlog zu Satz 3.3.2 für die aussagenlogischen Definitionen. Sie können ähnlich bewiesen werden wie dieser.

Satz 3.5.1 (strukturelle Rekursion [Term, PL1S]). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe. Um eine Funktion $\Phi : \mathcal{T}_\mathcal{L} \to B$ von der Menge der $\mathcal{L}$-Terme in einen Wertebereich B eindeutig zu definieren, genügen folgende Angaben für Φ:

1. Basisfälle:
   1.1 Der Wert von Φ auf jeder Variablen wird explizit angegeben.
   1.2 Der Wert von Φ auf jeder Konstanten wird explizit angegeben.
2. Rekursionsfälle:
   Der Wert von Φ auf zusammengesetzten Termen der Gestalt $f(t_1, \ldots, t_n)$ wird für jedes n-stellige ($n \geq 1$) Funktionssymbol f in Abhängigkeit von den Werten von Φ auf den Termen t_1 und … und t_n definiert. ∎

Satz 3.5.2 (strukturelle Rekursion [Formel, PL1S]). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe. Um eine Funktion $\Phi : \mathcal{F}_\mathcal{L} \to B$ von der Menge der $\mathcal{L}$-Formeln in einen Wertebereich B eindeutig zu definieren, genügen folgende Angaben für Φ:

1. Basisfälle:
   1.1 Die Werte von Φ auf $\top$ und $\bot$ werden explizit angegeben.
   1.2 Der Wert von Φ auf jeder atomaren $\mathcal{L}$-Formel wird explizit angegeben.
2. Rekursionsfälle:
   2.1 Der Wert von Φ auf zusammengesetzten Formeln der Gestalt $\neg F$ wird in Abhängigkeit von dem Wert von Φ auf der Teilformel F definiert.

2.2 Der Wert von Φ auf zusammengesetzten Formeln der Gestalt $(F\ \theta\ G)$ wird für jeden zweistelligen Junktor θ in Abhängigkeit von den Werten von Φ auf den Teilformeln F und G definiert.

2.3 Der Wert von Φ auf zusammengesetzten Formeln der Gestalt $Qx\,F$ wird für jeden Quantor Q in Abhängigkeit von dem Wert von Φ auf der Teilformel F definiert. ∎

Eine Interpretation für eine Sprache $\mathcal{L}$ der Prädikatenlogik erster Stufe muss in erster Linie die Symbole der Signatur von $\mathcal{L}$ interpretieren. Die Grundidee ist, dass sie den Funktionssymbolen Funktionen zuordnet und den Relationssymbolen Relationen. Zusätzlich müssen aber auch die Variablen berücksichtigt werden. Dies geschieht durch eine sogenannte Umgebung.

Relation vs. Prädikat. Das n-fache kartesische Produkt $\overbrace{D \times \cdots \times D}^{n-\text{mal}} = D^n$ einer Menge D ist die Menge aller n-Tupel $(d_1, \ldots, d_n)$ mit $d_i \in D$.
Eine n-stellige Relation über D ist eine Teilmenge $R \subseteq D^n$, also eine Menge von n-Tupeln. Ein n-stelliges Prädikat über D ist eine n-stellige Funktion $P : D^n \to \{f, w\}$ mit Boole'schem Ergebnis, also eine Funktion, die jedes n-Tupel auf einen Wahrheitswert abbildet.

Zu jeder Relation R gibt es ein charakteristisches Prädikat, das die Tupel in R auf w abbildet und die anderen Tupel auf f. Zu jedem Prädikat P gibt es eine charakteristische Relation, die genau die Tupel enthält, die von P auf w abgebildet werden. Jede Relation ist die charakteristische Relation ihres charakteristischen Prädikats und umgekehrt.

Das gilt auch für $n = 0$, wobei D^0 die Menge aller 0-stelligen Tupel über D ist: $D^0 = \{\,()\,\}$. Diese Menge hat ein einziges Element $()$, das gar nicht von D abhängt. Also hat sie genau zwei Teilmengen, die leere Menge und sich selbst, die somit die einzigen 0-stelligen Relationen über D sind. Das charakteristische Prädikat der Relation $\emptyset$ ist das 0-stellige Prädikat mit konstantem Wert f, das der Relation $\{\,()\,\}$ das 0-stellige Prädikat mit konstantem Wert w.

Da Relationen und Prädikate einander eineindeutig entsprechen, ist es Geschmackssache, mit welchem von beiden man die Interpretationen formalisiert. Von dieser Entscheidung hängt dann ab, welche der synonymen Bezeichnungen *Relationssymbol* und *Prädikatssymbol* für die Signaturelemente bevorzugt wird. Der Name „Prädikatenlogik" deutet darauf hin, dass ihre Entwickler ursprünglich eine Formalisierung mit Prädikaten im Sinn gehabt haben dürften. Es ist aber eher ein historischer Zufall, dass für diese Logik nicht die Bezeichnung „Relationenlogik" eingeführt wurde.

Die Definition 3.3.1 einer aussagenlogischen Interpretation M ist auf Prädikaten aufgebaut. Sie ordnet jedem 0-stelligen Relationssymbol p ein 0-stelliges Prädikat $M^*(p)$ zu, also einen Wahrheitswert. Man könnte die Definition auf Relationen umschreiben, indem man eine Notation p^M für die Relation einführt, die M dem Relationssymbol p zuordnet, und dann statt $M^*(p) = w$ überall schreibt $() \in p^M$. Diese Formalisierung würde für die Aussagenlogik unnötig umständlich wirken, entspricht aber genau der Formalisierung mit Prädikaten.

In diesem Abschnitt wird die Semantik der Prädikatenlogik erster Stufe auf Relationen aufgebaut, wobei konsequenterweise auch 0-stellige Relationssymbole mit 0-stelligen Relationen statt mit Wahrheitswerten interpretiert werden.

Definition 3.5.3 (Umgebung). Eine D-Umgebung (oder D-Variablenbelegung, kurz D-Belegung) in eine nichtleere Menge D ist eine Funktion U, die jeder Variablen x einen Wert $U(x) \in D$ zuordnet. ∎

Definition 3.5.4 (Interpretation [PL1S]). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe. Eine $\mathcal{L}$-*Interpretation* M ist ein Tripel (D, Ab, U) mit:

1. D ist eine nichtleere Menge, genannt das Universum von M (auch: Diskursuniversum, Diskursbereich, Domäne).
2. Ab ist eine auf der Signatur von $\mathcal{L}$ definierte Abbildung mit:

   2.1 Für jedes n-stellige Funktionssymbol f von $\mathcal{L}$ ist $Ab(f) : D^n \to D$ eine n-stellige Funktion. Für $n = 0$ ist $Ab(f) \in D$.

   2.2 Für jedes n-stellige Relationssymbol p von $\mathcal{L}$ ist $Ab(p) \subseteq D^n$ eine n-stellige Relation. Für $n = 0$ ist $Ab(p) = \emptyset$ oder $Ab(p) = \{\,()\,\}$.

3. U ist eine D-Umgebung.

Ist $M = (D, Ab, U)$ eine $\mathcal{L}$-Interpretation, so bezeichnet

$Univ(M)$	$:=$	D	das Universum von M
f^M	$:=$	$Ab(f)$	die Funktion zum Funktionssymbol f
p^M	$:=$	$Ab(p)$	die Relation zum Relationssymbol p

■

Das Universum einer Interpretation darf nicht leer sein, damit U und Ab überhaupt definierbar sind. Außerdem würde sonst der (uninteressante) Sonderfall auftreten, dass jede Existenzaussage von vornherein falsch wäre.

Definition 3.5.5. Der Wert t^M eines $\mathcal{L}$-Terms t unter einer $\mathcal{L}$-Interpretation $M = (D, Ab, U)$ ist rekursiv über den Aufbau von t definiert:

x^M	$:=$	$U(x)$	(Variable)
c^M	$:=$	$Ab(c)$	(Konstante)
$f(t_1, \dots, t_n)^M$	$:=$	$Ab(f)(t_1^M, \dots, t_n^M)$	(zusammengesetzter Term)

■

Diese Definition ist eine Anwendung des Definitionsprinzips der strukturellen Rekursion für Terme. Der Wert eines Terms unter einer Interpretation ist immer ein Objekt aus dem Universum der Interpretation.

Notation 3.5.6. Ist D eine nichtleere Menge, U eine D-Umgebung, $d \in D$ und x eine Variable, so bezeichnet $U[d/x]$ die folgende D-Umgebung:

$$U[d/x](y) := \begin{cases} U(y) & \text{falls } y \neq x \\ d & \text{falls } y = x \end{cases}$$

Ist $M = (D, Ab, U)$ eine $\mathcal{L}$-Interpretation und $d \in D$, so bezeichnet $M[d/x]$ die $\mathcal{L}$-Interpretation $(D, Ab, U[d/x])$ ■

Die Umgebung $U[d/x]$ stimmt also mit der Umgebung U überein, außer dass sie der Variablen x den Wert d zuordnet, der ein anderer sein kann als $U(x)$. Die $\mathcal{L}$-Interpretation $M[d/x]$ stimmt mit der $\mathcal{L}$-Interpretation M überein, außer dass sie die Variable x mit dem Wert d interpretiert.

Definition 3.5.7 (Modellbeziehung [PL1S]). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe. Für eine $\mathcal{L}$-Interpretation $M = (D, Ab, U)$ und eine $\mathcal{L}$-Formel F ist die Modellbeziehung $M \models F$ rekursiv über den Aufbau von F definiert:

$$
\begin{array}{lll}
M \models \top & & \text{gilt} \\
M \models \bot & & \text{gilt nicht} \\
M \models p & \text{gdw.} & () \in p^M \qquad\qquad\qquad (p \text{ 0-stellig}) \\
M \models p(t_1, \ldots, t_n) & \text{gdw.} & (t_1^M, \ldots, t_n^M) \in p^M \;\; (p\ n\text{-stellig}, n \geq 1) \\[4pt]
M \models \neg G & \text{gdw.} & M \models G \text{ nicht gilt} \\
M \models (G_1 \land G_2) & \text{gdw.} & M \models G_1 \text{ und } M \models G_2 \\
M \models (G_1 \lor G_2) & \text{gdw.} & M \models G_1 \text{ oder } M \models G_2 \\
M \models (G_1 \Rightarrow G_2) & \text{gdw.} & \text{wenn } M \models G_1, \text{ so } M \models G_2 \\
M \models (G_1 \Leftrightarrow G_2) & \text{gdw.} & \text{entweder } M \models G_1 \text{ und } M \models G_2, \\
 & & \text{oder weder } M \models G_1 \text{ noch } M \models G_2 \\[4pt]
M \models \forall x\, G & \text{gdw.} & \text{für alle } d \in D \text{ gilt } M[d/x] \models G \\
M \models \exists x\, G & \text{gdw.} & \text{es gibt ein } d \in D \text{ mit } M[d/x] \models G
\end{array}
$$

■

Eine $\mathcal{L}$-Interpretation M ordnet jedem $\mathcal{L}$-Term t ein Element t^M ihres Universums zu. Für eine Formel F ist die Notation F^M nicht erklärt, aber in einer Formalisierung mit Prädikaten würde man sie für den (hier nicht eingeführten) Wahrheitswert der Formel F verwenden und $F^M = w$ anstelle von $M \models F$ schreiben. Im Prinzip kann man eine $\mathcal{L}$-Interpretation also als Funktion auf $\mathcal{L}$-Termen und auf $\mathcal{L}$-Formeln ansehen. Die Definitionsprinzipien der strukturellen Rekursion für Terme und Formeln der PL1S stellen sicher, dass die Funktionswerte stets eindeutig bestimmt sind.

Definition 3.5.8 (semantische Begriffe [PL1S]). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe. M bezeichne eine $\mathcal{L}$-Interpretation, F und G bezeichne $\mathcal{L}$-Formeln, S bezeichne eine Menge von $\mathcal{L}$-Formeln.

- M *erfüllt* F (oder M ist ein *Modell* von F) gdw. $M \models F$.
  Andernfalls: M *falsifiziert* F, notiert $M \not\models F$.

  M *erfüllt* S (oder M ist *Modell* von S), notiert $M \models S$, gdw. $M \models F$ für jedes $F \in S$.
  Andernfalls: M *falsifiziert* S, notiert $M \not\models S$.

- Eine $\mathcal{L}$-Formel (oder Menge von $\mathcal{L}$-Formeln) heißt

  allgemeingültig gdw. sie von jeder $\mathcal{L}$-Interpretation erfüllt wird
  erfüllbar gdw. sie von einer $\mathcal{L}$-Interpretation erfüllt wird
  falsifizierbar gdw. sie von einer $\mathcal{L}$-Interpretation falsifiziert wird
  unerfüllbar gdw. sie von jeder $\mathcal{L}$-Interpretation falsifiziert wird

  Eine allgemeingültige Formel nennt man auch Tautologie, eine unerfüllbare Formel(menge) auch widersprüchlich, eine erfüllbare Formel(menge) auch widerspruchsfrei.

- Aus F *folgt* G, notiert $F \models G$, gdw. für jede $\mathcal{L}$-Interpretation M gilt:
  wenn $M \models F$ dann $M \models G$ (jedes Modell von F ist auch Modell von G).

  F, G sind *logisch äquivalent*, notiert $F \models\!\!\!\!\dashv G$, gdw. $F \models G$ und $G \models F$
  (F und G haben die selben Modelle).

- Die Folgerungsbeziehung $F \models G$ und die logische Äquivalenz $F \models\!\!\!\!\dashv G$ sind genauso definiert, wenn F oder G oder sowohl F als auch G eine Formelmenge ist. ■

Die Definition dieser semantischen Begriffe ist genau gleich wie für die Aussagenlogik (Definition 3.3.4). Die Bemerkungen, dass das Zeichen $\models$ überladen ist und dass eine Menge S nur im endlichen Fall durch die Konjunktion ihrer Elemente ersetzt werden kann, gelten hier ganz genau so. Ferner gilt wie für die Aussagenlogik der Satz:

Satz 3.5.9. Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe, seien F und G $\mathcal{L}$-Formeln.

1. Ist F allgemeingültig, so ist $\neg F$ unerfüllbar.
2. Ist F unerfüllbar, so ist $\neg F$ allgemeingültig.
3. Ist F erfüllbar und falsifizierbar, so ist $\neg F$ erfüllbar und falsifizierbar.
4. $F \models G$ gdw. $\neg G \models \neg F$
   gdw. $(F \Rightarrow G)$ allgemeingültig ist
   gdw. $(F \wedge \neg G)$ unerfüllbar ist.
5. $F \models\!\!\!\!\dashv G$ gdw. $\neg F \models\!\!\!\!\dashv \neg G$
   gdw. $(F \Leftrightarrow G)$ allgemeingültig ist
   gdw. für jede $\mathcal{L}$-Interpretation M gilt: $M \models F$ gdw. $M \models G$. ■

Die folgenden Sätze haben dagegen keine Entsprechung in der Aussagenlogik, da sie sich auf die Struktur von Termen beziehen.

Satz 3.5.10 (Koinzidenzsatz). Seien $M_1 = (D, Ab, U_1)$ und $M_2 = (D, Ab, U_2)$ $\mathcal{L}$-Interpretationen mit dem selben Universum D und der selben Funktion Ab.
Sei t ein $\mathcal{L}$-Term und F eine $\mathcal{L}$-Formel

1. Gilt $U_1(x) = U_2(x)$ für alle $x \in \mathit{var}(t)$ so ist $t^{M_1} = t^{M_2}$.
2. Gilt $U_1(x) = U_2(x)$ für alle $x \in \mathit{fvar}(F)$ so gilt $M_1 \models F$ gdw. $M_2 \models F$.

Beweis: Strukturelle Induktion über den Term- bzw. Formelaufbau.

1. *Basisfall* Variable x: nach Voraussetzg $x^{M_1} = U_1(x) = U_2(x) = x^{M_2}$.

   Basisfall Konstante c: nach Definition $c^{M_1} = Ab(c) = c^{M_2}$.

   Induktionsfall $f(t_1, \ldots, t_n)$: Es gelte $t_1^{M_1} = t_1^{M_2}, \ldots, t_n^{M_1} = t_n^{M_2}$, dann gilt

$$\begin{array}{rcll} f(t_1,\ldots,t_n)^{M_1} & = & Ab(f)(t_1^{M_1},\ldots,t_n^{M_1}) & \text{nach Definition} \\ & = & Ab(f)(t_1^{M_2},\ldots,t_n^{M_2}) & \text{nach Indukt.annahme} \\ & = & f(t_1,\ldots,t_n)^{M_2} & \text{nach Definition} \end{array}$$

2. Ähnlich wie 1. ■

Geschlossene Formeln enthalten nach Definition keine freien Variablen. Dafür bedeutet der Koinzidenzsatz: Ist eine $\mathcal{L}$-Interpretation $M = (D, Ab, U)$ ein Modell einer geschlossenen Formel F, so ist für jede beliebige D-Umgebung U' die $\mathcal{L}$-Interpretation $M = (D, Ab, U')$ ebenfalls ein Modell von F.

Die Umgebung dient also nur dazu, nichtgeschlossene Formeln zu interpretieren, wie etwa die nichtgeschlossene Teilformel G von $\forall x\, G$ und $\exists x\, G$ in Definition 3.5.7. Für geschlossene Formeln ist die Umgebung irrelevant.

Variablenersetzungen und gebundene Variablenumbenennung. Es ist häufig erforderlich, einen Term s für eine Variable x in einem anderen Term t oder in einer Formel F einzusetzen.

In einen Term t einzusetzen, ist unproblematisch: alle Vorkommen der Variablen x im Term t werden durch den Term s ersetzt. Für Formeln sind dagegen einige Besonderheiten zu beachten, die am besten an Beispielen illustriert werden.

	F	s	s eingesetzt für x in F
1.	$(\exists y[p(x) \wedge \neg p(f(y))] \wedge \forall x\, q(x))$	$f(a)$	$(\exists y[p(f(a)) \wedge \neg p(f(y))] \wedge \forall x\, q(x))$
2.	$\sum_{y=1}^{2}(x+y) \neq 0$ für jedes $x \in \mathbb{Z}$ denn $(x+1)+(x+2) = 2x+3$	$-y$	$\sum_{y=1}^{2}(-y+y) \neq 0$ ist aber falsch denn $(-1+1)+(-2+2) = 0$
2′.			$\sum_{y'=1}^{2}(-y+y') \neq 0$ für jedes $y \in \mathbb{Z}$ denn $(-y+1)+(-y+2) = -2y+3$
3.	$\exists y[p(x) \wedge \neg p(f(y))]$	$f(y)$	$\exists y[p(f(y)) \wedge \neg p(f(y))]$
3′.			$\exists y'[p(f(y)) \wedge \neg p(f(y'))]$

Beispiel 1 zeigt, dass nur die freien Vorkommen von x durch s ersetzt werden. Es wäre nicht sinnvoll, das durch den Allquantor gebundene Vorkommen von x zu ersetzen.

Beispiel 2 ist kein Beispiel der Prädikatenlogik, aber es zeigt ein Phänomen, das im Zusammenhang mit gebundenen Variablen immer auftreten kann. In dem Summenausdruck ist die Laufariable y durch das Summenzeichen gebunden. Ersetzt man das freie x im Summenausdruck durch einen Term, der selbst ein Vorkommen von y enthält, wird dieses durch das Summenzeichen „eingefangen“. Dadurch kann die vorher richtige Ungleichung falsch werden.

Statt der naiven Ersetzung nach 2 kann man aber ausnutzen, dass die Summe nicht davon abhängt, wie ihre Laufvariable heißt. Wenn man diese vor der Ersetzung zunächst in y' umbenennt, wird der Wert der Summe nicht beeinflusst, aber der Konflikt mit den Variablen von s beseitigt. Die Ersetzung nach $2'$ ergibt das richtige Ergebnis.

Beispiel 3 zeigt das Phänomen des „Einfangens“ in der Prädikatenlogik. Die Formel F ist erfüllbar. Wird das freie x wie in Variante 3 durch $f(y)$ ersetzt, entsteht eine unerfüllbare Formel. Die Ersetzung nach $3'$ vermeidet das Problem durch vorherige Umbenennung der gebundenen Variablen.

Für die Bedingung, dass keine Variablen „eingefangen“ werden, ist folgende Bezeichnung üblich:

Definition 3.5.11. Sei s ein $\mathcal{L}$-Term, F eine $\mathcal{L}$-Formel und x eine Variable. Der Term s heißt *frei für x in F*, wenn kein freies Vorkommen von x in F im Bereich eines Quantors für eine Variable steht, die in s vorkommt (vergleiche Definition 2.3.6). ■

Zur Vereinfachung werden wir von jetzt an Formeln identifizieren, die sich nur durch Umbenennung von gebundenen Variablen unterscheiden. Semantisch ist das gerechtfertigt, weil in Definition 3.5.7 die Modellbeziehung nicht davon abhängt, wie die gebundenen Variablen heißen. Der Nutzen ist, dass wir immer erreichen können, dass ein gegebener Term frei für eine Variable in einer Formel ist, indem wir stillschweigend zu einer Variante der Formel übergehen, in der die gebundenen Variablen geeignet umbenannt sind.

Definition 3.5.12 (elementare Substitution). Seien s und t $\mathcal{L}$-Terme, F eine $\mathcal{L}$-Formel, x eine Variable. Dann heißt $[s/x]$ eine *elementare Substitution*. Sie wird wie folgt angewandt:

- $t[s/x]$ ist der Term, der entsteht, wenn man *alle* Vorkommen von x in t durch s ersetzt.
- $F[s/x]$ ist die Formel, die entsteht, wenn man *alle* freien Vorkommen von x in F durch s ersetzt, wobei o.B.d.A. s frei für x in F ist. ∎

Bemerkung: Im Ersetzungssatz 3.3.6 wurde die Notation $F\langle G_2/V_{G_1}\rangle$ für die Ersetzung *eines* Vorkommens der Formel G_1 in F durch G_2 benutzt. Im Gegensatz dazu bezeichnet $F[s/x]$ die Ersetzung *aller* freien Vorkommen der Variablen x in F durch s. Die unterschiedlichen Notationen $\langle\ldots\rangle$ und $[\ldots]$ sollen diesen Unterschied hervorheben. ∎

Die Notation $F[s/x]$ erinnert an $M[d/x]$ (Notation 3.5.6 für Interpretationen und Umgebungen). Aber s ist ein Term, also etwas syntaktisches, und d ein Objekt des Universums von M, also etwas semantisches. Der folgende Satz stellt den Zusammenhang her.

Satz 3.5.13 (Substitutionssatz). Sei $M = (D, Ab, U)$ eine $\mathcal{L}$-Interpretation, seien s und t $\mathcal{L}$-Terme, F eine $\mathcal{L}$-Formel, x eine Variable.

1. $t[s/x]^M = t^{M[s^M/x]}$
2. $M \models F[s/x]$ gdw. $M[s^M/x] \models F$ $\qquad$ o.B.d.A. s frei für x in F

Beweis: Strukturelle Induktion über den Term- bzw. Formelaufbau. ∎

Der Substitutionssatz erlaubt eine Art Übersetzung zwischen Syntax und Semantik: der syntaktische Term s entspricht semantisch einem Objekt s^M des Universums. Es ist egal, ob man einer Variablen dieses Objekt semantisch über die Umgebung zuordnet (mittels $M[s^M/x]$) oder die Variable syntaktisch durch den Term ersetzt (mittels $t[s/x]$ bzw. $F[s/x]$).

Satz 3.5.14. Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe.

1. Der Ersetzungssatz 3.3.6 gilt auch für $\mathcal{L}$-Formeln.
2. Die Regeln von Satz 3.3.7 gelten auch für $\mathcal{L}$-Formeln.
3. Regeln der überflüssigen Quantoren: falls $x \notin \mathit{fvar}(F)$, gilt
   $\forall xF \models\!\!\!\!\;\!\!=\!\!| \; F$ und $\exists xF \models\!\!\!\!\;\!\!=\!\!| \; F$.
4. Variablenumbenennungsregeln: falls x' frei für x in F ist, gilt
   $\forall xF \models\!\!\!\!\;\!\!=\!\!| \; \forall x'F[x'/x]$ und $\exists xF \models\!\!\!\!\;\!\!=\!\!| \; \exists x'F[x'/x]$.
5. Quantorvertauschungsregeln:
   $\forall x\forall yF \models\!\!\!\!\;\!\!=\!\!| \; \forall y\forall xF$ und $\exists x\exists yF \models\!\!\!\!\;\!\!=\!\!| \; \exists y\exists xF$.

   Achtung: im allgemeinen aber $\forall x\exists yF \not\models\!\!\!\!\;\!\!=\!\!| \; \exists y\forall xF$
6. Regeln der negierten Quantoren:
   $\neg\forall xF \models\!\!\!\!\;\!\!=\!\!| \; \exists x\neg F$ und $\neg\exists xF \models\!\!\!\!\;\!\!=\!\!| \; \forall x\neg F$.
7. Quantor/Junktor-Verträglichkeitsregeln:
   $\forall x(F \wedge G) \models\!\!\!\!\;\!\!=\!\!| \; ((\forall xF) \wedge (\forall xG))$ und $\exists x(F \vee G) \models\!\!\!\!\;\!\!=\!\!| \; ((\exists xF) \vee (\exists xG))$
8. Variablenverschmelzungsregeln:
   $\forall x\forall yF \models \forall xF[x/y]$ und $\exists x\exists yF \;=\!\!| \; \exists xF[x/y]$

   Achtung: im allgemeinen gilt $\models\!\!\!\!\;\!\!=\!\!|$ hier nicht. ∎

Ganz analog wie für die Aussagenlogik gilt auch für die Prädikatenlogik erster Stufe ein Endlichkeits-/Kompaktheitssatz.

Satz 3.5.15 (Endlichkeits- oder Kompaktheitssatz [PL1S]). Sei S eine unendliche Menge von geschlossenen Formeln einer Sprache $\mathcal{L}$ der Prädikatenlogik erster Stufe. Ist jede endliche Teilmenge von S erfüllbar, so ist S erfüllbar. ∎

Sein Beweis benötigt aber Hilfsmittel, die an dieser Stelle noch nicht zur Verfügung stehen. Er wird erst in Kapitel 4 (Abschnitt 4.9) behandelt.

3.6 Gleichheit

Wir haben in Kapitel 2 ein spezielles zweistelliges Gleichheitsrelationssymbol $\doteq$ eingeführt. Wie kann man aber erreichen, dass dieses Relationssymbol gerade mit der Gleichheitsrelation und nicht mit einer anderen zweistelligen Relation auf dem Universum interpretiert wird?

Die Eigenschaften eines Relationssymbols werden normalerweise dadurch festgelegt, dass man sie mittels Formeln axiomatisiert. Die Interpretationen, die Modelle dieser Axiome sind, ordnen dem Relationssymbol dann eine Relation mit den axiomatisierten Eigenschaften zu. Im Falle der Gleichheit ist diese Vorgehensweise aber aus verschiedenen Gründen nicht zufriedenstellend:

1. Man kann mit Formeln der PL1S nicht ausschließen, dass Modelle dieser Formeln zwei verschiedene Elemente des Universums als „gleich“ interpretieren.
2. Der Begriff Gleichheit ist in sehr vielen Anwendungen der PL1S zu finden. Die Gleichheit hat also eine besondere Stellung, die eine besondere Behandlung rechtfertigt.

Der folgende Satz liefert die Begründung für die erste Behauptung:

Satz 3.6.1 (Modellerweiterungssatz). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe. Sei $M = (D, Ab, U)$ eine $\mathcal{L}$-Interpretation, sei $d' \notin D$ und $D' = D \cup \{d'\}$.
Dann gibt es eine $\mathcal{L}$-Interpretation M' mit Universum D', so dass für jede $\mathcal{L}$-Formel F gilt: $M \models F$ gdw. $M' \models F$.

Beweis: Sei $d \in D$ beliebig, aber fest gewählt. Wir konstruieren Ab' aus Ab, so dass es das neue Element d' als „unscheinbaren Zwilling“ von d behandelt: „Zwilling“, weil d' als Argument von Funktionen und Relationen genauso wirkt als stünde d an seiner Stelle; „unscheinbar“, weil die den Funktionssymbolen zugeordneten Funktionen die gleichen Ergebnisse liefern wie mit Ab, also nie d'. Zur Vereinfachung der Notation dient eine Funktion π, die d' auf seinen „Zwilling“ d abbildet und alle anderen Elemente auf sich selbst. Seien also:

$$\begin{array}{lll}
\pi : D' \to D & \pi(a) := & \begin{cases} d & \text{falls } a = d' \\ a & \text{falls } a \neq d' \end{cases} \\[2ex]
Ab'(p) \subseteq D'^{\,0} & Ab'(p) := & Ab(p) \text{ für jedes 0-stellige Relationssymbol } p \\
Ab'(p) \subseteq D'^{\,n} & Ab'(p) := & \{\, (d_1, \ldots, d_n) \in D'^{\,n} \mid (\, \pi(d_1), \ldots, \pi(d_n)\,) \in Ab(p) \,\} \\
 & & \text{für jedes } n\text{-stellige } (n \geq 1) \text{ Relationssymbol } p \\[2ex]
Ab'(c) \in D' & Ab'(c) := & Ab(c) \text{ für jedes 0-stellige Funktionssymbol } c \\
Ab'(f) : D'^{\,n} \to D' & Ab'(f)(d_1, \ldots, d_n) := & Ab(f)(\, \pi(d_1), \ldots, \pi(d_n)\,) \\
 & & \text{für jedes } n\text{-stellige } (n \geq 1) \text{ Funktionssymbol } f
\end{array}$$

Damit gilt:

1. Für jede D'-Umgebung U' ist $\pi \circ U'$ eine D-Umgebung
   mit $(\pi \circ U')(x) = \pi(U'(x)) = \begin{cases} d & \text{falls } U'(x) = d' \\ U'(x) & \text{falls } U'(x) \neq d' \end{cases}$

2. Für jede Konstante c ist $\pi(Ab'(c)) = Ab(c)$. Für jedes n-stellige Funktionssymbol f ist $\pi(\, Ab'(f)(d_1, \ldots, d_n)\,) = Ab(f)(\,\pi(d_1), \ldots, \pi(d_n)\,)$

   Das gilt weil $Ab(c) \in D$ und $Ab(f)(\ldots) \in D$ ist, also $\neq d'$.

3. Für jeden $\mathcal{L}$-Term t und jede D'-Umgebung U' ist $\pi(t^{(D',Ab',U')}) = t^{(D,Ab,\,\pi\circ U')}$

   Durch strukturelle Induktion.

   Basisfall Variable x: $\pi(x^{(D',Ab',U')}) = \pi(U'(x)) = (\pi \circ U')(x) = x^{(D,Ab,\,\pi\circ U')}$

   Basisfall Konstante c: $\pi(c^{(D',Ab',U')}) = \pi(Ab'(c)) \overset{2.}{=} Ab(c) = c^{(D,Ab,\,\pi\circ U')}$

   Induktionsfall $f(t_1, \ldots, t_n)$: Es gelte $\pi(t_i^{(D',Ab',U')}) = t_i^{(D,Ab,\,\pi\circ U')}$, dann gilt

   $$\begin{array}{lll} \multicolumn{2}{l}{\pi(\, f(t_1, \ldots, t_n)^{(D',Ab',U')}\,)} & \\ = & \pi(\, Ab'(f)(t_1^{(D',Ab',U')}, \ldots, t_n^{(D',Ab',U')})\,) & \text{nach Definition} \\ = & Ab(f)(\,\pi(t_1^{(D',Ab',U')}), \ldots, \pi(t_n^{(D',Ab',U')})\,) & \text{nach 2.} \\ = & Ab(f)(\, t_1^{(D,Ab,\,\pi\circ U')}, \ldots, t_n^{(D,Ab,\,\pi\circ U')}\,) & \text{nach Indukt.annahme} \\ = & f(t_1, \ldots, t_n)^{(D,Ab,\,\pi\circ U')} & \text{nach Definition} \end{array}$$

4. Für alle $a \in D'$ gilt $\pi \circ (U'[a/x]) = (\pi \circ U')[\pi(a)/x]$

   Durch ausrechnen für Variable x und Variable $y \neq x$:
   $(\pi \circ (U'[a/x]))(x) = \pi(U'[a/x](x)) = \pi(a) = (\pi \circ U')[\pi(a)/x](x)$
   $(\pi \circ (U'[a/x]))(y) = \pi(U'[a/x](y)) = \pi(U'(y)) = (\pi \circ U')(y) = (\pi \circ U')[\pi(a)/x](y)$

5. Für jede $\mathcal{L}$-Formel F und jede D'-Umgebung U' gilt
   $(D', Ab', U') \models F$ gdw. $(D, Ab,\, \pi \circ U') \models F$

   Durch strukturelle Induktion.

   Basisfall $\top$ oder $\bot$: ergibt sich unmittelbar aus der Definition von $\models$.

   Basisfall nullstelliges Relationssymbol p: gilt wegen der Definition $Ab'(p) = Ab(p)$.

   Basisfall atomare Formel $p(t_1, \ldots, t_n)$:

   $$\begin{array}{lll} \multicolumn{2}{l}{(D', Ab', U') \models p(t_1, \ldots, t_n)} & \\ \text{gdw.} & (t_1^{(D',Ab',U')}, \ldots, t_n^{(D',Ab',U')}) \in Ab'(p) & \text{nach Definition } \models \\ \text{gdw.} & (\,\pi(t_1^{(D',Ab',U')}), \ldots, \pi(t_n^{(D',Ab',U')})\,) \in Ab(p) & \text{nach Definition } Ab' \\ \text{gdw.} & (\, t_1^{(D,Ab,\,\pi\circ U')}, \ldots, t_n^{(D,Ab,\,\pi\circ U')}\,) \in Ab(p) & \text{nach 3.} \\ \text{gdw.} & (D, Ab,\, \pi \circ U') \models p(t_1, \ldots, t_n) & \text{nach Definition } \models \end{array}$$

   Induktionsfall Junktor: Anwendung der Definition von $\models$ auf die Induktionsannahme.

Induktionsfall $\forall x\, G$: Für alle U' gelte $(D', Ab', U') \models G$ gdw. $(D, Ab, \pi \circ U') \models G$, dann gilt

$$\begin{array}{lll} \multicolumn{2}{l}{(D', Ab', U') \models \forall x\, G} & \\ \text{gdw.} & \text{für alle } a \in D' \text{ gilt } (D', Ab', U'[a/x]) \models G & \text{nach Definition } \models \\ \text{gdw.} & \text{für alle } a \in D' \text{ gilt } (D, Ab, \pi \circ (U'[a/x])) \models G & \text{nach Indukt.annahme} \\ \text{gdw.} & \text{für alle } a \in D' \text{ gilt } (D, Ab, (\pi \circ U')[\pi(a)/x]) \models G & \text{nach 4.} \\ \text{gdw.} & \text{für alle } b \in D \text{ gilt } (D, Ab, (\pi \circ U')[b/x]) \models G & \pi \text{ surjektiv, } \pi(d') \in D \\ \text{gdw.} & (D, Ab, \pi \circ U') \models \forall x\, G & \text{nach Definition } \models \end{array}$$

Induktionsfall $\exists x\, G$: wie im Fall $\forall x\, G$, mit „es gibt“ statt „für alle“.

6. Für die gegebene $\mathcal{L}$-Interpretation $M = (D, Ab, U)$ sei $M' = (D', Ab', U)$. Die D-Umgebung U ist auch eine D'-Umgebung, weil $D \subseteq D'$ ist. Da sie das Element d' nicht enthält, ist obendrein $\pi \circ U = U$. Damit gilt für jede $\mathcal{L}$-Formel F:

   $M \models F$ gdw. $(D, Ab, U) \models F$ gdw. $(D, Ab, \pi \circ U) \models F$ $\overset{5.}{\text{gdw.}}$ $(D', Ab', U) \models F$ gdw. $M' \models F$ ∎

Die Bedeutung des Modellerweiterungssatzes liegt zunächst darin, dass er eine eindeutige Axiomatisierung der Gleichheitsrelation durch eine Menge von $\mathcal{L}$-Formeln ausschließt.

Angenommen, es gäbe eine solche Menge, die erzwingt, dass ihre Modelle das Relationssymbol $\doteq$ mit der Gleichheitsrelation auf dem Universum interpretieren. Dann fügen wir die Formel $\forall x \forall y\; x \doteq y$ hinzu. Wenn $\doteq$ wirklich mit der Gleichheitsrelation interpretiert würde, müsste jedes Modell der erweiterten Formelmenge ein einelementiges Universum haben. Nach dem Modellerweiterungssatz gilt aber: wenn es ein Modell mit einem einelementigen Universum gibt, dann gibt es auch ein Modell mit einem zweielementigen (und folglich auch mit noch größerem) Universum, dessen Elemente alle zueinander in der Relation stehen, mit der $\doteq$ interpretiert wird. Also gibt es keine Formelmenge, die erzwingt, dass ihre Modelle das Gleichheitsrelationssymbol $\doteq$ nur mit der Gleichheitsrelation interpretieren können.

Eine weitere Folge des Modellerweiterungssatzes ist, dass keine Formelmenge erzwingen kann, dass die Universen ihrer Modelle eine Höchstzahl von Elementen haben. Man kann also für keine natürliche Zahl n ausdrücken, dass es „höchstens n“ Objekte geben soll. Dagegen ist es bemerkenswerterweise möglich, mit $\mathcal{L}$-Formeln zu erzwingen, dass es „mindestens n“ Objekte gibt. Ebenso ist ausdrückbar, dass es unendlich viele Objekte geben soll.

Mit der im folgenden vorgestellten in die Logik „eingebauten“ Gleichheitsrelation lässt sich immerhin erreichen, dass für jede vorgegebene natürliche Zahl n „höchstens n“ ausdrückbar wird. Allerdings ermöglicht auch die PL1S mit eingebauter Gleichheit nicht, „höchstens endlich viele“ auszudrücken, ohne eine konkrete Zahl als Obergrenze vorzugeben (wir werden in Abschnitt 4.9 auf dieses Thema zurückkommen). Diese Nichtausdrückbarkeit einer Obergrenze setzt sich übrigens im Unendlichen fort. Nach dem sogenannten „aufsteigenden Satz von Löwenheim und Skolem“ gilt für jede Formelmenge der PL1S mit eingebauter Gleichheit, die ein Modell mit unendlichem Universum hat: für jede beliebige Menge hat die Formelmenge auch ein Modell, dessen Universum mindestens so groß ist wie diese Menge.

Doch nun zur angekündigten eingebauten Gleichheitsrelation.

Definition 3.6.2 (normale Interpretation). Eine $\mathcal{L}$-Interpretation $M = (D, Ab, U)$ heißt *normal*, wenn $Ab(\doteq)$ die Gleichheitsrelation auf D ist. Ist eine normale Interpretation M ein

Modell einer Formel F, schreibt man $M \models_= F$. Die Folgerungsbeziehung bezüglich normaler Modelle wird ebenfalls $\models_=$ notiert. ■

Damit ist die Semantik von Sprachen der PL1S mit Gleichheit festgelegt. Zwar ist die Gleichheitsrelation durch Formeln der PL1S nicht eindeutig axiomatisierbar, doch bleibt noch zu untersuchen, wie weit man mit einer solchen Axiomatisierung genau kommt.

Definition 3.6.3 (Gleichheitsaxiome). Das Gleichheitsaxiomensystem für eine Sprache $\mathcal{L}$ der PL1S ist

$$\begin{aligned} GAS_{\mathcal{L}} := \{\mathit{Refl}\} &\cup \{Sub_f \mid f \ n\text{-stelliges Funktionssymbol in } \mathcal{L}, n \geq 1\} \\ &\cup \{Sub_p \mid p \ n\text{-stelliges Relationssymbol in } \mathcal{L}, n \geq 1\} \end{aligned}$$

Reflexivitätsaxiom:

$$\mathit{Refl} := \forall x\, x \doteq x$$

Verträglichkeitsaxiom für ein n-stelliges ($n \geq 1$) Funktionssymbol:

$$Sub_f := \forall x_1 \ldots \forall x_n \forall x_1' \ldots \forall x_n'((x_1 \doteq x_1' \land \ldots \land x_n \doteq x_n') \Rightarrow f(x_1, \ldots, x_n) \doteq f(x_1', \ldots, x_n'))$$

Verträglichkeitsaxiom für ein n-stelliges ($n \geq 1$) Relationssymbol:

$$Sub_p := \forall x_1 \ldots \forall x_n \forall x_1' \ldots \forall x_n'((x_1 \doteq x_1' \land \ldots \land x_n \doteq x_n') \Rightarrow (p(x_1, \ldots, x_n) \Rightarrow p(x_1', \ldots, x_n')))$$

■

Bemerkungen:

1. Die Verträglichkeitsaxiome drücken das Leibniz'sche Prinzip „identitas indiscernibilium", also der Identität der ununterscheidbaren Dinge aus: das Gleiche darf durch das Gleiche in einer Aussage ersetzt werden, ohne die Gültigkeit der Aussage zu verändern.
2. In der Regel ist $GAS_{\mathcal{L}}$ unendlich. (Warum?) ■

Mit $GAS_{\mathcal{L}}$ wird das Prinzip „identitas indiscernibilium" für die einfachsten Terme und Formeln sichergestellt. Der folgende Satz zeigt, dass es auch für komplexere Terme und Formeln, also für die PL1S allgemein, gilt.

Satz 3.6.4.

1. Sei t ein $\mathcal{L}$-Term und $var(t) = \{x_1, \ldots, x_n\}$. Seien $x_1', \ldots, x_m'$ mit $1 \leq m \leq n$ paarweise verschiedene Variablen mit $\{x_1, \ldots, x_n\} \cap \{x_1', \ldots, x_m'\} = \emptyset$. Sei $t' = t[x_1'/x_1] \ldots [x_m'/x_m]$. Dann gilt:
$$\begin{aligned} &\{\mathit{Refl}\} \cup \{Sub_f \mid f \ n\text{-stelliges Funktionssymbol in } \mathcal{L}, n \geq 1\} \\ &\quad \models \ \forall x_1 \ldots \forall x_n \forall x_1' \ldots \forall x_m'(x_1 \doteq x_1' \land \ldots \land x_m \doteq x_m' \Rightarrow t \doteq t') \end{aligned}$$
2. Sei F eine $\mathcal{L}$-Formel, $fvar(F) = \{x_1, \ldots, x_n\}$. Seien $x_1', \ldots, x_m'$ mit $1 \leq m \leq n$ paarweise verschiedene Variablen mit $\{x_1, \ldots, x_n\} \cap \{x_1', \ldots, x_m'\} = \emptyset$ und x_i' frei für x_i in F. Sei $F' = F[x_1'/x_1] \ldots [x_m'/x_m]$. Dann gilt:
$$GAS_{\mathcal{L}} \models \forall x_1 \ldots \forall x_n \forall x_1' \ldots \forall x_m'(x_1 \doteq x_1' \land \ldots \land x_m \doteq x_m' \Rightarrow (F \Rightarrow F'))$$

Beweis: Strukturelle Induktion über den Aufbau von Termen und Formeln. ■

Die Gleichheitsaxiome werden häufig so formuliert, dass auch ein Symmetrieaxiom und ein Transitivitätsaxiom enthalten sind. Diese folgen aus den Gleichheitsaxiomen in der obigen Formulierung.

Satz 3.6.5.

1. $GAS_{\mathcal{L}} \models Sym$ wobei $Sym := \forall x \forall y\, (x \doteq y \Rightarrow y \doteq x)$
2. $GAS_{\mathcal{L}} \models Trans$ wobei $Trans := \forall x \forall y \forall z\, (x \doteq y \wedge y \doteq z \Rightarrow x \doteq z)$

Beweis:

1. $GAS_{\mathcal{L}}$

$$
\begin{array}{lll}
\models & \forall x \forall y \forall x' \forall y'\, (x \doteq x' \wedge y \doteq y' \Rightarrow (x \doteq y \Rightarrow x' \doteq y')) & \text{das ist } Sub_{\doteq} \\
\models & \forall y' \forall x'\, (y' \doteq x' \wedge y' \doteq y' \Rightarrow (y' \doteq y' \Rightarrow x' \doteq y')) & \text{Verschmelzung von } x, y, y' \text{ zu } y' \\
\models\!\!\!\!\dashv & \forall y' \forall x'\, (y' \doteq y' \wedge y' \doteq y' \wedge y' \doteq x' \Rightarrow x' \doteq y') & \text{Satz 3.5.14, 3.3.7} \\
\models\!\!\!\!\dashv & \forall x \forall y\, (x \doteq x \wedge x \doteq x \wedge x \doteq y \Rightarrow y \doteq x) & \text{Umbenennung}
\end{array}
$$

   Mit $GAS_{\mathcal{L}} \models Refl$ gilt damit auch $GAS_{\mathcal{L}} \models \forall x \forall y\, (x \doteq y \Rightarrow y \doteq x)$.

2. Wir wenden Satz 3.6.4 (2) auf $F = y \doteq z$ an und erhalten

   $GAS_{\mathcal{L}}$

$$
\begin{array}{lll}
\models & \forall y \forall z \forall y'\, (y \doteq y' \Rightarrow (y \doteq z \Rightarrow y' \doteq z)) & \\
\models & \forall y \forall z \forall y'\, (y' \doteq y \wedge y \doteq z \Rightarrow y' \doteq z) & \text{da } GAS_{\mathcal{L}} \models Sym \\
\models\!\!\!\!\dashv & \forall y \forall z \forall x\, (x \doteq y \wedge y \doteq z \Rightarrow x \doteq z) & \text{Umbenennung}
\end{array}
$$

■

Man beachte, dass sowohl Satz 3.6.4 als auch Satz 3.6.5 keine Aussagen über normale Modelle von $GAS_{\mathcal{L}}$ sind, sondern über beliebige Modelle von $GAS_{\mathcal{L}}$, also auch nichtnormale. Auch die Folgerungsbeziehung ist jeweils $\models$ und nicht $\models_{=}$. Den Zusammenhang zwischen beliebigen Modellen von $GAS_{\mathcal{L}}$ und normalen Interpretationen stellt der folgende Satz her.

Satz 3.6.6.

1. Jede normale Interpretation ist ein Modell von $GAS_{\mathcal{L}}$.
2. Zu jedem Modell M von $GAS_{\mathcal{L}}$ gibt es ein normales Modell M' von $GAS_{\mathcal{L}}$, so dass für jede $\mathcal{L}$-Formel F gilt $M \models F$ gdw. $M' \models_{=} F$.
3. Für jede Formelmenge S und jede Formel F gilt
   $S \cup GAS_{\mathcal{L}} \models F$ gdw. $S \models_{=} F$.

Beweis: (skizziert)

1. Im wesentlichen durch Ausrechnen. Die Gleichheitsrelation besitzt die von $GAS_{\mathcal{L}}$ verlangten Eigenschaften.

2. Sei $M = (D, Ab, U)$ das Modell von $GAS_{\mathcal{L}}$. Wir schreiben $\sim$ für die zweistellige Relation $Ab(\doteq)$ auf dem Universum D.

   Da M die Formeln in $GAS_{\mathcal{L}}$ erfüllt, kann man leicht nachweisen, dass $\sim$ eine Äquivalenzrelation ist. Obendrein besitzt sie folgende Strukturverträglichkeitseigenschaften, ist also eine Kongruenzrelation:

   - Für jedes n-stellige Funktionssymbol f:
     wenn $d_1 \sim e_1, \ldots, d_n \sim e_n$, dann $Ab(f)(d_1, \ldots, d_n) \sim Ab(f)(e_1, \ldots, e_n)$.
   - Für jedes n-stellige Relationssymbol p:
     wenn $d_1 \sim e_1, \ldots, d_n \sim e_n$, dann $(d_1, \ldots, d_n) \in Ab(p)$ gdw. $(e_1, \ldots, e_n) \in Ab(p)$.

Für $d \in D$ bezeichne $[d]$ die Äquivalenzklasse bezüglich $\sim$, also $d \sim e$ gdw. $[d] = [e]$. Wir definieren:

$$
\begin{array}{lll}
D' & := \{[d] \mid d \in D\} & \\
Ab'(c) & := [Ab(c)] & \text{0-stelliges Funktionssymbol } c \\
Ab'(f)([d_1], \ldots, [d_n]) & := [Ab(f)(d_1, \ldots, d_n)] & n\text{-stelliges } (n \geq 1) \text{ Funktionssymbol } f \\
Ab'(p) & := Ab(p) & \text{0-stelliges Relationssymbol } p \\
Ab'(p) & := \{([d_1], \ldots, [d_n]) \mid (d_1, \ldots, d_n) \in Ab(p)\} & \\
 & & n\text{-stelliges } (n \geq 1) \text{ Relationssymbol } p \\
U'(x) & := [U(x)] & \\
M' & := (D', Ab', U') &
\end{array}
$$

Dass Ab' wohldefiniert ist, ergibt sich unmittelbar aus den obigen Kongruenzeigenschaften von $\sim$. Durch strukturelle Induktion zeigt man nun, dass für jede Formel F gilt $M \models F$ gdw. $M' \models F$, also, weil M' normal ist, $M \models F$ gdw. $M' \models_= F$

3. „$\longrightarrow$“: Wir nehmen an, es gilt $S \cup GAS_{\mathcal{L}} \models F$. Sei M' ein normales Modell von S. Nach 1. ist M' auch ein Modell von $GAS_{\mathcal{L}}$. Nach Annahme gilt dann auch $M' \models_= F$.

   „$\longleftarrow$“: Wir nehmen an, es gilt $S \models_= F$. Sei M ein Modell von $S \cup GAS_{\mathcal{L}}$. Nach 2. gibt es ein normales Modell M' von $S \cup GAS_{\mathcal{L}}$, das die selben Formeln erfüllt wie M. Nach Annahme gilt $M' \models F$. Also gilt auch $M \models F$. ∎

Dieses Ergebnis mildert die Bemerkungen im Anschluss an den Modellerweiterungssatz wieder ab. Zwar kann man mit $GAS_{\mathcal{L}}$ oder mit einer anderen Formelmenge nicht verhindern, dass das Gleichheitsrelationssymbol mit einer größeren Relation als der Gleichheit interpretiert wird. Aber in diesen größeren Modellen gilt auch nicht mehr und nicht weniger als in dem entsprechenden normalen Modell, und die Folgerungsbeziehung verändert sich dadurch auch nicht.

3.7 Exkurs: Natürlichsprachliche Interpretationen der Quantoren

In der natürlichen Sprache werden die Quantoren manchmal, aber nicht immer, anders interpretiert als in der (klassischen) Logik.

Wenn ein Prüfer sagt:

„Alle Studenten, die die Prüfung abgelegt haben, haben bestanden.“

erwartet man, dass mindestens ein Student die Prüfung abgelegt hat. In der klassischen Logik ist der Satz aber auch wahr, wenn gar kein Student zur Prüfung erschienen ist.

Wenn der Prüfer mitteilt:

„Einige Studenten haben die Prüfung bestanden.“

versteht man implizit, dass einige Unglückliche dabei waren, die nicht bestanden haben. Diese Konsequenz folgt aber nicht mit der klassisch-logischen Interpretation der Quantoren.

Andererseits kann die Negation der Aussage

„Alle Studenten haben die Prüfung bestanden.“

weder natürlichsprachlich noch klassisch-logisch heißen:

„Kein Student hat die Prüfung bestanden.“

sondern nur:

„Einige Studenten haben die Prüfung nicht bestanden.“

Hier stimmt also die natürlichsprachliche Interpretation mit der klassisch-logischen überein.

3.8 Herbrand-Interpretationen und Skolemisierung

Die Folgerungsbeziehung und andere semantische Begriffe sind mit den bisherigen Definitionen zwar formal präzise charakterisiert, aber auf eine Art, die keine Grundlage für eine algorithmische Behandlung bietet. Betrachten wir zum Beispiel die Frage, ob zwischen zwei $\mathcal{L}$-Formeln F und G die Folgerungsbeziehung besteht. Nach Definition besteht sie genau dann, wenn jede Interpretation, die F erfüllt, auch G erfüllt. Eine Interpretation wiederum basiert auf einer beliebigen nichtleeren Menge als Universum. Es gibt also mindestens so viele Interpretationen wie es Mengen gibt, und für jede davon wäre zu überprüfen, ob sie, wenn sie F erfüllt, auch G erfüllt. Darauf lässt sich kein Algorithmus aufbauen.

Damit liegt die Frage nahe, ob man gewisse „Referenzinterpretationen" auszeichnen kann, so dass man zum Nachweis der Folgerungsbeziehung die genannte Überprüfung nicht für jede überhaupt mögliche Interpretation durchführen muss, sondern nur für diese Referenzinterpretationen. Das wäre ein wichtiger Zwischenschritt in Richtung Algorithmisierung.

Tatsächlich haben wir im Zusammenhang mit der Gleichheit bereits etwas ähnliches kennengelernt. In Satz 3.6.6 wurde die Folgerungsbeziehung $\models$ ja auf die Beziehung $\models_=$ zurückgeführt. Wenn die Gleichheitsaxiome beteiligt sind, braucht man zum Nachweis der Folgerungsbeziehung die Überprüfung also nicht für sämtliche Interpretationen durchzuführen, sondern kann sich auf die normalen Interpretationen beschränken.

Natürlich nützt die Beschränkung auf normale Interpretationen nicht viel im Hinblick auf die Algorithmisierbarkeit, da ja auch die normalen Interpretationen jede beliebige nichtleere Menge als Universum benutzen können. Doch zeigt das Ergebnis immerhin, dass die Idee der Referenzinterpretationen nicht von vornherein aussichtslos ist. Die angestrebten Referenzinterpretationen sollten aber in erster Linie bewirken, dass weniger Universen betrachtet werden müssen.

Genau das ist für Herbrand-Interpretationen der Fall. Alle Herbrand-Interpretationen basieren auf einem einzigen ausgezeichneten Universum, dem sogenannten Herbrand-Universum.

Wenn die beteiligten Formeln eine bestimmte syntaktische Eigenschaft haben, nämlich universell sind, reicht es tatsächlich aus, zum Nachweis der Folgerungsbeziehung die Überprüfung nur für Herbrand-Interpretationen durchzuführen statt für sämtliche Interpretationen. Zudem können Formeln, die nicht universell sind, mit Hilfe der Skolemisierung in universelle Formeln transformiert werden, die in einem gewissen Sinn äquivalent zu den ursprünglichen Formeln sind. Insgesamt wird das angestrebte Ziel damit für alle Formeln erreicht.

Im folgenden werden wir zunächst universelle Formeln behandeln, danach Herbrand-Interpretationen und schließlich die Skolemisierung.

Pränexform und universelle Formeln

Eigenschaften von Formeln der Prädikatenlogik erster Stufe, insbesondere im Zusammenhang mit Herbrand-Interpretationen, hängen oft davon ab, welche Quantoren in den Formeln vorkommen und wo sie relativ zu Negationsjunktoren stehen. Sei zum Beispiel F die Formel $\neg(\exists x\, p(x) \vee \neg\forall y\, q(y))$. Es gilt:

$$\begin{array}{rcll} F & = & \neg(\exists x\, p(x) \vee \neg\forall y\, q(y)) & \\ & \models\!\!\!\dashv & (\neg\exists x\, p(x) \wedge \neg\neg\forall y\, q(y)) & \text{de Morgan'sche Regel (Satz 3.3.7)} \\ & \models\!\!\!\dashv & (\neg\exists x\, p(x) \wedge \forall y\, q(y)) & \text{Regel der doppelten Negation (Satz 3.3.7)} \\ & \models\!\!\!\dashv & (\forall x\, \neg p(x) \wedge \forall y\, q(y)) & \text{Regel der negierten Quantoren (Satz 3.5.14)} \end{array}$$

In F kommen zwar rein syntaktisch ein Existenzquantor und ein Allquantor vor, aber „eigentlich“ sind beide Variablen in F allquantifiziert.

Dieses „eigentlich“ wird durch die sogenannte Polarität formalisiert. Die Teilformel $\exists x\, p(x)$ hat negative Polarität in F, weil die Teilformel in F im Bereich einer Negation steht. Die Teilformel $\forall y\, q(y)$ hat dagegen positive Polarität in F, weil sie im Bereich von zwei Negationen steht, die einander ja aufheben. Existenzquantoren mit negativer Polarität in einer Formel wirken wie Allquantoren mit positiver Polarität.

Definition 3.8.1 (Polarität). Für eine Formel F sind die Polaritäten der Vorkommen von Teilformeln von F wie folgt rekursiv definiert.

Statt „G hat positive Polarität in F“ sagt man auch „G kommt positiv in F vor“ oder kürzer „G ist positiv in F“, und entsprechend für negativ.

1. F ist positiv in F.
2. Sei G ein Vorkommen einer Teilformel von F der Gestalt $\neg G_1$:
   Ist G positiv in F, so ist G_1 negativ in F.
   Ist G negativ in F, so ist G_1 positiv in F.
3. Sei G ein Vorkommen einer Teilformel von F der Gestalt $(G_1 \wedge G_2)$ oder $(G_1 \vee G_2)$:
   Ist G positiv in F, so sind G_1 und G_2 positiv in F.
   Ist G negativ in F, so sind G_1 und G_2 negativ in F.
4. Sei G ein Vorkommen einer Teilformel von F der Gestalt $(G_1 \Rightarrow G_2)$:
   Ist G positiv in F, so ist G_1 negativ in F und G_2 positiv in F.
   Ist G negativ in F, so ist G_1 positiv in F und G_2 negativ in F.
5. Sei G ein Vorkommen einer Teilformel von F der Gestalt $(G_1 \Leftrightarrow G_2)$:
   G_1 und G_2 sind sowohl positiv in F als auch negativ in F.
6. Sei G ein Vorkommen einer Teilformel von F der Gestalt $Qx\, G_1$ für einen Quantor Q:
   Ist G positiv in F, so ist G_1 positiv in F.
   Ist G negativ in F, so ist G_1 negativ in F.

   In diesem Fall sprechen wir auch von der Polarität des Vorkommens des Quantors Q und meinen damit die Polarität des Vorkommens G der Teilformel, die mit Q beginnt. ∎

Für Informatiker ist diese Definition am einfachsten dadurch zu veranschaulichen, dass sie sich die Formel F durch ihren Syntaxbaum dargestellt denken, also einen Baum, dessen Blätter die Vorkommen von atomaren Teilformeln von F sind und dessen innere Knoten mit Junktoren und Quantoren beschriftet sind. Die Wurzel dieses Baums wird wegen 1. als positiv markiert. Der Rest der Definition gibt an, wie das Vorzeichen auf alle Nachfolgerknoten bis hin zu den Blättern propagiert wird.

Bei dieser Propagation wechselt das Vorzeichen bei jedem Negationsjunktor. Sei zum Beispiel F die Formel $\neg\neg\neg p$, dann haben $\neg\neg\neg p$ und $\neg p$ positive Polarität in F, während $\neg\neg p$ und p negative Polarität in F haben. Die Polarität eines Vorkommens einer Teilformel drückt aus, ob dieses Vorkommen im Bereich einer geraden oder einer ungeraden Anzahl von Negationen steht, ob diese Negationen einander also aufheben oder nicht.

Das Vorzeichen wechselt außerdem auf der linken Seite des Implikationsjunktors $\Rightarrow$. Nach Satz 3.5.14 und Satz 3.3.7 gilt $(G_1 \Rightarrow G_2) \;\models\!\!\!\!\mid\;\; (\neg G_1 \vee G_2)$. Die linke Teilformel G_1 steht also „eigentlich“ im Bereich einer impliziten Negation, die bei der Definition der Polarität mit berücksichtigt wird.

Ist F eine Formel, in der der Äquivalenzjunktor $\Leftrightarrow$ nicht auftritt, so ist jedes Vorkommen einer Teilformel von F eindeutig als ein Vorkommen positiver oder negativer Polarität klassifiziert.

Dagegen haben alle Teilformeln, die im Syntaxbaum unterhalb eines Äquivalenzjunktors vorkommen, sowohl positive als auch negative Polarität. Die Begründung dafür liefert die Regel zur Auflösung des Äquivalenzjunktors: $(G_1 \Leftrightarrow G_2) \models\!\!\!\!\!\!\!\;\;\; ((\neg G_1 \vee G_2) \wedge (G_1 \vee \neg G_2))$ nach Satz 3.5.14 und Satz 3.3.7. Jede der beiden unmittelbaren Teilformeln von $(G_1 \Leftrightarrow G_2)$ kommt also „eigentlich“ einmal negiert und einmal unnegiert vor. Das gleiche gilt mit der zweiten Regel $(G_1 \Leftrightarrow G_2) \;\rightleftharpoons\; ((G_1 \wedge G_2) \vee (\neg G_1 \wedge \neg G_2))$.

Tritt eine Teilformel mehrfach in einer Formel auf, so können verschiedene Vorkommen der Teilformel unterschiedliche Polaritäten haben. Ist zum Beispiel F die Formel $(p \vee \neg p)$, hat das erste Vorkommen von p positive Polarität in F und das zweite negative.

Definition 3.8.2 (Pränexform). Eine Formel F ist in *Pränexform*, wenn sie die Gestalt $Q_1x_1 \ldots Q_nx_n\, G$ hat, $n \geq 0$, wo die Q_i Quantoren sind und in G kein Quantor vorkommt. ■

Satz 3.8.3. Sei F eine $\mathcal{L}$-Formel, in der kein Quantor in einer Teilformel vorkommt, die mit dem Äquivalenzjunktor $\Leftrightarrow$ gebildet ist. Seien $Q_1x_1 \ldots Q_nx_n$ die Vorkommen von Quantoren mit ihren Variablen in textueller Reihenfolge in F, und es gelte, dass die Variablen x_i paarweise verschieden sind und keine der Variablen x_i frei in F vorkommt.

Sei F' die Formel, die durch Streichen aller Vorkommen Q_ix_i aus F entsteht. Zu jedem der Q_i sei Q'_i der gleiche Quantor wie Q_i, falls Q_i positive Polarität in F hat, und der entgegengesetzte Quantor, falls Q_i negative Polarität in F hat.

Dann gilt $F \;\rightleftharpoons\; Q'_1x_1 \ldots Q'_nx_n\, F'$.

Beispiel: Für $F = \neg(\exists x_1\, p(x_1) \vee \neg\forall x_2\, q(x_2))$ ist $F' = \neg(p(x_1) \vee \neg q(x_2))$, und es gilt $F \;\rightleftharpoons\; \forall x_1 \forall x_2\, \neg(p(x_1) \vee \neg q(x_2))$.

Beweis: Durch vollständige Induktion über die Anzahl n der Vorkommen von Quantoren in F.

Induktionsbasis $n = 0$: Dann ist $F = F'$ und die Behauptung gilt.

Induktionsschritt $n \to n + 1$: Die Behauptung gelte für Formeln mit n Vorkommen von Quantoren. Seien $Q_1x_1Q_2x_2 \ldots Q_{n+1}x_{n+1}$ die Vorkommen von Quantoren mit ihren Variablen in textueller Reihenfolge in F. Davon ist Q_1x_1 also das textuell erste Vorkommen.

Für beliebige Formeln G und H, wo die Variable x_1 nicht in H vorkommt, gelten folgende logische Äquivalenzen:

$$
\begin{array}{lcl@{\qquad\qquad}lcl}
\neg\forall x_1 G & \rightleftharpoons & \exists x_1 \neg G & \neg\exists x_1 G & \rightleftharpoons & \forall x_1 \neg G \\[1ex]
(\forall x_1 G \wedge H) & \rightleftharpoons & \forall x_1 (G \wedge H) & (\exists x_1 G \wedge H) & \rightleftharpoons & \exists x_1 (G \wedge H) \\
(\forall x_1 G \vee H) & \rightleftharpoons & \forall x_1 (G \vee H) & (\exists x_1 G \vee H) & \rightleftharpoons & \exists x_1 (G \vee H) \\
(\forall x_1 G \Rightarrow H) & \rightleftharpoons & \exists x_1 (G \Rightarrow H) & (\exists x_1 G \Rightarrow H) & \rightleftharpoons & \forall x_1 (G \Rightarrow H) \\[1ex]
(H \wedge \forall x_1 G) & \rightleftharpoons & \forall x_1 (H \wedge G) & (H \wedge \exists x_1 G) & \rightleftharpoons & \exists x_1 (H \wedge G) \\
(H \vee \forall x_1 G) & \rightleftharpoons & \forall x_1 (H \vee G) & (H \vee \exists x_1 G) & \rightleftharpoons & \exists x_1 (H \vee G) \\
(H \Rightarrow \forall x_1 G) & \rightleftharpoons & \forall x_1 (H \Rightarrow G) & (H \Rightarrow \exists x_1 G) & \rightleftharpoons & \exists x_1 (H \Rightarrow G)
\end{array}
$$

Hat eine Teilformel von F eine der Gestalten auf der linken Seite von $\rightleftharpoons$ in dieser Liste, ist sichergestellt, dass die Variable nicht in der jeweiligen Teilformel H vorkommt, weil die quantifizierten Variablen von F paarweise verschieden sind und außerdem nicht frei in F vorkommen.

Wenn das erste Quantorvorkommen $Q_1 x_1$ nicht am Anfang von F steht, gibt es ein Vorkommen einer Teilformel von F, das eine der Gestalten auf der linken Seite von $\models\mid$ in dieser Liste hat (nach Voraussetzung kommt $Q_1 x_1$ nicht in einer mit $\Leftrightarrow$ gebildeten Teilformel vor). Aufgrund des Ersetzungssatzes (Satz 3.5.14 und Satz 3.3.6) können wir dieses Vorkommen durch die entsprechende Formel auf der rechten Seite von $\models\mid$ ersetzen und erhalten mit diesem Ersetzungsschritt eine zu F logisch äquivalente Formel.

Dieser Ersetzungsschritt verschiebt das erste Quantorvorkommen $Q_1 x_1$ nach links, so dass es weiterhin das erste bleibt, wobei sich möglicherweise der Quantor ändert. Ansonsten bleibt aber die Struktur der Formel unverändert.

Durch Wiederholung derartiger Ersetzungsschritte erhalten wir eine Formel $Q'_1 x_1\, F_1$ mit $F \models\mid Q'_1 x_1\, F_1$, wobei F_1 durch Streichen von $Q_1 x_1$ aus F entsteht.

Ein Vergleich der obigen Liste mit Definition 3.8.1 zeigt, dass sich beim Verschieben nach links der Quantor genau in den Fällen ändert, in denen sich auch die Polarität des Quantors ändert. Q'_1 hat positive Polarität in $Q'_1 x_1\, F_1$, also ist Q'_1 der gleiche Quantor wie Q_1, falls Q_1 positive Polarität in F hat, und der entgegengesetzte Quantor, falls Q_1 negative Polarität in F hat.

Nun ist F_1 eine Formel mit n Quantorvorkommen $Q_2 x_2 \ldots Q_{n+1} x_{n+1}$ in dieser textuellen Reihenfolge, die die gleichen Voraussetzungen erfüllt wie F. Nach Induktionsannahme gilt $F_1 \models\mid Q'_2 x_2 \ldots Q'_{n+1} x_{n+1}\, F'_1$ mit den geforderten Eigenschaften. Insbesondere ist $F'_1 = F'$. Mit $F \models\mid Q'_1 x_1\, F_1$ gilt $F \models\mid Q'_1 x_1 Q'_2 x_2 \ldots Q'_{n+1} x_{n+1}\, F'$ mit allen geforderten Eigenschaften. ∎

Wegen der Regel zur Auflösung des Äquivalenzjunktors (Satz 3.5.14 und Satz 3.3.7) ist jede geschlossene $\mathcal{L}$-Formel logisch äquivalent zu einer geschlossenen $\mathcal{L}$-Formel, in der der Äquivalenzjunktor $\Leftrightarrow$ nicht vorkommt. Mit den Variablenumbenennungsregeln (Satz 3.5.14) kann außerdem stets erreicht werden, dass die quantifizierten Variablen in einer Formel paarweise verschieden und verschieden von den Variablen sind, die frei in der Formel vorkommen. Insgesamt erhalten wir damit:

Folgesatz 3.8.4 (Pränexform). Jede geschlossene $\mathcal{L}$-Formel ist logisch äquivalent zu einer geschlossenen $\mathcal{L}$-Formel in Pränexform. Enthält die Formel keinen Quantor in einer mit $\Leftrightarrow$ gebildeten Teilformel, so ist ihre Pränexform bis auf Variablenumbenennung eindeutig bestimmt. ∎

Definition 3.8.5 (universell). Eine geschlossene $\mathcal{L}$-Formel heißt universell, wenn

1. keine Teilformel der Gestalt $\exists x\, G$ positiv in ihr vorkommt und
2. keine Teilformel der Gestalt $\forall x\, G$ negativ in ihr vorkommt. ∎

Bemerkung: In der Vorlesung „Theorie der Logikprogrammierung“ werden universelle Formeln in etwas anderer Weise definiert, weil die zugrundeliegende Syntax eine andere ist. ∎

Aus der Definition der Polarität ergibt sich unmittelbar, dass eine Formel, die einen Quantor in einer mit $\Leftrightarrow$ gebildeten Teilformel enthält, nicht universell sein kann.

Folgesatz 3.8.6. Eine geschlossene $\mathcal{L}$-Formel, die keinen Quantor in einer mit $\Leftrightarrow$ gebildeten Teilformel enthält, ist universell genau dann wenn ihre Pränexform keine Existenzquantoren enthält. ∎

Herbrand-Interpretationen

Wenn Herbrand-Interpretationen, wie in der Einleitung erläutert, auf einem ausgezeichneten Universum basieren, stellt sich die Frage, woraus so ein ausgezeichnetes Universum überhaupt bestehen kann. Da es stellvertretend für alle denkbaren Universen stehen soll, müsste es „Universalobjekte“ enthalten in dem Sinn, dass Objekte jeglicher anderer Universen damit repräsentiert werden können. Ob es so ein „Universaluniversum“ überhaupt gibt, erscheint auf den ersten Blick äußerst fraglich.

Bei näherer Betrachtung sind uns solche „Universalobjekte“ aber längst wohlvertraut. Wir benutzen doch ständig die syntaktischen Terme einer Sprache der Prädikatenlogik erster Stufe, um damit Objekte beliebiger Universen zu repräsentieren. Mit anderen Worten, die Menge aller Terme, oder vielleicht gewisser Terme, einer Sprache der Prädikatenlogik erster Stufe wäre ein naheliegender Kandidat für das angestrebte ausgezeichnete Universum.

Dieser Gedanke kann natürlich Verwirrung stiften. Bisher haben wir Syntax und Semantik sorgfältig auseinandergehalten. Wenn die Grundlage aller semantischen Begriffe, das Universum, jetzt aus Termen bestehen soll, also aus syntaktischen Elementen, ist eine gewisse Vermischung von Syntax und Semantik unvermeidlich. Diese begriffliche Verwirrung ist sicherlich ein Nachteil des Ansatzes.

Andererseits kennen wir aus der Informatik analoge Fälle von scheinbarer begrifflicher Verwirrung, die sich als äußerst vorteilhaft erweisen. Zum Beispiel muss man bei einem Übersetzer Quellsprache, Zielsprache und Implementierungssprache sorgfältig auseinanderhalten. Wenn man dann Übersetzer betrachtet, deren Implementierungssprache gleich ihrer Quellsprache ist, wirkt dies zunächst verwirrend oder sogar unsinnig. Dabei stellt sich aber heraus, dass solche Übersetzer sogar besonders nützliche Eigenschaften haben. In ähnlicher Weise erweist sich die Vermischung von Syntax und Semantik in der Logik als nützlich.

Zur Realisierung des Ansatzes bleibt noch zu klären, aus welcher Teilmenge von Termen der Sprache das ausgezeichnete Universum bestehen soll. Auf Jacques Herbrand (Paris 1908 – La Bérarde 1931) geht die Idee zurück, die Menge aller Grundterme, also aller variablenfreien Terme, als Universum zu benutzen.

Das Herbrand-Universum einer Sprache $\mathcal{L}$ der Prädikatenlogik erster Stufe ist die Menge aller Grundterme der Sprache. Wir erinnern an eine Voraussetzung in der Definition von $\mathcal{L}$ (Definition 2.3.1): „Es wird vorausgesetzt, dass $\mathcal{L}$ mindestens eine Konstante enthält.“ Dank dieser Voraussetzung ist die Menge der Grundterme nicht leer und kann daher als Universum dienen. (Das war der einzige Grund für die Voraussetzung.)

Definition 3.8.7 (Herbrand-Interpretation). Sei $\mathcal{L}$ eine Sprache der PL1S.

- Das *Herbrand-Universum* $HU_\mathcal{L}$ von $\mathcal{L}$ ist die Menge aller Grundterme von $\mathcal{L}$.
  Die *Herbrand-Basis* $HB_\mathcal{L}$ von $\mathcal{L}$ ist die Menge aller Grundatome von $\mathcal{L}$.
- Eine $\mathcal{L}$-Interpretation $M = (D, Ab, U)$ heißt Herbrand-Interpretation, wenn:
  1. $D = HU_\mathcal{L}$ ist, also die Menge aller Grundterme von $\mathcal{L}$ und
  2. für jeden Grundterm t gilt $Ab(t) = t$, also:
     $Ab(c) = c$ für jedes 0-stellige Funktionssymbol c von $\mathcal{L}$.
     $Ab(f)(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ für alle Grundterme $t_1, \ldots, t_n$
     für jedes n-stellige $(n \geq 1)$ Funktionssymbol f von $\mathcal{L}$.
- Eine Herbrand-Interpretation, die ein Modell einer Formel oder Formelmenge F ist, heißt ein Herbrand-Modell von F. ■

Von den Bestandteilen einer Interpretation ist für Herbrand-Interpretationen also der überwiegende Teil von vornherein vorgegeben. Das Universum ist immer die Menge der Grundterme. Die Funktion Ab ist für Konstanten und Funktionssymbole immer so festgelegt, dass jeder Grundterm „mit sich selbst“ interpretiert wird. Das einzige, was einer Herbrand-Interpretation überhaupt noch an eigenen Definitionsmöglichkeiten offensteht, ist die relativ uninteressante Umgebung sowie die Definition der Funktion Ab für Relationssymbole.

Um letztere zu definieren, muss die Herbrand-Interpretation festlegen, welche Relation, also welche Menge von Tupeln von Grundtermen, jedem Relationssymbol zugeordnet ist. Wenn man vor ein Tupel von Grundtermen das Relationssymbol schreibt, erhält man ein Grundatom. Das, was eine Herbrand-Interpretation neben der Umgebung überhaupt noch festlegen muss, lässt sich damit bequem durch eine Menge B von Grundatomen repräsentieren, also eine Teilmenge der Herbrandbasis $HB_{\mathcal{L}}$.

Definition 3.8.8 (Grundatome/Herbrand-Interpretation). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe. Sei $B \subseteq HB_{\mathcal{L}}$. Sei U eine $HU_{\mathcal{L}}$-Umgebung. Die von B *repräsentierte* Herbrand-Interpretation mit Umgebung U ist die Herbrand-Interpretation $(HU_{\mathcal{L}}, Ab, U)$ mit:

1. $Ab(p) = \{\,()\,\}$ gdw. $p \in B$ und $Ab(p) = \emptyset$ gdw. $p \notin B$
   für jedes 0-stellige Relationssymbol p von $\mathcal{L}$.
2. $Ab(p) = \{(t_1, \dots, t_n) \mid p(t_1, \dots, t_n) \in B\}$
   für jedes n-stellige ($n \geq 1$) Relationssymbol p von $\mathcal{L}$.

Für eine fest gewählte Konstante c von $\mathcal{L}$ sei U_c die Umgebung, die jeder Variablen c zuordnet. $\mathcal{H}_{\mathcal{L}}(B)$ bezeichnet die von B repräsentierte Herbrand-Interpretation mit Umgebung U_c. ■

Ist M die von B repräsentierte Herbrand-Interpretation mit einer beliebigen Umgebung U, gilt mit dieser Definition für jedes Grundatom A, dass $M \models A$ gdw. $A \in B$. Somit erfüllt M also genau die Grundatome in B und falsifiziert alle anderen Grundatome. Für jede andere Formel ist aufgrund der Definition der Modellbeziehung dann ebenfalls festgelegt, ob M die Formel erfüllt oder falsifiziert.

Da wir Herbrand-Interpretationen nur im Zusammenhang mit geschlossenen Formeln betrachten werden, spielt die Umgebung nach dem Koinzidenzsatz 3.5.10 keine Rolle. Durch Vorgabe der festen Umgebung U_c (die auch anders hätte gewählt sein können und die für geschlossene Formeln sowieso irrelevant ist) ist für jedes $B \subseteq HB_{\mathcal{L}}$ die Herbrand-Interpretation $\mathcal{H}_{\mathcal{L}}(B)$ eindeutig bestimmt. Umgekehrt bestimmt jede Herbrand-Interpretation M eindeutig eine Teilmenge $B_M = \{A \in HB_{\mathcal{L}} \mid M \models A\} \subseteq HB_{\mathcal{L}}$. Wegen dieser eineindeutigen Entsprechung bezeichnet man gelegentlich eine Menge von Grundatomen sogar selbst als eine Herbrand-Interpretation bzw. ein Herbrand-Modell.

Satz 3.8.9. Sei $M = (HU_{\mathcal{L}}, Ab, U)$ eine Herbrand-Interpretation einer Sprache $\mathcal{L}$ der PL1S, sei t ein $\mathcal{L}$-Term, F eine $\mathcal{L}$-Formel, x eine Variable.

1. $t^M = t[U(x)/x]^M$.
2. ist $var(t) \subseteq \{x_1, \dots, x_n\}$, dann ist $t^M = t[U(x_1)/x_1] \dots [U(x_n)/x_n]$.
3. $M \models F$ gdw. $M \models F[U(x)/x]$.

Beweis:

1. Es gilt stets $M = M[U(x)/x]$. Da M eine Herbrand-Interpretation und $U(x)$ ein Grundterm ist, gilt $U(x) = U(x)^M$. Also ist $t^M = t^{M[U(x)^M/x]}$. Damit ist die Behauptung ein Spezialfall des Substitutionssatzes 3.5.13.
2. $t[U(x_1)/x_1] \ldots [U(x_n)/x_n]$ ist ein Grundterm, also gleich $t[U(x_1)/x_1] \ldots [U(x_n)/x_n]^M$. Damit folgt die Behauptung aus 1.
3. Da M eine Herbrand-Interpretation ist, ist $U(x)$ ein Grundterm und somit frei für x in F. Wie in 1. gezeigt ist $M = M[U(x)^M/x]$. Damit ist die Behauptung ein Spezialfall des Substitutionssatzes 3.5.13. ∎

Für beliebige Interpretationen übersetzt der Substitutionssatz zwischen einer Umgebung, die einer Variablen semantisch ein Objekt des Universums zuordnet, und einer Substitution, die eine Variable syntaktisch durch einen Term ersetzt. Der obige Satz zeigt, dass für Herbrand-Interpretationen, wo die Objekte des Universums Terme sind, Umgebungen und Substitutionen zusammenfallen. Die Vermischung von Syntax und Semantik bei Herbrand-Interpretationen ist hier also günstig, weil sie eine Vereinfachung bewirkt.

Satz 3.8.10. Sei M eine Herbrand-Interpretation einer Sprache $\mathcal{L}$ der PL1S, sei F eine $\mathcal{L}$-Formel (in der x frei vorkommen kann, aber nicht muss).

1. $M \models \forall x F$ gdw. für jeden Grundterm t gilt $M \models F[t/x]$.
2. $M \models \exists x F$ gdw. es gibt einen Grundterm t mit $M \models F[t/x]$.

Beweis: Satz 3.8.9 (3) anwenden. ∎

Der Satz 3.8.10 gilt für Nicht-Herbrand-Interpretationen im allgemeinen nicht. Wir betrachten zwei Gegenbeispiele.

Beispiel 3.8.11. Die Sprache $\mathcal{L}$ bestehe aus einer Konstanten a und einem einstelligen Relationssymbol p. Sei $M = (\{1, 2\}, Ab, U)$ mit $Ab(a) = 1$ und $Ab(p) = \{1\}$.
Es gilt $M \models p(a)$ und $M[1/x] \models p(x)$ und $M[2/x] \not\models p(x)$. Der einzige Grundterm ist a.

1. Für jeden Grundterm t gilt $M \models p(x)[t/x]$, aber $M \not\models \forall x\, p(x)$.
2. $M \models \exists x\, \neg p(x)$, aber es gibt keinen Grundterm t mit $M \models \neg p(x)[t/x]$. ∎

Hier ist der Satz deshalb verletzt, weil es nur einen einzigen Grundterm gibt, aber zwei Objekte im Universum, so dass das Objekt 2 des Universums durch gar keinen Grundterm repräsentiert ist. Man könnte vermuten, dass dieses Phänomen leicht zu vermeiden sei, sobald die Sprache Funktionssymbole enthält, denn dann gibt es ja unendlich viele Grundterme. Das nächste Beispiel zeigt aber, dass das Phänomen prinzipiell nicht vermeidbar ist.

Beispiel 3.8.12. Die Sprache $\mathcal{L}$ bestehe aus abzählbar unendlich vielen Konstanten und zu jeder Stelligkeit abzählbar unendlich vielen Funktionssymbolen sowie aus einem einstelligen Relationssymbol p. Sei $\mathbb{R}$ die Menge der reellen Zahlen und $M = (\mathbb{R}, Ab, U)$ mit $Ab(p) = \{t^M \mid t \text{ Grundterm}\}$. Da es nur abzählbar viele Grundterme gibt, gibt es $d \in \mathbb{R}$ mit $d \notin Ab(p)$, so dass $M[d/x] \not\models p(x)$.

1. Für jeden Grundterm t gilt $M \models p(x)[t/x]$, aber $M \not\models \forall x\, p(x)$.
2. $M \models \exists x\, \neg p(x)$, aber es gibt keinen Grundterm t mit $M \models \neg p(x)[t/x]$. ∎

Man kann zu jeder beliebigen Interpretation M auf natürliche Weise eine Herbrand-Interpretation konstruieren, die M weitgehend nachbildet.

Satz 3.8.13. Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe, sei M eine $\mathcal{L}$-Interpretation, sei $B_M := \{A \in HB_\mathcal{L} \mid M \models A\}$. Die von B_M repräsentierte Herbrand-Interpretation $\mathcal{H}_\mathcal{L}(B_M)$ hat folgende Eigenschaften:

1. Für jede quantorfreie geschlossene Formel F gilt: $M \models F$ gdw. $\mathcal{H}_\mathcal{L}(B_M) \models F$.
2. Für jede universelle geschlossene Formel F gilt: wenn $M \models F$, dann $\mathcal{H}_\mathcal{L}(B_M) \models F$.

Beweis:

1. Durch strukturelle Induktion.

   Basisfall $\top$ oder $\bot$: ergibt sich unmittelbar aus der Definition von $\models$.

   Basisfall Atom: da F quantorfrei und geschlossen ist, ist F ein Grundatom, also gilt $M \models F$ gdw. $F \in B_M$ gdw. $\mathcal{H}_\mathcal{L}(B_M) \models F$.

   Induktionsfall Junktor: ergibt sich unmittelbar durch Anwendung der Definition von $\models$ auf die Induktionsannahme.

   Induktionsfall Quantor: tritt nicht auf, da F quantorfrei ist.

2. Es gelte $M \models F$. Da F universell ist, enthält seine nach Satz 3.8.3 gebildete Pränexform $F_{prä}$ keine Existenzquantoren. Wegen $F \models\!\!\mid F_{prä}$ gilt $M \models F_{prä}$.

   Wir zeigen durch vollständige Induktion über die Anzahl n der Quantoren in $F_{prä}$:

   $$\text{wenn } M \models F_{prä}, \text{ dann } \mathcal{H}_\mathcal{L}(B_M) \models F_{prä}$$

   Mit $F \models\!\!\mid F_{prä}$ erhalten wir dann $\mathcal{H}_\mathcal{L}(B_M) \models F$, was zu zeigen war.

   Induktionsbasis $n = 0$: Es gelte $M \models F_{prä}$. Enthält $F_{prä}$ keinen Quantor, ist es eine quantorfreie geschlossene Formel, und nach 1. gilt $\mathcal{H}_\mathcal{L}(B_M) \models F_{prä}$.

   Induktionsschritt $n \to n+1$: Enthält $F_{prä}$ $n+1$ Quantoren, hat es die Gestalt $\forall x\, G_{prä}$. Es gelte $M \models F_{prä}$, also $M \models \forall x\, G_{prä}$. Nach Definition bedeutet dies, wenn D das Universum von M bezeichnet, dass für jedes $d \in D$ gilt $M[d/x] \models G_{prä}$.

   Angenommen, es gäbe einen Grundterm t mit $\mathcal{H}_\mathcal{L}(B_M)[t/x] \not\models G_{prä}$, das heißt nach Satz 3.8.9, $\mathcal{H}_\mathcal{L}(B_M)[t/x] \not\models G_{prä}[t/x]$. Da $x \notin fvar(G_{prä}[t/x])$ ist, folgt daraus nach dem Koinzidenzsatz 3.5.10 $\mathcal{H}_\mathcal{L}(B_M) \not\models G_{prä}[t/x]$. Die Formel $G_{prä}[t/x]$ ist eine geschlossene universelle Formel in Pränexform mit n Quantoren, also folgt weiter mit der Induktionsannahme dass $M \not\models G_{prä}[t/x]$. Nach dem Substitutionssatz 3.5.13 müsste dann gelten $M[t^M/x] \not\models G_{prä}$, im Widerspruch dazu dass für jedes $d \in D$ gilt $M[d/x] \models G_{prä}$.

   Also gibt es keinen Grundterm t mit $\mathcal{H}_\mathcal{L}(B_M)[t/x] \not\models G_{prä}$, das heißt, für jeden Grundterm t gilt $\mathcal{H}_\mathcal{L}(B_M)[t/x] \models G_{prä}$. Das ist genau die Definition von $\mathcal{H}_\mathcal{L}(B_M) \models \forall x\, G_{prä}$, also $\mathcal{H}_\mathcal{L}(B_M) \models F_{prä}$. ∎

Es mag verwundern, dass der zweite Teil dieses Satzes nur mit „wenn dann“ formuliert ist und nicht mit „genau dann wenn“. Tatsächlich gilt die Gegenrichtung im allgemeinen nicht. Der Grund sind genau die obigen Beispiele 3.8.11 und 3.8.12. Für die darin angegebene Interpretation M gilt jeweils $\mathcal{H}_\mathcal{L}(B_M) \models \forall x\, p(x)$, aber $M \not\models \forall x\, p(x)$.

Die Beispiele zeigen außerdem die Notwendigkeit der Voraussetzung, dass F universell ist. In den Beispielen gilt jeweils $M \models \exists x\, \neg p(x)$, aber $\mathcal{H}_{\mathcal{L}}(B_M) \not\models \exists x\, \neg p(x)$.

Der Zusammenhang zwischen der Erfüllung einer Formel durch eine beliebige Interpretation und der Erfüllung der Formel durch die aus der Interpretation konstruierte Herbrand-Interpretation ist also nicht sehr stark. Trotzdem hat er weitreichende Konsequenzen.

Folgesatz 3.8.14. Sei $F_\forall$ eine geschlossene universelle Formel. Sei $S_\forall$ eine Menge von solchen Formeln. Sei $G_\exists$ eine geschlossene Formel derart dass $\neg G_\exists$ universell ist.

1. $G_\exists$ ist allgemeingültig gdw. jede Herbrand-Interpretation erfüllt $G_\exists$.
2. $G_\exists$ ist falsifizierbar gdw. eine Herbrand-Interpretation falsifiziert $G_\exists$.
3. $F_\forall$ ist erfüllbar gdw. eine Herbrand-Interpretation erfüllt $F_\forall$.
4. $F_\forall$ ist unerfüllbar gdw. jede Herbrand-Interpretation falsifiziert $F_\forall$.
5. $S_\forall$ ist erfüllbar gdw. $S_\forall$ ein Herbrand-Modell besitzt.
6. $F_\forall \models G_\exists$ gdw. jedes Herbrand-Modell von $F_\forall$ auch $G_\exists$ erfüllt.
7. $S_\forall \models G_\exists$ gdw. jedes Herbrand-Modell von $S_\forall$ auch $G_\exists$ erfüllt. ∎

Für die Formeln und Formelmengen mit den angegebenen syntaktischen Eigenschaften ist damit das eingangs motivierte Ziel erreicht. Zum Nachweis der Folgerungsbeziehung reicht es wegen der letzten beiden Ergebnisse aus, die Überprüfung auf Herbrand-Interpretationen und damit auf ein einziges Universum zu beschränken.

Übrigens hätte die Motivation in der Einleitung zu Abschnitt 3.8 nicht unbedingt auf der Folgerungsbeziehung aufgebaut werden müssen. Eine analoge Motivation wäre auch mit der Erfüllbarkeit oder einem anderen der semantischen Begriffe möglich gewesen. Die ersten Ergebnisse des Satzes zeigen, dass das Ziel auch für diese anderen semantischen Begriffe jetzt erreicht ist: zum Nachweis der jeweiligen semantischen Eigenschaft müssen nicht mehr sämtliche Interpretationen betrachtet werden, sondern nur noch die Herbrand-Interpretationen, jedenfalls für Formeln und Formelmengen mit den angegebenen syntaktischen Eigenschaften.

Definition 3.8.15 (Grundinstanz). Sei $F = \forall x_1 \ldots \forall x_n\, G$ eine geschlossene universelle Formel in Pränexform. Sei S eine Menge von solchen Formeln.
Die Menge der *Grundinstanzen* von F ist $\{G[t_1/x_1] \ldots [t_n/x_n] \mid t_i \text{ Grundterm}\}$.
Die Menge der Grundinstanzen von S ist die Vereinigung der Mengen von Grundinstanzen der Formeln in S. ∎

Aus Satz 3.8.10 folgt unter Zuhilfenahme von Satz 3.8.14 unmittelbar ein weiteres Ergebnis mit sehr weitreichenden Konsequenzen, auf die Kapitel 4 näher eingehen wird.

Folgesatz 3.8.16. Sei S eine Menge von geschlossenen universellen Formeln in Pränexform. S ist erfüllbar gdw. die Menge der Grundinstanzen von S ein Herbrand-Modell besitzt. ∎

Skolemisierung

In diesem Abschnitt zeigen wir, dass zu jeder endlichen Menge S von (beliebigen) geschlossenen Formeln eine Menge S^* von *universellen* geschlossenen Formeln konstruiert werden kann, so dass S^* genau dann ein Modell hat, wenn S ein Modell hat. Dieses Ergebnis reicht aus, um die Folgerungsbeziehung für beliebige Formeln durch die Überprüfung von Herbrand-Interpretationen nachweisen zu können. Das Verfahren, mit dem S^* aus S konstruiert wird, nennt man Skolemisierung, nach seinem Erfinder Albert Thoralf Skolem (Sandsvær 1887 – Oslo 1963).

Der genaue Zusammenhang zwischen S und S^* ist allerdings etwas komplizierter. Die Menge S enthält $\mathcal{L}$-Formeln für eine gegebene Sprache $\mathcal{L}$ der Prädikatenlogik erster Stufe. Diese Sprache wird zunächst um zusätzliche Funktionssymbole erweitert zu einer Sprache $\mathcal{L}^*$. Die Menge S^* besteht dann aus $\mathcal{L}^*$-Formeln. Man muss deshalb auch zwischen $\mathcal{L}$-Interpretationen und $\mathcal{L}^*$-Interpretationen unterscheiden. Das ist der wesentliche Grund, dass S und S^* nicht einfach logisch äquivalent sind. Aber die Beziehung zwischen den Modellen von S (bezüglich $\mathcal{L}$) und den Modellen von S^* (bezüglich $\mathcal{L}^*$) wird sich als sehr viel enger erweisen, als es für das obengenannte Gesamtergebnis erforderlich ist.

Um die Grundidee der Skolemisierung informell zu verstehen, betrachten wir noch einmal die Axiome einer Boole'schen Algebra nach Definition 3.2.1.

Das erste Axiom verlangt unter anderem, dass es ein neutrales Element für die Operation $\times$ gibt. Die unmittelbare Übersetzung dieser Forderung ergäbe folgende Formel der Prädikatenlogik erster Stufe: $\exists y \forall x\,(y \times x \doteq x)$. Tatsächlich wurde das Axiom aber so formuliert, dass die entsprechende Formel der Prädikatenlogik erster Stufe lautet: $\forall x\,(1 \times x \doteq x)$. Das als existent geforderte Element wurde also durch eine Konstante repräsentiert statt durch eine existentiell quantifizierte Variable.

Das letzte Axiom verlangt unter anderem, dass es zu jedem Element ein komplementäres Element bezüglich der Operation $+$ gibt. Diese Forderung lässt sich darstellen durch die Formel $\forall x \exists y\,(x + y \doteq 1)$, aber auch durch $\forall x\,(x + \overline{x} \doteq 1)$. Dabei ist $\overline{}$ ein Funktionssymbol, das für eine Funktion steht, die zu jedem x ein komplementäres Element liefert.

Ein als existent gefordertes Objekt kann also statt durch eine existentiell quantifizierte Variable auch durch einen Term repräsentiert werden, der mit einem Funktionssymbol gebildet ist. Die Unterterme in diesem Term repräsentieren Objekte, von denen das als existent geforderte abhängt. Im ersten Beispiel bestand so eine Abhängigkeit nicht, so dass der Term mit einem nullstelligen Funktionssymbol, also einer Konstanten, gebildet wurde.

Die Skolemisierung basiert auf dem Prinzip, für einen Existenzquantor positiver Polarität oder Allquantor negativer Polarität ein neues Funktionssymbol einzuführen, damit einen Term zu bilden, und alle durch den Quantor gebundenen Variablenvorkommen durch diesen Term zu ersetzen. Dadurch wird der Quantor überflüssig. Nach diesem Prinzip können systematisch alle störenden Quantoren beseitigt werden, bis eine universelle Formel entsteht.

Welcher Zusammenhang besteht nun zwischen einer Formel QxG, die mit einem störenden Quantor beginnt, und der daraus konstruierten Formel G', in der der störende Quantor beseitigt ist?

Der folgende Satz zeigt, dass abhängig von Q entweder $G' \models QxG$ oder $QxG \models G'$ gilt. Die Formeln sind aber nicht logisch äquivalent, das heißt, die jeweilige Gegenrichtung gilt nicht. Es gilt jedoch eine abgeschwächte Gegenrichtung, die durch folgende Betrachtung erläutert werden kann. Wenn zwei Formeln G_1 und G_2 die Eigenschaft haben, dass für jede

Interpretation M gilt $M \models (G_1 \Rightarrow G_2)$, dann besteht die Folgerungsbeziehung $G_1 \models G_2$. Wenn $M \models (G_1 \Rightarrow G_2)$ nun nicht für jedes M gilt, sondern nur für gewisse M', die man aber aus jedem gegebenen M konstruieren kann, dann kann man von einer Art schwacher Folgerungsbeziehung zwischen G_1 und G_2 sprechen. Eine derartige schwache Folgerungsbeziehung besteht jeweils in der Gegenrichtung, wobei die gewissen M' überhaupt nicht von den Umgebungen abhängen.

Notation 3.8.17. Für jede $\mathcal{L}$-Interpretation $M = (D, Ab, U_0)$ und jede D-Umgebung U bezeichne $M[U]$ die $\mathcal{L}$-Interpretation (D, Ab, U). ∎

Satz 3.8.18 (elementarer Skolemisierungsschritt). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe. Sei G eine $\mathcal{L}$-Formel mit $\mathit{fvar}(G) = \{x, y_1, \ldots, y_n\}$, $n \geq 0$, so dass jedes y_i frei für x in G ist. Sei f ein n-stelliges Funktionssymbol, das nicht in G vorkommt. Sei $G' = G[f(y_1, \ldots, y_n)/x]$. Dann gilt:

1. Zu jeder $\mathcal{L}$-Interpretation $M = (D, Ab, U_0)$ gibt es eine $\mathcal{L}$-Interpretation $M' = (D, Ab', U_0)$, die sich höchstens darin von M unterscheidet, dass $Ab'(f) \neq Ab(f)$ sein kann, so dass für jede D-Umgebung U gilt: $M'[U] \models (\exists xG \Rightarrow G')$.
2. $G' \models \exists xG$.
3. $\forall xG \models G'$
4. Zu jeder $\mathcal{L}$-Interpretation $M = (D, Ab, U_0)$ gibt es eine $\mathcal{L}$-Interpretation $M' = (D, Ab', U_0)$, die sich höchstens darin von M unterscheidet, dass $Ab'(f) \neq Ab(f)$ sein kann, so dass für jede D-Umgebung U gilt: $M'[U] \models (G' \Rightarrow \forall xG)$.

Beweis:

1. Sei $d_0 \in D$ fest gewählt. Für jedes n-Tupel $(d_1, \ldots, d_n) \in D^n$ gibt es zwei Fälle:
   Falls $M[d_1/y_1] \ldots [d_n/y_n] \not\models \exists xG$, definieren wir $Ab'(f)(d_1, \ldots, d_n) := d_0$.
   Falls $M[d_1/y_1] \ldots [d_n/y_n] \models \exists xG$, gibt es $d \in D$ mit $M[d_1/y_1] \ldots [d_n/y_n][d/x] \models G$, wir wählen ein solches d und definieren $Ab'(f)(d_1, \ldots, d_n) := d$.

   Sei U eine beliebige D-Umgebung.

   Falls $M'[U] \not\models \exists xG$, gilt $M'[U] \models (\exists xG \Rightarrow G')$ nach Definition von $\models$.
   Falls $M'[U] \models \exists xG$, ist $f(y_1, \ldots, y_n)$ nach Konstruktion ein Term mit der Eigenschaft $M'[U][f(y_1, \ldots, y_n)^{M'}/x] \models G$. Dieser Term ist frei für x in G, weil das für jedes y_i gilt, und nach dem Substitutionssatz 3.5.13 folgt daraus $M'[U] \models G[f(y_1, \ldots, y_n)/x]$, also $M'[U] \models G'$. Nach Definition von $\models$ gilt dann auch $M'[U] \models (\exists xG \Rightarrow G')$.
2. Sei $M = (D, Ab, U)$ eine $\mathcal{L}$-Interpretation mit $M \models G'$, also $M \models G[f(y_1, \ldots, y_n)/x]$. Nach dem Substitutionssatz 3.5.13 gilt $M[f(y_1, \ldots, y_n)^M/x] \models G$. Also gibt es $d \in D$, nämlich $d = f(y_1, \ldots, y_n)^M$, mit $M[d/x] \models G$. Nach Definition gilt $M \models \exists xG$.
3. Folgt unmittelbar aus (2) für $\neg G$ und $\neg G'$ anstelle von G und G' (Satz 3.5.9 (4) und Satz 3.5.14).
4. Folgt unmittelbar aus (1) für $\neg G$ und $\neg G'$ anstelle von G und G' (Satz 3.5.14). ∎

Die Formeln $\exists xG$ und G' (bzw. G' und $\forall xG$) sind also nicht logisch äquivalent, aber fast. In der einen Richtung gilt die Folgerungsbeziehung immer. In der anderen gilt sie nicht immer, weil manche Interpretationen das Funktionssymbol f mit einer „unpassenden" Funktion interpretieren können. Da nach Voraussetzung f nicht in G vorkommt, kann man in diesen Fällen aber immer eine „passende" Definition für $Ab'(f)$ angeben, ohne irgend etwas anderes an der Interpretation zu ändern, so dass die Folgerungsbeziehung bezüglich dieser „angepassten" Interpretationen auch in der Gegenrichtung gilt.

Als nächstes muss folgende Situation genauer untersucht werden: Gegeben ist eine „große" Formel F. Darin kommt eine „kleine" Teilformel vor, die mit einem störenden Quantor beginnt. Wenn man diese Teilformel durch eine andere „kleine" Formel ersetzt, in der der störende Quantor beseitigt ist, entsteht die „große" Formel F'.

$$\begin{array}{rcl} F &=& (\ldots \quad QxG \quad \ldots) \\ && \qquad\ \ \blacktriangledown \\ F' &=& (\ldots \quad\ \ G' \quad\ \ \ldots) \end{array}$$

Um die Folgerungsbeziehungen, die gemäß Satz 3.8.18 zwischen den „kleinen" Formeln QxG und G' bestehen, auf Beziehungen zwischen den „großen" Formeln F und F' zu übertragen, benötigen wir einige Verallgemeinerungen der Ersetzungssätze 3.3.6 und 3.5.14.

Satz 3.8.19 (Ersetzungssatz für Folgerungen). F, G_1, G_2 seien $\mathcal{L}$-Formeln, V_{G_1} sei ein Vorkommen von G_1 als Teilformel in F mit eindeutiger Polarität. Sei $F\langle G_2/V_{G_1}\rangle$ die $\mathcal{L}$-Formel, die aus F entsteht, wenn das Vorkommen V_{G_1} von G_1 in F durch G_2 ersetzt wird. Dann gilt:

1. Wenn V_{G_1} positiv in F und $G_1 \models G_2$, dann $F \models F\langle G_2/V_{G_1}\rangle$.
2. Wenn V_{G_1} positiv in F und $G_2 \models G_1$, dann $F\langle G_2/V_{G_1}\rangle \models F$.
3. Wenn V_{G_1} negativ in F und $G_1 \models G_2$, dann $F\langle G_2/V_{G_1}\rangle \models F$.
4. Wenn V_{G_1} negativ in F und $G_2 \models G_1$, dann $F \models F\langle G_2/V_{G_1}\rangle$.

Beweis: Strukturelle Induktion. ∎

Bei Ersetzung eines Vorkommens einer Teilformel überträgt sich demnach die Folgerungsbeziehung von der Teilformel auf die Gesamtformel, sofern das Vorkommen positive Polarität hat. Hat das Vorkommen negative Polarität, überträgt sich die Folgerungsbeziehung in entgegengesetzter Richtung auf die Gesamtformel.

Hat ein Vorkommen sowohl positive als auch negative Polarität, überträgt sich die Folgerungsbeziehung im allgemeinen weder in die eine noch in die andere Richtung. Sei zum Beispiel $G_1 = (p \wedge q)$ und $G_2 = (p \vee q)$, dann gilt $G_1 \models G_2$. Sei $F = ((p \wedge q) \Leftrightarrow \neg q)$, dann ist $F\langle G_2/V_{G_1}\rangle = ((p \vee q) \Leftrightarrow \neg q)$. Man rechnet leicht nach, dass sowohl F als auch $F\langle G_2/V_{G_1}\rangle$ ein Modell haben, dass aber keine Interpretation beide Formeln erfüllt, so dass die Folgerungsbeziehung in keiner Richtung besteht.

Mit diesem Ersetzungssatz übertragen sich die Folgerungsbeziehungen wie folgt von den Teilformeln auf die „großen" Formeln.

Wenn die Teilformel QxG positiv in F ist, hat sie die Gestalt $\exists xG$.

Satz 3.8.18 (2) garantiert:	Satz 3.8.19 (2) besagt, dass daraus folgt:
$$\begin{array}{rcl} && \qquad\ \ + \\ F &=& (\ldots \quad \exists xG \quad \ldots) \\ && \qquad\ \ \Perp \\ F' &=& (\ldots \quad\ \ G' \quad\ \ \ldots) \end{array}$$	$$\begin{array}{rcl} && \qquad\ \ + \\ F &=& (\ldots \quad \exists xG \quad \ldots) \\ \Perp && \\ F' &=& (\ldots \quad\ \ G' \quad\ \ \ldots) \end{array}$$

Wenn die Teilformel QxG negativ in F ist, hat sie die Gestalt $\forall xG$.

Satz 3.8.18 (3) garantiert: Satz 3.8.19 (3) besagt, dass daraus folgt:

$$\begin{array}{rcllcl} & & & \overset{-}{} & & \\ F & = & (\dots & \forall xG & \dots) \\ & & & \text{⫪} & & \\ F' & = & (\dots & G' & \dots) \end{array} \qquad\qquad \begin{array}{rcllcl} & & & \overset{-}{} & & \\ F & = & (\dots & \forall xG & \dots) \\ \text{⫫} & & & & & \\ F' & = & (\dots & G' & \dots) \end{array}$$

In beiden Fällen gilt also $F' \models F$. Wird ein Existenzquantor positiver Polarität oder ein Allquantor negativer Polarität durch Skolemisierung beseitigt, so folgt in jedem Fall die ursprüngliche Formel F aus der neuen Formel F'. (Es ist übrigens Absicht, dass gemeinsam benutzte Unterfälle der beiden Sätze jeweils die gleiche Nummer (2) bzw. (3) haben. Das gilt auch für die weiteren Sätze mit Unterfällen.)

Für die Gegenrichtung besteht keine Folgerungsbeziehung zwischen QxG und G', sondern die oben erläuterte schwache Form davon. Man kann zu jedem Modell M von F ein anderes Modell M' von F so konstruieren, dass diese schwache Folgerungsbeziehung zwischen QxG und G' besteht. Betrachten wir zunächst nur den Fall, dass die Teilformel positiv in F ist.

$$\begin{array}{cccclcl} & & M' & & & M' & \\ & & \text{⫪} & & & \text{⫪}\;^{+} & \\ M & \models & F & = & (\dots & \exists xG & \dots) \\ & & & & & \Downarrow & \\ & & F' & = & (\dots & G' & \dots) \end{array}$$

Wenn man in dieser Situation schließen kann, dass auch $M' \models (F \Rightarrow F')$ gilt, dann gilt auch $M' \models F'$. Damit hätte man die schwache Folgerungsbeziehung von den Teilformeln auf die „großen“ Formeln übertragen. Diese Übertragung ist tatsächlich möglich, aber nur für Interpretationen, die völlig unabhängig von den Umgebungen sind. Die nächste, noch allgemeinere, Variante des Ersetzungssatzes erlaubt im obigen Fall die angestrebte Übertragung auf die „große“ Formel, wenn M' die Eigenschaft hat, dass für jede D-Umgebung U gilt $M'[U] \models (\exists xG \Rightarrow G')$. Dann gilt auch für jedes U, dass $M'[U] \models (F \Rightarrow F')$.

Satz 3.8.20 (interpretationsspezifischer Ersetzungssatz). Mit den Voraussetzungen wie in Satz 3.8.19 sei zusätzlich M eine $\mathcal{L}$-Interpretation. Dann gilt:

1. Wenn V_{G_1} positiv in F und für jedes U gilt $M[U] \models (G_1 \Rightarrow G_2)$,
dann gilt für jedes U: $M[U] \models (F \Rightarrow F\langle G_2/V_{G_1}\rangle)$.
2. Wenn V_{G_1} positiv in F und für jedes U gilt $M[U] \models (G_2 \Rightarrow G_1)$,
dann gilt für jedes U: $M[U] \models (F\langle G_2/V_{G_1}\rangle \Rightarrow F)$.
3. Wenn V_{G_1} negativ in F und für jedes U gilt $M[U] \models (G_1 \Rightarrow G_2)$,
dann gilt für jedes U: $M[U] \models (F\langle G_2/V_{G_1}\rangle \Rightarrow F)$.
4. Wenn V_{G_1} negativ in F und für jedes U gilt $M[U] \models (G_2 \Rightarrow G_1)$,
dann gilt für jedes U: $M[U] \models (F \Rightarrow F\langle G_2/V_{G_1}\rangle)$.

Beweis: Strukturelle Induktion. ∎

Auch für die schwache Folgerungsbeziehung dreht sich also die Richtung um, wenn sie von einer Teilformel negativer Polarität auf die „große“ Formel übertragen wird.

Dieser Ersetzungssatz gilt nicht, wenn die Modellbeziehung von der Umgebung abhängt: Sei $G = p(x,x)$ und $G' = p(x,y)$. Sei $F = \forall x \forall y\, p(x,x)$ und $F' = F[G'/G] = \forall x \forall y\, p(x,y)$. Es gibt eine Interpretation $M = (D, Ab, U)$ mit $M \models (G \Rightarrow G')$, nämlich $D = \mathbb{N}$ und

$Ab(p) = \leq_{\mathbb{N}}$ und $U(x){=}U(y){=}5$. Die schwache Folgerungsbeziehung zwischen G und G' besteht also nur deswegen, weil x und y in der Umgebung U den selben Wert haben. Aber für dieses M gilt $M \not\models (F \Rightarrow F')$, denn $M \models \forall x \forall y\, p(x,x)$ aber $M \not\models \forall x \forall y\, p(x,y)$. Die schwache Folgerungsbeziehung überträgt sich also nicht auf die „großen" Formeln.

Die Interpretationen, die mit Satz 3.8.18 konstruiert werden, hängen nicht von der Umgebung ab, deshalb kann Satz 3.8.20 darauf angewandt werden. Das Gesamtbild für die Gegenrichtung ist wie folgt.

Wenn die Teilformel QxG positiv in F ist, hat sie die Gestalt $\exists xG$.

Satz 3.8.18 (1) garantiert:

für jede Umgebung U gilt

$$\begin{array}{ccccccc}
 & & M' & & & M'[U] & \\
 & & \rotatebox{-90}{$\models$} & & & \rotatebox{-90}{$\models$} & \\
M & \models & F & = & (\ldots & \exists xG^{+} & \ldots) \\
 & & & & & \Downarrow & \\
 & & F' & = & (\ldots & G' & \ldots)
\end{array}$$

Satz 3.8.20 (1) besagt, dass daraus folgt:

für jede Umgebung U gilt

$$\begin{array}{ccccccc}
 & & M'[U] & & & & \\
 & & \rotatebox{-90}{$\models$} & & & & \\
M & \models & F & = & (\ldots & \exists xG^{+} & \ldots) \\
 & & \Downarrow & & & & \\
 & & F' & = & (\ldots & G' & \ldots)
\end{array}$$

Wenn die Teilformel QxG negativ in F ist, hat sie die Gestalt $\forall xG$.

Satz 3.8.18 (4) garantiert:

für jede Umgebung U gilt

$$\begin{array}{ccccccc}
 & & M' & & & M'[U] & \\
 & & \rotatebox{-90}{$\models$} & & & \rotatebox{-90}{$\models$} & \\
M & \models & F & = & (\ldots & \forall xG^{-} & \ldots) \\
 & & & & & \Uparrow & \\
 & & F' & = & (\ldots & G' & \ldots)
\end{array}$$

Satz 3.8.20 (4) besagt, dass daraus folgt:

für jede Umgebung U gilt

$$\begin{array}{ccccccc}
 & & M'[U] & & & & \\
 & & \rotatebox{-90}{$\models$} & & & & \\
M & \models & F & = & (\ldots & \forall xG^{-} & \ldots) \\
 & & \Downarrow & & & & \\
 & & F' & = & (\ldots & G' & \ldots)
\end{array}$$

In beiden Fällen bedeutet die Übertragung, dass $M'[U] \models (F \Rightarrow F')$ für jede Umgebung U gilt. Also gilt auch $M' \models (F \Rightarrow F')$ und wegen $M' \models F$ auch $M' \models F'$. Insgesamt ist damit aus dem Modell M von F ein Modell M' von F konstruiert, das auch Modell von F' ist.

Jetzt müssen die Bausteine nur noch zusammengefügt werden.

Satz 3.8.21 (Skolemisierung). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe und sei F eine geschlossene $\mathcal{L}$-Formel, in der jedes Quantorvorkommen eine eindeutige Polarität hat. Dann gibt es eine Erweiterung $\mathcal{L}^*$ der Sprache $\mathcal{L}$ um endlich viele zusätzliche Funktionssymbole und eine universelle geschlossene $\mathcal{L}^*$-Formel F^*, so dass gilt:

1. $F^* \models F$
2. Zu jeder $\mathcal{L}$-Interpretation M mit $M \models F$ gibt es eine $\mathcal{L}^*$-Interpretation M^*, deren Restriktion auf die Symbole von $\mathcal{L}$ gleich M ist, so dass $M^* \models F$ und $M^* \models F^*$.

Beweis: Durch vollständige Induktion über die Anzahl k der Vorkommen von Quantoren falscher Polarität, also von Existenzquantoren positiver Polarität in F und von Allquantoren negativer Polarität in F.

Induktionsbasis $k = 0$*:* Dann ist F universell, und die Behauptung gilt für $\mathcal{L}^* = \mathcal{L}$ und $F^* = F$.

Induktionsschritt $k \to k+1$*:* Die Behauptung gelte für Formeln mit k Vorkommen von Quantoren falscher Polarität. Sei F eine $\mathcal{L}$-Formel mit $k{+}1$ solchen Vorkommen und sei $V_{Qx\,G}$

eines davon. Alle Vorkommen von Quantoren in F seien o.B.d.A. Quantoren für paarweise verschiedene Variablen (das ist durch geeignete Variablenumbenennung stets erreichbar).

Falls $x \notin \mathit{fvar}(G)$, sei $F' = F\langle G/V_{Qx\,G}\rangle$. Nach den Regeln der überflüssigen Quantoren (Satz 3.5.14) gilt $Qx\ G \models\!\!\!| \ G$, also $F \models\!\!\!| \ F'$. Die Formel F' enthält k Vorkommen von Quantoren falscher Polarität. Die Behauptung ergibt sich direkt aus der Induktionsannahme.

Andernfalls sei $\mathit{fvar}(G) = \{x, y_1, \ldots, y_n\}$. Jedes dieser y_i ist frei für x in G, weil jedes y_i in F durch einen Quantor gebunden ist und in G kein zweiter Quantor für die gleiche Variable vorkommt. Sei f ein n-stelliges Funktionssymbol, das nicht zu der Sprache $\mathcal{L}$ gehört, und sei $\mathcal{L}'$ die Erweiterung von $\mathcal{L}$ um f. Sei $G' = G[f(y_1, \ldots, y_n)/x]$ und $F' = F\langle G'/V_{Qx\,G}\rangle$. Dann ist F' eine geschlossene $\mathcal{L}'$-Formel.

Mit den vorigen Sätzen gilt $F' \models F$, im positiven Fall nach Satz 3.8.18 (2) und 3.8.19 (2), im negativen Fall nach Satz 3.8.18 (3) und 3.8.19 (3).

Zu jeder $\mathcal{L}$-Interpretation M mit $M \models F$ gibt es, im positiven Fall nach Satz 3.8.18 (1) und 3.8.20 (1), im negativen Fall nach Satz 3.8.18 (4) und 3.8.20 (4), eine $\mathcal{L}'$-Interpretation M', deren Restriktion auf die Symbole von $\mathcal{L}$ gleich M ist, so dass für jede D-Umgebung U gilt $M'[U] \models (F \Rightarrow F')$. Da F und damit auch F' geschlossen ist, gilt nach dem Koinzidenzsatz 3.5.10 auch $M' \models (F \Rightarrow F')$. Wegen $M \models F$ gilt auch $M' \models F$ und nach Definition von $\models$ auch $M' \models F'$.

F' ist eine geschlossene $\mathcal{L}'$-Formel mit k Quantorvorkommen falscher Polarität. Der Rest ergibt sich unmittelbar aus der Induktionsannahme. ∎

Bei der Skolemisierung einer Formel nach der Konstruktion in diesem Beweis werden falsch quantifizierte Variablen der Reihe nach durch Terme ersetzt, die mit neu eingeführten Funktionssymbolen, sogenannten Skolem-Funktionssymbolen, gebildet werden. Eine aus F durch Skolemisierung gewonnene universelle Formel nennt man eine *Skolemform* von F. Die mit den Skolem-Funktionssymbolen gebildeten Terme kann man als eine Umbenennung der Objekte auffassen, deren Existenz in der Formel gefordert wird.

Die Skolemform ist nicht eindeutig, da sie von der Reihenfolge abhängt, in der die Quantoren falscher Polarität ersetzt werden. Sei F die geschlossene Formel $\forall u \exists v \exists w\, p(u, v, w)$. Ersetzen wir zuerst $\exists v$ und dann $\exists w$, entsteht erst $\forall u \exists w p(u, f(u), w)$ und dann $\forall u p(u, f(u), g(u))$. Das ist eine Skolemform von F. In der anderen Reihenfolge entsteht erst $\forall u \exists v p(u, v, h(u, v))$, dann $\forall u\, p(u, k(u), h(u, k(u)))$. Das ist auch eine Skolemform von F.

Es gibt verschiedene Verfahren, um eine Skolemform aus einer Formel zu gewinnen. Neben der Festlegung der Reihenfolge der Ersetzungen kann man auch festlegen, dass vor den Ersetzungen zunächst die Quantoren in der Formel verschoben werden. Am einfachsten zu verstehen ist die Skolemisierung, wenn man die Formel zunächst in eine Pränexform umwandelt. Um die Komplexität der neuen Terme gering zu halten, ist es dagegen günstiger, die Quantoren zunächst möglichst weit nach innen zu ziehen und eine sogenannte Antipränexform oder Miniscope-Form herzustellen.

Die Voraussetzung des obigen Satzes, dass jedes Quantorvorkommen eine eindeutige Polarität hat, lässt sich mit der Regel zur Auflösung des Äquivalenzjunktors (Satz 3.5.14 und Satz 3.3.7) immer herstellen. Also gilt:

Folgesatz 3.8.22.

1. Zu jeder geschlossenen $\mathcal{L}$-Formel F gibt es eine Erweiterung $\mathcal{L}_{sko}$ von $\mathcal{L}$ um endlich viele Funktionssymbole und eine universelle geschlossene $\mathcal{L}_{sko}$-Formel $\mathit{sko}(F)$, so dass F genau dann ein Modell hat, wenn $\mathit{sko}(F)$ ein Modell hat.

2. Zu jeder abzählbaren Menge S von geschlossenen $\mathcal{L}$-Formeln gibt es eine Erweiterung $\mathcal{L}_{sko}$ von $\mathcal{L}$ um abzählbar viele Funktionssymbole und eine abzählbare Menge $sko(S)$ von universellen geschlossenen $\mathcal{L}_{sko}$-Formeln, so dass S genau dann ein Modell hat, wenn $sko(S)$ ein Modell hat.

   Zwischen S und $sko(S)$ gibt es eine Bijektion, so dass jedem $F \in S$ eineindeutig eine Formel $sko(F) \in sko(S)$ zugeordnet ist.

   Ist S endlich, ist $\mathcal{L}_{sko}$ eine Erweiterung von $\mathcal{L}$ um endlich viele Funktionssymbole. ■

Kombiniert mit Satz 3.8.14 ergibt sich daraus die Lösung für das Gesamtziel dieses Abschnitts: alle semantischen Begriffe auf Herbrand-Interpretationen zurückzuführen.

Folgesatz 3.8.23. Seien F und G geschlossene Formeln und S eine endliche Menge von geschlossenen Formeln.

1. F ist allgemeingültig gdw. jede Herbrand-Interpretation erfüllt $\neg sko(\neg F)$.
2. F ist falsifizierbar gdw. eine Herbrand-Interpretation falsifiziert $\neg sko(\neg F)$.
3. F ist erfüllbar gdw. eine Herbrand-Interpretation erfüllt $sko(F)$.
4. F ist unerfüllbar gdw. jede Herbrand-Interpretation falsifiziert $sko(F)$.
5. S ist erfüllbar gdw. $sko(S)$ ein Herbrand-Modell besitzt.
6. $F \models G$ gdw. jede Herbrand-Interpretation falsifiziert $sko(F \wedge \neg G)$.
7. $S \models G$ gdw. jede Herbrand-Interpretation falsifiziert $sko(S \cup \{\neg G\})$.

Dabei beziehen sich die genannten Herbrand-Interpretationen jeweils auf das Herbrand-Universum einer Erweiterung der ursprünglichen Sprache um endlich viele Funktionssymbole. ■

Aus Satz 3.8.21 folgt ein klassisches Ergebnis.

Satz 3.8.24 (Satz von Löwenheim und Skolem). Jede erfüllbare, geschlossene Formel einer Sprache $\mathcal{L}$ der PL1S ist erfüllbar durch eine $\mathcal{L}$-Interpretation, deren Universum höchstens abzählbar ist.

Beweis: Sei $\mathcal{L}$ eine Sprache der PL1S, und sei F eine geschlossene $\mathcal{L}$-Formel, die erfüllbar ist. Nach den Regeln zur Auflösung des Äquivalenzjunktors (Satz 3.3.7 (8)) gibt es eine geschlossene $\mathcal{L}$-Formel F' mit $F \models\mid F'$, in der jedes Quantorvorkommen eine eindeutige Polarität hat.

Nach Satz 3.8.21 gibt es eine Erweiterung $\mathcal{L}^*$ von $\mathcal{L}$ um endlich viele Funktionssymbole und eine geschlossene universelle $\mathcal{L}^*$-Formel F^*, so dass $F^* \models F'$ und F^* erfüllbar ist (ergibt sich unmittelbar aus Teil (2) von Satz 3.8.21). Nach Satz 3.8.14, da F^* universell und erfüllbar ist, wird F^* von einer Herbrand-Interpretation M erfüllt. Da $F^* \models F$, erfüllt M auch F.

Das Ergebnis folgt daraus, dass nach Definition 2.3.1 ein Herbrand-Universum höchstens abzählbar ist. ■

Folgesatz 3.8.25. Jede erfüllbare, endliche Menge von geschlossenen Formeln einer Sprache der PL1S ist erfüllbar durch eine Interpretation mit höchstens abzählbarem Universum. ■

Bemerkungen: Es gibt weitere Sätze, die den obigen ähneln. In Abschnitt 3.6 wurde bereits der „aufsteigende Satz von Löwenheim und Skolem“ erwähnt:

> Jede Menge von geschlossenen Formeln einer Sprache erster Stufe mit Gleichheit, die ein Modell mit unendlichem Universum besitzt, besitzt auch ein Modell, dessen Universum mindestens so groß wie jede beliebige Menge ist.

Satz 3.8.25 wird durch den „absteigenden Satz von Löwenheim und Skolem“ verallgemeinert:

> Jede erfüllbare Menge von geschlossenen Formeln einer Sprache $\mathcal{L}_{\doteq}$ mit Gleichheit besitzt ein Modell, dessen Universum nicht größer als die Menge der $\mathcal{L}_{\doteq}$-Formeln ist.

Aus diesem Satz folgt, dass die Überabzählbarkeit in der PL1S nicht ausdrückbar ist. Dies hat schwerwiegende Folgen in der Mathematik, in der überabzählbare Mengen wie z.B. $\mathbb{R}$ eine zentrale Rolle spielen. Für die praktische Informatik ist diese Einschränkung aber nicht so bedeutsam, weil die Informatik vorwiegend höchstens abzählbare Mengen untersucht. ∎

3.9 Exkurs: Relationale Datenbanken

Eine relationale Datenbank besteht aus einem Datenbank-Schema und den eigentlichen Daten. Das Schema beschreibt die syntaktische Form und sonstige Bedingungen, die von den Daten eingehalten werden müssen. Mit Begriffen der Logik kann eine relationale Datenbank formalisiert werden als Tripel $D = (\mathcal{L}, \mathcal{C}, \mathcal{F})$ mit:

1. $\mathcal{L}$ ist eine Sprache der Prädikatenlogik erster Stufe, die keine Funktionssymbole außer Konstanten enthält.
2. $\mathcal{C}$ ist eine endliche Menge von geschlossenen, beschränkt quantifizierten $\mathcal{L}$-Formeln, *Integritätsbedingungen* genannt.
3. $\mathcal{F}$ ist eine endliche Menge von $\mathcal{L}$-Grundatomen, auch *Fakten* genannt.

Das Datenbank-Schema $(\mathcal{L}, \mathcal{C})$ ist also einfach eine Signatur zusammen mit Integritätsbedingungen. $\mathcal{F}$ ist die Menge der eigentlichen Daten. Die Teilmenge der Daten mit dem selben Relationssymbol p nennt man auch die Relation p.

Beispiel 3.9.1 (Relationale Datenbank). $D = (\mathcal{L}, \mathcal{C}, \mathcal{F})$ mit folgenden Definitionen ist eine relationale Datenbank:

$$\mathcal{L}: \quad Rel^3_{\mathcal{L}} = \{stud\} \qquad Rel^2_{\mathcal{L}} = \{fach\} \qquad Rel^n_{\mathcal{L}} = \emptyset \text{ für } n \neq 2, n \neq 3$$

$$\mathcal{C} = \left\{ \quad \forall x \forall y \forall z\, (stud(x,y,z) \Rightarrow \exists v\, fach(x,v)) \quad \right\}$$

$$\mathcal{F} = \left\{ \begin{array}{ll} stud(45001, Maier, Anna), & fach(45001, Informatik), \\ stud(45002, Müller, Bernd), & fach(45002, Mathematik), \\ stud(45003, Schulz, Clara), & fach(45003, Mathematik), \\ & fach(45003, Informatik) \end{array} \right\}$$

Diese Datenbank repräsentiert Matrikelnummer, Name, Vorname von eingeschriebenen Studenten in der Relation *stud* und die jeweiligen Studienfächer in der Relation *fach*. Die einzige Integritätsbedingung verlangt, dass jedem eingeschriebenen Studenten ein Studienfach zugeordnet sein muss. Für die Daten in $\mathcal{F}$ ist das der Fall. Das Studienfach muss aber nicht eindeutig sein. Beispielsweise enthält die Datenbank die Information, dass Clara Schulz sowohl Mathematik als auch Informatik studiert. ∎

Die Menge $Fun^0_{\mathcal{L}}$ der Konstanten der Sprache $\mathcal{L}$ ist in diesem Beispiel nicht angegeben. Es wäre auch kaum praktikabel, zum Beispiel alle in der Datenbank zulässigen Familiennamen a priori im Datenbank-Schema vorzugeben. Deshalb wird die Menge der Konstanten vom Datenbank-Schema normalerweise offengelassen und lediglich durch Wertebereiche wie *string* oder *integer* für die einzelnen Argumente der Relationen eingegrenzt.

An eine Datenbank können Anfragen gestellt werden. Zum Beispiel ist

$$?y\, \exists x \exists z (\mathit{stud}(x, y, z) \land \neg \mathit{fach}(x, \mathit{Mathematik}))$$

eine Anfrage zur Bestimmung der Namen aller in der obigen Datenbank gespeicherten Studenten, die nicht Mathematik studieren. Das Zeichen ? wird „select" ausgesprochen. Die einzige Antwort auf die Anfrage ist in diesem Fall Maier/y.

Eine Anfrage hat allgemein die Form $?x_1 \ldots ?x_n\, F$, wobei F eine $\mathcal{L}$-Formel ist mit $\mathit{fvar}(F) = \{x_1, \ldots, x_n\}$. Allerdings sind gewisse Formeln in Anfragen problematisch. Was soll man etwa als Antwort auf die Anfrage $?y\, \neg \exists x \exists z\, \mathit{stud}(x, y, z)$ erwarten? Die Namen der Personen, die nicht als eingeschriebene Studenten in der Datenbank gespeichert sind, sind ja unbekannt. Und die Elemente des Wertebereichs *string*, die von *Maier*, *Müller*, *Schulz* verschieden sind, wären schon deshalb keine brauchbare Antwort, weil es unendlich viele davon gibt. Um derartige Probleme zu vermeiden, schränkt man die Formeln syntaktisch ein.

Definition 3.9.2 (DB-Formel). Für eine relationale Datenbank $D = (\mathcal{L}, \mathcal{C}, \mathcal{F})$ sind die Datenbank-Formeln, kurz DB-Formeln, wie folgt induktiv definiert:

1. Jede atomare $\mathcal{L}$-Formel ist eine DB-Formel.
2. Sind G und G_1 und G_2 DB-Formeln, so sind auch $\neg G$ und $(G_1 \land G_2)$ und $(G_1 \lor G_2)$ DB-Formeln.
3. Ist A eine atomare DB-Formel mit $\{x_1, \ldots, x_n\} \subseteq \mathit{var}(A)$ und ist G eine DB-Formel, so sind auch $\forall x_1 \ldots \forall x_n\, (A \Rightarrow G)$ und $\exists x_1 \ldots \exists x_n\, A$ und $\exists x_1 \ldots \exists x_n\, (A \land G)$ DB-Formeln.

Eine *Integritätsbedingung* ist eine geschlossene DB-Formel. Eine *Anfrage* ist entweder eine geschlossene DB-Formel oder hat die Form $?x_1 \ldots ?x_n\, F$, derart dass $\exists x_1 \ldots \exists x_n\, F$ eine geschlossene DB-Formel ist. Anfragen, die geschlossene DB-Formeln sind, werden auch Ja/Nein-Anfragen genannt. ∎

Die nach dieser Definition erlaubten Gestalten von Anfragen entsprechen Konstrukten der relationalen Datenbank-Anfragesprache SQL. Eine Anfrage $?x\, p(\ldots)$ entspricht dem SQL-Konstrukt `SELECT` x `FROM` p. Eine Anfrage $?x\, (p(\ldots) \land G)$ entspricht dem SQL-Konstrukt `SELECT` x `FROM` p `WHERE` G. Allerdings beruht SQL auf einem Tupel-Kalkül, so dass die Variablen eine etwas andere Bedeutung haben.

Die oben definierten syntaktischen Einschränkungen für Anfragen und Integritätsbedingungen ähneln den beschränkt quantifizierten Formeln aus Abschnitt 2.7, sind aber enger als notwendig und für praktische Zwecke wünschenswert. Weniger restriktive Definitionen finden sich in der Literatur unter Bezeichnungen wie „sichere" Formeln, „erlaubte" Formeln, „bereichsbeschränkte" Formeln.

Definition 3.9.3 (Gültigkeitsbeziehung). Sei $D = (\mathcal{L}, \mathcal{C}, \mathcal{F})$ eine relationale Datenbank. Die Gültigkeitsbeziehung $D \approx F$ („in D gilt F") ist für eine Anfrage F wie folgt rekursiv definiert. Es bezeichne A_{grund} ein Grundatom, A eine atomare DB-Formel, und G, G_1, G_2 DB-Formeln.

$$
\begin{array}{lll}
D \approx A_{grund} & \text{gdw.} & \mathcal{F} \models A_{grund} \\
D \approx \neg G & \text{gdw.} & D \not\approx G \\
D \approx (G_1 \wedge G_2) & \text{gdw.} & D \approx G_1 \text{ und } D \approx G_2 \\
D \approx (G_1 \vee G_2) & \text{gdw.} & D \approx G_1 \text{ oder } D \approx G_2 \\
D \approx \forall x_1 \ldots \forall x_n\, (A \Rightarrow G) & \text{gdw.} & D \approx \neg\exists x_1 \ldots \exists x_n\, (A \wedge \neg G) \\
D \approx \exists x_1 \ldots \exists x_n\, A & \text{gdw.} & \text{es gibt Grundterme } c_1, \ldots, c_n \text{ mit} \\
 & & D \approx A[c_1/x_1] \ldots [c_n/x_n] \\
D \approx \exists x_1 \ldots \exists x_n\, (A \wedge G) & \text{gdw.} & \text{es gibt Grundterme } c_1, \ldots, c_n \text{ mit} \\
 & & D \approx (A \wedge G)[c_1/x_1] \ldots [c_n/x_n] \\
D \approx ?x_1 \ldots ?x_n\, G & \text{gdw.} & D \approx \exists x_1 \ldots \exists x_n\, G
\end{array}
$$

Im Fall $D \approx F$ mit $F = ?x_1 \ldots ?x_n G$ nennt man jede Menge von Paaren $\{c_1/x_1, \ldots, c_n/x_n\}$ mit $D \approx G[c_1/x_1] \ldots [c_n/x_n]$ eine *Antwort* zu der Anfrage F. Ist F eine Ja/Nein-Anfrage, also geschlossen, ist die Antwort „ja" falls $D \approx F$, andernfalls „nein". ■

Diese Definition erklärt die Gültigkeitsbeziehung nicht für DB-Formeln mit freien Variablen, sondern nur für geschlossene DB-Formeln sowie für die mit dem Zeichen ? gebildeten Anfragen. Letztere sind im wesentlichen auch geschlossene DB-Formeln, wobei ? wie ein Existenzquantor behandelt wird, für den zusätzlich die Antworten geliefert werden.

Die Gültigkeitsbeziehung $D \approx F$ erinnert rein äußerlich an die Folgerungsbeziehung, da auf der linken Seite mit D genaugenommen die Formelmenge $\mathcal{F}$ gemeint ist und rechts eine Formel steht. Aber die Unterschiede sind beträchtlich, vor allem im Hinblick auf die Behandlung der Negation.

Betrachten wir die obige Beispieldatenbank und die Anfrage $A = \mathit{fach}(45001, \mathit{Mathematik})$. Für dieses Grundatom A besteht weder die Folgerungsbeziehung $\mathcal{F} \models A$ noch die Folgerungsbeziehung $\mathcal{F} \models \neg A$, denn es gibt Modelle von $\mathcal{F}$, in denen Anna Maier nur Informatik studiert und auch Modelle von $\mathcal{F}$, in denen sie sowohl Informatik als auch Mathematik studiert. Vereinfacht gesagt: über das, was nicht in der Datenbank steht, weiß man mit der Folgerungsbeziehung nichts.

Da die Folgerungsbeziehung $\mathcal{F} \models A$ nicht besteht, besteht nach Definition 3.9.3 auch nicht die Gültigkeitsbeziehung $D \approx A$, also besteht die Gültigkeitsbeziehung $D \approx \neg A$. Mit anderen Worten, in der Datenbank gilt, dass Anna Maier nicht Mathematik studiert. Die Gültigkeitsbeziehung legt fest, dass von allem, was nicht in der Datenbank steht (und auch nicht daraus folgt), die Negation gilt.

Die Folgerungsbeziehung $\models$ besitzt die sogenannte Monotonie-Eigenschaft: wenn $S \models F$ und $S \subseteq S'$, dann auch $S' \models F$, wobei S, S' Formelmengen bezeichnen und F eine Formel. Die Gültigkeitsbeziehung $\approx$ besitzt diese Eigenschaft dagegen nicht. Wenn man sich aber auf negationsfreie Anfragen beschränkt, ist die Gültigkeitsbeziehung $\approx$ doch monoton. Da die Nichtmonotonie von $\approx$ nur durch die Behandlung der Negation verursacht wird, sagt man vereinfachend auch, diese Behandlung der Negation sei nichtmonoton, manchmal auch, die dadurch definierte Variante der Negation sei nichtmonoton.

Die Behandlung der Negation mit der nichtmonotonen Gültigkeitsbeziehung $\approx$ ist für viele Anwendungen nützlich und natürlich, manchmal sogar natürlicher als die Behandlung mit der Folgerungsbeziehung. Typische Beispiele dafür sind Fahrpläne. Normalerweise interpretiert man sie so, dass Verkehrsverbindungen, die nicht im Fahrplan stehen, auch nicht existieren.

Nach Definition 3.9.2 sind Integritätsbedingungen einfach Sonderfälle von Anfragen, nämlich geschlossene DB-Formeln. Die Komponente $\mathcal{C}$ einer relationalen Datenbank $D = (\mathcal{L}, \mathcal{C}, \mathcal{F})$ besteht aus solchen Integritätsbedingungen. Diese haben aber für die Definition der Gültigkeitsbeziehung keine Rolle gespielt. Sie dienen nur dazu, unter den rein syntaktisch möglichen Datenbankinhalten zwischen zulässigen und unzulässigen zu unterscheiden.

Definition 3.9.4 (konsistent [Datenbank]). Eine relationale Datenbank $D = (\mathcal{L}, \mathcal{C}, \mathcal{F})$ heißt *konsistent*, wenn $D \mathrel{|}\!\approx F$ für jedes $F \in \mathcal{C}$. ∎

Die obige Beispieldatenbank ist konsistent. Zum Nachweis, dass die einzige Integritätsbedingung in D gilt, wenden wir Definition 3.9.3 an:

$$
\begin{array}{ll}
 & D \mathrel{|}\!\approx \forall x \forall y \forall z\, (stud(x,y,z) \Rightarrow \exists v\, fach(x,v)) \\
\text{gdw.} & D \mathrel{|}\!\approx \neg\exists x \exists y \exists z\, (stud(x,y,z) \wedge \neg\exists v\, fach(x,v)) \\
\text{gdw.} & D \mathrel{|}\!\not\approx \exists x \exists y \exists z\, (stud(x,y,z) \wedge \neg\exists v\, fach(x,v)) \\
\text{gdw.} & \text{es gibt keine Grundterme } c_1,\ c_2,\ c_3 \text{ mit} \\
 & D \mathrel{|}\!\approx (stud(c_1,c_2,c_3) \wedge \neg\exists v\, fach(c_1,v))
\end{array}
$$

Diese letzte Konjunktion kann nur dann in D gelten, wenn ihre linke Teilformel in D gilt. Die einzigen Grundterme, für die das der Fall ist, können direkt der Relation *stud* entnommen werden: $stud(45001, Maier, Anna)$, $stud(45002, Müller, Bernd)$, $stud(45003, Schulz, Clara)$. Für alle anderen Grundterme ist mit Sicherheit ausgeschlossen, dass die Konjunktion in D gelten kann. Die Definition der DB-Formeln bewirkt, dass die Grundterme für quantifizierte Variablen immer auf diese Weise direkt aus einer Relation entnommen werden können. Wir können die obige Argumentation also fortsetzen:

$$
\begin{array}{ll}
\text{gdw.} & D \mathrel{|}\!\not\approx (stud(45001, Maier, Anna) \wedge \neg\exists v\, fach(45001,v)) \text{ und} \\
 & D \mathrel{|}\!\not\approx (stud(45002, Müller, Bernd) \wedge \neg\exists v\, fach(45002,v)) \text{ und} \\
 & D \mathrel{|}\!\not\approx (stud(45003, Schulz, Clara) \wedge \neg\exists v\, fach(45003,v)) \\
\text{gdw.} & D \mathrel{|}\!\not\approx \neg\exists v\, fach(45001,v) \text{ und} \\
 & D \mathrel{|}\!\not\approx \neg\exists v\, fach(45002,v) \text{ und} \\
 & D \mathrel{|}\!\not\approx \neg\exists v\, fach(45003,v) \\
\text{gdw.} & D \mathrel{|}\!\approx \exists v\, fach(45001,v) \text{ und} \\
 & D \mathrel{|}\!\approx \exists v\, fach(45002,v) \text{ und} \\
 & D \mathrel{|}\!\approx \exists v\, fach(45003,v)
\end{array}
$$

Jetzt können die erforderlichen Grundterme direkt aus der Relation *fach* in der Beispieldatenbank abgelesen werden, womit der Nachweis der Konsistenz sofort abgeschlossen wird.

Definition 3.9.5 (Intendiertes Modell [Datenbank]). Das *intendierte Modell* einer relationalen Datenbank $D = (\mathcal{L}, \mathcal{C}, \mathcal{F})$ ist die von der Menge $\mathcal{F}$ von Grundatomen repräsentierte Herbrand-Interpretation $\mathcal{H}_\mathcal{L}(\mathcal{F})$. ∎

Satz 3.9.6. Sei $D = (\mathcal{L}, \mathcal{C}, \mathcal{F})$ eine relationale Datenbank und F eine geschlossene DB-Formel. Es gilt $D \mathrel{|}\!\approx F$ gdw. $\mathcal{H}_\mathcal{L}(\mathcal{F}) \models F$.

Beweis: Strukturelle Induktion und Satz 3.8.10. ∎

Die Gültigkeitsbeziehung ist also keine Folgerungsbeziehung, sondern einfach die Modellbeziehung bezüglich des intendierten Modells der Datenbank. Dementsprechend werden die eigentlichen Daten einer Datenbank in moderneren theoretischen Arbeiten auch nicht als eine Menge von Grundatomen formalisiert, sondern als eine Interpretation: $D = (\mathcal{L}, \mathcal{C}, M)$ wobei M eine $\mathcal{L}$-Interpretation ist mit $M \models \mathcal{C}$. In diesem Rahmen besteht die Anfrageauswertung also darin, zu überprüfen, ob und für welche Grundterme eine Anfrage von dieser gegebenen Interpretation erfüllt wird.

Das intendierte Modell der Datenbank kann wie in diesem Abschnitt durch eine Menge von Grundatomen spezifiziert werden, aber auch durch andere Formeln. Damit sind insbesondere auch die sogenannten Sichten oder „Views" von relationalen Datenbanksystemen erfasst.

Betrachten wir wieder die obige Beispieldatenbank D, aber mit der Erweiterung, dass die Sprache $\mathcal{L}$ ein weiteres zweistelliges Relationssymbol *infostud* enthält. Als zusätzliche Komponente enthalte D die Formelmenge

$$\mathcal{V} = \Big\{ \forall y \forall z (\ \exists x [\mathit{stud}(x, y, z) \wedge \mathit{fach}(x, \mathit{Informatik})] \Rightarrow \mathit{infostud}(y, z)\ ) \Big\}$$

Intuitiv definiert diese Sicht eine abgeleitete Relation *infostud*, die Namen und Vornamen der Personen repräsentiert, die als eingeschriebene Informatikstudenten in der Datenbank gespeichert sind. Auf die Anfrage $?y?z\ \mathit{infostud}(y, z)$ werden also genau die Antworten $\{\mathit{Maier}/y, \mathit{Anna}/z\}$ und $\{\mathit{Schulz}/y, \mathit{Clara}/z\}$ erwartet.

Das intendierte Modell dieser erweiterten Datenbank ist die Herbrand-Interpretation $\mathcal{H}_{\mathcal{L}}(B)$ mit der kleinsten Menge B von Grundatomen, für die $\mathcal{H}_{\mathcal{L}}(B)$ die Formelmenge $\mathcal{F} \cup \mathcal{V}$ erfüllt. In diesem Beispiel ist $B = \mathcal{F} \cup \mathcal{F}'$ mit $\mathcal{F}' = \{\mathit{infostud}(\mathit{Maier}, \mathit{Anna}),\ \mathit{infostud}(\mathit{Schulz}, \mathit{Clara})\}$.

Man nennt $\mathcal{F}$ auch die *extensionalen* Daten und $\mathcal{F}'$ die *intensionalen* Daten der Datenbank D. Semantisch repräsentieren $\mathcal{F} \cup \mathcal{V}$ und $\mathcal{F} \cup \mathcal{F}'$ das selbe intendierte Modell. Das bedeutet, dass ein Datenbanksystem die Wahl hat, ob es tatsächlich $\mathcal{F}$ und $\mathcal{V}$ abspeichert und die Formeln in $\mathcal{V}$ bei jeder Anfrage geeignet auswertet, oder ob es in einem Vorverarbeitungsschritt die Menge $\mathcal{F}'$ von Grundatomen berechnet und abspeichert („materialisiert") und Anfragen mittels $\mathcal{F} \cup \mathcal{F}'$ beantwortet.

Üblicherweise ist für die Formeln in $\mathcal{V}$ die gleiche syntaktische Gestalt zugelassen wie für Integritätsbedingungen. Die obige Formel ist keine DB-Formel gemäß Definition 3.9.2, da die Teilformel links vom Implikationsjunktor keine atomare Formel ist. Die Definition der DB-Formeln kann aber entsprechend angepasst werden. Von der zugelassenen syntaktischen Gestalt der Formeln in $\mathcal{V}$ hängt auch ab, wie das intendierte Modell der Datenbank genau definiert ist.

3.10 Die natürlichen Zahlen und das Induktionsaxiom

In diesem Abschnitt betrachten wir die Menge $\mathbb{N}$ der natürlichen Zahlen mit der Zahl 0 und der Nachfolgerfunktion $1+ : \mathbb{N} \to \mathbb{N}$, $n \mapsto 1+(n) := 1 + n$ und folgenden Eigenschaften:

1. Kein Wert der Nachfolgerfunktion 1+ ist 0.
2. Die Nachfolgerfunktion 1+ ist injektiv.
3. Für jede Teilmenge X der Menge der natürlichen Zahlen gilt: Enthält X die Null und mit jedem Element auch dessen Nachfolger, dann ist X die gesamte Menge der natürlichen Zahlen.

Ob die intuitiven natürlichen Zahlen diese drei Eigenschaften tatsächlich besitzen, insbesondere die dritte, kann selbstverständlich hinterfragt werden. Diese philosophische Frage soll uns aber hier nicht weiter beschäftigen. Aus mathematischer Sicht werden sie einfach postuliert und sogar zur Grundlage einer Formalisierung der natürlichen Zahlen in der Prädikatenlogik gemacht. Für eine solche Formalisierung benutzen wir folgende Hilfsmittel.

Definition 3.10.1. Sei $\mathcal{L}_{o,s}$ die Sprache der Prädikatenlogik erster bzw. zweiter Stufe mit Gleichheit, die aus einer Konstanten o und einem einstelligen Funktionssymbol s besteht.

Es bezeichne $(\mathbb{N}, 0, 1+)$ die normale $\mathcal{L}_{o,s}$-Interpretation $(\mathbb{N}, Ab_{\mathbb{N}}, U_{\mathbb{N}})$ mit $Ab_{\mathbb{N}}(o) = 0$ und $Ab_{\mathbb{N}}(s) = 1+$ und $U_{\mathbb{N}}(x) = 0$ für jede Variable x. ■

In der Sprache $\mathcal{L}_{o,s}$ soll also die Konstante o für die natürliche Zahl 0 stehen und das Funktionssymbol s für die Nachfolgerfunktion 1+. Damit können die obigen drei Eigenschaften durch drei geschlossene $\mathcal{L}_{o,s}$-Formeln formalisiert werden.

Definition 3.10.2 (Peano'sches Axiomensystem).

$P1$: $\forall x\, \neg(s(x) \doteq o)$

$P2$: $\forall x \forall y\, (s(x) \doteq s(y) \;\Rightarrow\; x \doteq y)$

$P3$: $\forall X \left[\Big(X(o) \;\wedge\; \forall y\, [\, X(y) \Rightarrow X(s(y))\,] \Big) \quad\Rightarrow\quad \forall z\, X(z) \right]$
(Induktionsaxiom) ■

Da wir postulieren, dass die natürlichen Zahlen die genannten drei Eigenschaften besitzen, ist die $\mathcal{L}_{o,s}$-Interpretation $(\mathbb{N}, 0, 1+)$ ein Modell der Formelmenge $\{P1, P2, P3\}$. Wir werden sehen, dass alle anderen normalen Modelle des Peano'schen Axiomensystems zu diesem Modell isomorph sind, das heißt, im wesentlichen identisch.

Definition 3.10.3 (isomorphe Interpretationen). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster oder zweiter Stufe und seien $M_1 = (D_1, Ab_1, U_1)$ und $M_2 = (D_2, Ab_2, U_2)$ zwei $\mathcal{L}$-Interpretationen.

Eine Funktion $\mu : D_1 \to D_2$ heißt ein Isomorphismus von M_1 auf M_2, wenn:

1. $\mu : D_1 \to D_2$ ist eine Bijektion.
2. Für jede Konstante c gilt $\mu(c^{M_1}) = c^{M_2}$.
3. Für jedes n-stellige ($n \geq 1$) Funktionssymbol f und alle $d_1, \ldots, d_n \in D_1$ gilt $\mu(f^{M_1}(d_1, \ldots, d_n)) = f^{M_2}(\mu(d_1), \ldots, \mu(d_n))$.

4. Für jedes nullstellige Relationssymbol p gilt $p^{M_1} = p^{M_2}$.
5. Für jedes n-stellige ($n \geq 1$) Relationssymbol p und alle $d_1, \ldots, d_n \in D_1$ gilt $(d_1, \ldots, d_n) \in p^{M_1}$ gdw. $(\mu(d_1), \ldots, \mu(d_n)) \in p^{M_2}$.

Gibt es einen Isomorphismus von M_1 auf M_2, dann heißen M_1 und M_2 isomorph. ■

Genaugenommen müsste die Definition auch Bedingungen für die Umgebungen enthalten. Wir betrachten aber nur geschlossene Formeln, so dass die Umgebungen keine Rolle spielen.

Satz 3.10.4 (Isomorphiesatz). Sind M_1 und M_2 isomorphe $\mathcal{L}$-Interpretationen, so gilt für jede geschlossene $\mathcal{L}$-Formel F: $M_1 \models F$ gdw. $M_2 \models F$.

Beweis: strukturelle Induktion. ■

Bevor wir die Modelle des Peano'schen Axiomensystems untersuchen, betrachten wir die Formel $P3$ genauer. Sie ist eine Formel der Prädikatenlogik zweiter Stufe. Diese Formel wird als Induktionsaxiom bezeichnet, weil sie die Grundlage für Induktionsbeweise liefert.

Satz 3.10.5 (vollständige Induktion). Sei $M = (D, Ab, U)$ eine $\mathcal{L}_{o,s}$-Interpretation, die das Induktionsaxiom $P3$ erfüllt. Um zu zeigen, dass jedes Element von D eine Eigenschaft $\mathcal{E}$ besitzt, genügt es, zu zeigen:

1. **Induktionsbasis:** $Ab(o)$ besitzt die Eigenschaft $\mathcal{E}$.
2. **Induktionsschritt:** Für jedes $d \in D$, das die Eigenschaft $\mathcal{E}$ besitzt, besitzt auch $Ab(s)(d)$ die Eigenschaft $\mathcal{E}$.

Beweis: Sei $D_\mathcal{E}$ die Relation (das heißt, Teilmenge) derjenigen Elemente von D, die die Eigenschaft $\mathcal{E}$ besitzen. M erfüllt $P3$ insbesondere auch für den Fall, dass X mit $D_\mathcal{E}$ interpretiert wird, mit anderen Worten:

$$M[D_\mathcal{E}/X] \quad \models \quad \Big[\Big(X(o) \ \wedge \ \forall y \, [\, X(y) \Rightarrow X(s(y)) \,] \Big) \quad \Rightarrow \quad \forall z \, X(z) \Big] \tag{$\star$}$$

Wegen der Induktionsbasis gilt $Ab(o) \in D_\mathcal{E}$, also $M[D_\mathcal{E}/X] \models X(o)$. Wegen des Induktionsschritts gilt für jedes $d \in D$: wenn $d \in D_\mathcal{E}$ so $Ab(s)(d) \in D_\mathcal{E}$, das heißt, für jedes $d \in D$ gilt $M[D_\mathcal{E}/X][d/y] \models [\, X(y) \Rightarrow X(s(y)) \,]$, also $M[D_\mathcal{E}/X] \models \forall y \, [\, X(y) \Rightarrow X(s(y)) \,]$. Mit diesen beiden Ergebnissen gilt wegen ($\star$) auch $M[D_\mathcal{E}/X] \models \forall z \, X(z)$, für jedes $d \in D$ gilt damit $d \in D_\mathcal{E}$, also besitzt jedes $d \in D$ die Eigenschaft $\mathcal{E}$. ■

Dieses Beweisprinzip ist für jede $\mathcal{L}_{o,s}$-Interpretation anwendbar, die $P3$ erfüllt, also insbesondere auch für $(\mathbb{N}, 0, 1{+})$. Spezialisiert für diese Interpretation lautet die Induktionsbasis, dass 0 die Eigenschaft $\mathcal{E}$ hat und der Induktionsschritt, dass für jedes $n \in \mathbb{N}$ mit Eigenschaft $\mathcal{E}$ auch $1{+}(n)$ die Eigenschaft $\mathcal{E}$ hat. Das ist genau das übliche Beweisprinzip der vollständigen Induktion für die natürlichen Zahlen, das wir schon mehrfach benutzt haben.

Aus der Tatsache, dass $(\mathbb{N}, 0, 1{+})$ ein normales Modell der Formelmenge $\{P1, P2, P3\}$ ist, kann man weitere bekannte Eigenschaften der natürlichen Zahlen ableiten, die wir bisher schon benutzt haben. Dazu gehört zum Beispiel ein Prinzip der rekursiven Definition von Funktionen auf $\mathbb{N}$ oder auch, dass es zu jedem $n \in \mathbb{N}$ mit $n \neq 0$ ein $k \in \mathbb{N}$ gibt mit $n = 1{+}(k)$.

Das soll aber hier nicht im einzelnen durchgeführt werden. Wir werden derartige Eigenschaften der natürlichen Zahlen weiterhin, wie bisher, einfach voraussetzen und benutzen, ohne sie formal daraus abzuleiten, dass $(\mathbb{N}, 0, 1{+})$ das Peano'sche Axiomensystem erfüllt.

Satz 3.10.6 (Satz von Dedekind). Jedes normale Modell der Formelmenge $\{P1, P2, P3\}$ ist mit $(\mathbb{N}, 0, 1+)$ isomorph.

Beweis: Sei $M = (D, Ab, U)$ ein normales Modell von $\{P1, P2, P3\}$. Wir definieren eine Funktion $\mu : \mathbb{N} \to D$ rekursiv für alle natürlichen Zahlen:

$$\begin{array}{lcl} \mu(0) & := & Ab(o) \\ \mu(1+(n)) & := & Ab(s)(\mu(n)) \quad \text{für alle } n \in \mathbb{N} \end{array}$$

Diese Funktion erfüllt nach Konstruktion die Forderungen 2. und 3. von Definition 3.10.3. Da $\doteq$ das einzige Relationssymbol in $\mathcal{L}_{o,s}$ ist, gilt trivialerweise auch Forderung 4. Die Forderung 5. reduziert sich wegen der Normalität beider Modelle darauf, dass für alle $m, n \in \mathbb{N}$ gilt $m = n$ gdw. $\mu(m) = \mu(n)$. Das folgt unmittelbar, wenn die Funktion auch Forderung 1. erfüllt, also eine Bijektion ist, was noch zu zeigen bleibt.

Für die Injektivität von μ zeigen wir durch vollständige Induktion, dass jede natürliche Zahl n folgende Eigenschaft besitzt:

$E(n)$: Für alle $m \in \mathbb{N}$ gilt, wenn $m \neq n$, dann $\mu(m) \neq \mu(n)$.

Induktionsbasis $n = 0$: Sei $m \neq 0$. Dann ist $m = 1+(k)$ für ein $k \in \mathbb{N}$. Also ist $\mu(m) = \mu(1+(k)) = Ab(s)(\mu(k))$. Da M das Axiom $P1$ erfüllt, gilt $Ab(s)(\mu(k)) \neq Ab(o)$. Wegen $Ab(o) = \mu(0)$ gilt also $\mu(m) \neq \mu(0)$.

Induktionsschritt $n \to 1+(n)$: Es gelte $E(n)$. Sei $m \neq 1+(n)$. Falls $m = 0$, wird wie im Basisfall gezeigt, dass $\mu(m) \neq \mu(1+(n))$. Andernfalls ist $m = 1+(k)$ für ein $k \in \mathbb{N}$, also $1+(k) \neq 1+(n)$ und damit $k \neq n$. Nach Induktionsannahme gilt dann $\mu(k) \neq \mu(n)$. Da M das Axiom $P2$ erfüllt, ist $Ab(s)$ injektiv, so dass auch $Ab(s)(\mu(k)) \neq Ab(s)(\mu(n))$ gilt. Anwendung der Definition von μ auf beiden Seiten ergibt $\mu(1+(k)) \neq \mu(1+(n))$, das heißt, $\mu(m) \neq \mu(1+(n))$.

Für die Surjektivität von μ nutzen wir aus, dass M das Induktionsaxiom $P3$ erfüllt. Wir zeigen durch vollständige Induktion die Surjektivität von μ, dass also jedes Element von D zum Bild von μ gehört.

Induktionsbasis $Ab(o)$: Da $\mu(0) = Ab(o)$, gehört $Ab(o)$ zum Bild von μ.

Induktionsschritt $d \to Ab(s)(d)$: Sei d im Bild von μ, das heißt, es gibt ein $n \in \mathbb{N}$ mit $\mu(n) = d$. Nach Definition ist $\mu(1+(n)) = Ab(s)(\mu(n))$, also $\mu(1+(n)) = Ab(s)(d)$, das heißt, $Ab(s)(d)$ gehört zum Bild von μ. ∎

Das Peano'sche Axiomensystem hat also die Eigenschaft, dass alle seine normalen Modelle isomorph zu $(\mathbb{N}, 0, 1+)$ sind. Unter Anwendung des Endlichkeitssatzes kann gezeigt werden, dass keine Menge von geschlossenen $\mathcal{L}_{o,s}$-Formeln der Prädikatenlogik erster Stufe diese Eigenschaft hat. Das bedeutet, dass das Induktionsaxiom $P3$ nicht in der Prädikatenlogik erster Stufe ausdrückbar ist.

Da der Endlichkeitssatz für die Prädikatenlogik erster Stufe erst in Kapitel 4 bewiesen wird, wird auch die Nichtausdrückbarkeit des Induktionsaxioms erst dort behandelt.

3.11 Exkurs: Semantik von Modal- und Temporallogiken

Modallogik. Um die Bedeutung von modallogischen Formeln festzulegen, müssen neben den Funktions- und Relationssymbolen und den klassischen Junktoren und Quantoren auch die Modaljunktoren $\Box$ und $\Diamond$ berücksichtigt werden.

Eine Modalinterpretation M für eine Sprache $\mathcal{L}$ der Modallogik erster Stufe basiert auf einer Menge W von „möglichen Welten". Zu jeder Welt $w \in W$ gehört eine klassisch-logische $\mathcal{L}$-Interpretation M_w – also eine Interpretation für die Sprache $\mathcal{L}$ im Sinne von Definition 3.5.4, ohne Berücksichtigung von Modaljunktoren. Die verschiedenen Welten mit ihren Interpretationen können zum Beispiel verschiedene Sichten über die Wirklichkeit repräsentieren.

Man legt zunächst fest, was es heißen soll, dass die Modalinterpretation M eine Formel F in einer Welt $w \in W$ erfüllt. Die Notation dafür ist:

$$M \models_w F$$

Ist F eine Formel, in der kein Modaljunktor vorkommt, so soll $M \models_w F$ einfach bedeuten, dass $M_w \models F$ im Sinne von Definition 3.5.7 gilt. Die Beziehung $M \models_w F$ ist in diesem Fall also die klassische Modellbeziehung für die Interpretation, die zur Welt w gehört.

Hat F die Gestalt $\Box G$, wird auf eine „Erreichbarkeitsrelation" E über W zurückgegriffen.

$$M \models_w \Box G$$

soll gelten, wenn $M \models_v G$ für alle Welten v gilt, die aus w erreichbar sind, d.h., für alle v mit $(w, v) \in E$.

$$M \models_w \Diamond G$$

soll gelten, wenn $M \models_v G$ für mindestens eine Welt v gilt, die aus w erreichbar ist.

Die Junktoren $\Box$ und $\Diamond$ ermöglichen also eine Art Quantifizierung über die erreichbaren Welten. Die sonstigen Symbole werden wie in der klassischen Logik interpretiert.

Eine Modalinterpretation M erfüllt eine Formel F, geschrieben wie üblich $M \models F$, wenn die Formel in jeder Welt von M erfüllt ist, d.h., wenn $M \models_w F$ für alle $w \in W$.

Um gewisse technische Schwierigkeiten zu vermeiden, wird oft – aber nicht immer! – verlangt, dass die Universen der klassisch-logischen Interpretationen M_w bezüglich der Erreichbarkeitsrelation monoton sind, d.h.,

$$\text{wenn } (w_1, w_2) \in E, \text{ dann } \mathit{Univ}(M_{w_1}) \subseteq \mathit{Univ}(M_{w_2})$$

und dass die Konstanten in zwei klassisch-logischen Interpretationen gleich interpretiert werden, wenn die eine aus der anderen erreichbar ist, d.h.,

$$\text{wenn } c \text{ eine Konstante ist und } (w_1, w_2) \in E, \text{ dann } c^{M_{w_1}} = c^{M_{w_2}}$$

Man sagt dann, dass die Konstanten in der Modalinterpretation *rigide* sind. Wir werden im folgenden beide Eigenschaften voraussetzen.

Definition 3.11.1 (Interpretation [Modallogik]). Sei $\mathcal{L}$ eine Sprache der Modallogik erster Stufe. Eine $\mathcal{L}$-Modalinterpretation ist ein Tripel $M = (W, E, \{M_w\}_{w \in W})$ mit:

1. W ist eine nichtleere Menge, die Menge der Welten.
2. E ist eine zweistellige Relation über W, d.h., $E \subseteq W \times W$.
3. Für jedes $w \in W$ ist M_w eine $\mathcal{L}$-Interpretation gemäß Definition 3.5.4.
4. Wenn $(w_1, w_2) \in E$, dann $\mathit{Univ}(M_{w_1}) \subseteq \mathit{Univ}(M_{w_2})$.
5. Wenn $(w_1, w_2) \in E$ und c eine Konstante ist, dann $c^{M_{w_1}} = c^{M_{w_2}}$.

Ist $(W, E, \{M_w\}_{w \in W})$ eine Modalinterpretation, so wird das Paar (W, E), also der gerichtete Graph der Welten mit der Erreichbarkeitsrelation, auch „Frame" genannt. ∎

Modalinterpretationen werden auch „Kripke-Interpretationen“ genannt, nach ihrem Erfinder Saul Kripke von der Princeton University. In Abgrenzung zur „Tarski-Semantik“ der Prädikatenlogik spricht man oft auch von der „Kripke-Semantik“ der Modallogik.

Definition 3.11.2. Sei $\mathcal{L}$ eine Sprache der Modallogik erster Stufe, $M = (W, E, \{M_w\}_{w \in W})$ eine $\mathcal{L}$-Modalinterpretation, und für jedes $w \in W$ sei $M_w = (D_w, Ab_w, U_w)$ gemäß Definition 3.5.4. Sei x eine Variable von $\mathcal{L}$ und $d \in \bigcup_{w \in W} D_w$ ein Element eines Universums von M.

$$M_w[d/x] \quad := \quad \begin{cases} M_w & \text{falls } d \notin D_w \\ (D_w, Ab_w, U_w[d/x]) \text{ gemäß Notation 3.5.6} & \text{falls } d \in D_w \end{cases}$$

$$M[d/x] \quad := \quad (W, E, \{M_w[d/x]\}_{w \in W})$$ ■

Damit ist $M[d/x]$ die $\mathcal{L}$-Modalinterpretation, die aus M entsteht, wenn die Umgebung jeder Welt, deren Universum d enthält, für die Variable x umdefiniert wird zu d. Die Welten, in deren Universen d nicht vorkommt, bleiben unverändert, und das ist der wesentliche Punkt dieser Definition. Für eine Welt w mit $d \notin D_w$ wäre nämlich $(D_w, Ab_w, U_w[d/x])$ keine $\mathcal{L}$-Interpretation nach Definition 3.5.4, und die Auswertung von Termen, die Modellbeziehung und ähnliches wären für dieses Gebilde nicht vollständig definiert.

So aber ist sichergestellt, dass $M[d/x]$ jeder Welt eine $\mathcal{L}$-Interpretation zuordnet und alle bisherigen Definitionen anwendbar sind.

Definition 3.11.3 (Modellbeziehung [Modallogik]). Sei $\mathcal{L}$ eine Sprache der Modallogik erster Stufe. Sei $M = (W, E, \{M_w\}_{w \in W})$ eine $\mathcal{L}$-Modalinterpretation, $w \in W$ und F eine $\mathcal{L}$-Formel.

- Die Beziehung $M \models_w F$ ist rekursiv über den Aufbau von F definiert wie folgt:

$$\begin{array}{lll}
M \models_w A & \text{gdw.} & M_w \models A \qquad\qquad \text{für } A \text{ Atom}, \top \text{ oder } \bot \\
M \models_w \neg G & \text{gdw.} & M \models_w G \text{ nicht gilt} \\
M \models_w (G_1 \wedge G_2) & \text{gdw.} & M \models_w G_1 \text{ und } M \models_w G_2 \\
M \models_w (G_1 \vee G_2) & \text{gdw.} & M \models_w G_1 \text{ oder } M \models_w G_2 \\
M \models_w (G_1 \Rightarrow G_2) & \text{gdw.} & \text{wenn } M \models_w G_1, \text{ so } M \models_w G_2 \\
M \models_w (G_1 \Leftrightarrow G_2) & \text{gdw.} & \text{entweder } M \models_w G_1 \text{ und } M \models_w G_2, \\
 & & \text{oder weder } M \models_w G_1 \text{ noch } M \models_w G_2 \\
M \models_w \forall x\, G & \text{gdw.} & \text{für alle } d \in \mathit{Univ}(M_w) \text{ gilt } M[d/x] \models_w G \\
M \models_w \exists x\, G & \text{gdw.} & \text{es gibt } d \in \mathit{Univ}(M_w) \text{ mit } M[d/x] \models_w G \\
M \models_w \Box G & \text{gdw.} & \text{für alle } v \in W \text{ mit } (w, v) \in E \text{ gilt } M \models_v G \\
M \models_w \Diamond G & \text{gdw.} & \text{es gibt } v \in W \text{ mit } (w, v) \in E, \text{ so dass } M \models_v G
\end{array}$$

- Die Modellbeziehung $M \models F$ gilt genau dann, wenn für alle $w \in W$ die Beziehung $M \models_w F$ gilt. Man sagt dann auch, M erfüllt F oder M ist ein Modell von F. ■

Man muss also drei Beziehungen auseinanderhalten: $M_w \models F$ und $M \models_w F$ und $M \models F$.
$M_w \models F$ ist die Modellbeziehung der klassischen Prädikatenlogik erster Stufe nach Definition 3.5.7. Sie bezieht sich auf eine Welt w von M und ist nur erklärt, wenn in F keine Modaljunktoren vorkommen.

$M \models_w F$ bezieht sich ebenfalls auf eine Welt w von M, ist aber auch dann erklärt, wenn die Modaljunktoren $\Box$ und $\Diamond$ in F vorkommen. Die obige Definition von $\models_w$ unterscheidet sich von Definition 3.5.7 nur deshalb, weil die Fälle für die Modaljunktoren hinzukommen. Enthält eine Formel F keine Modaljunktoren, kommen die zusätzlichen Fälle „$M \models_w \Box G$“ und „$M \models_w \Diamond G$“ nicht zur Anwendung, und $M \models_w F$ gilt genau dann, wenn $M_w \models F$ gilt.

Die Modellbeziehung $M \models F$ schließlich ist für Formeln mit Modaljunktoren erklärt und unabhängig von einer Welt von M. Die Bedingung „für alle $w \in W \ldots$“ in der obigen Definition hat zur Folge, dass diese Beziehung nicht die gleichen Eigenschaften hat wie die Modellbeziehung der klassischen Prädikatenlogik, obwohl sie völlig gleich geschrieben wird. Wir gehen auf diese Unterschiede ein, nachdem wir zunächst ein Beispiel für die Anwendung der Definition betrachten.

Beispiel 3.11.4. Sei $M = (W, E, \{M_w\}_{w \in W})$ mit $W = \{w_0, w_1, w_2\}$ und Erreichbarkeitsrelation $E = \{(w_0, w_1),\ (w_0, w_2)\}$. Für die zugehörigen $\mathcal{L}$-Interpretationen gelte $M_{w_0} \models p$, $M_{w_0} \models q$ und $M_{w_1} \models p$, $M_{w_1} \not\models q$ und $M_{w_2} \not\models p$, $M_{w_2} \not\models q$.

Es gilt $M \models_{w_0} \Diamond p$ (wegen w_1) und $M \models_{w_0} \Diamond\neg p$ (wegen w_2) und $M \not\models_{w_0} \Box p$ (wegen w_2). Da aus w_1 gar keine Welt erreichbar ist, gilt $M \models_{w_1} \Box p$, aber $M \not\models_{w_1} \Diamond p$.

Ferner gilt weder $M \models p$ noch $M \models \neg p$, weil keine der beiden Formeln in allen Welten von M erfüllt ist, aber zum Beispiel gilt $M \models \Box\neg q$, weil $\Box\neg q$ in jeder Welt von M erfüllt ist: in w_0 wegen $M \models_{w_1} \neg q$ und $M \models_{w_2} \neg q$, und in w_1 und in w_2, weil aus diesen Welten gar keine Welt erreichbar ist. ■

Satz 3.11.5. Seien $\mathcal{L}$ eine Sprache der Modallogik erster Stufe. Sei $M = (W, E, \{M_w\}_{w \in W})$ eine $\mathcal{L}$-Modalinterpretation. Seien F und G geschlossene $\mathcal{L}$-Formeln.

1. Wenn $M \models \neg F$ dann $M \not\models F$.
   Die Gegenrichtung gilt im allgemeinen nicht.
2. Wenn $M \models F$ oder $M \models G$ dann $M \models (F \vee G)$.
   Die Gegenrichtung gilt im allgemeinen nicht.
3. $M \models F$ und $M \models G$ gdw. $M \models (F \wedge G)$.

Beweis:

1. Es gelte $M \models \neg F$. Nach Definition 3.11.3 gilt für alle $w \in W$ die Beziehung $M \models_w \neg F$. Das heißt nach Definition, dass für alle $w \in W$ die Beziehung $M \models_w F$ nicht gilt. Da W nicht leer ist, gibt es somit mindestens ein $w \in W$ für das $M \models_w F$ nicht gilt. Also gilt nicht für alle $w \in W$ die Beziehung $M \models_w F$, das heißt nach Definition, dass $M \models F$ nicht gilt.

   Die Gegenrichtung wird durch Beispiel 3.11.4 widerlegt. Dort gilt $M \models p$ nicht, aber $M \models \neg p$ gilt auch nicht.

2.,3. Übung. ■

In der klassischen Prädikatenlogik erster Stufe gilt in allen diesen Fällen „gdw.“, und zwar einfach nach Definition der Modellbeziehung $\models$. Der Unterschied kommt daher, dass die Modellbeziehung $\models$ der Modallogik mit der Bedingung „für alle $w \in W \ldots$“ definiert ist. Man kann also Eigenschaften, die die Modellbeziehung in der Prädikatenlogik hat, in der Modallogik nicht einfach voraussetzen – manche übertragen sich, manche nicht.

Für eine Formel mit Modaljunktoren und Variablen hängt die Definition auf eine nicht ganz offensichtliche Weise von der Monotonie-Annahme ab. Betrachten wir die Beziehung $M \models_w \forall x \Box p(x)$. Sie gilt genau dann, wenn für jedes d aus dem Universum der Welt w die Beziehung $M[d/x] \models_w \Box p(x)$ gilt, also genau dann, wenn für jede aus w erreichbare Welt v gilt $M_v[d/x] \models p(x)$. Die Monotonie-Annahme stellt sicher, dass jedes dieser d im Universum jeder dieser Welten v vorkommt, so dass in allen Fällen die Umgebung tatsächlich der Variablen x den Wert d zuordnet. Diejenigen Welten, deren Umgebungen durch $[d/x]$ nicht beeinflusst werden, können nicht aus w erreicht werden und sind deshalb irrelevant.

Definition 3.11.6 (semantische Begriffe [Modallogik]). Sei $\mathcal{L}$ eine Sprache der Modallogik erster Stufe.

- Eine $\mathcal{L}$-Formel F heißt *allgemeingültig*, wenn sie von jeder $\mathcal{L}$-Modalinterpretation erfüllt wird, andernfalls *falsifizierbar*.
- Aus F *folgt* G, notiert $F \models G$, gdw. für jede Modalinterpretation M gilt: wenn $M \models F$ dann $M \models G$ (jedes Modell von F ist auch Modell von G). ■

Diese Definitionen sind völlig analog zu den entsprechenden Definitionen der klassischen Prädikatenlogik erster Stufe (Definition 3.5.8). Aber da die Modellbeziehung $\models$ andere Eigenschaften hat, hat auch die Folgerungsbeziehung andere Eigenschaften.

Satz 3.11.7. Sei $\mathcal{L}$ eine Sprache der Modallogik erster Stufe. Seien F und G geschlossene $\mathcal{L}$-Formeln.

Wenn $(F \Rightarrow G)$ allgemeingültig ist, dann $F \models G$.

Die Gegenrichtung gilt im allgemeinen nicht.

Beweis: Vorausgesetzt sei, dass $(F \Rightarrow G)$ allgemeingültig ist, das heißt, für jede Modalinterpretation M gelte $M \models (F \Rightarrow G)$.
Sei $M = (W, E, \{M_w\}_{w \in W})$ eine Modalinterpretation mit $M \models F$. Nach Voraussetzung gilt auch $M \models (F \Rightarrow G)$. Sei $w \in W$ beliebig, aber fest. Nach Definition gilt $M \models_w F$ und $M \models_w (F \Rightarrow G)$. Nach Definition 3.11.3 gilt dann auch $M \models_w G$. Da w beliebig ist, gilt für alle $w \in W$ die Beziehung $M \models_w G$, das heißt, $M \models G$.

Zur Widerlegung der Gegenrichtung betrachten wir eine Sprache $\mathcal{L}$ mit einem nullstelligen Relationssymbol p und die Formeln p für F und $\Box p$ für G.

Um zu zeigen, dass die Folgerungsbeziehung $p \models \Box p$ gilt, sei $M = (W, E, \{M_w\}_{w \in W})$ eine Modalinterpretation mit $M \models p$. Sei $w_1 \in W$. Weil $M \models p$ gilt, gilt $M \models_w p$ für alle $w \in W$, unter anderem für alle $w_2 \in W$, so dass $(w_1, w_2) \in E$. Also gilt $M \models_{w_1} \Box p$. Da w_1 beliebig ist, gilt $M \models \Box p$. Damit gilt die Folgerungsbeziehung $p \models \Box p$.

Es bleibt zu zeigen, dass $(p \Rightarrow \Box p)$ nicht allgemeingültig ist. Sei M die Modalinterpretation aus Beispiel 3.11.4. In diesem Beispiel gilt $M_{w_0} \models p$ und $M_{w_0} \not\models \Box p$, also $M \not\models_{w_0} (p \Rightarrow \Box p)$, also $M \not\models (p \Rightarrow \Box p)$. Damit ist $(p \Rightarrow \Box p)$ nicht allgemeingültig. ■

Im Fall der klassischen Prädikatenlogik erster Stufe gilt auch die Gegenrichtung (vergleiche Satz 3.5.9). Im Fall der Modallogik ist dagegen die Eigenschaft, dass zwischen zwei Formeln F und G die Folgerungsbeziehung $F \models G$ gilt, schwächer als die Eigenschaft, dass ihre Implikation $(F \Rightarrow G)$ allgemeingültig ist. Aus diesem Grund führen wir die logische Äquivalenz $F \models\!\!\!\!\!\dashv G$ gar nicht erst ein.

„Rechenregeln“ analog zu Satz 3.5.14 werden für die Modallogik üblicherweise nicht in der Form „$F \models\!\mid G$“ formuliert, sondern in der Form „$(F \Leftrightarrow G)$ ist allgemeingültig“. Wir zeigen exemplarisch einige dieser Rechenregeln für Formeln mit Modaljunktoren.

Satz 3.11.8. Sei $\mathcal{L}$ eine Sprache der Modallogik erster Stufe. Sei F eine geschlossene $\mathcal{L}$-Formel, $G[x]$ eine $\mathcal{L}$-Formel, in der nur die Variable x frei vorkommt.

1. $(\Box F \Leftrightarrow \neg\Diamond\neg F)$ ist allgemeingültig.
2. $(\Box F \Rightarrow \Diamond F)$ ist falsifizierbar, also nicht allgemeingültig.
3. $(\Box\forall x G[x] \Rightarrow \forall x \Box G[x])$ ist allgemeingültig.
4. $(\forall x \Box G[x] \Rightarrow \Box\forall x G[x])$ ist falsifizierbar, also nicht allgemeingültig.

Beweis:

1. Angenommen, die Formel sei nicht allgemeingültig. Dann gibt es eine Modalinterpretation $M = (W, E, \{M_w\}_{w\in W})$ und eine Welt $w \in W$, so dass einer der beiden folgenden Fälle gilt:

   1. Fall: $M \models_w \Box F$ und $M \not\models_w \neg\Diamond\neg F$. Nach Definition 3.11.3 gilt $M \models_w \Diamond\neg F$, und es gibt $v \in W$ mit $(w, v) \in E$, so dass $M \models_v \neg F$. Wegen $M \models_w \Box F$ gilt aber $M \models_v F$, Widerspruch.

   2. Fall: $M \models_w \neg\Diamond\neg F$ und $M \not\models_w \Box F$. Nach Definition 3.11.3 gilt nicht für alle $v \in W$ mit $(w, v) \in E$, dass $M \models_v F$, es gibt also ein solches v mit $M \not\models_v F$, das heißt, $M \models_v \neg F$. Wegen $M \models_w \neg\Diamond\neg F$ gilt $M \not\models_w \Diamond\neg F$, es gib also kein $v \in W$ mit $(w, v) \in E$, so dass $M \models_v \neg F$, Widerspruch.

2. Sei $M = (W, E, \{M_w\}_{w\in W})$ eine Modalinterpretation mit $W = \{w\}$ und $E = \emptyset$ und M_w beliebig. Da aus der einzigen Welt w gar keine Welt erreichbar ist, gilt $M \models_w \Box F$ und $M \not\models_w \Diamond F$, egal ob $M_w \models F$ gilt oder nicht. Also gilt $M \models_w (\Box F \Rightarrow \Diamond F)$ nicht, also auch nicht $M \models (\Box F \Rightarrow \Diamond F)$.

3. Angenommen, die Formel sei nicht allgemeingültig. Dann gibt es eine Modalinterpretation $M = (W, E, \{M_w\}_{w\in W})$ und eine Welt $w \in W$ mit $M \models_w \Box\forall x G[x]$ und $M \not\models_w \forall x \Box G[x]$.

   Nach Definition 3.11.3 gibt es $d \in \mathit{Univ}(w)$ mit $M[d/x] \not\models_w \Box G[x]$ und $v \in W$ mit $(w, v) \in E$, so dass $M[d/x] \not\models_v G[x]$. Wegen $M \models_w \Box\forall x G[x]$ gilt $M \models_v \forall x G[x]$, und wegen der Monotonie-Annahme ist $d \in \mathit{Univ}(v)$, also $M[d/x] \models_v G[x]$, Widerspruch.

4. Sei $M = (\{w_1, w_2\}, \{(w_1, w_2)\}, \{M_{w_1}, M_{w_2}\})$, $\mathit{Univ}(M_{w_1}) = \{1\}$, $\mathit{Univ}(M_{w_2}) = \{1, 2\}$. Außerdem sei $p^{M_{w_1}} = p^{M_{w_2}} = \{1\}$. Damit gilt $M[1/x] \models_{w_1} p(x)$ und $M[1/x] \models_{w_2} p(x)$ und $M[2/x] \not\models_{w_2} p(x)$.

   Es gilt $M \models_{w_1} \forall x \Box p(x)$, weil für das einzige Element $1 \in \mathit{Univ}(M_{w_1})$ und für die einzige erreichbare Welt w_2 gilt $M[1/x] \models_{w_2} p(x)$. Außerdem gilt $M \models_{w_2} \forall x \Box p(x)$, weil keine Welt aus w_2 erreichbar ist. Also gilt $M \models \forall x \Box p(x)$.
   Andererseits gilt $M[2/x] \not\models_{w_2} p(x)$, also $M \not\models_{w_2} \forall x\, p(x)$, also $M \not\models_{w_1} \Box\forall x\, p(x)$, und somit $M \not\models \Box\forall x\, p(x)$. ∎

Als die Modallogik zu Beginn des 20. Jahrhunderts eingeführt wurde, war zwar die Syntax festgelegt, aber es gab keine Modelltheorie dafür. Deshalb untersuchte man, welche Anforderungen an die Modallogik sich aus ihren Anwendungen ergeben und formalisierte diese Anforderungen durch Axiome. Für alethische Auslegungen ist es zum Beispiel sinnvoll, das Formelschema $(\Box F \Rightarrow F)$ als Axiom vorauszusetzen. Für deontische Auslegungen wäre dieses Axiom dagegen weniger sinnvoll (vergleiche Abschnitt 2.13).

Je nachdem, welche Axiome jeweils vorausgesetzt wurden, unterschied man verschiedene Varianten der Modallogik, für die sich historisch gewachsene Bezeichnungen eingebürgert haben: die Modallogik mit dem Axiom $(\Box F \Rightarrow F)$ heißt zum Beispiel T, die Modallogik mit den beiden Axiomen $(\Box F \Rightarrow F)$ und $(\Box F \Rightarrow \Box\Box F)$ heißt S_4. Auf diese Weise entstand ein schwer überschaubarer „Zoo“ verschiedener Varianten der Modallogik.

Die Entwicklung der Kripke-Semantik in den frühen 1960-er Jahren war ein wesentlicher Durchbruch, diesen „Zoo“ übersichtlicher zu machen. Es stellte sich nämlich heraus, dass die meisten Axiome einfachen Eigenschaften der Erreichbarkeitsrelation entsprechen.

Der folgende Satz zeigt zwei typische Zusammenhänge. Er verwendet den Begriff „Frame“ aus Definition 3.11.1, also den gerichteten Graph der Welten mit der Erreichbarkeitsrelation einer Modalinterpretation. Außerdem setzt er eine Sprache der Modallogik erster Stufe voraus, deren Signatur mindestens ein Relationssymbol enthält. Das bedeutet, dass es neben $\perp$ und $\top$ auch atomare $\mathcal{L}$-Formeln gibt, die sowohl erfüllbar als auch falsifizierbar sind.

Satz 3.11.9. Sei $\mathcal{L}$ eine Sprache der Modallogik erster Stufe, deren Signatur mindestens ein Relationssymbol enthält. Sei (W, E) ein Frame.

1. E ist reflexiv gdw. für jede $\mathcal{L}$-Modalinterpretation $M = (W, E, \{M_w\}_{w \in W})$ mit dem gegebenen Frame und für jede geschlossene $\mathcal{L}$-Formel F gilt $M \models (\Box F \Rightarrow F)$.
2. E ist transitiv gdw. für jede $\mathcal{L}$-Modalinterpretation $M = (W, E, \{M_w\}_{w \in W})$ mit dem gegebenen Frame und für jede geschlossene $\mathcal{L}$-Formel F gilt $M \models (\Box F \Rightarrow \Box\Box F)$.

Beweis:

1. „$\longrightarrow$“:
   Voraussetzung: E ist reflexiv.
   Sei $M = (W, E, \{M_w\}_{w \in W})$ eine $\mathcal{L}$-Modalinterpretation mit dem gegebenen Frame, und sei F eine geschlossene $\mathcal{L}$-Formel.

   Annahme: $M \not\models (\Box F \Rightarrow F)$.
   Dann gibt es eine Welt $w \in W$ mit $M \not\models_w (\Box F \Rightarrow F)$, das heißt, $M \models_w \Box F$ und $M \not\models_w F$. Wegen der Reflexivität von E ist $(w, w) \in E$, und wegen $M \models_w \Box F$ gilt $M \models_w F$. Widerspruch.

   „$\longleftarrow$“:
   Voraussetzung: für jede $\mathcal{L}$-Modalinterpretation $M = (W, E, \{M_w\}_{w \in W})$ mit dem gegebenen Frame und für jede geschlossene $\mathcal{L}$-Formel F gilt $M \models (\Box F \Rightarrow F)$.

   Annahme: E ist nicht reflexiv, das heißt, es gibt eine Welt $w_0 \in W$ mit $(w_0, w_0) \notin E$.

   Sei F eine atomare geschlossene $\mathcal{L}$-Formel. Sie ist also erfüllbar und falsifizierbar. Sei $M = (W, E, \{M_w\}_{w \in W})$ eine $\mathcal{L}$-Modalinterpretation mit $M_{w_0} \not\models F$ und $M_w \models F$ für alle $w \in W \setminus \{w_0\}$. Dann gilt insbesondere $M \models_w F$ für alle w mit $(w_0, w) \in E$. Das bedeutet, $M \models_{w_0} \Box F$. Wegen $M \not\models_{w_0} F$ gilt dann $M \not\models_{w_0} (\Box F \Rightarrow F)$, also $M \not\models (\Box F \Rightarrow F)$. Widerspruch.

2. „$\longrightarrow$“:
   Voraussetzung: E ist transitiv.
   Sei $M = (W, E, \{M_w\}_{w \in W})$ eine $\mathcal{L}$-Modalinterpretation mit dem gegebenen Frame, und sei F eine geschlossene $\mathcal{L}$-Formel.

   Annahme: $M \not\models (\Box F \Rightarrow \Box\Box F)$.
   Dann gibt es eine Welt $w_1 \in W$ mit $M \not\models_{w_1} (\Box F \Rightarrow \Box\Box F)$, das heißt, $M \models_{w_1} \Box F$ und $M \not\models_{w_1} \Box\Box F$. Also gibt es Welten $w_2, w_3 \in W$ mit $(w_1, w_2) \in E$ und $(w_2, w_3) \in E$ und $M \not\models_{w_3} F$. Wegen der Transitivität von E ist $(w_1, w_3) \in E$, und wegen $M \models_{w_1} \Box F$ gilt $M \models_{w_3} F$. Widerspruch.

   „$\longleftarrow$“:
   Voraussetzung: für jede $\mathcal{L}$-Modalinterpretation $M = (W, E, \{M_w\}_{w \in W})$ mit dem gegebenen Frame und für jede geschlossene $\mathcal{L}$-Formel F gilt $M \models (\Box F \Rightarrow \Box\Box F)$.

   Annahme: E ist nicht transitiv, das heißt, es gibt Welten $w_1, w_2, w_3 \in W$ mit $(w_1, w_2) \in E$ und $(w_2, w_3) \in E$ und $(w_1, w_3) \notin E$.

   Sei F eine atomare geschlossene $\mathcal{L}$-Formel. Sie ist also erfüllbar und falsifizierbar. Sei $M = (W, E, \{M_w\}_{w \in W})$ eine $\mathcal{L}$-Modalinterpretation mit $M_{w_3} \not\models F$ und $M_w \models F$ für alle $w \in W \setminus \{w_3\}$. Dann gilt insbesondere $M \models_w F$ für alle w mit $(w_1, w) \in E$. Das bedeutet, $M \models_{w_1} \Box F$. Wegen $(w_2, w_3) \in E$ und $M_{w_3} \not\models F$ gilt $M_{w_2} \not\models \Box F$ und wegen $(w_1, w_2) \in E$ auch $M_{w_1} \not\models \Box\Box F$. Damit gilt $M_{w_1} \not\models (\Box F \Rightarrow \Box\Box F)$, also $M \not\models (\Box F \Rightarrow \Box\Box F)$. Widerspruch. ∎

Wenn eine Modalinterpretation eine reflexive Erreichbarkeitsrelations hat, dann erfüllt sie jede Formel der Gestalt $(\Box F \Rightarrow F)$. Umgekehrt, wenn ein Frame (W, E) gegeben ist, für das jede Familie $\{M_w\}_{w \in W}$ von $\mathcal{L}$-Interpretationen, mit denen (W, E) zu einer Modalinterpretation ergänzt werden kann, jede Formel der Gestalt $(\Box F \Rightarrow F)$ erfüllt, dann ist die Erreichbarkeitsrelation im gegebenen Frame reflexiv.

Das bedeutet, dass die Klasse aller Modelle der Formeln der Gestalt $(\Box F \Rightarrow F)$ genau die Klasse der Modalinterpretationen mit reflexiver Erreichbarkeitsrelation ist. Man sagt, dass das Formelschema $(\Box F \Rightarrow F)$ diese Klasse von Modalinterpretationen eineindeutig charakterisiert.

Entsprechendes gilt für das Formelschema $(\Box F \Rightarrow \Box\Box F)$. Es charakterisiert eineindeutig die Klasse der Modalinterpretationen mit transitiver Erreichbarkeitsrelation.

Auch andere mögliche Eigenschaften der Erreichbarkeitsrelation lassen sich eineindeutig durch Formelschemata charakterisieren, zum Beispiel die Symmetrie durch $(F \Rightarrow \Box\Diamond F)$, die Linkstotalität durch $(\Box F \Rightarrow \Diamond F)$, die Rechtseindeutigkeit durch $(\Diamond F \Rightarrow \Box F)$. Es gibt viele Ergebnisse dieser Art.

Diese Zusammenhänge sind einer der Gründe für die große Beachtung, die die Semantik der möglichen Welten seit ihrer Einführung erfahren hat. Die Zusammenhänge gelten nämlich auch für die modale Aussagenlogik. In der Modallogik kann man also mit aussagenlogischen Mitteln Sachverhalte modellieren, die in der klassischen Logik über die Aussagenlogik hinausgehen und Prädikatenlogik erster Stufe erfordern.

Es gibt allerdings auch Eigenschaften der Erreichbarkeitsrelation, die sich nicht durch Formelschemata charakterisieren lassen, zum Beispiel die Irreflexivität oder die Antisymmetrie. Da viele dieser Eigenschaften in der klassischen Prädikatenlogik erster Stufe formalisiert werden können, liegt die Ausdrucksstärke der modalen Aussagenlogik zwischen der klassischen Aussagenlogik und der klassischen Prädikatenlogik erster Stufe.

Temporallogik. Der Semantik einer Temporallogik liegt eine Repräsentation der Zeit mit folgenden Merkmalen zu Grunde:

- Die Zeit ist diskret.
- Die Zeit hat einen Ursprung, d.h., einen initialen Zeitpunkt, dem alle anderen Zeitpunkte folgen und der selbst keinem Zeitpunkt folgt.
- Die Zeit ist in die Zukunft unbegrenzt, kann sich aber wiederholen.
- Die Zeit kann sich in alternative Zeitströme verzweigen.

Definition 3.11.10 (Zeitmodell). Sei Z eine Menge. Ihre Elemente heißen Zeitpunkte. Ein *diskretes Zeitmodell mit Ursprung* ist eine zweistellige Relation $\rho_Z \subseteq Z \times Z$ mit:

1. ρ_Z ist diskret: für je zwei Zeitpunkte $z, z' \in Z$ gibt es höchstens endlich viele ρ_Z-Pfade von z nach z'. Ein ρ_Z-Pfad von z nach z' ist eine endliche Sequenz $\langle z_0, \ldots, z_n \rangle$ mit:
   a) $z = z_0$ und $z_n = z'$
   b) $z_i \neq z_j$ für $i \neq j$
   c) $(z_{i-1}, z_i) \in \rho_Z$ für alle $i \in \{1, \ldots, n\}$
2. ρ_Z hat einen Ursprung: es gibt $z_{init} \in Z$, so dass
   a) es kein $z \in Z$ gibt mit $(z, z_{init}) \in \rho_Z$
   b) es für jedes $z \in Z$ einen ρ_Z-Pfad von z_{init} nach z gibt.
3. ρ_Z ist linkstotal: zu jedem $z \in Z$ gibt es $z' \in Z$ mit $(z, z') \in \rho_Z$.

Ist die Relation ρ_Z zusätzlich rechtseindeutig, d.h., gibt es für jedes $z \in Z$ nicht mehr als ein $z' \in Z$ mit $(z, z') \in \rho_Z$, so heißt das diskrete Zeitmodell mit Ursprung *linear*. Andernfalls heißt es *verzweigt*.

Im linearen Fall ist das Zeitmodell also eine Funktion $\rho_Z : Z \to Z$, die jeden Zeitpunkt auf einen Nachfolgerzeitpunkt abbildet. ∎

Bemerkung: ρ_Z kann Zyklen enthalten, auch im linearen Fall. Bedingung 1. schließt nur aus, dass es unendlich viele (zyklusfreie) ρ_Z-Pfade zwischen zwei Zeitpunkten gibt, wie zum Beispiel für $Z = \mathbb{Q}$ und $\rho_Z = <_{\mathbb{Q}}$. ∎

Eine Temporalinterpretation ist eine Modalinterpretation, deren Definition sich auf ein diskretes Zeitmodell mit Ursprung ρ_Z bezieht. Die reflexive und transitive Hülle ${\rho_Z}^\star$ des Zeitmodells ρ_Z bildet die Erreichbarkeitsrelation. Wegen des Temporaljunktors $\circ$ ist es günstig, ρ_Z statt ${\rho_Z}^\star$ zur Grundlage der Definition zu machen. Bei einer Temporalinterpretation spricht man von Zeitpunkten statt von Welten.

Definition 3.11.11 (Interpretation [Temporallogik]). Sei $\mathcal{L}$ eine Sprache der Temporallogik erster Stufe und ρ_Z ein diskretes Zeitmodell mit Ursprung über einer Menge Z von Zeitpunkten. Eine $\mathcal{L}$-Temporalinterpretation mit Zeitmodell ρ_Z ist eine Modalinterpretation $M = (W, E, \{M_w\}_{w \in W})$, so dass $W = Z$ und $E = {\rho_Z}^\star$ ist.

Ist ρ_Z linear, heißt die $\mathcal{L}$-Temporalinterpretation ebenfalls *linear,* andernfalls *verzweigt.* ∎

Definition 3.11.12 (Modellbeziehung [Temporallogik]). Sei $\mathcal{L}$ eine Sprache der Temporallogik erster Stufe. Sei $M = (Z, {\rho_Z}^\star, \{M_z\}_{z \in Z})$ eine $\mathcal{L}$-Temporalinterpretation mit Zeitmodell ρ_Z, sei $z \in Z$ und F eine $\mathcal{L}$-Temporalformel.

- Die Beziehung $M \models_z F$ ist rekursiv über den Aufbau von F definiert wie folgt:

$M \models_z A$	gdw.	$M_z \models A$	für A Atom, $\top$ oder $\bot$
$M \models_z \neg G$	gdw.	$M \models_z G$ nicht gilt	
$M \models_z (G_1 \wedge G_2)$	gdw.	$M \models_z G_1$ und $M \models_z G_2$	
$M \models_z (G_1 \vee G_2)$	gdw.	$M \models_z G_1$ oder $M \models_z G_2$	
$M \models_z (G_1 \Rightarrow G_2)$	gdw.	wenn $M \models_z G_1$, so $M \models_z G_2$	
$M \models_z (G_1 \Leftrightarrow G_2)$	gdw.	entweder $M \models_z G_1$ und $M \models_z G_2$, oder weder $M \models_z G_1$ noch $M \models_z G_2$	
$M \models_z \forall x\, G$	gdw.	für alle $d \in \mathit{Univ}(M_z)$ gilt $M[d/x] \models_z G$	
$M \models_z \exists x\, G$	gdw.	es gibt $d \in \mathit{Univ}(M_z)$ mit $M[d/x] \models_z G$	
$M \models_z \Box G$	gdw.	für alle $z' \in Z$ mit $(z, z') \in {\rho_Z}^\star$ gilt $M \models_{z'} G$	
$M \models_z \Diamond G$	gdw.	es gibt $z' \in Z$ mit $(z, z') \in {\rho_Z}^\star$, so dass $M \models_{z'} G$	
$M \models_z \circ G$	gdw.	für alle $z' \in Z$ mit $(z, z') \in \rho_Z$ gilt $M \models_{z'} G$	

- Die Modellbeziehung $M \models F$ gilt genau dann, wenn für alle $z \in Z$ die Beziehung $M \models_z F$ gilt. Man sagt dann auch, M erfüllt F oder M ist ein Modell von F. ■

Bemerkung: Der Fall „$M \models_z \circ G$“ bezieht sich auf das Zeitmodell ρ_Z und nicht auf die Erreichbarkeitsrelation ${\rho_Z}^\star$, die die reflexive und transitive Hülle des Zeitmodells ist. ■

Bis auf den Fall „$M \models_z \circ G$“ ist Definition 3.11.12 identisch mit Definition 3.11.3, abgesehen natürlich von der Vorgabe der speziellen Erreichbarkeitsrelation.

Zur Definition der Semantik einer Temporallogik wurde ein ähnlicher Ansatz angewandt wie zur Definition der normalen Modelle. In beiden Fällen wurden aus allen Interpretationen einige ausgewählt, die in einem Fall das Gleichheitsrelationssymbol $\doteq$, im anderen Fall die Modaljunktoren $\Box$ und $\Diamond$ wunschgemäß interpretieren.

Die beiden Fälle unterscheiden sich ansonsten aber wesentlich. Im Falle der Gleichheit kann die Einschränkung auf die gewünschten Interpretationen auch durch Axiome erfolgen. Es gibt aber kein Axiomensystem, das Modalinterpretationen auf diejenigen einschränkt, die für Temporallogiken sinnvoll sind. Zwar kann man, wie wir gesehen haben, Reflexivität, Transitivität, Linkstotalität und einige andere Eigenschaften der Erreichbarkeitsrelation mit Modalformeln ausdrücken, aber zum Beispiel nicht die Existenz eines Ursprungs.

4 Beweistheorie

Dieses Kapitel beschäftigt sich damit, wie die im vorigen Kapitel eingeführten semantischen Begriffe algorithmisch behandelt werden können. Diese Frage erfordert zunächst eine Präzisierung des Beweisbegriffs. Danach werden je eine Beweismethode für die Aussagenlogik und für die Prädikatenlogik erster Stufe eingeführt und ihre Eigenschaften untersucht.

4.1 Was ist eine Beweismethode? Was ist ein Beweis?

Angenommen, für zwei aussagenlogische Formeln F und G soll überprüft werden, ob die Folgerungsbeziehung $F \models G$ besteht. Dazu muss man nach Definition für jede Interpretation überprüfen, ob sie, wenn sie F erfüllt, auch G erfüllt. Da alle aussagenlogischen Interpretationen durch die Zeilen der entsprechenden Wahrheitstafel repräsentiert sind, kann die Frage mit folgender Methode beantwortet werden: man prüft in der Wahrheitstafel für F und G, ob jede Zeile, die der Formel F den Wahrheitswert w zuordnet, auch der Formel G den Wahrheitswert w zuordnet.

Diese Methode mag ineffizient sein. Immerhin hat eine Wahrheitstafel für n Aussagensymbole 2^n Zeilen, das heißt, ihre Größe wächst exponentiell mit der Anzahl der Aussagensymbole. Jedoch zeigt die Methode, dass die aussagenlogische Folgerungsbeziehung überhaupt automatisch überprüft werden kann.

Was heißt dabei „automatisch"? Zur Klärung dieser Frage formulieren wir die Methode etwas genauer.

Algorithmus $\mathit{folgt}(F, G)$:

1. Baue mit den Aussagensymbolen, die in F oder in G vorkommen, die Wahrheitstafel für F und für G auf.
2. Für jede Zeile der Wahrheitstafel prüfe, ob sie entweder in Spalte F den Eintrag f enthält oder sowohl in Spalte F als auch in Spalte G den Eintrag w enthält.
3. Wenn das für jede Zeile der Fall ist, liefere das Ergebnis *ja*, andernfalls liefere das Ergebnis *nein*. ■

Dies ist ein Algorithmus in dem Sinn, dass er die charakteristischen Anforderungen an Algorithmen erfüllt. Erstens ist er endlich formalisierbar. Das heißt, er kann als endlich lange Zeichenreihe in einem geeigneten Formalismus beschrieben werden. Die obige Beschreibung ist zwar nicht formal, aber hinreichend detailliert, dass man sie zum Beispiel in den Formalismus einer Programmiersprache übersetzen kann. Zweitens ist die Ausführung des Algorithmus allein anhand seiner Beschreibung möglich und erfordert kein zusätzliches Wissen. Jeder der obigen Schritte kann durchgeführt werden, ohne dass man zum Beispiel wissen muss, welche Sachverhalte mit den Formeln F und G modelliert wurden. Drittens ist das Ergebnis der Ausführung deterministisch. Ob der Algorithmus für gegebene Formeln F und G das Ergebnis *ja* oder *nein* liefert, hängt nur von diesen Formeln ab und nicht von Zufällen. Anders ausgedrückt, wenn man den Algorithmus mehrmals mit den selben Formeln F und G ausführt, erhält man jedesmal das selbe Ergebnis.

Darüberhinaus hat der obige Algorithmus zwei wichtige Eigenschaften:

- *Korrektheit:* Wenn $\mathit{folgt}(F, G)$ das Ergebnis *ja* liefert, gilt $F \models G$.
  Wenn $\mathit{folgt}(F, G)$ das Ergebnis *nein* liefert, gilt $F \not\models G$.

- *Vollständigkeit:* Für Formeln F, G der Aussagenlogik gilt:
  Wenn $F \models G$ gilt, liefert *folgt*(F, G) nach endlich vielen Schritten ein Ergebnis.
  Wenn $F \not\models G$ gilt, liefert *folgt*(F, G) nach endlich vielen Schritten ein Ergebnis.

Man beachte, dass die Vollständigkeitseigenschaft nicht etwa von einer festen Obergrenze für die Anzahl der Schritte abhängt. Der wesentliche Grund für die Vollständigkeit ist, dass in zwei beliebigen aussagenlogischen Formeln F, G nur endlich viele Aussagensymbole vorkommen können. Damit ist die Wahrheitstafel, die in Schritt 1. aufgebaut wird, endlich groß, und für Schritt 2. gibt es insgesamt nur endlich viele Zeilen zu prüfen.

In der Standardterminologie der Programmverifikation wird die obige Korrektheitseigenschaft als *partielle Korrektheit* bezeichnet, die Vollständigkeitseigenschaft als *Terminierung*. Der Begriff „Korrektheit" ohne Adjektiv steht meist als Abkürzung für *totale Korrektheit*, das heißt, partielle Korrektheit zusammen mit Terminierung.

Der Grund für den etwas abweichenden Sprachgebrauch in der Logik liegt darin, dass die Terminierung nicht immer möglich ist. Der Algorithmus *folgt*(F, G) terminiert für beliebige aussagenlogische Formeln. Für die Prädikatenlogik erster Stufe ist aber die Frage, ob $F \models G$ gilt, nicht entscheidbar, so dass es gar keinen entsprechenden Algorithmus geben kann, der für alle Formeln terminiert. Jedoch ist die Frage semi-entscheidbar, so dass folgende abgeschwächte Eigenschaft für einen Algorithmus möglich ist:

- *Vollständigkeit bezüglich* $\models$*:* Für Formeln F, G der Prädikatenlogik erster Stufe gilt:
  Wenn $F \models G$ gilt, liefert *folgt*(F, G) nach endlich vielen Schritten ein Ergebnis.

Wenn es einen Algorithmus mit dieser Eigenschaft gibt, ist seine Terminierung also nur für den Fall garantiert, dass $F \models G$ tatsächlich gilt. Wird er dagegen für zwei Formeln mit $F \not\models G$ aufgerufen, kann es sein, dass er ein Ergebnis liefert (wenn er außerdem korrekt ist, ist dieses Ergebnis *nein*), es kann aber auch sein, dass er nicht terminiert.

Dieses Phänomen ist auch für andere semantische Begriffe prinzipiell unvermeidbar. Zum Beispiel kann ein Algorithmus $\mathit{unerfüllbar}(F)$ für die Prädikatenlogik erster Stufe im günstigsten Fall korrekt sein und vollständig bezüglich der Unerfüllbarkeit: für jedes unerfüllbare F liefert er nach endlicher Zeit das Ergebnis *ja*, für manche erfüllbaren F liefert er das Ergebnis *nein*, für manche erfüllbaren F terminiert er nicht.

Man spricht also von einer „Beweismethode" für einen semantischen Begriff einer Logik, wenn es einen Algorithmus im obigen Sinn gibt, der korrekt ist und vollständig zumindest bezüglich des jeweiligen semantischen Begriffs.

Wenn der Begriff „Beweismethode" damit geklärt ist, was ist dann also ein „Beweis"? Eine einfache – aber passende! – Antwort ist: ein Protokoll („Trace" oder „Script") der Ausführung des Algorithmus oder die Datenstruktur, die von dem Algorithmus erzeugt und verwendet wird, oder zumindest hinreichende Ausschnitte davon, aus denen die Schritte des Algorithmus rekonstruierbar sind. Es gibt also keinen Beweisbegriff, der von einer Beweismethode unabhängig wäre.

Im Folgenden wird deshalb den Beweismethoden der Vorrang gegeben. Der zugehörige Beweisbegriff ergibt sich jeweils aus der Definition der Beweismethode. Für die oben beschriebene Methode der Wahrheitstafeln sind zum Beispiel die aufgebauten Wahrheitstafeln die Beweise.

4.2 Entscheidbarkeitsergebnisse für die Aussagenlogik

Die Tatsache, dass der Algorithmus $folgt(F, G)$ für aussagenlogische Formeln F und G korrekt und vollständig ist, bedeutet insbesondere, dass die Folgerungsbeziehung $\models$ für die Aussagenlogik entscheidbar ist.

Entsprechende auf Wahrheitstafeln aufbauende korrekte und vollständige Algorithmen sind für andere semantische Begriffe genauso offensichtlich wie für die Folgerungsbeziehung. Damit gilt:

Satz 4.2.1 (Entscheidbarkeit im endlichen Fall). Die Unerfüllbarkeit einer aussagenlogischen Formel oder einer endlichen Menge von aussagenlogischen Formeln ist entscheidbar. Ebenso die Erfüllbarkeit, Falsifizierbarkeit, Allgemeingültigkeit. ∎

Satz 4.2.2 (Semi-Entscheidbarkeit im unendlichen Fall). Die Unerfüllbarkeit einer unendlichen, aber aufzählbaren Menge von aussagenlogischen Formeln ist semi-entscheidbar.

Beweis: Sei S eine aufzählbare, unendliche Menge von aussagenlogischen Formeln. Es gibt also einen Algorithmus $Formel_S(i)$, der eine natürliche Zahl als Argument hat, für jedes Argument terminiert und eine Formel aus S als Ergebnis liefert, derart dass gilt $S = \{Formel_S(i) \mid i \in \mathbb{N}\}$.

Dieser Algorithmus ist eine Repräsentation der unendlichen Menge S, und deshalb ist er das Argument des zu konstruierenden Algorithmus für die Unerfüllbarkeit.

Algorithmus $unerfüllbar(Formel_S)$:

1. Initialisierung: $i := 0$ und $S_i := \emptyset$.
2. Schleife:
   a) $S_i := S_i \cup \{Formel_S(i)\}$
   b) Falls S_i unerfüllbar ist: Schleifenabbruch, liefere Ergebnis *ja*
   andernfalls $i := i + 1$ und Fortsetzung der Schleife.

Jedes in der Schleife gebildete S_i ist eine endliche Teilmenge von S. Die Unerfüllbarkeit von S_i in Schritt 2(b) ist also nach dem vorigen Satz entscheidbar. Sowohl Schritt 2(a) als auch Schritt 2(b) terminiert.

Wir betrachten zunächst den Fall, dass die durch $Formel_S$ repräsentierte unendliche Menge S erfüllbar ist. Da jede Teilmenge einer erfüllbaren Menge ebenfalls erfüllbar ist, wird die Schleife für kein i abgebrochen. Das heißt, $unerfüllbar(Formel_S)$ terminiert für erfüllbares S nicht.

Falls S unerfüllbar ist, gibt es nach dem Endlichkeitssatz für die Aussagenlogik (Satz 3.3.9) eine unerfüllbare endliche Teilmenge von S. Sei $\{Formel_S(i_1), \ldots, Formel_S(i_k)\}$ eine solche Teilmenge, und sei m das Maximum der Indizes $i_1, \ldots, i_k$. Im Schleifendurchgang mit $i = m$ enthält S_i also die unerfüllbare Teilmenge $\{Formel_S(i_1), \ldots, Formel_S(i_k)\}$ und ist damit selbst unerfüllbar. Damit wird spätestens für $i = m$ die Schleife beendet und das Ergebnis *ja* geliefert.

Da der Algorithmus das Ergebnis *nein* gar nicht liefert und das Ergebnis *ja* nur bei unerfüllbarem S, ist er korrekt. Außerdem ist er vollständig bezüglich der Unerfüllbarkeit, aber nicht bezüglich der Erfüllbarkeit. ∎

Dieser Semi-Entscheidungs-Algorithmus für die Unerfüllbarkeit ist selbstverständlich kein Beweis dafür, dass es keinen Entscheidungs-Algorithmus gibt, der auch für erfüllbares S terminiert (und zwar mit dem Ergebnis *nein*). Das folgende Beispiel liefert wenigstens eine anschauliche Plausibilitäts-Begründung dafür.

Angenommen, die erste Formel in der Aufzählung von S ist $\neg p$. Es könnte sein, dass an einer Stelle k in der Aufzählung auch die Formel p vorkommt. In diesem Fall kann man die Unerfüllbarkeit im k-ten Schritt feststellen, ohne die (unendlich vielen) weiteren Formeln betrachten zu müssen. Wenn man aber die Formeln bis zu einer Stelle k in der Aufzählung betrachtet hat, ohne dass p vorgekommen ist (und ohne dass ein anderer Grund für die Unerfüllbarkeit aufgetreten ist), weiß man nicht, ob p vielleicht später noch vorkommt. Das Nicht-Vorkommen der Formel p in der Aufzählung kann man an keiner Stelle feststellen, ohne die (unendlich vielen) weiteren Formeln zu betrachten. Alle Formeln der Aufzählung zu betrachten ist aber mit endlich vielen Schritten nicht möglich.

Es gibt hinreichende Kriterien für die Erfüllbarkeit in Spezialfällen. Man kann den Algorithmus so erweitern, dass er in derartigen Spezialfällen das korrekte Ergebnis *nein* liefert. Aber derartige Kriterien können nicht alle erfüllbaren Fälle abdecken.

4.3 Exkurs: Logikkalküle

In der mathematischen Logik werden Beweismethoden traditionell durch sogenannte *Kalküle* spezifiziert. Ein Kalkül definiert *Schlussregeln* (auch: Inferenzregeln), mit denen rein syntaktisch aus Formeln weitere Formeln gewonnen werden können. Die wohl bekannteste Schlussregel trägt den Namen *Modus Ponens* und kommt in verschiedenen Kalkülen vor:

$$\frac{F \qquad (F \Rightarrow G)}{G}$$

Diese Schreibweise bedeutet: zu zwei Formeln, die die Gestalt oberhalb des Strichs haben, darf eine zusätzliche Formel hinzugefügt werden, die die Gestalt unterhalb des Strichs hat.

Zum Beispiel kann diese Schlussregel folgendermaßen auf die Formeln in der Menge $S = \{(p \wedge q),\ ((p \wedge q) \Rightarrow (r \Rightarrow q)),\ r\}$ angewandt werden:

$$\dfrac{\dfrac{(p \wedge q) \qquad ((p \wedge q) \Rightarrow (r \Rightarrow q))}{(r \Rightarrow q)} \qquad r}{q}$$

Man sagt, dass damit aus der Formelmenge S die Formel q *hergeleitet* wird.

Man beachte, dass in der Schlussregel keine Formeln vorkommen, sondern sogenannte *Formelschemata*, das sind Ausdrücke einer Metasprache, die die Gestalt von Formeln beschreiben. Wir haben ja im ersten Schritt die Schlussregel für $F = (p \wedge q)$ und $G = (r \Rightarrow q)$ angewandt und im zweiten Schritt für $F = r$ und $G = q$. Mit beliebigen anderen Formeln für F und für G ist die Schlussregel ebenso anwendbar. Jedes Formelschema stellt also unendlich viele Formeln dar.

Es kann auch Schlussregeln wie die folgende geben:

$$\frac{}{(F \Rightarrow (G \Rightarrow F))}$$

In diesem Fall darf eine Formel der gegebenen Gestalt zu beliebigen anderen Formeln hinzugefügt werden. Man nennt eine solche Formel auch ein *logisches Axiom* des Kalküls. In der obigen Herleitung könnte damit zum Beispiel die Formel $(q \Rightarrow (p \Rightarrow q))$ hinzugefügt werden, so dass mit einer weiteren Anwendung der Modus-Ponens-Regel die Formel $(p \Rightarrow q)$ aus S hergeleitet werden kann.

Zu jedem Kalkül gibt es eine Herleitbarkeitsbeziehung: $S \vdash F$ bedeutet, dass mit den Schlussregeln des Kalküls in endlich vielen Schritten aus der Formelmenge S die Formel F hergeleitet werden kann. Die bereits für Algorithmen definierten Eigenschaften sind damit ganz einfach auf Kalküle übertragbar:

- *Korrektheit (des Kalküls):* wenn $S \vdash F$ dann $S \models F$.
- *Vollständigkeit (des Kalküls):* wenn $S \models F$ dann $S \vdash F$.

Die eigentliche Beweismethode bleibt bei einem Kalkül implizit. Man kann den Kalkül mit einer Grammatik und die Schlussregeln mit den Produktionen der Grammatik vergleichen. Durch die Grammatik sind die möglichen Syntaxbäume von Ausdrücken festgelegt. Es gibt verschiedene Verfahren, diese Syntaxbäume aufzubauen, aber jedes derartige Verfahren beruht letztlich auf einer systematischen Anwendung der Produktionen. Durch den Kalkül sind ganz analog die möglichen Strukturen von Herleitungen festgelegt (meistens handelt es sich dabei auch um Bäume). Dazu denkt man sich einen beliebigen geeigneten Algorithmus, der Herleitungen durch systematische Anwendung der Schlussregeln des Kalküls aufbaut.

Bei einer Beweismethode, die durch einen Kalkül spezifiziert ist, versteht man unter einem Beweis natürlich eine Herleitung, bzw. eine linearisierte Darstellung einer Herleitung.

Die bekanntesten Kalküle für die Prädikatenlogik erster Stufe stammen von David Hilbert (Königsberg 1862 – Göttingen 1943) und Gerhard Gentzen (Greifswald 1909 – Prag 1945). Inzwischen gibt es sehr viele Kalküle für diese und für andere Logiken, und viele Merkmale, in denen sie sich unterscheiden können.

Manche Kalküle überprüfen die Folgerungsbeziehung, andere die Allgemeingültigkeit oder Unerfüllbarkeit oder ähnliche semantische Eigenschaften von Formeln. Kalküle zur Überprüfung der Allgemeingültigkeit nennt man auch *positiv*, Kalküle zur Überprüfung der Unerfüllbarkeit *negativ*. Es liegt nahe, dass die logischen Axiome von positiven Kalkülen allgemeingültige Formeln sind (wie im obigen Beispiel $(q \Rightarrow (p \Rightarrow q))$), die von negativen Kalkülen dagegen unerfüllbare Formeln (zum Beispiel Formeln der Gestalt $(F \wedge \neg F)$).

Ein davon unabhängiger Unterschied betrifft die „Richtung“ der Herleitungen. Sogenannte *synthetische* Kalküle haben Schlussregeln, die ausgehend von den logischen Axiomen angewandt werden, bis die zu untersuchende Formel hergeleitet ist. Bei *analytischen* Kalkülen beginnt man mit der zu untersuchenden Formel und leitet daraus logische Axiome her.

Weitere Unterschiede sind: einige Kalküle benutzen viele Schlussregeln, andere wenige; manche haben möglichst anschauliche Schlussregeln, manche möglichst leicht implementierbare; meistens besteht eine Herleitung, so wie hier beschrieben, aus Formeln, aber gelegentlich auch aus komplexeren Gebilden, die Formeln als Bestandteile enthalten.

Es ist, wie gesagt, eine bewährte Tradition, Beweismethoden durch Kalküle zu spezifizieren. Inzwischen gibt es allerdings auch Beweismethoden, die sich ziemlich weit vom klassischen Kalkülbegriff entfernt haben.

4.4 Normalformen

Viele Beweismethoden sind der Einfachheit halber nur für Formeln von eingeschränkter syntaktischer Gestalt definiert. Formeln einer anderen Gestalt müssen zunächst „normalisiert“ werden, das heißt, in die entsprechende syntaktische Gestalt umgewandelt. Wir haben zum Beispiel in Kapitel 3 gesehen, dass jede Formel der Prädikatenlogik erster Stufe äquivalent zu einer Formel in Pränexform ist. Die Pränexform ist eine typische „Normalform“, die von einer Beweismethode vorausgesetzt werden kann.

Dieser Abschnitt behandelt einige dieser Normalformen, die für die zu besprechenden Beweismethoden von Bedeutung sind.

Definition 4.4.1 (Polyadische Junktoren). Für jedes $n \geq 0$ seien $F_1, \ldots, F_n$ Formeln einer Sprache der Aussagenlogik oder der Prädikatenlogik erster Stufe.

- $\wedge^\star(F_1, \ldots, F_n)$ bezeichnet die *polyadische Konjunktion* von $F_1, \ldots, F_n$.

  Die Modellbeziehung wird für jede Interpretation M darauf erweitert:
  $M \models \wedge^\star(F_1, \ldots, F_n)$ gdw. für alle i mit $1 \leq i \leq n$ gilt $M \models F_i$.

- $\vee^\star(F_1, \ldots, F_n)$ bezeichnet die *polyadische Disjunktion* von $F_1, \ldots, F_n$.

  Die Modellbeziehung wird für jede Interpretation M darauf erweitert:
  $M \models \vee^\star(F_1, \ldots, F_n)$ gdw. es gibt ein i mit $1 \leq i \leq n$ und $M \models F_i$. ■

Bemerkungen:

1. $\wedge^\star$ und $\vee^\star$ haben beliebig viele (aber endlich viele) Argumente und nicht etwa stets n Argumente für ein festes n (dafür wäre die Schreibweise $\wedge^n$ und $\vee^n$).

2. Bei null Argumenten bedeuten die obige Definitionen, dass für jede Interpretation M gilt $M \models \wedge^\star()$ und $M \not\models \vee^\star()$.

3. $$\begin{aligned} \wedge^\star() &\models\!\!\!\!\mid\ \top & \qquad \vee^\star() &\models\!\!\!\!\mid\ \bot \\ \wedge^\star(F_1) &\models\!\!\!\!\mid\ F_1 & \vee^\star(F_1) &\models\!\!\!\!\mid\ F_1 \\ \wedge^\star(F_1, F_2) &\models\!\!\!\!\mid\ (F_1 \wedge F_2) & \vee^\star(F_1, F_2) &\models\!\!\!\!\mid\ (F_1 \vee F_2) \\ \wedge^\star(F_1, F_2, F_3) &\models\!\!\!\!\mid\ (F_1 \wedge (F_2 \wedge F_3)) & \vee^\star(F_1, F_2, F_3) &\models\!\!\!\!\mid\ (F_1 \vee (F_2 \vee F_3)) \end{aligned}$$

4. Man könnte also auch definieren
   $\wedge^\star(F_1, \ldots, F_n) := (F_1 \wedge \ldots \wedge F_n)$ und $\vee^\star(F_1, \ldots, F_n) := (F_1 \vee \ldots \vee F_n)$
   mit zum Beispiel rechtsassoziativer Klammerung. Damit wären aber die Fälle mit null und mit einem Argument nicht erklärt. ■

Das König'sche Unendlichkeitslemma

Dieser Einschub behandelt ein bekanntes Ergebnis über Bäume, das auch außerhalb der Logik nützlich ist, insbesondere für Terminierungsnachweise.

Definition 4.4.2 (endlicher Verzweigungsgrad). Wenn es in einem Baum einen Knoten gibt, der unendlich viele direkte Nachfolger hat, sagt man, der Baum hat unendlichen Verzweigungsgrad. Andernfalls hat er endlichen Verzweigungsgrad. ■

Wenn es eine Obergrenze n gibt, so dass jeder Knoten im Baum höchstens n direkte Nachfolger hat, dann hat der Baum endlichen Verzweigungsgrad. Aber er kann auch endlichen Verzweigungsgrad haben, ohne dass so eine Obergrenze existiert. Zum Beispiel kann die Wurzel zwei direkte Nachfolger haben, der erste davon drei direkte Nachfolger, der erste davon vier usw.

Satz 4.4.3 (König'sches Unendlichkeitslemma). Ein Baum mit unendlich vielen Knoten hat unendlichen Verzweigungsgrad oder einen unendlichen Ast.

Beweis: Sei T ein Baum mit unendlich vielen Knoten und endlichem Verzweigungsgrad. Wir müssen zeigen, dass T einen unendlichen Ast hat.

Sei K ein beliebiger Knoten in T. Wegen des endlichen Verzweigungsgrads hat er endlich viele direkte Nachfolger. Falls jeder davon Wurzel eines Teilbaums mit endlich vielen Knoten ist, dann ist auch K selbst Wurzel eines Teilbaums mit endlich vielen Knoten.

Durch Kontraposition, und weil K beliebig ist, erhalten wir: wenn ein Knoten in T Wurzel eines Teilbaums mit unendlich vielen Knoten ist, dann hat er mindestens einen direkten Nachfolger, der ebenfalls Wurzel eines Teilbaums mit unendlich vielen Knoten ist.

Sei K_0 die Wurzel von T, also die Wurzel eines Teilbaums mit unendlich vielen Knoten. K_0 hat mindestens einen direkten Nachfolger, der die Wurzel eines Teilbaums mit unendlich vielen Knoten ist, also sei K_1 ein solcher Nachfolger. K_1 hat wieder mindestens einen solchen Nachfolger, also sei K_2 einer davon, und so weiter. Auf diese Weise kann ein unendlicher Ast $K_0, K_1, K_2, \ldots$ konstruiert werden. ■

Es gibt ein gedankliches Spiel, das eine hübsche Illustration für die Anwendung des König'schen Unendlichkeitslemmas liefert. Gegeben sei ein Vorrat von Münzen, die mit natürlichen Zahlen beschriftet sind. Für jede Zahl seien unbegrenzt viele Münzen mit dieser Zahl verfügbar. Gegeben sei außerdem ein Sparschwein, das am Anfang genau eine Münze enthält.

Ein Spielzug besteht darin, eine Münze aus dem Sparschwein herauszunehmen (und der eigenen Geldbörse einzuverleiben) und dafür beliebig viele, aber endlich viele, Münzen aus dem Vorrat in das Sparschwein hineinzulegen. Diese Münzen können mit gleichen oder verschiedenen Zahlen beschriftet sein, aber jede dieser Zahlen muss echt kleiner sein als die Zahl auf der herausgenommenen Münze.

Das Spiel kann immer mit dem ersten Spielzug beendet werden, indem man die ursprüngliche Münze herausnimmt und null Münzen dafür hineinlegt. Offensichtlich kann man auf diese Weise von jedem Zustand aus das Spiel mit endlich vielen Spielzügen beenden. Weniger offensichtlich ist die Frage, ob man die Spielzüge so wählen kann, dass das Spiel nicht endet, dass man sich also unendlich oft aus dem Sparschwein bedienen kann, ohne dass es je leer wird. Man könnte ja zum Beispiel die erste Münze herausnehmen und dafür 10 neue hineinlegen; dann nimmt man eine davon wieder heraus und legt dafür 100 neue hinein; bei jedem Schritt verzehnfacht man die Anzahl der Münzen, die man hineinlegt.

Jeder mögliche Spielverlauf kann durch einen Baum dargestellt werden: die Wurzel ist die Münze, die am Anfang im Sparschwein ist, die direkten Nachfolger einer Münze sind die Münzen, durch die sie bei einem Spielzug ersetzt wird. Die Knoten des Baums repräsentieren also sämtliche Münzen, die bei diesem Spielverlauf jemals in das Sparschwein kommen.

Jeder dieser Bäume hat endlichen Verzweigungsgrad, da bei jedem Spielzug nur endlich viele neue Münzen in das Sparschwein kommen. Außerdem ist jeder Knoten außer der Wurzel mit einer kleineren Zahl beschriftet als sein Vorgänger. Also kann kein Ast eines solchen Baums länger sein als die Zahl, mit der die Wurzel beschriftet ist, das heißt, jeder Ast in einem solchen Baum ist endlich.

Nach dem König'schen Unendlichkeitslemma hat jeder dieser Bäume nur endlich viele Knoten. Also terminiert jeder mögliche Spielverlauf.

Das Spiel entspricht einem nichtdeterministischen Algorithmus mit sehr vielen Wahlmöglichkeiten, nicht einmal eine Obergrenze für den Verzweigungsgrad ist vorgegeben. Das König'sche Unendlichkeitslemma ermöglicht in solchen Fällen Ergebnisse der Art: „wie auch immer diese Wahlmöglichkeiten ausgenutzt werden, der Algorithmus terminiert in jedem Fall.“

Aussagenlogische Normalformen

In diesem Abschnitt ist mit der Bezeichnung „Konjunktion“ immer eine Formel der Gestalt $\wedge^\star(\dots)$ gemeint, mit „Disjunktion“ eine Formel der Gestalt $\vee^\star(\dots)$, einschließlich der Fälle mit null Argumenten.

Definition 4.4.4.

- Sei A eine atomare Formel der Aussagenlogik. Ein *Literal* hat eine der Gestalten A oder $\neg A$. Ein Literal der Gestalt A heißt *positiv*, ein Literal der Gestalt $\neg A$ *negativ*. Das *Komplement* eines positiven Literals A ist $\neg A$, das Komplement eines negativen Literals $\neg A$ ist A.
- Eine *Klausel* ist eine Disjunktion von Literalen.

  Eine aussagenlogische Formel heißt in *konjunktiver Normalform*, wenn sie eine Konjunktion von Klauseln ist.
- Eine Klausel in *Implikationsnormalform* hat die Gestalt $(R \Rightarrow K)$, wobei R (für *Rumpf*) eine Konjunktion und K (für *Kopf*) eine Disjunktion von positiven Literalen, also von atomaren Formeln, ist.

  Eine aussagenlogische Formel heißt in *Implikationsnormalform*, wenn sie eine Konjunktion von Klauseln in Implikationsnormalform ist. ∎

Ein Beispiel für eine Klausel ist $\vee^\star(p, \neg q, r, \neg s)$. Die Implikationsnormalform dieser Klausel ist $(\wedge^\star(q, s) \Rightarrow \vee^\star(p, r))$. Eine andere Klausel ist $\vee^\star()$. Für diese „leere Klausel“ ist auch die Notation $\Box$ verbreitet. Sie ist, wie bereits erwähnt, logisch äquivalent mit $\bot$.

Bemerkungen: In Kapitel 2 wurden $\top$ und $\bot$ als nullstellige Junktoren eingeführt und nicht als atomare Formeln. Die Formeln $\top$, $\bot$, $\neg\top$, $\neg\bot$ sind also keine Literale und können deshalb nicht in einer Klausel vorkommen.

Eine Klausel kann auch als Menge von Literalen definiert werden. Dann sind zum Beispiel die logisch äquivalenten Klauseln $\vee^\star(p, \neg q, p)$, $\vee^\star(p, p, \neg q)$ und $\vee^\star(p, \neg q)$ alle durch die selbe Menge $\{p, \neg q\}$ repräsentiert. Für Implementierungen ist es aber oft günstiger, solche Klauseln unterscheiden zu können.

Man kann natürlich ganz analog die disjunktive Normalform definieren als eine Disjunktion von Konjunktionen von Literalen. Diese Normalform spielt aber für die hier behandelten Beweismethoden keine Rolle. ∎

Definition 4.4.5 (Transformation in Implikationsnormalform). Die Implikationsnormalform $(\bigwedge^\star(R_1, \ldots, R_n) \Rightarrow \bigvee^\star(K_1, \ldots, K_m))$ einer Klausel $\bigvee^\star(L_1, \ldots, L_k)$ wird wie folgt gebildet: $K_1, \ldots, K_m$ ist die Sequenz von Atomen, die aus $L_1, \ldots, L_k$ durch Streichen aller negativen Literale entsteht. $R_1, \ldots, R_n$ ist die Sequenz von Atomen, die aus $L_1, \ldots, L_k$ durch Streichen aller positiven Literale und (danach) aller Negationsjunktoren entsteht. ∎

Sowohl $n = 0$ als auch $m = 0$ sind dabei möglich. Dass diese Transformation die logische Äquivalenz erhält, folgt unmittelbar aus den Definitionen.

Definition 4.4.6 (Umschreibungsregeln für Disjunktionen).

$$\frac{\bigvee^\star(\ldots\ \neg\neg G\ \ldots)}{\bigvee^\star(\ldots\ G\ \ldots)}$$

$$\frac{\bigvee^\star(\ldots\ \neg(G_1 \wedge G_2)\ \ldots)}{\bigvee^\star(\ldots\ \neg G_1, \neg G_2\ \ldots)} \qquad \frac{\bigvee^\star(\ldots\ (G_1 \wedge G_2)\ \ldots)}{\bigvee^\star(\ldots\ G_1\ \ldots),\ \bigvee^\star(\ldots\ G_2\ \ldots)}$$

$$\frac{\bigvee^\star(\ldots\ (G_1 \vee G_2)\ \ldots)}{\bigvee^\star(\ldots\ G_1, G_2\ \ldots)} \qquad \frac{\bigvee^\star(\ldots\ \neg(G_1 \vee G_2)\ \ldots)}{\bigvee^\star(\ldots\ \neg G_1\ \ldots),\ \bigvee^\star(\ldots\ \neg G_2\ \ldots)}$$

$$\frac{\bigvee^\star(\ldots\ (G_1 \Rightarrow G_2)\ \ldots)}{\bigvee^\star(\ldots\ \neg G_1, G_2\ \ldots)} \qquad \frac{\bigvee^\star(\ldots\ \neg(G_1 \Rightarrow G_2)\ \ldots)}{\bigvee^\star(\ldots\ G_1\ \ldots),\ \bigvee^\star(\ldots\ \neg G_2\ \ldots)}$$

$$\frac{\bigvee^\star(\ldots\ (G_1 \Leftrightarrow G_2)\ \ldots)}{\bigvee^\star(\ldots\ \neg G_1, G_2\ \ldots), \bigvee^\star(\ldots\ G_1, \neg G_2\ \ldots)} \qquad \frac{\bigvee^\star(\ldots\ \neg(G_1 \Leftrightarrow G_2)\ \ldots)}{\bigvee^\star(\ldots\ G_1, G_2\ \ldots), \bigvee^\star(\ldots\ \neg G_1, \neg G_2\ \ldots)}$$

$$\frac{\bigvee^\star(\ldots\ \bot\ \ldots)}{\bigvee^\star(\ldots\ \ \ldots)} \qquad \frac{\bigvee^\star(\ldots\ \neg\bot\ \ldots)}{}$$

$$\frac{\bigvee^\star(\ldots\ \top\ \ldots)}{} \qquad \frac{\bigvee^\star(\ldots\ \neg\top\ \ldots)}{\bigvee^\star(\ldots\ \ \ldots)}$$

∎

Diese Regeln dienen als Hilfsmittel zur Berechnung einer konjunktiven Normalform einer aussagenlogischen Formel F. Sie sind dafür vorgesehen, auf eine Disjunktion angewandt zu werden, die in einer Sequenz von Argumenten von $\bigwedge^\star$ vorkommt. Bei der Anwendung einer Umschreibungsregel wird eine Disjunktion durch eine Teilsequenz ersetzt, die aus einer einzigen Disjunktion bestehen kann (z.B. im obersten linken Fall) oder aus zwei Disjunktionen (z.B. im obersten rechten Fall) oder aus null Disjunktionen (z.B. im untersten linken Fall).

Definition 4.4.7 (Transformation in konjunktive Normalform).

Nichtdeterministischer Algorithmus *konjunktive_normalform*(F):

1. Initialisierung: $K := \bigwedge^\star(\bigvee^\star(F))$
2. Solange es in $K = \bigwedge^\star(D_1, \ldots, D_m)$ eine Disjunktion gibt, die keine Klausel ist:
   a) Wähle eine solche Disjunktion $D_j = \bigvee^\star(F_1, \ldots, F_n)$ aus K aus.
   b) Wähle eine Formel F_i aus D_j aus, die kein Literal ist.
   c) Bestimme die Umschreibungsregel für D_j gemäß Definition 4.4.6, die auf die Gestalt von F_i passt.
   d) Wende die Umschreibung auf D_j innerhalb von K an.
   (Dadurch ändert sich der Wert von K.)
3. Liefere K als Ergebnis.

∎

Zum Beispiel entstehen für $F = (p \vee (q \wedge (r \vee \bot)))$ der Reihe nach folgende Werte für K:

$$\begin{array}{l}
\bigwedge^\star\Big(\ \bigvee^\star[\ (p \vee (q \wedge (r \vee \bot)))\ ]\ \Big)\\
\bigwedge^\star\Big(\ \bigvee^\star[\ p,\ \ (q \wedge (r \vee \bot))\ ]\ \Big)\\
\bigwedge^\star\Big(\ \bigvee^\star[\ p,\ \ q\ ],\quad \bigvee^\star[\ p,\ \ (r \vee \bot)\ ]\ \Big)\\
\bigwedge^\star\Big(\ \bigvee^\star[\ p,\ \ q\ ],\quad \bigvee^\star[\ p,\ \ r,\ \ \bot\ ]\ \Big)\\
\bigwedge^\star\Big(\ \bigvee^\star[\ p,\ \ q\ ],\quad \bigvee^\star[\ p,\ \ r\ ]\ \Big)
\end{array}$$

In diesem Beispiel ist keine andere Folge von Werten für K möglich. In der letzten Zeile sind beide Disjunktionen Klauseln. Jede andere Zeile enthält genau eine Disjunktion, die keine Klausel ist, und jede davon enthält genau eine Formel, die kein Literal ist. Im allgemeinen kann es aber in beiden Fällen mehrere Möglichkeiten geben. Die Schritte 2(a) und 2(b) sind nichtdeterministisch. Dagegen ist die Umschreibungsregel in Schritt 2(c) eindeutig bestimmt, sobald F_i gewählt ist. Dies ergibt sich unmittelbar aus dem Eindeutigkeitssatz 2.1.7 (erweitert auf $\bigwedge^\star$ und $\bigvee^\star$).

Satz 4.4.8. Egal, wie die Wahl in den Schritten 2(a) und 2(b) getroffen wird, gilt für jede aussagenlogische Formel F:

1. *konjunktive_normalform*(F) terminiert.
2. *konjunktive_normalform*(F) ist in konjunktiver Normalform.
3. *konjunktive_normalform*$(F) \models\!\!\!\dashv F$.

Beweis:

1. Wir definieren nach dem Prinzip der strukturellen Rekursion eine Funktion rg, genannt Rang, die jeder aussagenlogischen Formel eine natürliche Zahl zuordnet:

   $$\begin{array}{lll}
   rg(A) & := & 0 & A \text{ atomare Formel}\\
   rg(\top) := rg(\bot) & := & 1 & \\
   rg(\neg G) & := & 1 + rg(G) & \\
   rg((G_1\ \theta\ G_2)) & := & 2 + rg(G_1) + rg(G_2) & \theta \text{ zweistelliger Junktor}
   \end{array}$$

   Intuitiv ist der Rang die Anzahl der Junktoren in einer Formel, wobei zweistellige Junktoren doppelt zählen. Wir erweitern die Funktion auf Disjunktionen wie folgt: $rg(\bigvee^\star(F_1, \dots, F_n)) := rg(F_1) + \cdots + rg(F_n)$.

   Man prüft leicht nach, dass für jede Umschreibungsregel in Definition 4.4.6 gilt, dass jede Disjunktion unterhalb des Strichs einen echt kleineren Rang hat als die Disjunktion oberhalb des Strichs.

   Die Konjunktion $K = \bigwedge^\star(\dots)$ enthält initial genau eine Disjunktion. Bei jedem Schleifendurchgang wird eine der Disjunktionen in K durch maximal zwei Disjunktionen mit echt kleinerem Rang ersetzt. Der so konstruierbare Baum aller Disjunktionen, die jemals in Schritt 2 umgeschrieben werden, hat endlichen Verzweigungsgrad und nur endliche Äste, also nach dem König'schen Unendlichkeitslemma endlich viele Knoten.

2. K ist initial eine Konjunktion von Disjunktionen und bleibt bei jedem Durchgang durch Schritt 2 eine Konjunktion von Disjunktionen.

   Nach Beendigung der Schleife (Schritt 2) ist jede Disjunktion in K eine Klausel (sonst wäre die Schleife noch nicht beendet), das heißt, K ist in konjunktiver Normalform.

3. Initial gilt $K = \bigwedge^\star(\bigvee^\star(F)) \models\!\mid F$. Man prüft leicht nach, dass die logische Äquivalenz bei jeder Anwendung einer Umschreibungsregel auf eine Disjunktion in K erhalten bleibt. ∎

Das Terminierungsargument ist analog zu dem Spiel mit den Münzen: $K = \bigwedge^\star(\ldots)$ entspricht dem Sparschwein, die Disjunktionen den Münzen.

Der Nichtdeterminismus beim Ablauf des Algorithmus hat auch einen Nichtdeterminismus beim Ergebnis zur Folge. Zum Beispiel kann *konjunktive_normalform*($((p \land q) \lor (r \land s))$) je nach der getroffenen Wahl in Schritt 2(a) entweder $\bigwedge^\star(\ \bigvee^\star(p,r),\ \bigvee^\star(p,s),\ \bigvee^\star(q,r),\ \bigvee^\star(q,s)\ )$ ergeben oder $\bigwedge^\star(\ \bigvee^\star(p,r),\ \bigvee^\star(q,r),\ \bigvee^\star(p,s),\ \bigvee^\star(q,s)\ )$. Dieser Unterschied in der Reihenfolge der Klauseln ist aber uninteressant.

Der obige Satz garantiert, dass jeder deterministische Algorithmus, der durch geeignete Festlegung der Auswahl in den Schritten 2(a) und 2(b) definiert werden kann, die relevanten Eigenschaften besitzt.

Man kann als Datenstruktur für das K im Algorithmus statt einer Konjunktion eine Menge wählen (also Mehrfachvorkommen von Disjunktionen vermeiden). Dann liefert der Algorithmus als Ergebnis eine endliche Menge von Klauseln.

Folgesatz 4.4.9 (Normalformensatz [Aussagenlogik]). Sei F eine aussagenlogische Formel und S eine endliche Menge von aussagenlogischen Formeln. Es gibt Algorithmen zur effektiven Berechnung:

1. Einer Formel F' in konjunktiver Normalform oder Implikationsnormalform mit $F \models\!\mid F'$.
2. Einer endlichen Menge S' von Klauseln oder von Klauseln in Implikationsnormalform mit $S \models\!\mid S'$. ∎

Normalformen für die Prädikatenlogik erster Stufe

Die Begriffe Literal und Klausel werden völlig analog zur Aussagenlogik definiert, nur dass die Grundlage jetzt atomare Formeln im Sinne der Prädikatenlogik statt der Aussagenlogik sind. Die konjunktive Normalform wird als spezielle Pränexform definiert. Zusätzlich gibt es Normalformen, die gar keine expliziten Quantoren enthalten.

Definition 4.4.10.

- Sei A eine atomare Formel der Prädikatenlogik erster Stufe. Ein *Literal* hat eine der Gestalten A oder $\neg A$. Ein Literal der Gestalt A heißt *positiv*, ein Literal der Gestalt $\neg A$ *negativ*. Das *Komplement* eines positiven Literals A ist $\neg A$, das Komplement eines negativen Literals $\neg A$ ist A.

- Eine Formel der Prädikatenlogik erster Stufe heißt *rektifiziert*, wenn die Variablen, die unmittelbar hinter einem Vorkommen eines Quantors in der Formel stehen, paarweise verschieden und verschieden von allen freien Variablen in der Formel sind. (Siehe Kapitel 3)

- Eine Formel der Prädikatenlogik erster Stufe heißt in *Pränexform*, wenn sie die Gestalt $Q_1 x_1 \ldots Q_n x_n\ G$ hat, $n \geq 0$, wobei die Q_i Quantoren sind und in G kein Quantor vorkommt. (Siehe Kapitel 3)

- Eine *Klausel* ist eine Disjunktion von Literalen.

  Eine Formel der Prädikatenlogik erster Stufe heißt in *konjunktiver Normalform*, wenn sie in Pränexform ist und ihr quantorfreier Teil eine Konjunktion von Klauseln ist.

- Eine Formel der Prädikatenlogik erster Stufe heißt in *Klauselnormalform*, wenn sie eine Konjunktion von Klauseln ist.

- Eine Klausel in *Implikationsnormalform* hat die Gestalt $(R \Rightarrow K)$, wobei R (für *Rumpf*) eine Konjunktion und K (für *Kopf*) eine Disjunktion von positiven Literalen, also von atomaren Formeln, ist.

  Eine Formel der Prädikatenlogik erster Stufe heißt in *Implikationsnormalform*, wenn sie eine Konjunktion von Klauseln in Implikationsnormalform ist. ∎

Die Formeln $\top$, $\bot$, $\neg\top$, $\neg\bot$ kommen auch in prädikatenlogischen Klauseln nicht vor.

Ein Beispiel für eine Klausel ist $\vee^\star(\neg p(x,y),\ \neg q(x,y),\ r(x),\ s(x,z))$, ihre Implikationsnormalform ist $(\wedge^\star(p(x,y),\ q(x,y)) \Rightarrow \vee^\star(r(x),\ s(x,z)))$. Ein Beispiel für eine Formel in konjunktiver Normalform ist $\forall x \exists y \forall z \wedge^\star\Big(\vee^\star(p(a,x),\ p(x,a)),\ \vee^\star(\neg p(x,y),\ \neg q(x,y),\ r(x),\ s(x,z))\Big)$. Sie ist obendrein geschlossen.

Eine Formel in Klauselnormalform oder in Implikationsnormalform ist nach Definition eine quantorfreie Darstellung einer Formel. Sie dient als abkürzende Schreibweise für die konjunktive Normalform einer universellen geschlossenen Formel.

Notation 4.4.11 (Allabschluss).

- Sei F eine Formel. Der Allabschluss $\forall(F)$ ist die geschlossene Formel
  $$\forall(F) := \begin{cases} F & \text{falls } F \text{ geschlossen ist} \\ \forall x_1 \dots \forall x_n F & \text{falls } \mathit{fvar}(F) = \{x_1, \dots, x_n\} \end{cases}$$

- Sei S eine Menge von Formeln. Der Allabschluss $\forall(S)$ ist die Menge von geschlossenen Formeln $\forall(S) := \{\forall(F) \mid F \in S\}$. ∎

Satz 4.4.12 (Transformation in konjunktive Normalform).

1. Die aussagenlogische Transformation in konjunktive Normalform hat die gleichen Eigenschaften, wenn sie auf eine quantorfreie Formel der Prädikatenlogik erster Stufe angewandt wird.

2. Ebenso für die Transformation in Implikationsnormalform.

3. Es gibt Algorithmen, die zu jeder geschlossenen Formel F der Prädikatenlogik erster Stufe effektiv eine geschlossene Formel F' in konjunktiver Normalform berechnen mit $F \models\!\mid F'$.

Beweis: Die genannten Verfahren hängen nicht davon ab, welche Struktur die atomaren Formeln haben, damit gelten die ersten beiden Aussagen.

Nach Satz 3.8.4 ist jede geschlossene Formel F logisch äquivalent zu einer geschlossenen Formel $Q_1x_1 \dots Q_nx_n\, G$ in Pränexform. Satz 3.8.3 gibt ein Verfahren zur Berechnung dieser Formel an. Zu der quantorfreien Formel G kann eine konjunktive Normalform G' effektiv berechnet werden, damit ist $F' = Q_1x_1 \dots Q_nx_n\, G'$. ∎

Folgesatz 4.4.13 (Normalformensatz [universell]). Sei F eine universelle geschlossene Formel und S eine endliche Menge von solchen Formeln. Es gibt Algorithmen zur effektiven Berechnung:

1. Einer Formel F' in Klauselnormalform oder in Implikationsnormalform mit $F \models\!\!\mid \forall(F')$.
2. Einer endlichen Menge S' von Klauseln oder von Klauseln in Implikationsnormalform mit $S \models\!\!\mid \forall(S')$. ■

Für eine universelle geschlossene Formel in Pränexform enthält der Quantorpräfix nur Allquantoren, und zwar einen Allquantor für jede Variable in der Formel. Der Quantorpräfix kann aus dem quantorfreien Teil der Formel immer durch Bildung des Allabschlusses rekonstruiert werden. Die Klauselnormalform bzw. Implikationsnormalform nutzt das aus, indem sie die Quantoren implizit lässt.

Die Skolemisierung ermöglicht eine Umwandlung von nichtuniversellen Formeln in universelle, für die die semantischen Begriffe auf Herbrand-Interpretationen zurückgeführt werden können (Satz 3.8.23). Damit ergibt sich:

Folgesatz 4.4.14 (Normalformensatz [PL1S]). Sei F eine geschlossene Formel und S eine endliche Menge von geschlossenen Formeln. Es gibt Algorithmen zur effektiven Berechnung:

1. Einer Formel F' in Klauselnormalform oder in Implikationsnormalform, so dass F erfüllbar ist genau dann wenn $\forall(F')$ ein Herbrand-Modell hat.
2. Einer endlichen Menge S' von Klauseln oder von Klauseln in Implikationsnormalform, so dass S erfüllbar ist genau dann wenn $\forall(S')$ ein Herbrand-Modell hat.

Die Herbrand-Modelle beziehen sich dabei auf das Herbrand-Universum einer Erweiterung der ursprünglichen Sprache um endlich viele Funktionssymbole. ■

4.5 Exkurs: Die Davis-Putnam-Beweismethode

Für Beweisprobleme der Aussagenlogik wird in der Praxis am häufigsten die Methode von Davis und Putnam eingesetzt. Sie dient zum Nachweis der Unerfüllbarkeit bzw. Erfüllbarkeit einer endlichen aussagenlogischen Klauselmenge. Das reicht nach dem Normalformensatz für die Aussagenlogik (Satz 4.4.9) aus, um die Frage der Unerfüllbarkeit bzw. Erfüllbarkeit einer beliebigen aussagenlogischen Formel, oder auch einer endlichen Menge von aussagenlogischen Formeln, zu beantworten.

Eine aussagenlogische Klauselmenge S ist erfüllbar, wenn es eine Interpretation (Wahrheitsbelegung) gibt, die jede Klausel in S erfüllt. Das heißt, dass die Interpretation mindestens ein Literal aus jeder Klausel erfüllt.

Sei nun L ein Literal, das in einer Klausel in S vorkommt. Angenommen, wir betrachten nur Interpretationen, die das Literal L erfüllen. In einem solchen Kontext ist folgende Vereinfachung von S bezüglich L möglich:

Algorithmus $\mathit{boolesche\_vereinfachung}(S, L)$:

1. S_1 sei die Klauselmenge, die aus S durch Streichen aller Klauseln entsteht, die das Literal L enthalten.
2. S_2 sei die Klauselmenge, die aus S_1 entsteht, wenn das Komplement von L aus jeder der übrigen Klauseln gestrichen wird.
3. Liefere S_2 als Ergebnis. ■

Satz 4.5.1. Für jede aussagenlogische Interpretation M mit $M \models L$ gilt $M \models S$ gdw. $M \models \mathit{boolesche\_vereinfachung}(S, L)$.

Beweis: M erfüllt jede Klausel, die L enthält, also $M \models S$ gdw. $M \models S_1$. Da M das Komplement von L nicht erfüllt, gilt für jede Klausel aus S_1, dass M sie genau dann erfüllt, wenn M ein anderes Literal der Klausel erfüllt. Das heißt, $M \models S_1$ gdw. $M \models S_2$. ■

Falls jede Klausel in S das Literal L enthält, entsteht im ersten Schritt die leere Klauselmenge. Falls eine Klausel in S nur das Komplement von L enthält, entsteht im zweiten Schritt eine Klauselmenge, die die leere Klausel $\vee^\star()$ enthält. Das sind die beiden Basisfälle der Beweismethode.

Die Beweismethode besteht aus einer systematischen Suche nach Interpretationen, die S erfüllen. Falls keiner der genannten Basisfälle vorliegt, wählt man ein Literal L aus einer der Klauseln in S und untersucht in einer Fallunterscheidung zum einen die Interpretationen, die L erfüllen, zum anderen die Interpretationen, die das Komplement von L erfüllen. Jeder der beiden Fälle erlaubt eine Vereinfachung der Klauselmenge, und jede der beiden vereinfachten Klauselmengen wird entsprechend weiterbearbeitet. Dadurch ergibt sich insgesamt ein Baum von Klauselmengen.

Die jeweils betrachtete Menge von Interpretationen kann durch eine Menge W von Literalen dargestellt werden. W repräsentiert die Menge derjenigen Interpretationen, die alle Literale in W erfüllen, also alle M mit $M \models W$.

Betrachten wir als Beispiel eine Klauselmenge, in der die atomaren Formeln p, q, r und s vorkommen. Die Menge $W = \{p, \neg q\}$ repräsentiert alle Interpretationen, die p erfüllen und q falsifizieren. Dazu gehören Interpretationen, die r erfüllen, und Interpretationen, die r falsifizieren. Diese beiden disjunkten Teilmengen von Interpretationen werden gerade durch die Mengen $W \cup \{r\}$ und $W \cup \{\neg r\}$ repräsentiert.

Die Knoten des Baums sind also Paare (W, S) mit einer Literalmenge W und einer Klauselmenge S.

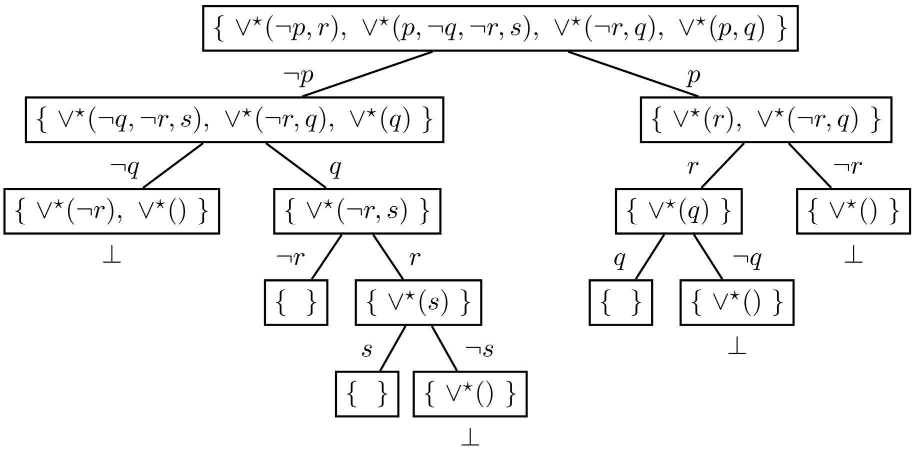
Nichtdeterministischer Algorithmus $\mathit{eigenschaft}(S_0)$:

1. Erzeuge einen Baum T, der nur aus dem Knoten $(\emptyset, S_0)$ besteht.
   Markiere diesen Knoten als unbearbeitet.
2. Solange es in T einen unbearbeiteten Knoten gibt:
   a) Wähle einen solchen Knoten (W, S), markiere ihn als bearbeitet.
   b) Falls $S = \emptyset$ oder $\vee^\star() \in S$, ist der Knoten (W, S) ein Blatt.
      Falls $S \neq \emptyset$ und $\vee^\star() \notin S$
      i. Wähle ein Literal L aus einer Klausel in S. Sei $\overline{L}$ das Komplement von L.
      ii. Erzeuge zwei unbearbeitete Nachfolgerknoten von (W, S):
          $(W \cup \{L\},\ \mathit{boolesche\_vereinfachung}(S, L))$ und
          $(W \cup \{\overline{L}\},\ \mathit{boolesche\_vereinfachung}(S, \overline{L}))$
3. Wenn jedes Blatt von T die leere Klausel $\vee^\star()$ enthält, liefere das Ergebnis *unerfüllbar*, sonst liefere für jedes Blatt $(W, \emptyset)$ ein Ergebnis *erfüllbar mit W*. ■

Selbstverständlich braucht eine Implementierung der Methode den Baum nicht explizit als Datenstruktur aufzubauen, so wie es hier beschrieben ist. Normalerweise wird der Baum implizit durch die Aufrufstruktur von rekursiven Prozeduren gegeben sein.

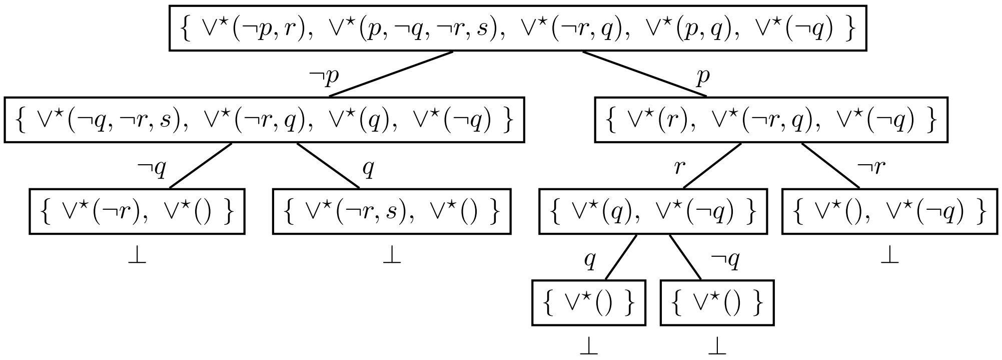
Der Algorithmus ist aus zwei Gründen nichtdeterministisch. Schritt 2(a) legt nicht fest, welcher Knoten als nächstes bearbeitet wird. Diese Wahl beeinflusst aber nicht die Struktur des Baums T, sondern nur die Reihenfolge, in der er aufgebaut wird. Im Schritt 2(b)i ist die Wahl des Literals L nicht festgelegt. Hier hängt die Struktur des Baums davon ab, wie gewählt wird.

Sei zum Beispiel $S_0 = \{\ \vee^\star(\neg p, r),\ \vee^\star(p, \neg q, \neg r, s),\ \vee^\star(\neg r, q),\ \vee^\star(p, q)\ \}$. Im Schritt 2 entsteht der folgende Baum, wenn im Schritt 2(b)i jeweils das erste Literal aus der ersten Klausel in S gewählt wird. Der Übersichtlichkeit halber sind die Literalmengen W als Kantenbeschriftungen dargestellt. Die Menge W eines Knotens in diesem Baum ist also die Menge aller Kantenbeschriftungen auf dem Pfad vom Knoten zur Wurzel.



Die Blätter, die die leere Klausel $\vee^\star()$ enthalten, sind hier mit dem Symbol $\perp$ gekennzeichnet. Der Baum hat drei Blätter mit der leeren Klauselmenge. Im Schritt 3 werden also die Ergebnisse *erfüllbar mit* $\{\neg p, q, \neg r\}$ und *erfüllbar mit* $\{\neg p, q, r, s\}$ und *erfüllbar mit* $\{p, r, q\}$ geliefert.

Als zweites Beispiel enthalte S_0 zusätzlich zu den obigen Klauseln noch die Klausel $\vee^\star(\neg q)$. Wenn die Auswahl der Literale wieder auf die gleiche Weise erfolgt, entsteht folgender Baum.



Diesmal enthält jedes Blatt die leere Klausel. Der Algorithmus liefert also das Ergebnis *unerfüllbar*.

Satz 4.5.2 (Terminierung). Für jede endliche aussagenlogische Klauselmenge S_0 terminiert *eigenschaft*(S_0), egal, wie die Wahl in den nichtdeterministischen Schritten getroffen wird.

Beweis: Der Rang einer Klausel sei definiert als die Anzahl der Literale in der Klausel, der Rang einer endlichen Klauselmenge sei die Summe der Ränge der Klauseln in der Menge, der Rang eines Knotens im Baum T sei der Rang seiner Klauselmenge. Man überprüft leicht, dass der Rang jedes Nachfolgerknotens im Baum T um mindestens 1 kleiner ist als der Rang seines Vorgängers. Also ist jeder Ast im Baum endlich. Da der Baum auch endlichen Verzweigungsgrad hat, besitzt er nach dem König'schen Unendlichkeitslemma endlich viele Knoten.

Die Schleife in Schritt (2) bearbeitet keinen Knoten des Baums mehrmals, also terminiert sie. ∎

Hilfssatz 4.5.3. Sei T ein in Schritt 2 von *eigenschaft*(S_0) aufgebauter Baum. Für jeden Knoten (W, S) in T und für jede Interpretation M gilt:

$M \models W$ und $M \models S$ genau dann, wenn es im Teilbaum mit Wurzel (W, S) ein Blatt $(W', \emptyset)$ gibt mit $M \models W'$.

Beweis: Durch strukturelle Induktion über den Aufbau des Baums T. Dies ist möglich, weil T nach Satz 4.5.2 keine unendlichen Äste hat.

Basisfall: Sei (W, S) ein Blatt von T. Dann ist $S = \emptyset$ oder $\vee^\star() \in S$. Im Fall $S = \emptyset$ gilt $M \models S$ für jedes M, und die Behauptung gilt für $W = W'$. Im Fall $\vee^\star() \in S$ gilt $M \not\models S$ für jedes M, so dass die linke Seite des „genau dann wenn“ nicht gilt, und die rechte gilt wegen $S \neq \emptyset$ auch nicht.

Induktionsfall: Sei (W, S) ein Knoten mit Nachfolgern, und die Behauptung gelte für die Teilbäume, deren Wurzeln diese Nachfolger sind. Es bleibt zu zeigen, dass sie auch für (W, S) gilt.

Nach Konstruktion gibt es ein Literal L mit Komplement $\overline{L}$, so dass die Nachfolgerknoten von (W, S) die beiden Knoten $(W \cup \{L\}, \textit{boolesche\_vereinfachung}(S, L))$ sowie $(W \cup \{\overline{L}\}, \textit{boolesche\_vereinfachung}(S, \overline{L}))$ sind.

Für die Richtung von links nach rechts sei M eine Interpretation mit $M \models W$ und $M \models S$. Jede Interpretation erfüllt entweder L oder $\overline{L}$, und o.B.d.A. gelte $M \models L$ (der andere Fall ist dazu symmetrisch). Dann gilt $M \models W \cup \{L\}$ und nach Satz 4.5.1 auch $M \models \textit{boolesche\_vereinfachung}(S, L)$. Nach Induktionsannahme gibt es im Teilbaum, dessen Wurzel dieser Knoten ist, ein Blatt $(W', \emptyset)$ mit $M \models W'$. Dieses Blatt ist auch ein Blatt des Teilbaums mit Wurzel (W, S).

Für die Gegenrichtung sei $(W', \emptyset)$ ein Blatt im Teilbaum mit Wurzel (W, S), und sei M eine Interpretation mit $M \models W'$. Dann ist $(W', \emptyset)$ ein Blatt eines Teilbaums, dessen Wurzel ein Nachfolger von (W, S) ist. O.B.d.A. sei $(W \cup \{L\}, \textit{boolesche\_vereinfachung}(S, L))$ dieser Nachfolger. Dann gilt $M \models W \cup \{L\}$ und $M \models \textit{boolesche\_vereinfachung}(S, L)$ nach Induktionsannahme. Daraus folgt $M \models W$ und $M \models L$, und mit Satz 4.5.1 auch $M \models S$. ∎

Satz 4.5.4 (Korrektheit und Vollständigkeit). Für jede endliche aussagenlogische Klauselmenge S_0 und für jede Wahl in den nichtdeterministischen Schritten gilt

1. (Korrektheit bezüglich Modellen)
   Wenn *eigenschaft*(S_0) ein Ergebnis *erfüllbar mit* W liefert, ist jede von W repräsentierte Interpretation ein Modell von S_0.

2. (Vollständigkeit bezüglich Modellen)
   Wenn S_0 erfüllbar ist, gilt für jedes Modell M von S_0: *eigenschaft*(S_0) liefert ein Ergebnis *erfüllbar mit* W, so dass W unter anderem die Interpretation M repräsentiert.
3. (Korrektheit bezüglich Unerfüllbarkeit)
   Wenn *eigenschaft*(S_0) das Ergebnis *unerfüllbar* liefert, ist S_0 unerfüllbar.
4. (Vollständigkeit bezüglich Unerfüllbarkeit)
   Wenn S_0 unerfüllbar ist, liefert *eigenschaft*(S_0) das Ergebnis *unerfüllbar*.

Beweis:

1. Wenn der Algorithmus in Schritt 3 ein Ergebnis *erfüllbar mit* W' liefert, gibt es ein Blatt $(W', \emptyset)$ im Baum T. Sei M eine von W' repräsentierte Interpretation, das heißt, $M \models W'$. Nach Satz 4.5.3 gilt für dieses M und für den Knoten $(\emptyset, S_0)$ an der Wurzel von T, dass $M \models \emptyset$ und $M \models S_0$.
2. Sei M ein Modell von S_0. Für den Knoten $(\emptyset, S_0)$ an der Wurzel von T gilt also $M \models \emptyset$ und $M \models S_0$. Nach Satz 4.5.3 gibt es ein Blatt $(W', \emptyset)$ im Baum mit $M \models W'$. Dann liefert der Algorithmus in Schritt 3 ein Ergebnis *erfüllbar mit* W'.
3. Die Korrektheit bezüglich der Unerfüllbarkeit ergibt sich unmittelbar aus der Kontraposition der Vollständigkeit bezüglich Modellen.
4. Die Vollständigkeit bezüglich der Unerfüllbarkeit ergibt sich unmittelbar aus der Kontraposition der Korrektheit bezüglich Modellen. ∎

Optimierungen

Die Davis-Putnam-Beweismethode basiert auf dem oben beschriebenen Algorithmus, angereichert mit einer Vielzahl von Optimierungen. Diese Optimierungen in ihrer Gesamtheit tragen wesentlich zur Leistungsfähigkeit bei. Sie führen dazu, dass die Methode in vielen Fällen erheblich effizienter ist als die Methode der Wahrheitstafeln. Allerdings gibt es immer auch Beispiele, für die der Aufwand ähnlich ist wie mit Wahrheitstafeln.

Tautologie-Vereinfachung. Eine Klausel heißt eine Tautologie, wenn sie ein Literal sowie dessen Komplement enthält. Sei *tautologie_vereinfachung*(S) die Klauselmenge, die durch Streichen aller Tautologien aus S entsteht.

Für jede Interpretation M gilt, dass sie jede Tautologie erfüllt, also gilt $M \models S$ gdw. $M \models$ *tautologie_vereinfachung*(S).

Es reicht aus, statt $(\emptyset, S_0)$ den Knoten $(\emptyset,$ *tautologie_vereinfachung*$(S_0))$ zur Wurzel des Baums zu machen. Man kann sich leicht klarmachen, dass die Nachfolger eines tautologiefreien Knotens ebenfalls tautologiefrei sind.

Subsumption-Vereinfachung. Eine Klausel C subsumiert eine Klausel C', wenn jedes Literal von C auch in C' enthalten ist. Zum Beispiel gilt $\vee^\star(p, \neg q)$ subsumiert $\vee^\star(\neg q, r, p)$. Sei *subsumption_vereinfachung*(S) die Klauselmenge, die durch Streichen jeder Klausel aus S entsteht, die von einer der verbleibenden Klauseln subsumiert wird.

Für jede Interpretation M gilt, wenn $M \models C$ und C subsumiert C', dann $M \models C'$, also gilt $M \models S$ gdw. $M \models$ *subsumption_vereinfachung*(S).

Damit lassen sich die obigen Beweise so erweitern, dass alle bisherigen Ergebnisse weiterhin gelten, wenn im Algorithmus bei der Erzeugung eines Knotens (W, S) statt dessen der Knoten $(W, \mathit{subsumption\_vereinfachung}(S))$ erzeugt wird.

Unär-Vereinfachung. Eine Klausel heißt unär, wenn sie genau ein Literal enthält. Wird im Schritt 2(b) das Literal L einer unären Klausel gewählt, ist $\mathit{boolesche\_vereinfachung}(S, \overline{L})$ für das Komplement $\overline{L}$ immer eine Klauselmenge, die die leere Klausel enthält. Den zugehörigen Nachfolgerknoten kann man deshalb gleich weglassen.

Dazu wird in Schritt 2(b) vor der Wahl des Literals einfach folgender Fall eingefügt: Falls S eine unäre Klausel $\vee^\star(L)$ enthält, erzeuge *einen einzigen* unbearbeiteten Nachfolgerknoten $(W \cup \{L\},\ \mathit{boolesche\_vereinfachung}(S, L))$.

Alle bisherigen Ergebnisse gelten auch mit dieser Optimierung.

Es ist gar nicht so selten, dass die Unär-Vereinfachung eine neue unäre Klausel im Nachfolger erzeugt, so dass die Unär-Vereinfachung gleich wieder anwendbar ist. An den obigen Beispielen kann man nachvollziehen, dass diese „Kettenreaktion" zu drastisch verkleinerten Bäumen führt. Im zweiten Beispiel entsteht ein Baum mit einem einzigen Ast ohne Verzweigungen.

In einer Implementierung kann man obendrein den Übergang zu einem einzigen Nachfolgerknoten durch „Wiederverwendung" des Vorgängerknotens realisieren, der ja – bis auf die Erweiterung von W – nur verkleinert wird.

Modellgenerierung vs. Widerlegung

In der bisher beschriebenen Form, auch mit den obigen Optimierungen, ist die Methode korrekt und vollständig bezüglich der Modelle. Das ist für viele Anwendungen nützlich. Wenn man zum Beispiel einen digitalen Schaltkreis analysieren will, interessiert man sich für die Frage, welche Kombinationen von Werten an den Eingängen zu welchen Werten am Ausgang führen. Modelliert man den Schaltkreis durch eine aussagenlogische Formel, entspricht dies gerade der Frage, welche Interpretationen die Formel erfüllen.

In anderen Fällen reicht aber eine einfachere Antwort. Angenommen wir haben zwei Schaltkreise, modelliert durch zwei Formeln F_1 und F_2, und wollen lediglich wissen, ob wir den einen durch den anderen ersetzen dürfen, ohne die Funktionalität zu verändern. Das ist genau dann der Fall, wenn die Formel $((F_1 \wedge \neg F_2) \vee (\neg F_1 \wedge F_2))$ unerfüllbar ist. Hier interessiert nur die Antwort *unerfüllbar* oder *erfüllbar*, aber nicht die Modelle.

Beweismethoden für Fragen der ersten Art nennt man Modellgenerierungsmethoden, Beweismethoden für Fragen der zweiten Art Widerlegungsmethoden (oder Refutationsmethoden). Eine Modellgenerierungsmethode mit den erforderlichen Korrektheits- und Vollständigkeitseigenschaften kann auch als Widerlegungsmethode eingesetzt werden, aber nicht umgekehrt.

Dafür können Widerlegungsmethoden effizienter sein, weil sie nur korrekt und vollständig bezüglich der Unerfüllbarkeit bzw. Erfüllbarkeit sein müssen, aber nicht bezüglich der Modelle.

Deshalb kann eine Variante des Algorithmus $\mathit{eigenschaft}(S)$ als Widerlegungsmethode den Aufbau des Baums abbrechen, sobald das erste Blatt mit leerer Klauselmenge entstanden ist, und einfach das Ergebnis *erfüllbar* liefern. Auch die Buchführung für die Mengen W kann vereinfacht werden.

Darüberhinaus sind zusätzliche Optimierungen möglich. Wird eine Klauselmenge S zu einer Klauselmenge S' vereinfacht, muss nicht mehr sichergestellt werden, dass $M \models S$ gdw. $M \models S'$ für jedes M (oder jedes gerade betrachtete M) gilt. Es reicht, dass S irgend ein Modell hat genau dann wenn S' irgend ein (möglicherweise anderes) Modell hat. „Die“ Beweismethode von Davis und Putnam wurde ursprünglich als Widerlegungsmethode mit Optimierungen entwickelt, die nur diesen schwachen Zusammenhang zwischen S und S' erhalten müssen und deshalb viel stärker vereinfachen können als die bisher besprochenen Optimierungen.

Isolation-Vereinfachung. Eine Klausel heißt isoliert (englisch: pure) in einer Klauselmenge S, wenn sie ein Literal enthält, dessen Komplement in keiner Klausel von S vorkommt. Sei $\mathit{isolation\_vereinfachung}(S)$ die Klauselmenge, die durch wiederholtes Streichen von isolierten Klauseln aus S entsteht, solange dies möglich ist.

Zum Beispiel ist in $S = \{\vee^\star(p, \neg q),\ \vee^\star(\neg p, q),\ \vee^\star(r, s, t),\ \vee^\star(\neg r)\}$ die vorletzte Klausel wegen der Literale s und t isoliert. Durch das Streichen dieser Klausel wird die letzte Klausel isoliert (obwohl sie es vor dem Streichen nicht war) und ebenfalls gestrichen, so dass $\mathit{isolation\_vereinfachung}(S) = \{\vee^\star(p, \neg q),\ \vee^\star(\neg p, q)\}$ ist.

Nicht jedes Modell von $\mathit{isolation\_vereinfachung}(S)$ erfüllt auch S. Im Beispiel könnte ein Modell von $\mathit{isolation\_vereinfachung}(S)$ das Literal r erfüllen und wäre somit kein Modell von S. Man kann jedoch zeigen, dass S genau dann erfüllbar ist, wenn $\mathit{isolation\_vereinfachung}(S)$ erfüllbar ist.

Positiv-Negativ-Vereinfachung. Eine Klausel heißt positiv, wenn sie nur positive Literale enthält, und negativ, wenn sie nur negative Literale enthält. Die leere Klausel ist die einzige, die sowohl positiv als auch negativ ist. Wenn eine Klauselmenge unerfüllbar ist, enthält sie mindestens eine positive und mindestens eine negative Klausel.

Sobald also im Baum eine Klauselmenge entsteht, die keine positive oder keine negative Klausel enthält (wie $\{\vee^\star(p, \neg q),\ \vee^\star(\neg p, q)\}$ im obigen Isolationsbeispiel), kann die Bearbeitung mit dem Ergebnis *erfüllbar* abgebrochen werden. Dieses Ergebnis ist korrekt, aber nicht alle durch das zugehörige W repräsentierten Interpretationen sind Modelle der Klauselmenge.

4.6 Entscheidbarkeitsergebnisse für die PL1S

Für die Prädikatenlogik erster Stufe haben wir bisher noch keine Beweismethode kennengelernt, die zumindest im Prinzip funktioniert, so wie die Methode der Wahrheitstafeln für die Aussagenlogik. Die dazu erforderlichen Bausteine wurden allerdings bereits behandelt und brauchen jetzt nur noch zusammengesetzt zu werden.

Satz 4.6.1 (Semi-Entscheidbarkeit der Unerfüllbarkeit). Die Unerfüllbarkeit einer Formel oder einer endlichen Menge von Formeln der Prädikatenlogik erster Stufe ist semi-entscheidbar.

Beweis: Eine endliche Formelmenge ist genau dann unerfüllbar, wenn die Konjunktion aller Formeln dieser Menge unerfüllbar ist. Es reicht also aus, den Fall einer einzelnen Formel F zu betrachten.

Nach dem Normalformensatz für die PL1S (Satz 4.4.14) gibt es Algorithmen zur effektiven Berechnung einer Formel F' in Klauselnormalform, so dass F genau dann unerfüllbar ist, wenn $\forall(F')$ kein Herbrand-Modell hat. Nach Satz 3.8.16 gilt dies genau dann, wenn die Menge $S^\star$ der Grundinstanzen von $\forall(F')$ kein Herbrand-Modell hat.

Wenn die betrachtete Sprache der Prädikatenlogik erster Stufe nur endlich viele Konstanten und keine mindestens einstelligen Funktionssymbole enthält, gibt es nur endlich viele Grundterme, und $S^\star$ ist endlich. Im allgemeinen ist $S^\star$ aber abzählbar unendlich.

Wir können nun jede der höchstens abzählbar vielen atomaren Formeln in $S^\star$ als Namen eines Aussagensymbols ansehen. Da weder Quantoren noch Variablen in $S^\star$ vorkommen, entspricht jede Herbrand-Interpretation bei dieser Sichtweise einer Wahrheitsbelegung für diese Aussagensymbole. Damit ist F genau dann unerfüllbar, wenn die unendliche Menge $S^\star$, als Menge von aussagenlogischen Formeln betrachtet, unerfüllbar ist.

Diese Frage ist nach Satz 4.2.2 semi-entscheidbar, sofern $S^\star$ aufzählbar ist. Dies folgt aber unmittelbar daraus, dass die Menge aller Grundterme zu einer gegebenen Signatur aufzählbar ist. ∎

Das Prinzip ist also, die zu testende Formel in Klauselnormalform umzuwandeln und davon systematisch alle Grundinstanzen aufzuzählen. Nach jeder neuen Grundinstanz kann man zum Beispiel mit der Methode der Wahrheitstafeln testen, ob die bisher erzeugte Menge von Grundinstanzen unerfüllbar ist. Sobald das der Fall ist, ist auch die Unerfüllbarkeit der ursprünglichen Formel nachgewiesen.

Man beachte, dass der Satz bereits für eine einzige Formel nur die Semi-Entscheidbarkeit sicherstellt, nicht die Entscheidbarkeit. Die Unerfüllbarkeit im endlichen Fall ist zwar für die Aussagenlogik entscheidbar, aber nicht für die Prädikatenlogik erster Stufe.

Dies kann gezeigt werden, indem man eine als nicht entscheidbar bekannte Frage auf die Frage der Unerfüllbarkeit reduziert (siehe Vorlesung „Informatik IV"). Wenn es einen Entscheidungs-Algorithmus für die Unerfüllbarkeit von Formeln der Prädikatenlogik erster Stufe gäbe, könnte man daraus beispielsweise einen Entscheidungs-Algorithmus für das Halteproblem von Turingmaschinen konstruieren.

Es gibt jedoch einige syntaktische Einschränkungen, unter denen die Unerfüllbarkeit einer Formel doch entscheidbar ist: zum Beispiel, wenn alle Funktions- und Relationssymbole höchstens einstellig sind, oder auch, wenn die Formel universell ist und keine mindestens einstelligen Funktionssymbole enthält.

Mit derartigen syntaktischen Einschränkungen der Logik ist natürlich nicht mehr alles ausdrückbar, was mit der uneingeschränkten Prädikatenlogik erster Stufe ausdrückbar ist. Man verbessert also die algorithmische Behandelbarkeit auf Kosten der Ausdrucksstärke. Will man umgekehrt die Ausdrucksstärke verbessern, geht dies auf Kosten der algorithmischen Behandelbarkeit. Wenn die Logik nur so weit über die Prädikatenlogik erster Stufe hinausgeht, dass man die natürlichen Zahlen darin formalisieren kann, verliert man sogar die Semi-Entscheidbarkeit der Unerfüllbarkeit. Das folgt aus einem berühmten Ergebnis von Kurt Gödel (Brünn 1906 – Princeton 1978), dem sogenannten Unvollständigkeitssatz.

Die Prädikatenlogik erster Stufe ist in mancherlei Hinsicht der beste Kompromiss zwischen Ausdrucksstärke und algorithmischer Behandelbarkeit.

4.7 Exkurs: Die PUHR-Tableau-Beweismethode

Dieser Abschnitt führt zunächst den Angleichungsalgorithmus (Matching) ein, dann darauf aufbauend eine Beweismethode für bereichsbeschränkte Klauseln der PL1S in Implikationsnormalform. Zudem wird gezeigt, dass bereichsbeschränkte Klauseln in Implikationsnormalform gleich ausdrucksstark sind wie allgemeine Klauseln in Implikationsnormalform.

Substitutionen und Angleichungsalgorithmus

Definition 4.7.1 (Substitution). Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe, sei $\mathcal{V}$ die Menge aller Variablen von $\mathcal{L}$ und $\mathcal{T}_{\mathcal{L}}$ die Menge aller $\mathcal{L}$-Terme.

- Eine *Substitution* ist eine postfix-notierte Funktion $\sigma : \mathcal{V} \to \mathcal{T}_{\mathcal{L}}$, so dass die Menge $\{x \in \mathcal{V} \mid x\sigma \neq x\}$ endlich ist.

  Wenn $x\sigma$ für jedes x aus dieser Menge ein Grundterm (variablenfreier Term) ist, heißt σ eine *Grundsubstitution.*
- Eine Substitution σ wird repräsentiert als endliche Menge von Paaren $\{t_1/x_1, \ldots, t_n/x_n\}$, wo die Variablen x_i paarweise verschieden sind und $x_i\sigma = t_i$ für $i = 1, \ldots, n$ und $x\sigma = x$ für alle $x \in \mathcal{V} \setminus \{x_1, \ldots, x_n\}$. ■

Die identische Funktion, die jede Variable auf sich selbst abbildet, ist eine Substitution, sogar eine Grundsubstitution. Sie wird durch die leere Menge von Paaren repräsentiert und manchmal mit ε bezeichnet.

Eine Substitution bildet Variablen auf Terme ab. Sie wird auf kanonische Weise so erweitert, dass sie Terme auf Terme abbildet und quantorfreie Formeln auf quantorfreie Formeln.

Definition 4.7.2. Sei σ eine Substitution.

- σ induziert eine ebenfalls σ genannte und postfix-notierte Funktion $\mathcal{T}_{\mathcal{L}} \to \mathcal{T}_{\mathcal{L}}$, die wie folgt rekursiv über den Aufbau der Terme definiert ist:

  $$\begin{array}{lll} c\sigma & := c & c \text{ Konstante} \\ f(t_1, \ldots, t_n)\sigma & := f(t_1\sigma, \ldots, t_n\sigma) & f\ n\text{-stelliges Funktionssymbol}, t_1, \ldots, t_n\ \mathcal{L}\text{-Terme} \end{array}$$
- σ induziert eine ebenfalls σ genannte und postfix-notierte Funktion, die quantorfreie $\mathcal{L}$-Formeln auf quantorfreie $\mathcal{L}$-Formeln abbildet und wie folgt rekursiv definiert ist:

  $$\begin{array}{lll} \top\sigma & := \top & \\ \bot\sigma & := \bot & \\ p\sigma & := p & p \text{ 0-stelliges Relationssymbol} \\ p(t_1, \ldots, t_n)\sigma & := p(t_1\sigma, \ldots, t_n\sigma) & p\ n\text{-stelliges Relationssymbol} \\ (\neg F)\sigma & := \neg(F\sigma) & \\ (F\,\theta\,G)\sigma & := (F\sigma\,\theta\,G\sigma) & \theta \text{ zweistelliger Junktor} \\ \bigwedge^\star(F_1, \ldots, F_n)\sigma & := \bigwedge^\star(F_1\sigma, \ldots, F_n\sigma) & \\ \bigvee^\star(F_1, \ldots, F_n)\sigma & := \bigvee^\star(F_1\sigma, \ldots, F_n\sigma) & \end{array}$$
- $S\sigma := \{F\sigma \mid F \in S\}$ für eine Menge S von quantorfreien Formeln.
- Ist F eine Formel oder ein Term, heißt $F\sigma$ eine *Instanz* von F. ■

Damit ist insbesondere die Anwendung von Substitutionen auf Klauseln und Klauselmengen erklärt. Die Notation $F[t/x]$, die bisher schon häufig vorkam, ist ein Sonderfall der Anwendung einer Substitution auf eine Formel.

Beispiel 4.7.3. Sei $\sigma = \{f(a)/x,\ c/z\}$.

$$\begin{aligned}
& p(x, \quad y, z)\,\sigma \\
= \; & p(f(a), y, c)
\end{aligned}$$

$$\begin{aligned}
& \vee^\star\Big(p(f(z), c, z),\ \ \neg p(x, \quad y, z),\ \ q(x, \quad x) \quad \Big)\,\sigma \\
= \; & \vee^\star\Big(p(f(c), c, c),\ \ \neg p(f(a), y, c),\ \ q(f(a), f(a))\Big)
\end{aligned}$$

$$\begin{aligned}
& \Big[\wedge^\star\Big(p(x, \quad y, z),\ \ r(f(x)) \quad \Big) \Rightarrow \vee^\star\Big(p(f(z), c, z),\ \ q(x, \quad x) \quad \Big)\Big]\,\sigma \\
= \; & \Big[\wedge^\star\Big(p(f(a), y, c),\ \ r(f(f(a)))\Big) \Rightarrow \vee^\star\Big(p(f(c), c, c),\ \ q(f(a), f(a))\Big)\Big]
\end{aligned}$$

■

Das Beispiel zeigt auch, dass durch Anwendung einer Grundsubstitution auf eine Formel F nicht notwendigerweise eine Grundinstanz von F entsteht. Dies ist nur dann der Fall, wenn die Grundsubstitution alle Variablen von F instanziiert.

Definition 4.7.4 (Umschreibungsregeln für Angleichung). Sei x eine Variable, s_i, t_i, L_i, R_i für passende Indizes Terme, c eine Konstante, f ein k-stelliges Funktionssymbol, g ein davon verschiedenes Funktionssymbol. Die folgenden Regeln schreiben jeweils ein Paar aus einer Gleichungsmenge und einer Substitution wiederum in ein solches Paar um:

Zerlegung:

$$\begin{array}{lr}
\{f(s_1,\ldots,s_k)\doteq f(t_1,\ldots,t_k)\} \cup \{L_1\doteq R_1,\ \ldots,\ L_n\doteq R_n\} & \sigma \\
\hline
\{s_1\doteq t_1,\ldots,s_k\doteq t_k\} \cup \{L_1\doteq R_1,\ \ldots,\ L_n\doteq R_n\} & \sigma
\end{array}$$

Zerlegung nullstellig:

$$\begin{array}{lr}
\{c\doteq c\} \cup \{L_1\doteq R_1,\ \ldots,\ L_n\doteq R_n\} & \sigma \\
\hline
\{L_1\doteq R_1,\ \ldots,\ L_n\doteq R_n\} & \sigma
\end{array}$$

Anwendung:

$$\begin{array}{lr}
\{x\doteq t_0\} \cup \{L_1\doteq R_1,\ \ldots,\ L_n\doteq R_n\} & \sigma \\
\hline
\{L_1[t_0/x]\doteq R_1,\ \ldots,\ L_n[t_0/x]\doteq R_n\} & \sigma \cup \{t_0/x\}
\end{array}$$

Abbruch:

$$\begin{array}{lr}
\{f(s_1,\ldots,s_k)\doteq g(t_1,\ldots,t_h)\} \cup \{L_1\doteq R_1,\ \ldots,\ L_n\doteq R_n\} & \sigma \\
\hline
\bot & \sigma
\end{array}$$

Die Abbruchregel gilt auch für die Fälle mit nullstelligem f oder g. ■

Definition 4.7.5 (Angleichungsalgorithmus).
Nichtdeterministischer Algorithmus *angleichung*(s, t):

1. Initialisierung: $S := \{\, s\doteq t \,\}$ und $\sigma := \emptyset$.
2. Solange $S \neq \emptyset$ und $S \neq \bot$ ist:
   wähle eine anwendbare Umschreibungsregel aus und wende sie auf S und σ an.
3. Wenn $S = \emptyset$, liefere σ als Ergebnis.
   Wenn $S = \bot$, liefere *scheitert* als Ergebnis.

Dabei ist s ein Term, der Variablen enthalten kann, und t ein Grundterm. Es ist auch zulässig, dass s ein Atom und t ein Grundatom ist, dann wird einfach jedes Relationssymbol wie ein Funktionssymbol gleicher Stelligkeit behandelt. ■

Der Angleichungsalgorithmus berechnet zu s und t eine Substitution σ mit $s\sigma = t$, falls es eine solche Substitution gibt. Statt von Angleichung spricht man auch von Semi-Unifikation oder von Matching. Die Angleichung ist ein Spezialfall der sogenannten „Unifikation“.

Beispiel 4.7.6. Für $s = g(f(a), f(x), y, x)$ und $t = g(f(a), f(a), b, a)$, ergeben sich beim Aufruf von *angleichung*(s, t) bei entsprechender Auswahl der Umschreibungsregeln der Reihe nach folgende Werte:
Initialisierung:
$S = \{g(f(a), f(x), y, x) \doteq g(f(a), f(a), b, a)\}$ $\qquad \sigma = \{\ \}$
Zerlegung:
$S = \{f(a) \doteq f(a),\ f(x) \doteq f(a),\ y \doteq b,\ x \doteq a\}$ $\qquad \sigma = \{\ \}$
Zerlegung (zweimal angewandt):
$S = \{a \doteq a,\ x \doteq a,\ y \doteq b,\ x \doteq a\}$ $\qquad \sigma = \{\ \}$
Zerlegung nullstellig (auf a angewandt):
$S = \{x \doteq a,\ y \doteq b,\ x \doteq a\}$ $\qquad \sigma = \{\ \}$
Anwendung:
$S = \{y \doteq b,\ a \doteq a\}$ $\qquad \sigma = \{a/x\}$
Anwendung:
$S = \{a \doteq a\}$ $\qquad \sigma = \{a/x, b/y\}$
Zerlegung nullstellig:
$S = \{\}$ $\qquad \sigma = \{a/x, b/y\}$
Ergebnis: $\{a/x, b/y\}$. ■

Beispiel 4.7.7. Für $s = g(f(a), f(x), y, x)$ und $t = g(f(a), f(a), b, f(a))$ ergeben sich beim Aufruf von *angleichung*(s, t) bei entsprechender Auswahl der Umschreibungsregeln der Reihe nach folgende Werte:
Initialisierung:
$S = \{g(f(a), f(x), y, x) \doteq g(f(a), f(a), b, f(a))\}$ $\qquad \sigma = \{\ \}$
Zerlegung:
$S = \{f(a) \doteq f(a),\ f(x) \doteq f(a),\ y \doteq b,\ x \doteq f(a)\}$ $\qquad \sigma = \{\ \}$
Zerlegung (zweimal angewandt):
$S = \{a \doteq a,\ x \doteq a,\ y \doteq b,\ x \doteq f(a)\}$ $\qquad \sigma = \{\ \}$
Zerlegung nullstellig (auf a angewandt):
$S = \{x \doteq a,\ y \doteq b,\ x \doteq f(a)\}$ $\qquad \sigma = \{\ \}$
Anwendung:
$S = \{y \doteq b,\ a \doteq f(a)\}$ $\qquad \sigma = \{a/x\}$
Abbruch (auf $a \doteq f(a)$ angewandt):
$S = \bot$ $\qquad \sigma = \{a/x\}$
Ergebnis: *scheitert*.

Es wäre auch möglich gewesen, vor der Abbruchregel noch die Anwendungsregel auf $y \doteq b$ anzuwenden und dadurch σ zu verändern. Das Ergebnis wäre trotzdem *scheitert*. ■

Satz 4.7.8. Ist s ein Term und t ein Grundterm, so gilt für jede Auswahl der Umschreibungsregeln in Schritt 2 des Angleichungsalgorithmus:

1. (Terminierung) *angleichung*(s, t) terminiert und liefert als Ergebnis entweder eine Grundsubstitution oder *scheitert*.

2. (Korrektheit) Wenn *angleichung*(s, t) als Ergebnis eine Substitution σ liefert, so gilt $s\sigma = t$.
3. (Vollständigkeit) Wenn *angleichung*(s, t) als Ergebnis *scheitert* liefert, so gibt es keine Substitution σ mit $s\sigma = t$.

Beweis:

1. Da t ein Grundterm ist, gilt für den initialen Wert von S, dass die rechte Seite jeder Gleichung variablenfrei ist. Man überprüft leicht, dass jede Anwendung einer Umschreibungsregel diese Eigenschaft erhält. Also ist σ stets eine Grundsubstitution.

   Der Rang rg von Termen, Gleichungen usw. sei wie folgt definiert:

$rg(c)$	$:=$	1	Konstante
$rg(f(t_1, \dots, t_n))$	$:=$	$1 + rg(t_1) + \cdots + rg(t_n)$	
$rg(L \doteq R)$	$:=$	$rg(R)$	Gleichung
$rg(\{L_1 \doteq R_1, \ \dots, \ L_n \doteq R_n\})$	$:=$	$rg(R_1) + \cdots + rg(R_n)$	Menge
$rg(\perp)$	$:=$	0	

   Jede Anwendung einer Umschreibungsregel verkleinert $rg(S)$:

   Zerlegung:

   Rang vorher: $1 + rg(t_1) + \cdots + rg(t_k) + rg(R_1) + \cdots + rg(R_n)$
   Rang nachher: $rg(t_1) + \cdots + rg(t_k) + rg(R_1) + \cdots + rg(R_n)$
   Das ist eine Verkleinerung um 1.

   Zerlegung nullstellig:

   Rang vorher: $1 + rg(R_1) + \cdots + rg(R_n)$
   Rang nachher: $rg(R_1) + \cdots + rg(R_n)$
   Das ist eine Verkleinerung um 1.

   Anwendung:

   Rang vorher: $rg(t_0) + rg(R_1) + \cdots + rg(R_n)$
   Rang nachher: $rg(R_1) + \cdots + rg(R_n)$
   Das ist eine echte Verkleinerung, da stets $rg(t_0) > 0$ gilt.

   Abbruch:

   Rang vorher: ≥ 1
   Rang nachher: 0

   Für den initialen Wert gilt $rg(S) = rg(t)$. Dieser endliche Wert ist also eine Obergrenze für die Anzahl der Umschreibungsschritte, so dass die Schleife in Schritt 2 terminiert.

   Enthält S eine Gleichung, so ist mindestens eine der Umschreibungsregeln anwendbar. Die Schleife ist also immer fortsetzbar bis $S = \emptyset$ oder $S = \perp$ ist. Damit liefert der Algorithmus in Schritt 3 als Ergebnis entweder die Grundsubstitution σ oder *scheitert*.

2. Wir zeigen folgende Invariante des Angleichungsalgorithmus:

   > Sei $S = \{L_1 \doteq R_1, \ \dots, \ L_n \doteq R_n\}$ die aktuelle Gleichungsmenge und σ die aktuelle Grundsubstitution.
   > Für jede Grundsubstitution τ mit $L_1\tau = R_1, \ \dots, \ L_n\tau = R_n$ gilt $s\sigma\tau = t$.

Initialisierung:

Das initiale S ist $\{s\dot{=}t\}$, das initiale σ die Identität. Dafür gilt die Invariante.

Zerlegung:

Die Invariante gelte in einem Zustand vor Anwendung der Regel. Das heißt, für jede Grundsubstitution τ mit $f(s_1,\ldots,s_k)\tau = f(t_1,\ldots,t_k), L_1\tau{=}R_1,\ \ldots,\ L_n\tau{=}R_n$ gelte $s\sigma\tau = t$.
Nach Definition ist $f(s_1,\ldots,s_k)\tau = f(t_1,\ldots,t_k)$ gdw. $s_1\tau{=}t_1,\ \ldots,\ s_k\tau{=}t_k$ ist, so dass die Invariante auch im Zustand nach Anwendung der Regel gilt.

Zerlegung nullstellig:

analog, weil nach Definition $c\tau = c$ für jedes τ gilt.

Anwendung:

Die Invariante gelte in einem Zustand vor Anwendung der Regel. Das heißt, für jede Grundsubstitution τ mit $x\tau = t_0,\ \ L_1\tau{=}R_1,\ldots,L_n\tau{=}R_n$ gelte $s\sigma\tau = t$.
Für den Zustand nach Anwendung der Regel sei τ' eine Grundsubstitution mit $L_1[t_0/x]\tau' = R_1,\ \ldots,\ L_n[t_0/x]\tau' = R_n$. Da t_0 ein Grundterm ist, gilt für dieses τ' auch $t_0\tau' = t_0$ und damit $x[t_0/x]\tau' = t_0$. Die Grundsubstitution $\{t_0/x\} \cup \tau'$ ist somit ein τ, das die Bedingungen der Invariante vor Anwendung der Regel erfüllt. Also gilt $s\sigma[t_0/x]\tau' = t$.
Nun ist aber das neue σ' gerade $\sigma \cup \{t_0/x\}$, so dass $s\sigma'\tau' = t$ gilt. Damit gilt die Invariante auch im Zustand nach Anwendung der Regel.

Abbruch:

wird nicht angewandt, wenn der Algorithmus eine Substitution als Ergebnis liefert.

Wenn der Algorithmus eine Substitution als Ergebnis liefert, ist die Gleichungsmenge S leer, und die Identitätssubstitution $\tau = \varepsilon$ erfüllt trivialerweise die Voraussetzungen der Invarianten. Also gilt $s\sigma\varepsilon = t$ und damit $s\sigma = t$.

3. Wir zeigen folgende Invariante des Angleichungsalgorithmus:

   Sei $S = \{L_1\dot{=}R_1,\ \ldots,\ L_n\dot{=}R_n\}$ die aktuelle Gleichungsmenge und σ die aktuelle Grundsubstitution.
   Wenn es eine Grundsubstitution ϑ gibt mit $s\vartheta = t$, dann gibt es eine Grundsubstitution τ mit $L_1\tau{=}R_1,\ \ldots,\ L_n\tau{=}R_n$

Initialisierung:

Die initiale Gleichungsmenge ist $\{s\dot{=}t\}$, und die Invariante gilt.

Zerlegung:

Die Invariante gelte in einem Zustand vor Anwendung der Regel. Das heißt, wenn es eine Grundsubstitution ϑ gibt mit $s\vartheta = t$, dann gebe es eine Grundsubstitution τ mit $f(s_1,\ldots,s_k)\tau = f(t_1,\ldots,t_k),\ \ L_1\tau{=}R_1,\ \ldots,\ L_n\tau{=}R_n$.
Nach Definition ist $f(s_1,\ldots,s_k)\tau = f(t_1,\ldots,t_k)$ gdw. $s_1\tau{=}t_1,\ldots,s_k\tau{=}t_k$ ist, so dass die Invariante auch im Zustand nach Anwendung der Regel gilt.

Zerlegung nullstellig:

analog, weil nach Definition $c\tau = c$ für jedes τ gilt.

Anwendung:

Die Invariante gelte in einem Zustand vor Anwendung der Regel. Das heißt, wenn es eine Grundsubstitution ϑ gibt mit $s\vartheta = t$, dann gebe es eine Grundsubstitution τ mit $x\tau = t_0$, $L_1\tau{=}R_1, \ \ldots, \ L_n\tau{=}R_n$. Wegen der ersten Gleichung hat τ die Form $\{t_0/x\} \cup \tau'$. Es gibt also eine Grundsubstitution τ' mit $L_1[t_0/x]\tau'{=}R_1, \ \ldots, \ L_n[t_0/x]\tau'{=}R_n$. Die Invariante gilt somit auch nach Anwendung der Regel.

Abbruch:

Liefert der Algorithmus das Ergebnis *scheitert*, dann wurde unmittelbar davor die Abbruchregel angewandt.

Die Invariante gelte in einem Zustand vor Anwendung der Regel. Dann enthält die Gleichungsmenge eine Gleichung $f(\ldots) \doteq g(\ldots)$, und dafür gibt es kein τ mit $f(\ldots)\tau = g(\ldots)$. Nach der Invarianten gibt es dann auch kein ϑ mit $s\vartheta = t$. ■

PUHR-Tableaus

Definition 4.7.9 (bereichsbeschränkt [Klausel]). Eine Klausel in Implikationsnormalform $(R \Rightarrow K)$ heißt *bereichsbeschränkt*, wenn alle im Kopf K vorkommenden Variablen ebenfalls im Rumpf R vorkommen. ■

Zum Beispiel ist $(\wedge^\star(p(x,y),\, q(y,z)) \Rightarrow \vee^\star(r(x),\, s(x,z)))$ bereichsbeschränkt, wogegen $(\wedge^\star(p(a,y),\, q(y,z)) \Rightarrow \vee^\star(r(x),\, s(x,z)))$ nicht bereichsbeschränkt ist. Jede Klausel der Gestalt $(R \Rightarrow K)$, in deren Kopf K keine Variable vorkommt, ist bereichsbeschränkt. Eine Klausel der Gestalt $(\wedge^\star() \Rightarrow K)$ ist nur dann bereichsbeschränkt, wenn keine Variable in K vorkommt.

Ein PUHR-Tableau ist ein Baum, dessen Knoten Vorkommen von Grundformeln sind. Die folgende Schreibweise dient als Hilfsmittel für die Definition von PUHR-Tableaus.

Notation 4.7.10. Ist T ein Baum und N ein Knoten in T, so bezeichnet *Vorfahren*(N) die Menge aller Knoten auf dem Pfad von N zur Wurzel des Baums T, einschließlich der Wurzel und einschließlich N selbst. ■

Definition 4.7.11 (PUHR-Tableau). Sei S eine Menge von bereichsbeschränkten Klauseln in Implikationsnormalform. Ein PUHR-Tableau für S ist ein Baum, dessen Knoten Vorkommen von $\mathcal{L}$-Formeln sind und der wie folgt gebildet werden kann:

1. Der degenerierte Baum, der aus dem einzigen Knoten $\top$ besteht, ist ein PUHR-Tableau für S, genannt das initiale PUHR-Tableau.

2. Ist T ein PUHR-Tableau für S und ist $N \neq \vee^\star()$ ein Blatt von T, so ist jeder Baum ein PUHR-Tableau für S, der aus T durch Anwendung einer der folgenden Erweiterungsregeln auf N entsteht.

PUHR-Regel:

$$\frac{\begin{array}{c}A_1\\ \vdots\\ A_n\end{array}}{\vee^\star(K_1,\ldots,K_m)\sigma}$$

Splitting-Regel:

$$\frac{\vee^\star(A_1,\ldots,A_m)}{A_1 \;\big|\; \ldots \;\big|\; A_m}$$

Die PUHR-Regel ist anwendbar auf N, wenn gilt:
Es gibt eine Klausel $C = (\wedge^\star(R_1, \ldots, R_n) \Rightarrow \vee^\star(K_1, \ldots, K_m))$ in S, $n \geq 0$, $m \geq 0$, und Atome $A_1, \ldots, A_n \in \mathit{Vorfahren}(N)$ und eine Grundsubstitution σ mit $R_i\sigma = A_i$ für alle $i \in \{1, \ldots, n\}$.
Dann wird $\vee^\star(K_1, \ldots, K_m)\sigma$ als Nachfolgerknoten von N hinzugefügt.

Man sagt dann auch, dass die PUHR-Regel mit der Instanz $C\sigma$ der Klausel $C \in S$ angewandt wird.

Die Splitting-Regel ist anwendbar auf N, wenn gilt:
Es gibt eine Disjunktion $\vee^\star(A_1, \ldots, A_m) \in \mathit{Vorfahren}(N)$ mit $m \geq 1$.
Dann wird jedes A_j für $j \in \{1, \ldots, m\}$ als ein Nachfolgerknoten von N hinzugefügt.

3. Sei $\langle T_i \rangle_{i \in \mathbb{N}}$ eine unendliche Sequenz von endlichen PUHR-Tableaus für S, so dass T_0 das initiale Tableau ist und für jedes $i \in \mathbb{N}$ das Tableau T_{i+1} aus T_i durch Anwendung einer der obigen Erweiterungsregeln entsteht.

   Der Baum T, dessen Knotenmenge die Vereinigung der Knotenmengen aller T_i und dessen Kantenmenge die Vereinigung der Kantenmengen aller T_i ist, ist ein (unendliches) PUHR-Tableau für S.

   Zu jedem endlichen Anfangsstück P eines Asts von T gibt es also $i \in \mathbb{N}$, so dass P ein Ast von T_i ist. ■

Bemerkungen:

1. PUHR steht für Positive Unit Hyper-Resolution. Das ist eine Zusammensetzung von Bezeichnungen verschiedener anderer Beweismethoden, auf denen die PUHR-Tableau-Beweismethode aufbaut.
2. Der Name Tableau, französisch für Tabelle, stammt aus einer Zeit, als noch keine Software für Textverarbeitung und -layout verfügbar war und Bäume aus typographischen Gründen als Tabellen dargestellt wurden. Die Schreibweise der Splitting-Regel erinnert noch an diese Tabellen-Darstellung.
3. Die Knoten eines PUHR-Tableaus sind „Vorkommen von $\mathcal{L}$-Formeln“ und nicht einfach „$\mathcal{L}$-Formeln“, weil die gleiche Formel an verschiedenen Stellen im Baum als Knoten vorkommen kann. Das ist im folgenden Beispiel der Fall.
4. Die Bedingung „$N \neq \vee^\star()$“ im Punkt 2 der Definition verhindert, dass ein Vorkommen von $\vee^\star()$ im Tableau Nachfolgerknoten hat.
5. Aus der Gestalt der Erweiterungsregeln ergibt sich sofort, dass in einem PUHR-Tableau jeder Knoten außer der Wurzel entweder eine Disjunktion von Atomen oder ein Atom ist. Wir werden sehen, dass es sich dabei immer um Grundformeln handelt.
6. Weiter ergibt sich aus der Gestalt der Erweiterungsregeln, dass jedes PUHR-Tableau endlichen Verzweigungsgrad hat. (Es gibt aber im allgemeinen keine feste Obergrenze für den Verzweigungsgrad: S kann für jedes $m \in \mathbb{N}$ eine Klausel mit m Atomen im Kopf enthalten.) Nach dem König'schen Unendlichkeitslemma ist ein PUHR-Tableau genau dann endlich, wenn alle seine Äste endlich sind.

   Jedes nach 1. und 2. gebildete PUHR-Tableau ist endlich. Tableaus mit unendlichen Ästen können nur durch die Grenzwertbildung gemäß 3. entstehen. ■

Beispiel 4.7.12. Sei S die Menge der folgenden Grundklauseln. Die Namen der Klauseln sollen nur den Bezug darauf erleichtern.

$$
\begin{array}{lll}
C_1 = & \wedge^\star() & \Rightarrow \vee^\star(semester) \\
C_2 = & \wedge^\star() & \Rightarrow \vee^\star(dozent(bry)) \\
C_3 = & \wedge^\star(dozent(bry)) & \Rightarrow \vee^\star(lehrt(bry), forscht(bry)) \\
C_4 = & \wedge^\star(dozent(bry), semester) & \Rightarrow \vee^\star(lehrt(bry)) \\
C_5 = & \wedge^\star(lehrt(bry), forscht(bry)) & \Rightarrow \vee^\star() \\
C_6 = & \wedge^\star(forscht(bry)) & \Rightarrow \vee^\star(froh(bry))
\end{array}
$$

Das initiale Tableau $\top$ enthält keine Disjunktion, so dass die Splitting-Regel nicht anwendbar ist, und es enthält kein Atom, so dass die PUHR-Regel nur mit Klauseln anwendbar ist, deren Rumpf die leere Konjunktion ist. Das ist für die Klauseln C_1 und C_2 der Fall.

Eine Anwendung der PUHR-Regel mit C_1 fügt den Kopf $\vee^\star(semester)$ dieser Klausel als Nachfolger des Wurzelknotens in das Tableau ein:

$$
\begin{array}{c}
\top \\
| \\
\vee^\star(\, semester \,)
\end{array}
$$

Auf das einzige Blatt dieses Tableaus wäre wieder die PUHR-Regel mit C_1 anwendbar, so dass ein zweites Vorkommen von $\vee^\star(semester)$ als Nachfolgerknoten hinzukäme (es ist klar, dass derartige Wiederholungen nicht nützlich sind – man kann sie verhindern, indem man die Anwendbarkeitsbedingungen der Erweiterungsregeln geeignet verschärft). Außerdem anwendbar ist die PUHR-Regel mit C_2 (wie schon im initialen Tableau) sowie die Splitting-Regel (da der einzige Ast jetzt eine Disjunktion enthält).

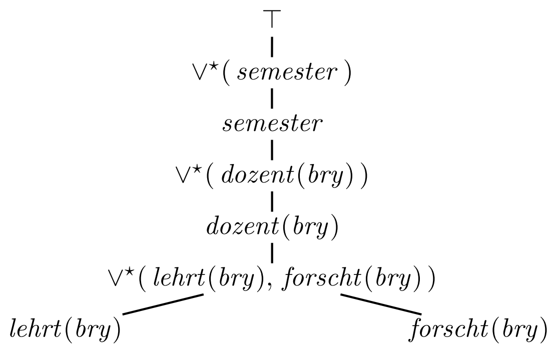
Wir wenden die Splitting-Regel an. Da die Disjunktion zufällig nur ein einziges Atom enthält, entsteht ein einziger Nachfolgerknoten:

$$
\begin{array}{c}
\top \\
| \\
\vee^\star(\, semester \,) \\
| \\
semester
\end{array}
$$

Abgesehen von Wiederholungen ist an dieser Stelle nur die PUHR-Regel mit C_2 anwendbar, danach die Splitting-Regel, und es entsteht:

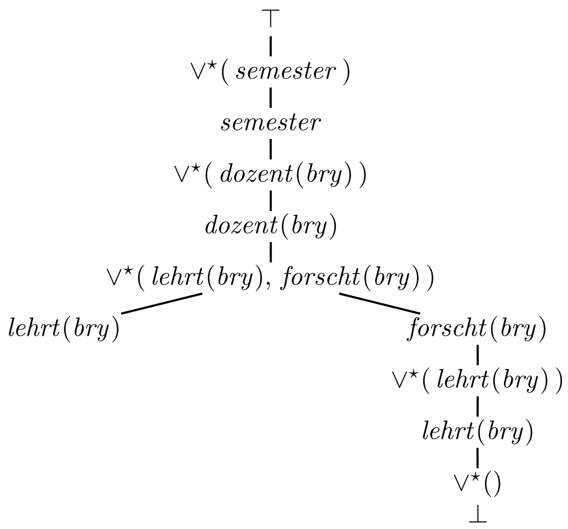
$$
\begin{array}{c}
\top \\
| \\
\vee^\star(\, semester \,) \\
| \\
semester \\
| \\
\vee^\star(\, dozent(bry) \,) \\
| \\
dozent(bry)
\end{array}
$$

Jetzt enthält der Ast alle Atome aus dem Rumpf der Klausel C_3, so dass die PUHR-Regel mit dieser Klausel anwendbar ist. Die dadurch eingefügte Disjunktion enthält zwei Atome, und die Anwendung der Splitting-Regel führt zu einer echten Verzweigung:



Im linken Ast wäre jetzt außer Wiederholungen noch eine Anwendung der PUHR-Regel mit C_4 möglich. Dadurch käme aber nach dem Splitting nur ein weiteres Vorkommen des Atoms *lehrt*(*bry*) hinzu, so dass diese Anwendung genausowenig nützlich wäre wie eine Wiederholung. Andere Möglichkeiten gibt es für diesen Ast nicht. Man überzeuge sich, dass die von der Menge der Atome in diesem Ast repräsentierte Herbrand-Interpretation ein Modell von S ist.

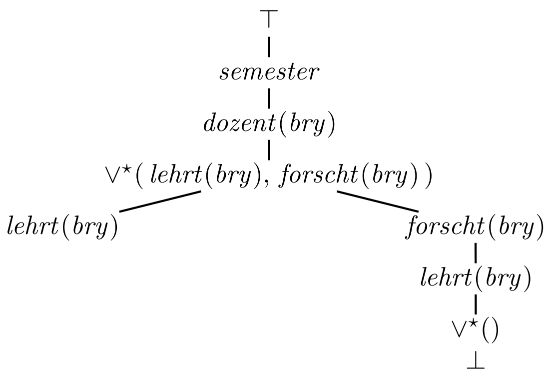
Im rechten Ast ist das Atom, das durch Anwendung der PUHR-Regel mit C_4 und anschließendem Splitting entsteht, noch nicht vorhanden, so dass die Anwendung in diesem Fall nützlich ist. Dadurch wird anschließend die PUHR-Regel mit C_5 anwendbar:



Der rechte Ast enthält zwar alle Atome aus dem Rumpf von C_6, aber nach Definition ist auf einen Knoten $\vee^\star()$ keine weitere Regelanwendung möglich. Deshalb ist dieser Ast mit dem Symbol $\perp$ gekennzeichnet. Wäre die PUHR-Regel mit C_5 erst nach allen anderen Möglichkeiten angewandt worden, würde der Ast aber zusätzlich das Atom *froh*(*bry*) enthalten.

Von dem Atom *lehrt*(*bry*) gibt es zwei Vorkommen in diesem Tableau.

Um die Bäume übersichtlicher zu halten, werden wir ab sofort Disjunktionen mit einem einzigen Atom nicht mehr explizit darstellen. Das obige Tableau hat dann folgende kompaktere Darstellung:

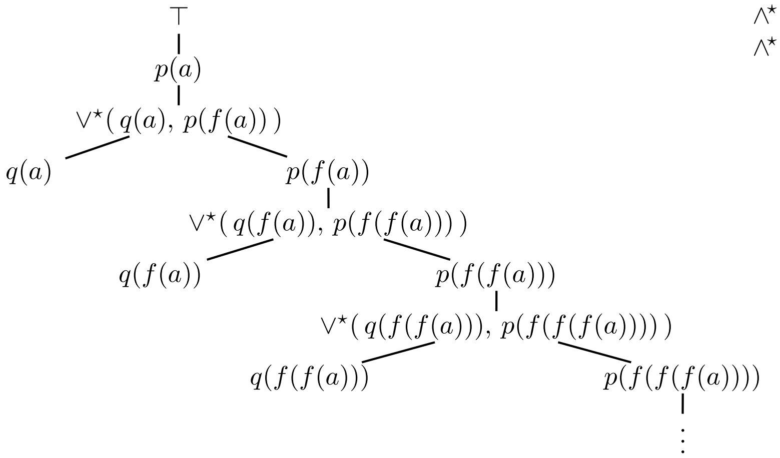


Enthält die Klauselmenge S keine Variablen, ist die PUHR-Regel mit einer Klausel anwendbar, wenn alle Atome des Klauselrumpfs im aktuellen Ast enthalten sind. Im Fall mit Variablen ist jeweils noch eine Grundsubstitution beteiligt, so dass nicht die Rumpfatome selbst, sondern ihre Instanzen im aktuellen Ast enthalten sein müssen. Diese Grundsubstitutionen können mit dem Angleichungsalgorithmus berechnet werden. Man überprüfe, dass der obige Baum auch ein PUHR-Tableau für folgende Klauselmenge ist:

$$\begin{array}{ll}
\wedge^\star() & \Rightarrow \vee^\star(semester) \\
\wedge^\star() & \Rightarrow \vee^\star(dozent(bry)) \\
\wedge^\star(dozent(x)) & \Rightarrow \vee^\star(lehrt(x), forscht(x)) \\
\wedge^\star(dozent(x), semester) & \Rightarrow \vee^\star(lehrt(x)) \\
\wedge^\star(lehrt(x), forscht(x)) & \Rightarrow \vee^\star()
\end{array}$$ ■

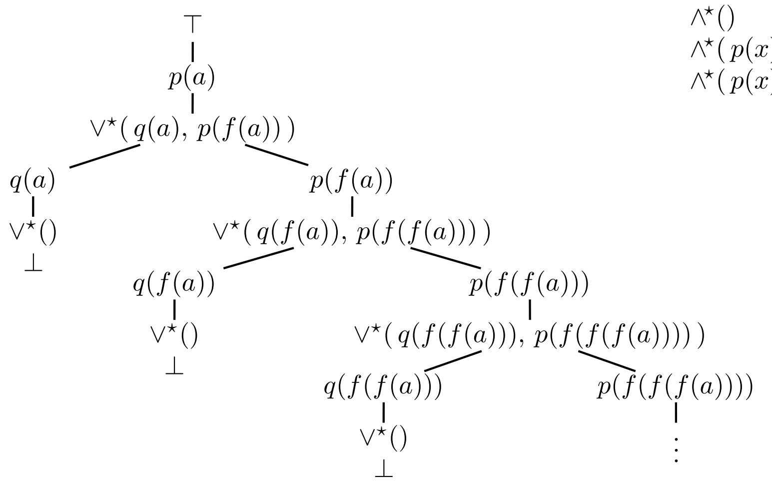
Die nächsten Beispiele illustrieren einige Phänomene im Zusammenhang mit unendlichen Tableaus.

Beispiel 4.7.13. Folgende Klauselmenge ergibt ein Tableau mit einem unendlichen Ast und unendlich vielen Blättern.



$$\begin{array}{ll}
\wedge^\star() & \Rightarrow \vee^\star(\, p(a)\,) \\
\wedge^\star(\, p(x)\,) & \Rightarrow \vee^\star(\, q(x), p(f(x))\,)
\end{array}$$

In diesem Beispiel kommt nirgends in dem unendlich großen Tableau ein Knoten $\vee^\star()$ vor. Aber mit einer zusätzlichen Klausel kann man das Beispiel so ändern, dass alle Blätter des Tableaus zu $\vee^\star()$ werden.



$$\begin{array}{ll}
\bigwedge^\star() & \Rightarrow \bigvee^\star(\,p(a)\,) \\
\bigwedge^\star(\,p(x)\,) & \Rightarrow \bigvee^\star(\,q(x),\, p(f(x))\,) \\
\bigwedge^\star(\,p(x),\, q(x)\,) & \Rightarrow \bigvee^\star()
\end{array}$$

■

Das folgende Beispiel zeigt, dass die Endlichkeit oder Unendlichkeit eines Tableaus auch davon abhängt, wie die Regeln und Klauseln ausgewählt werden.

Beispiel 4.7.14.

$$\begin{array}{c}
\top \\
| \\
p(a) \\
| \\
p(f(a)) \\
| \\
p(f(f(a))) \\
| \\
p(f(f(f(a)))) \\
\vdots
\end{array}$$

$$\begin{array}{ll}
\bigwedge^\star() & \Rightarrow \bigvee^\star(\,p(a)\,) \\
\bigwedge^\star(\,p(x)\,) & \Rightarrow \bigvee^\star(\,p(f(x))\,) \\
\bigwedge^\star(\,p(f(x)),\, p(g(x))\,) & \Rightarrow \bigvee^\star() \\
\bigwedge^\star(\,p(x)\,) & \Rightarrow \bigvee^\star(\,p(g(x))\,)
\end{array}$$

Es entsteht ein Tableau mit einem unendlichen Ast, wenn die PUHR-Regel nie mit den letzten beiden Klauseln angewandt wird. Eine Anwendung der PUHR-Regel mit der letzten Klausel ermöglicht eine Anwendung mit der dritten Klausel. Dann entsteht $\bigvee^\star()$, und damit ist das Tableau endlich. ■

Wir erinnern daran, dass ein Ast eines Baums ein von der Wurzel ausgehender Pfad maximaler Länge ist. Ein Ast ist also eine Sequenz von Knoten, die mit der Wurzel beginnt und entweder mit einem Blatt endet oder unendlich ist. Wir werden einen Ast A im folgenden meist als Menge seiner Knoten auffassen (also die Reihenfolge der Knoten in der Sequenz ignorieren) und die üblichen Mengenoperationen dafür verwenden.

Definition 4.7.15 (geschlossen [Tableau]).

1. Ein Ast eines PUHR-Tableaus heißt *geschlossen*, wenn er $\bigvee^\star()$ enthält und *offen*, wenn er $\bigvee^\star()$ nicht enthält.
2. Ein PUHR-Tableau heißt *geschlossen*, wenn alle seine Äste geschlossen sind und *offen*, wenn es wenigstens einen offenen Ast hat. ■

Alle Tableaus in den obigen Beispielen sind offen. In Beispiel 4.7.13 im zweiten Fall sind alle endlichen Äste des Tableaus geschlossen, aber der Ast ganz rechts ist unendlich und offen.

Satz 4.7.16.

1. Jedes geschlossene PUHR-Tableau ist endlich.
2. Ein offener Ast eines PUHR-Tableaus kann endlich oder unendlich sein.
3. Jeder Knoten eines PUHR-Tableaus außer der Wurzel ist eine Disjunktion von Grundatomen oder ein Grundatom.

Beweis:

1. Wenn ein Ast geschlossen ist, also $\vee^\star()$ enthält, ist dieser Knoten ein Blatt, weil nach Definition keine Erweiterungsregel auf ihn angewandt werden kann. Wenn ein Tableau geschlossen ist, endet somit jeder Ast mit einem Blatt.
2. In Beispiel 4.7.12 kommt ein endlicher offener Ast vor, in Beispiel 4.7.13 ein unendlicher offener Ast.
3. Das initiale Tableau hat die behauptete Eigenschaft.

   Sei T ein endliches Tableau, in dem jeder Knoten eine Disjunktion von Grundatomen oder ein Grundatom ist. Wird die Splitting-Regel auf T angewandt, sind die neuen Knoten Grundatome. Bei einer Anwendung der PUHR-Regel mit einer bereichsbeschränkten Klausel $(\wedge^\star(R_1, \ldots, R_n) \Rightarrow \vee^\star(K_1, \ldots, K_m))$ und Grundsubstitution σ kommt nach Definition jedes $R_i\sigma$ als Knoten in T vor und ist somit variablenfrei, also ist wegen der Bereichsbeschränktheit auch $\vee^\star(K_1, \ldots, K_m)\sigma$ variablenfrei, das heißt, der neue Knoten ist eine Disjunktion von Grundatomen. Damit besitzt jedes endliche PUHR-Tableau die behauptete Eigenschaft.

   Sei T ein unendliches Tableau, das aus einer Sequenz $\langle T_i \rangle_{i \in \mathbb{N}}$ von endlichen Tableaus gebildet ist. Da jeder Knoten von T in einem der endlichen T_i vorkommt, hat auch T die behauptete Eigenschaft. ∎

Eigenschaften der PUHR-Tableau-Beweismethode

Die PUHR-Tableau-Beweismethode ist eine Modellgenerierungsmethode für bereichsbeschränkte Klauselmengen in Implikationsnormalform. Wie jede Modellgenerierungsmethode kann sie auch als Widerlegungsmethode eingesetzt werden (siehe den Abschnitt zur Davis-Putnam-Beweismethode). Wenn ein geschlossenes PUHR-Tableau für eine Klauselmenge S konstruiert werden kann, ist S unerfüllbar. Offene Äste repräsentieren Modelle von S, sofern eine gewisse Fairness-Bedingung eingehalten wird.

Für den Rest dieses Abschnitts wird an einer Sprache $\mathcal{L}$ der Prädikatenlogik erster Stufe und an einer (möglicherweise unendlichen) Menge S von bereichsbeschränkten Klauseln in Implikationsnormalform festgehalten.

Hilfssatz 4.7.17.

1. Ist N ein Knoten in einem PUHR-Tableau für S, auf den die PUHR-Regel angewandt wurde, gilt für seinen direkten Nachfolgerknoten N' dass $\mathit{Vorfahren}(N) \cup \forall(S) \models N'$.
2. Ist N ein Knoten in einem PUHR-Tableau für S, auf den die Splitting-Regel angewandt wurde, dann gibt es zu jeder Interpretation M mit $M \models \mathit{Vorfahren}(N) \cup \forall(S)$ einen direkten Nachfolgerknoten N' von N mit $M \models N'$.

Beweis:

1. Sei M eine Interpretation mit $M \models \mathit{Vorfahren}(N) \cup \forall(S)$. Sei die PUHR-Regel angewandt mit der Instanz $C\sigma$ einer Klausel $C = (\wedge^\star(R_1, \ldots, R_n) \Rightarrow \vee^\star(K_1, \ldots, K_m)) \in S$, $n \geq 0$, $m \geq 0$. Der Nachfolgerknoten ist also $N' = \vee^\star(K_1, \ldots, K_m)\sigma$.

   Wegen $M \models \forall(S)$ gilt $M \models \forall(C)$, also erfüllt M auch die Grundinstanz $C\sigma$ von C (diese Richtung gilt für alle Interpretationen, nicht nur für Herbrand-Interpretationen). Nach Definition der PUHR-Regel ist $\{R_1\sigma, \ldots, R_n\sigma\} \subseteq \mathit{Vorfahren}(N)$, und damit gilt $M \models \wedge^\star(R_1, \ldots, R_n)\sigma$. Wegen $M \models C\sigma$ gilt auch $M \models \vee^\star(K_1, \ldots, K_m)\sigma$.
2. Sei M ein Modell von $\mathit{Vorfahren}(N) \cup \forall(S)$, und sei $\vee^\star(A_1, \ldots, A_m) \in \mathit{Vorfahren}(N)$ mit $m \geq 1$ die Disjunktion, mit der die Splitting-Regel angewandt wurde. Dann gilt $M \models \vee^\star(A_1, \ldots, A_m)$, und da es nach Satz 4.7.16 eine Disjunktion von Grundatomen ist, auch $M \models A_j$ für ein $j \in \{1, \ldots, m\}$. Die Behauptung gilt für $N' = A_j$. ∎

Satz 4.7.18. Sei M ein Modell von $\forall(S)$. Sei T ein PUHR-Tableau für S. Sei A ein endlicher Ast von T mit $M \models A$, und sei N das Blatt in A.

Bei Anwendung einer Erweiterungsregel auf N entsteht ein direkter Nachfolgerknoten N' von N, so dass $A' := A \cup \{N'\}$ ein endlicher Ast des resultierenden Tableaus ist mit $M \models A'$.

Beweis: Dass A' endlicher Ast ist, ist trivial. Da N ein Blatt in A ist, ist $\mathit{Vorfahren}(N) = A$. Unter den gegebenen Voraussetzungen gilt also $M \models \mathit{Vorfahren}(N) \cup \forall(S)$. Satz 4.7.17 garantiert die Existenz eines Nachfolgerknotens N' mit $M \models N'$ und damit $M \models A \cup \{N'\}$. ∎

Satz 4.7.19. Zu jedem Modell M von $\forall(S)$ hat jedes PUHR-Tableau für S einen Ast A mit $M \models A$.

Beweis: Sei M ein Modell von $\forall(S)$. Die Behauptung gilt für das initiale Tableau wegen $M \models \top$. Für endliche PUHR-Tableaus ergibt sie sich als Sonderfall der folgenden Konstruktion für unendliche Tableaus.

Sei T ein unendliches Tableau für S, das aus einer Sequenz $\langle T_i \rangle_{i \in \mathbb{N}}$ von endlichen Tableaus gebildet ist. Wir definieren für das gegebene M in jedem T_i einen Ast A_i wie folgt:

$A_0 \quad := \quad$ einziger Ast des initialen Tableaus T_0.

$$A_{i+1} := \begin{cases} A_i \cup \{N_i'\} & \text{falls } T_{i+1} \text{ aus } T_i \text{ entsteht durch Anwendung einer Erweiterungs-} \\ & \text{regel auf das Blatt } N_i \in A_i \text{, wobei } N_i' \text{ ein Nachfolgerknoten von} \\ & N_i \text{ gemäß Satz 4.7.18 ist.} \\ A_i & \text{falls } T_{i+1} \text{ aus } T_i \text{ entsteht durch Anwendung einer Erweiterungs-} \\ & \text{regel auf ein Blatt } N \notin A_i. \end{cases}$$

A_{i+1} ist jeweils ein endlicher Ast in T_{i+1}. Gilt $M \models A_i$, dann auch $M \models A_{i+1}$ (im oberen Fall nach Satz 4.7.18).

Für jedes $i \in \mathbb{N}$ ist damit A_i ein Ast in T_i mit $M \models A_i$. Da die Sequenz $\langle T_i \rangle_{i \in \mathbb{N}}$ beliebig ist, folgt daraus die Behauptung für jedes endliche Tableau.

Für das unendliche Tableau T berücksichtigen wir zusätzlich, dass nach Konstruktion $A_0 \subseteq A_1 \subseteq A_2 \subseteq \ldots$ gilt. Sei $A := \bigcup_{i \in \mathbb{N}} A_i$.

Dann ist A ein Ast von T mit $M \models A$. Denn angenommen, es gäbe eine Formel $F \in A$ mit $M \not\models F$, so gäbe es $i \in \mathbb{N}$ mit $F \in A_i$, also $M \not\models A_i$, Widerspruch. ∎

Satz 4.7.20 (Korrektheit bezüglich Unerfüllbarkeit). Gibt es ein geschlossenes PUHR-Tableau für S, so ist $\forall(S)$ unerfüllbar.

Beweis: Angenommen, es gäbe ein geschlossenes PUHR-Tableau T für S, aber ein Modell M von $\forall(S)$. Nach Satz 4.7.19 hat T einen Ast A mit $M \models A$. Da T geschlossen ist, enthält A die Formel $\vee^\star()$ und wird damit von keiner Interpretation erfüllt. Widerspruch. ∎

Für die übrigen Eigenschaften der PUHR-Tableau-Beweismethode ist noch eine Voraussetzung erforderlich. Offensichtlich kann nicht gelten, dass jeder offene Ast jedes beliebigen PUHR-Tableaus für S ein Modell von S repräsentiert, weil zum Beispiel das initiale Tableau unabhängig von S immer aus einem offenen Ast besteht. Offene Äste, in denen Regelanwendungen möglich, aber noch nicht durchgeführt sind, erlauben keine Aussage über die Erfüllbarkeit von S. Die Äste, die eine Aussage erlauben, werden durch die Fairness-Eigenschaft charakterisiert.

Definition 4.7.21 (fair [Tableau]).

1. Ein Ast A eines PUHR-Tableaus T für S heißt *fair*, wenn er geschlossen ist oder wenn er offen ist und folgende Eigenschaften hat:
   a) Für jede Klausel $(\wedge^\star(R_1, \ldots, R_n) \Rightarrow \vee^\star(K_1, \ldots, K_m)) \in S$ und für jede Grundsubstitution σ mit $\{R_1\sigma, \ldots, R_n\sigma\} \subseteq A$ gilt auch $\vee^\star(K_1, \ldots, K_m)\sigma \in A$.
   b) Wenn $\vee^\star(A_1, \ldots, A_m) \in A$ mit $m \geq 1$, dann auch $A_j \in A$ für ein $j \in \{1, \ldots, m\}$.
2. Ein PUHR-Tableau heißt *fair*, wenn alle seine Äste fair sind. ∎

Man sagt auch, dass ein fairer Ast bezüglich der Erweiterungsregeln *gesättigt* ist. Faire Äste können sowohl endlich als unendlich sein.

Die PUHR-Tableaus aus Beispiel 4.7.13 sind fair. Das unendliche, offene PUHR-Tableau aus Beispiel 4.7.14 ist nicht fair, wenn es in der angegebenen Weise weiterentwickelt wird. Ein faires PUHR-Tableau für die Klauselmenge dieses Beispiels wäre:

$$
\begin{array}{c}
\top \\
| \\
p(a) \\
| \\
p(f(a)) \\
| \\
p(f(f(a))) \\
| \\
p(f(f(f(a)))) \\
| \\
p(g(f(f(a)))) \\
| \\
\vee^\star() \\
\bot
\end{array}
$$

Dieses Tableau ist fair, weil es geschlossen ist. Es gibt unendlich viele andere geschlossene und damit faire PUHR-Tableaus, aber kein offenes faires PUHR-Tableau für diese Klauselmenge.

Notation 4.7.22. Ist A ein Ast eines PUHR-Tableaus für S, bezeichnet $\mathit{Atome}(A)$ die Menge der atomaren Formeln in A. ∎

Satz 4.7.23 (Korrektheit bezüglich Modellen). Für jeden offenen Ast A eines fairen PUHR-Tableaus für S gilt $\mathcal{H}_{\mathcal{L}}(Atome(A)) \models \forall(S)$.

Beweis: Zu zeigen ist, dass $\mathcal{H}_{\mathcal{L}}(Atome(A)) \models \forall(C)$ für jedes $C \in S$. Sei also $C = (\wedge^\star(R_1, \ldots, R_n) \Rightarrow \vee^\star(K_1, \ldots, K_m)) \in S$ beliebig. Seien $x_1, \ldots, x_k$ die Variablen in C.

Angenommen, $\mathcal{H}_{\mathcal{L}}(Atome(A)) \not\models \forall(C)$. Nach Satz 3.8.10 gibt es Grundterme $t_1, \ldots, t_k$ mit $\mathcal{H}_{\mathcal{L}}(Atome(A)) \not\models C[t_1/x_1] \ldots [t_k/x_k]$. Für die Grundsubstitution $\sigma = \{t_1/x_1, \ldots, t_k/x_k\}$ gilt damit $\mathcal{H}_{\mathcal{L}}(Atome(A)) \not\models C\sigma$, nach Definition von $\models$ also $\mathcal{H}_{\mathcal{L}}(Atome(A)) \models \wedge^\star(R_1, \ldots, R_n)\sigma$ und $\mathcal{H}_{\mathcal{L}}(Atome(A)) \not\models \vee^\star(K_1, \ldots, K_m)\sigma$.

Alle $R_i\sigma$ sind Grundatome, also gilt für die von $Atome(A)$ repräsentierte Herbrand-Interpretation $\mathcal{H}_{\mathcal{L}}(Atome(A)) \models R_i\sigma$ gdw. $R_i\sigma \in Atome(A)$. Wegen $\mathcal{H}_{\mathcal{L}}(Atome(A)) \models \wedge^\star(R_1\sigma, \ldots, R_n\sigma)$ gilt damit, dass $\{R_1\sigma, \ldots, R_n\sigma\} \subseteq Atome(A) \subseteq A$ ist. Da A fair ist, gilt auch $\vee^\star(K_1\sigma, \ldots, K_m\sigma) \in A$. Da A offen ist, ist $m \geq 1$, und die Fairness garantiert dass $K_j\sigma \in A$ für ein $j \in \{1, \ldots, m\}$.

Nun ist aber $K_j\sigma$ ein Grundatom, also $K_j\sigma \in Atome(A)$, somit gilt $\mathcal{H}_{\mathcal{L}}(Atome(A)) \models K_j\sigma$ und deshalb $\mathcal{H}_{\mathcal{L}}(Atome(A)) \models \vee^\star(K_1\sigma, \ldots, K_m\sigma)$, Widerspruch. ■

Satz 4.7.24 (Vollständigkeit bezüglich Unerfüllbarkeit). Ist $\forall(S)$ unerfüllbar, so ist jedes faire Tableau für S geschlossen.

Beweis: Die Kontraposition folgt unmittelbar aus der Korrektheit bezüglich Modellen. ■

Mit diesen Ergebnissen kann die PUHR-Tableau-Beweismethode als Semi-Entscheidungsverfahren für die Unerfüllbarkeit einer endlichen Menge von bereichsbeschränkten Klauseln in Implikationsnormalform benutzt werden.

Beginnend mit dem initialen Tableau wendet man wiederholt die Erweiterungsregeln an. Die Auswahl muss dabei sicherstellen, dass das entstehende Tableau fair ist. Das bedeutet unter anderem, dass jede Klausel und jede Grundsubstitution, mit der die PUHR-Regel in irgend einem Ast anwendbar ist, nach endlich vielen Schritten auch tatsächlich für diesen Ast ausgewählt wird. Für ein endliches S ist dies nicht besonders schwierig, und auch für ein unendliches, aufzählbares S wäre es möglich.

Ist S unerfüllbar, garantiert die Vollständigkeit bezüglich der Unerfüllbarkeit zusammen mit der Tatsache, dass jedes geschlossene PUHR-Tableau endlich ist, dass jede Auswahl, die die Fairness sicherstellt, nach endlich vielen Schritten zu einem geschlossenen Tableau führt.

Da die Erfüllbarkeit von S nicht entscheidbar ist, kann die PUHR-Tableau-Beweismethode nicht vollständig bezüglich der Modelle sein. Auf den ersten Blick scheint dies dem Satz 4.7.19 zu widersprechen, der doch sicherstellt, dass jedes Modell von S in jedem PUHR-Tableau einem Ast entspricht. Aber bei der Konstruktion eines fairen PUHR-Tableaus weiß man im allgemeinen nicht, ob ein Ast bei weiterer Anwendung der Erweiterungsregeln irgendwann geschlossen wird und damit kein Modell repräsentiert. Die Nicht-Entscheidbarkeit äußert sich hier in dem Phänomen, dass einige Modelle durch unendliche faire Äste repräsentiert sind, so dass die Ungewissheit, ob der Ast vielleicht irgendwann geschlossen wird, nie beseitigt wird. In Beispiel 4.7.13 ist die zweite Klauselmenge erfüllbar, aber nicht mit einer Herbrand-Interpretation, die durch einen endlichen Ast repräsentiert werden kann. In solchen Fällen terminiert eine faire Anwendung der Erweiterungsregeln nicht.

Für Spezialfälle, zum Beispiel wenn die Sprache $\mathcal{L}$ nur endlich viele Grundterme ermöglicht, kann man aus Satz 4.7.19 allerdings Vollständigkeitsergebnisse bezüglich der Modelle gewinnen.

Optimierungen

Die PUHR-Tableau-Beweismethode beruht auf der Konstruktion eines fairen PUHR-Tableaus für die gegebene Klauselmenge S. Diese Konstruktion kann durch eine Reihe von Optimierungen verbessert werden.

Anwendbarkeitsbedingungen. Wie bereits in Beispiel 4.7.12 erläutert, sind Wiederholungen von Regelanwendungen nutzlos. Man kann die Bedingungen für die Anwendbarkeit der Erweiterungsregeln wie folgt verschärfen. Die PUHR-Regel ist für eine Klauselinstanz mit Kopf $\vee^\star(K_1, \ldots, K_m)\sigma$ nur dann anwendbar, wenn weder diese Disjunktion noch eines der $K_j\sigma$ im Ast vorkommen. Die Splitting-Regel ist auf eine Disjunktion $\vee^\star(A_1, \ldots, A_m)$ nur dann anwendbar, wenn keines der A_j im Ast vorkommt. Man überprüft leicht, dass durch diese Einschränkungen die Fairness nicht verletzt wird.

Dadurch werden nicht nur Wiederholungen ausgeschlossen, sondern auch andere Redundanzen. Enthält die Klauselmenge zum Beispiel $\wedge^\star() \Rightarrow \vee^\star(p)$ und $\wedge^\star() \Rightarrow \vee^\star(p, q)$ und ist in einem Ast die PUHR-Regel mit der ersten Klausel angewandt worden, dann ist sie danach in diesem Ast nicht mehr mit der zweiten anwendbar. Die Optimierung ähnelt der Subsumption-Vereinfachung bei der Davis-Putnam-Beweismethode.

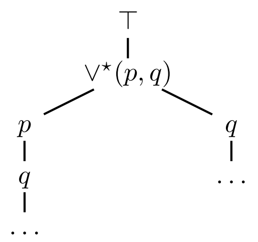
Bevorzugung kleinerer Disjunktionen. In Beispiel 4.7.12 wird gegen Ende ein Tableau erreicht, in dessen rechtem Ast die PUHR-Regel mit zwei Klauselinstanzen anwendbar ist: $C_5 = \wedge^\star(\mathit{lehrt}(\mathit{bry}), \mathit{forscht}(\mathit{bry})) \Rightarrow \vee^\star()$ und $C_6 = \wedge^\star(\mathit{forscht}(\mathit{bry})) \Rightarrow \vee^\star(\mathit{froh}(\mathit{bry}))$.

Im Beispiel wurde die PUHR-Regel mit C_5 angewandt und der Ast geschlossen. Wäre die Anwendung mit C_6 vorgezogen worden, wäre der Ast danach durch die Anwendung mit C_5 ebenfalls geschlossen worden. Es ist in solchen Fällen also offensichtlich günstiger, die Klauselinstanzen mit leerem Kopf vorzuziehen.

Analog verhält es sich in Fällen, wo eine Klauselinstanz ein einziges Atom im Kopf enthält: $\wedge^\star() \Rightarrow \vee^\star(p)$ und $\wedge^\star() \Rightarrow \vee^\star(q, r, s)$. Würde man hier die PUHR-Regel zuerst mit der zweiten Klausel anwenden, müsste nach dem Splitting in jedem der drei Äste die PUHR-Regel mit der ersten Klausel angewandt werden. Bei der umgekehrten Reihenfolge benötigt man mit jeder Klausel nur eine Regelanwendung.

Die Verallgemeinerung dieser Beobachtungen besteht darin, bei Anwendungen der PUHR-Regel die Klauselinstanzen mit möglichst wenigen Atomen im Kopf zu bevorzugen. Allerdings dürfen dadurch die „großen“ Klauseln von den „kleinen“ nicht so stark verdrängt werden, dass die Fairness verletzt würde.

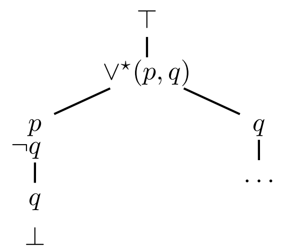
Komplement-Splitting. Es kann vorkommen, dass Äste eines PUHR-Tableaus bezüglich ihrer Atome Teilmengen voneinander sind:



$$
\begin{aligned}
\wedge^\star() &\Rightarrow \vee^\star(p, q)\\
\wedge^\star(p) &\Rightarrow \vee^\star(q)\\
\wedge^\star(q) &\Rightarrow \ldots
\end{aligned}
$$

Die Klauselmenge kann so erweitert werden, dass anstelle der drei Punkte ein beliebig großer Teilbaum entsteht, der zweimal bearbeitet wird.

Dies wird verhindert, wenn die Splitting-Regel dafür sorgt, dass bei Verzweigungen disjunkte Fälle entstehen (bei der Davis-Putnam-Beweismethode ergeben Verzweigungen ja auch stets disjunkte Fälle). Zum Beispiel kann der Fall p verschärft werden zu $p \wedge \neg q$. Das erfordert Erweiterungen, weil damit auch negative Grundliterale im Tableau vorkommen können und ein Ast auch dann als geschlossen gelten muss, wenn er zwei komplementäre Literale enthält. Mit diesen Modifikationen entsteht ein PUHR-Tableau, in dem der fragliche Teilbaum nur noch einmal vorkommt.



Die Komplement-Splitting-Regel hat folgende allgemeine Gestalt:

$$\begin{array}{c|c|c|c}
\multicolumn{4}{c}{\vee^\star(A_1, A_2, \ldots, A_m)} \\
\hline
A_1 & & & \\
\neg A_2 & A_2 & & \\
\vdots & \vdots & & \\
\neg A_m & \neg A_m & \ldots & A_m
\end{array}$$

Wie bei der einfachen Splitting-Regel wird jedes Atom der Disjunktion als Nachfolgerknoten hinzugefügt, aber zusätzlich kommen jeweils die Negationen aller Atome hinzu, die weiter rechts in der Disjunktion vorkommen.

Die Eigenschaften der PUHR-Tableau-Beweismethode bleiben erhalten, wenn anstelle der Splitting-Regel diese Komplement-Splitting-Regel verwendet wird. In vielen Fällen wird die Effizienz dadurch verbessert. Obendrein ist die Komplement-Splitting-Regel die Grundlage für eine Variante der Methode, mit der die sogenannten „minimalen Modelle“ generiert werden können.

Syntax-Erweiterungen. Die Konjunktion von zwei Klauseln $(\wedge^\star(R_1, \ldots, R_n) \Rightarrow K)$ und $(\wedge^\star(R_1, \ldots, R_n) \Rightarrow K')$, die den gleichen Rumpf haben, ist logisch äquivalent zu der Formel $(\wedge^\star(R_1, \ldots, R_n) \Rightarrow (K \wedge K'))$. Diese Formel ist aber keine Klausel.

Wären auch Formeln mit Konjunktionen im Kopf erlaubt, hätte dies aber operationale Vorteile. Für die Anwendbarkeit der PUHR-Regel muss mit Hilfe des Angleichungsalgorithmus eine Grundsubstitution σ berechnet werden, so dass $\{R_1\sigma, \ldots, R_n\sigma\} \subseteq A$ für den aktuellen Ast A gilt. Diese Berechnung wäre für die beiden Klauseln gleich und müsste für die Formel mit der Konjunktion im Kopf nur einmal durchgeführt werden.

Die PUHR-Tableau-Beweismethode lässt sich verallgemeinern, so dass sie andere Formeln als Klauseln in Implikationsnormalform behandeln kann, wobei diese Formeln eine entsprechende Verallgemeinerung der Bereichsbeschränktheit einhalten müssen.

Die Bereichsbeschränkungs-Transformation

Die behandelten Ergebnisse der PUHR-Tableau-Beweismethode gelten für eine Menge S von bereichsbeschränkten Klauseln in Implikationsnormalform. Für nicht-bereichsbeschränkte Klauseln gelten die Ergebnisse nicht. Es wäre nicht mehr garantiert, dass alle Knoten eines PUHR-Tableaus variablenfrei sind. Diese Eigenschaft war aber wesentlich, zum Beispiel für die Korrektheit bezüglich der Unerfüllbarkeit (genau: für Satz 4.7.17 (2)).

Nach dem Normalformensatz für die Prädikatenlogik erster Stufe (Satz 4.4.14) kann jede endliche Menge von $\mathcal{L}$-Formeln in eine im wesentlichen gleichwertige endliche Klauselmenge übersetzt werden. Für viele Anwendungen wird diese Klauselmenge bereichsbeschränkt sein. Bereichsbeschränkte Klauseln entsprechen beschränkt quantifizierten Formeln (Abschnitt 2.7) und ergeben sich deshalb häufig aus den „natürlichen“ Formalisierungen. Aber es gibt auch Formelmengen, deren Klauselform nicht bereichsbeschränkt ist.

Wir zeigen nun noch, dass jede Klauselmenge in eine im wesentlichen gleichwertige Klauselmenge übersetzt werden kann, die bereichsbeschränkt ist. Damit ist die Bereichsbeschränktheit keine echte Einschränkung und die PUHR-Tableau-Beweismethode zusammen mit dieser Übersetzung ist eine Beweismethode mit den genannten Eigenschaften für beliebige geschlossene Formeln der Prädikatenlogik erster Stufe.

Für den Rest dieses Abschnitts sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe, die mindestens eine Konstante enthält. Sei S eine Menge von $\mathcal{L}$-Klauseln in Implikationsnormalform. Sei $\mathcal{L}_{\dot{D}}$ eine Erweiterung der Sprache $\mathcal{L}$ um ein einstelliges Relationssymbol $\dot{D}$.

Definition 4.7.25 (Bereichsbeschränkungs-Transformation). Für jede Klausel $C \in S$ der Gestalt $(\wedge^\star(R_1, \dots, R_n) \Rightarrow \vee^\star(K_1, \dots, K_m))$ sei

$$BB(C) := \begin{cases} C & \text{falls } C \text{ bereichsbeschränkt ist} \\ (\wedge^\star(\dot{D}(x_1), \dots, \dot{D}(x_k), R_1, \dots, R_n) \Rightarrow \vee^\star(K_1, \dots, K_m)) & \\ \text{falls } x_1, \dots, x_k \text{ die Variablen sind, die im Kopf von } C \text{ vorkommen,} & \\ \text{aber nicht im Rumpf.} & \end{cases}$$

$$\begin{aligned} BB(\mathcal{L}) := \;& \{\ (\wedge^\star() \Rightarrow \vee^\star(\dot{D}(c)))\ \mid\ c \text{ Konstante in } \mathcal{L}\ \} \ \cup \\ & \{\ (\wedge^\star(\dot{D}(x_1), \dots, \dot{D}(x_n) \Rightarrow \vee^\star(\dot{D}(f(x_1, \dots, x_n))))\ \mid \\ & \qquad\qquad f\ n\text{-stelliges Funktionssymbol in } \mathcal{L}\ \} \\ BB(S) := \;& \{BB(C)\ \mid\ C \in S\} \ \cup\ BB(\mathcal{L}) \end{aligned}$$

■

Die Menge S von $\mathcal{L}$-Klauseln wird also übersetzt in eine Menge $BB(S)$ von $\mathcal{L}_{\dot{D}}$-Klauseln, die nach Konstruktion bereichsbeschränkt sind. Ist S endlich und besteht die Sprache $\mathcal{L}$ genau aus den Symbolen, die in S verwendet werden (plus einer Konstanten, falls in S keine vorkommt), dann ist $BB(S)$ ebenfalls endlich.

Satz 4.7.26. Sei M eine $\mathcal{L}$-Herbrand-Interpretation und sei $M_{\dot{D}}$ die $\mathcal{L}_{\dot{D}}$-Herbrand-Interpretation, die sämtliche Grundatome $\dot{D}(t)$ erfüllt und ansonsten mit M übereinstimmt.

Es gilt: wenn $M \models \forall(S)$, dann $M_{\dot{D}} \models \forall(BB(S))$.

Beweis: Es gelte $M \models \forall(S)$.

Nach Satz 3.8.10 erfüllt $M_{\dot{D}}$ den Allabschluss einer Klausel genau dann, wenn es jede Grundinstanz der Klausel erfüllt. Da $M_{\dot{D}}$ sämtliche Grundatome $\dot{D}(t)$ erfüllt, erfüllt $M_{\dot{D}}$ auch jede Grundinstanz jeder Klausel aus $BB(\mathcal{L})$, das heißt, $M_{\dot{D}} \models BB(\mathcal{L})$. Es bleibt zu zeigen, dass $M_{\dot{D}} \models \forall(BB(C))$ für jedes $C \in S$ gilt.

Falls C bereichsbeschränkt ist, gilt $BB(C) = C$, und das Relationssymbol $\dot{D}$ kommt nicht in C vor. Da in diesem Fall $M_{\dot{D}}$ mit M übereinstimmt und nach Voraussetzung $M \models \forall(C)$, gilt auch $M_{\dot{D}} \models \forall(BB(C))$.

Falls $BB(C) = (\wedge^\star(\dot{D}(x_1), \ldots, \dot{D}(x_k), R_1, \ldots, R_n) \Rightarrow \vee^\star(K_1, \ldots, K_m))$, sei σ eine beliebige Grundsubstitution, die alle Variablen in $BB(C)$ instanziiert. Da $M_{\dot{D}}$ jedes Grundatom $\dot{D}(x_i)\sigma$ erfüllt, gilt $M_{\dot{D}} \models BB(C)\sigma$ gdw. $M_{\dot{D}} \models C\sigma$, was mit der gleichen Begründung wie im vorigen Fall gilt. Da σ beliebig ist, gilt $M_{\dot{D}} \models \forall(BB(C))$. ■

Satz 4.7.27. Sei $M_{\dot{D}}$ eine $\mathcal{L}_{\dot{D}}$-Herbrand-Interpretation und sei M die Restriktion auf die Symbole von $\mathcal{L}$, also eine $\mathcal{L}$-Herbrand-Interpretation.

Es gilt: wenn $M_{\dot{D}} \models \forall(BB(S))$, dann $M \models \forall(S)$.

Beweis: Es gelte $M_{\dot{D}} \models \forall(BB(S))$.

Wegen $M_{\dot{D}} \models \forall(BB(\mathcal{L}))$ erhält man mit einer trivialen strukturellen Induktion, dass für jeden Grundterm t gilt $M_{\dot{D}} \models \dot{D}(t)$.

Sei $C \in S$. Ist C bereichsbeschränkt, so ist $C \in BB(S)$, und nach Voraussetzung gilt $M_{\dot{D}} \models \forall(C)$. Da in C das Relationssymbol $\dot{D}$ nicht vorkommt, gilt auch $M \models \forall(C)$.

Falls $C = (\wedge^\star(R_1, \ldots, R_n) \Rightarrow \vee^\star(K_1, \ldots, K_m))$ nicht bereichsbeschränkt ist, sei σ eine beliebige Grundsubstitution, die alle Variablen in C instanziiert. Damit instanziiert sie auch alle Variablen in $BB(C)$. Nach Voraussetzung gilt $M_{\dot{D}} \models BB(C)\sigma$. Da $M_{\dot{D}}$ jedes Grundatom $\dot{D}(x_i)\sigma$ erfüllt, gilt $M_{\dot{D}} \models BB(C)\sigma$ gdw. $M_{\dot{D}} \models C\sigma$. Da in C das Relationssymbol $\dot{D}$ nicht vorkommt, gilt auch $M \models C\sigma$. Da σ beliebig ist, gilt $M \models \forall(C)$. ■

Folgesatz 4.7.28. Zu jeder Menge S von Klauseln in Implikationsnormalform gibt es eine Menge S' von bereichsbeschränkten Klauseln in Implikationsnormalform, so dass $\forall(S)$ erfüllbar ist gdw. $\forall(S')$ erfüllbar ist. ■

Der Zusammenhang ist also ähnlich wie bei der Skolemisierung. Da die Transformation die Sprache $\mathcal{L}$ erweitert, sind S und S' nicht logisch äquivalent, aber „fast“.

Die hier beschriebene Transformation soll lediglich zeigen, dass die Bereichsbeschränktheit keine prinzipielle Einschränkung ist. Operational betrachtet hat die obige Transformation aber gewisse Nachteile. Es gibt andere Transformationen, die in dieser Hinsicht besser sind.

4.8 Exkurs: Deklarative Semantik von definiten Logikprogrammen

Im gesamten Abschnitt wird an einer Sprache $\mathcal{L}$ der Prädikatenlogik erster Stufe festgehalten.

Definition 4.8.1 (Programmklausel).

- Eine *definite Programmklausel* ist ein Ausdruck entweder der Gestalt „K :- $A_1, \ldots, A_n$.“ mit $n \geq 1$ oder der Gestalt „K.“, wobei K und die A_i $(i = 1, \ldots, n)$ $\mathcal{L}$-Atome sind.

  Hat eine Programmklausel die Gestalt „K :- $A_1, \ldots, A_n$.“, heißt das Atom K der Kopf der Programmklausel, der Ausdruck $A_1, \ldots, A_n$ der Rumpf der Programmklausel.
  Hat eine Programmklausel die Gestalt „K.“, heißt das Atom K der Kopf der Programmklausel.

- Ein *definites Logikprogramm* ist eine endliche Menge von definiten Programmklauseln.

■

Bemerkungen: $\top$ und $\perp$ sind keine Atome und kommen in definiten Programmklauseln nicht vor. Eine Programmklausel der Gestalt „K.“ wird Faktum (auch: Fakt) genannt. Man sagt sowohl, dass der Rumpf eines Faktums leer ist, als auch, dass ein Faktum keinen Rumpf besitzt. ∎

Definition 4.8.2 (Programmklausel/Formel).

- Eine definite Programmklausel G der Gestalt K :- $A_1, \dots, A_n$. repräsentiert die Klausel $F(G) := (\wedge^\star(A_1, \dots, A_n) \Rightarrow \vee^\star(K))$ und damit die geschlossene universelle $\mathcal{L}$-Formel $\forall(F(G))$.
- Eine definite Programmklausel G der Gestalt K. repräsentiert die unäre Klausel $F(G) := \vee^\star(K)$ und damit die geschlossene universelle $\mathcal{L}$-Formel $\forall(F(G))$.
- Für ein definites Logikprogramm P ist $F(P)$ die endliche Menge $\{F(G) \mid G \in P\}$ von Klauseln.
- Ein Ausdruck R der Gestalt $A_1, \dots, A_n$ (also keine Programmklausel) repräsentiert die $\mathcal{L}$-Formel $F(R) := \wedge^\star(A_1, \dots, A_n)$. ∎

Definition 4.8.3 (bereichsbeschränkt [Programmklausel]).

- Eine definite Programmklausel heißt bereichsbeschränkt, wenn jede Variable, die im Kopf der Programmklausel vorkommt, ebenfalls in ihrem Rumpf vorkommt.
- Ein definites Logikprogramm heißt bereichsbeschränkt, wenn alle seine Programmklauseln bereichsbeschränkt sind. ∎

Offensichtlich kommt in einem bereichsbeschränkten Fakt keine Variable vor.

Semantik des kleinsten Herbrand-Modells

Definition 4.8.4 (Herbrand-Modell [Logikprogramm]). Seien P ein definites Logikprogramm und M eine Teilmenge der Herbrand-Basis $HB_{\mathcal{L}}$ von $\mathcal{L}$. Die Menge M von Grundatomen wird Herbrand-Modell (kurz: Modell) von P genannt, wenn die von M repräsentierte Herbrand-Interpretation $\mathcal{H}_{\mathcal{L}}(M)$ die Formelmenge $\forall(F(P))$ erfüllt. ∎

Wir benutzen hier also die früher bereits erwähnte Sprechweise, eine Menge von Grundatomen mit der davon repräsentierten Herbrand-Interpretation gleichzusetzen und selbst als Interpretation bzw. Modell zu bezeichnen.

Satz 4.8.5. Sei P ein definites Logikprogramm. Seien M und N zwei Teilmengen von $HB_{\mathcal{L}}$, die Herbrand-Modelle von P sind.
$M \cap N$ ist ein Herbrand-Modell von P.

Beweis: Nach Satz 3.8.10 (1) ist eine Herbrand-Interpretation genau dann ein Modell von P, wenn sie für jede definite Programmklausel $G \in P$ und für jede Grundsubstitution σ, die alle Variablen in G instanziiert, die Formel $F(G)\sigma$ erfüllt. Sei also $G \in P$ und σ eine solche Grundsubstitution.

1. Hat G die Gestalt K., ist $F(G)\sigma = K\sigma$ ein Grundatom. Nach Voraussetzung gilt $\mathcal{H}_{\mathcal{L}}(M) \models K\sigma$ und $\mathcal{H}_{\mathcal{L}}(N) \models K\sigma$. Für ein Grundatom heißt das $K\sigma \in M$ und $K\sigma \in N$. Daraus folgt $K\sigma \in M \cap N$ und $\mathcal{H}_{\mathcal{L}}(M \cap N) \models K\sigma$, also $\mathcal{H}_{\mathcal{L}}(M \cap N) \models F(G)\sigma$.

2. Hat G die Gestalt K :- R. und gilt $\mathcal{H}_\mathcal{L}(M \cap N) \not\models F(R)\sigma$, gilt $\mathcal{H}_\mathcal{L}(M \cap N) \models F(G)\sigma$. Es bleibt also der Fall $\mathcal{H}_\mathcal{L}(M \cap N) \models F(R)\sigma$, für den zu zeigen ist $\mathcal{H}_\mathcal{L}(M \cap N) \models F(K)\sigma$.

   Da $F(R)$ eine Konjunktion von Atomen ist, gilt sowohl $\mathcal{H}_\mathcal{L}(M) \models F(R)\sigma$ als auch $\mathcal{H}_\mathcal{L}(N) \models F(R)\sigma$ (dies wäre im allgemeinen nicht der Fall, wenn $F(R)\sigma$ Teilformeln negativer Polarität enthielte). Da M sowie N Herbrand-Modelle von P sind und $F(G)\sigma$ erfüllen, gilt $\mathcal{H}_\mathcal{L}(M) \models K\sigma$ und $\mathcal{H}_\mathcal{L}(N) \models K\sigma$. Mit der gleichen Begründung wie im ersten Fall folgt daraus $\mathcal{H}_\mathcal{L}(M \cap N) \models K\sigma$. ∎

Definition 4.8.6 (Intendiertes Modell [Logikprogramm]). Sei P ein definites Logikprogramm. Das intendierte Modell von P ist das kleinste Herbrand-Modell von P, d.h., der Durchschnitt aller Herbrand-Modelle von P. ∎

Das kleinste Herbrand-Modell eines definiten Logikprogamms P ist nur dann wohldefiniert, wenn gilt:

1. Die Menge aller Herbrand-Modelle von P ist nicht leer.
2. Der Durchschnitt einer Menge von Herbrand-Modellen von P ist ein Herbrand-Modell von P.

Die erste Bedingung gilt, weil die Herbrand-Interpretation $\mathcal{H}_\mathcal{L}(HB_\mathcal{L})$, die alle Grundatome erfüllt, offensichtlich ein Modell von jedem definiten Logikprogramm ist. Die zweite Bedingung folgt aus Satz 4.8.5.

Fixpunkt-Semantik

Definite Programmklauseln repräsentieren Klauseln in Implikationsnormalform. Sind die Programmklauseln eines definiten Logikprogramms P bereichsbeschränkt, kann ein PUHR-Tableau für $F(P)$ expandiert werden.

Satz 4.8.7. Sei P ein definites Logikprogramm, dessen Programmklauseln bereichsbeschränkt sind. Sei T ein faires PUHR-Tableau für $F(P)$.

T ist offen, besitzt einen einzigen Ast A und $\mathcal{H}_\mathcal{L}(Atome(A))$ ist das kleinste Herbrand-Modell von P.

Beweis: Da in einem definiten Logikprogramm keine Disjunktion von mehr als einem Atom vorkommt, führt die Splitting-Regel in T nicht zu Verzweigungen. Also besitzt T nur einen einzigen Ast A. Da $\vee^\star()$ in einem definiten Logikprogramm nicht vorkommt, kann kein Knoten von T ein Vorkommen von $\vee^\star()$ sein, also ist A und damit T offen.

Sei M das kleinste Herbrand-Modell von P. Nach Satz 4.7.19 gilt, da A der einzige Ast von T ist, $\mathcal{H}_\mathcal{L}(M) \models A$. Wegen $Atome(A) \subseteq A$ folgt daraus $\mathcal{H}_\mathcal{L}(M) \models Atome(A)$ und damit $Atome(A) \subseteq M$.

Angenommen, es gäbe $K \in M$ mit $K \notin Atome(A)$. Dann wird $F(P) \cup \{(\wedge^\star(K) \Rightarrow \vee^\star())\}$ nicht von $\mathcal{H}_\mathcal{L}(M)$ erfüllt, und wegen der Minimalität von M ist diese Formelmenge unerfüllbar. Da $K \notin Atome(A)$ und A der einzige Ast von T ist, ist T ein faires offenes PUHR-Tableau nicht nur für $F(P)$, sondern auch für $F(P) \cup \{(\wedge^\star(K) \Rightarrow \vee^\star())\}$. Nach Satz 4.7.24 (Vollständigkeit bezüglich Unerfüllbarkeit) ist T geschlossen. Widerspruch. Also gilt auch $M \subseteq Atome(A)$. ∎

Dieses Ergebnis ermöglicht eine operationale Charakterisierung des intendierten Modells eines definiten Logikprogramms. Es entsteht durch wiederholte Anwendung der PUHR-Tableau-Erweiterungsregeln, solange dies möglich ist, bis also ein „Fixpunkt“ erreicht ist – was allerdings abzählbar unendlich viele Schritte erfordern kann.

Die Fixpunkt-Semantik von definiten Logikprogrammen, die nicht bereichsbeschränkt sind, lässt sich ähnlich definieren.

Nichtdefinite Logikprogramme

Nichtdefinite Logikprogramme sind endliche Mengen von Programmklauseln der Gestalt „K :- $L_1, \ldots, L_n$.“ mit $n \geq 1$ oder der Gestalt „K.“, wobei K ein Atom ist und die L_i ($i = 1, \ldots, n$) positive oder negative Literale sind. Ein negatives Literal wird in einer Programmklausel als `not` A dargestellt.

In der Logikprogrammierung wird die Negation ähnlich behandelt wie in relationalen Datenbanken (siehe Abschnitt 3.9). Wie der Gültigkeitsbegriff für nichtdefinite Logikprogramme formalisiert wird, ist aber eine viel schwierigere Frage als für relationale Datenbanken. Diese Frage kann in verschiedenen Weisen beantwortet werden, über die es keinen allgemeinen Konsens gibt. Für die meisten praktischen Logikprogramme ist die intendierte Semantik der Negation allerdings unumstritten und relativ einfach.

Zu den problematischen nichtdefiniten Programmen gehört das folgende:

```
mensch(a).
weiblich(x)  :- mensch(x), not maennlich(x).
maennlich(x) :- mensch(x), not weiblich(x).
```

Für dieses Logikprogramm gibt es zwei offensichtliche Herbrand-Modelle:

$$W = \{\texttt{mensch(a)}, \texttt{weiblich(a)}\} \quad \text{und} \quad M = \{\texttt{mensch(a)}, \texttt{maennlich(a)}\}$$

Aber $W \cap M = \{\texttt{mensch(a)}\}$ ist kein Herbrand-Modell des Logikprogramms. Satz 4.8.5 gilt also für nichtdefinite Logikprogramme nicht. W und M sind jedoch beide „minimal“ in dem Sinne, dass keine echte Teilmenge von W oder von M ein Herbrand-Modell des Logikprogramms ist.

Zur Bezeichnung „definit“

Die Bezeichnung „definit“ für Programmklauseln und Logikprogramme kommt daher, dass im Gegensatz zu dem vorangehenden Beispiel jedes definite Logikprogramm ein eindeutiges intendiertes Modell besitzt.

4.9 Der Endlichkeitssatz

Satz 4.9.1 (Endlichkeits- oder Kompaktheitssatz [PL1S]). Sei S eine unendliche Menge von geschlossenen Formeln einer Sprache $\mathcal{L}$ der Prädikatenlogik erster Stufe. Ist jede endliche Teilmenge von S erfüllbar, so ist S erfüllbar.

Beweis: Da $\mathcal{L}$ abzählbar ist und es somit insgesamt nur abzählbar viele $\mathcal{L}$-Formeln gibt, ist S abzählbar unendlich.

Nach Satz 3.8.22 gibt es eine Erweiterung $\mathcal{L}_{sko}$ von $\mathcal{L}$ um abzählbar viele Funktionssymbole und eine abzählbar unendliche Menge S_{sko} von universellen geschlossenen $\mathcal{L}_{sko}$-Formeln, so dass S genau dann ein Modell hat, wenn S_{sko} ein Modell hat. Zu jeder Formel $F \in S$ sei $F_{sko} \in S_{sko}$ die eineindeutig zugeordnete universelle Formel. Sei o.B.d.A. jedes F_{sko} in Pränexform.

Zu jedem $F \in S$ sei $F^\star$ die Menge der Grundinstanzen von F_{sko}. Sei $S^\star$ die Vereinigung all dieser Mengen. Nach Satz 3.8.16 ist S_{sko} und damit S genau dann erfüllbar, wenn $S^\star$ ein Herbrand-Modell besitzt. Wie im Beweis von Satz 4.6.1 können wir $S^\star$ als aussagenlogische Formelmenge ansehen.

Mit diesen Vorbereitungen zeigen wir jetzt die Kontraposition der Behauptung. Sei also S unerfüllbar. Dann ist $S^\star$ unerfüllbar. Nach dem Endlichkeitssatz für die Aussagenlogik (Satz 3.3.9) gibt es eine unerfüllbare endliche Teilmenge $\{G_1, \ldots, G_n\} \subseteq S^\star$.

Zu jeder der Formeln G_i sei $F_i \in S$ eine Formel mit $G_i \in F_i^\star$. Es gibt zu jedem G_i mindestens eine solche Formel (es kann auch beliebig viele geben). Nun ist $\{F_1, \ldots, F_n\}$ eine Formelmenge, deren Menge von Grundinstanzen $F_1^\star \cup \ldots \cup F_n^\star$ die unerfüllbare Teilmenge $\{G_1, \ldots, G_n\}$ enthält und damit selbst unerfüllbar ist. Nach Satz 3.8.16 ist $\{F_1, \ldots, F_n\}$ unerfüllbar, und es ist eine endliche Teilmenge von S. ■

Wie bereits in Kapitel 3 erwähnt, bezieht sich der Name „Kompaktheitssatz“ auf einen Beweis des Satzes, der auf der Kompaktheit eines topologischen Raums beruht. Eine andere verbreitete Technik zum Beweis des Endlichkeitssatzes setzt eine Beweismethode voraus. Zum Beispiel ist die PUHR-Tableau-Beweismethode vollständig bezüglich der Unerfüllbarkeit auch unendlicher bereichsbeschränkter Klauselmengen. Da jedes geschlossene PUHR-Tableau endlich ist, wird darin die PUHR-Regel nur mit endlich vielen Klauseln angewandt, und die entsprechende endliche Teilmenge von Klauseln ist unerfüllbar. Damit ist der Endlichkeitssatz für bereichsbeschränkte Klauseln eine einfache Folge aus diesem Vollständigkeitsergebnis.

Der Satz gilt auch für die Prädikatenlogik erster Stufe mit Gleichheit:

Folgesatz 4.9.2. Sei S eine unendliche Menge von geschlossenen Formeln einer Sprache $\mathcal{L}$ der Prädikatenlogik erster Stufe mit Gleichheit. Hat jede endliche Teilmenge von S ein normales Modell, so hat S ein normales Modell.

Beweis: Hat jede endliche Teilmenge von S ein normales Modell, hat nach Satz 3.6.6 (1) auch jede endliche Teilmenge von $S \cup GAS_{\mathcal{L}}$ ein Modell.

Nach dem Endlichkeitssatz besitzt folglich auch $S \cup GAS_{\mathcal{L}}$ ein Modell M, also nach Satz 3.6.6 (2) auch ein normales Modell M'. Da $S \subseteq S \cup GAS_{\mathcal{L}}$, ist M' ein normales Modell von S. ■

Folgesatz 4.9.3. Sei S eine unendliche Menge von geschlossenen Formeln einer Sprache $\mathcal{L}$ der Prädikatenlogik erster Stufe, und sei F eine geschlossene $\mathcal{L}$-Formel. $S \models F$ genau dann, wenn es eine endliche Teilmenge E von S gibt, so dass $E \models F$.

Beweis: Unmittelbare Folgerung aus Satz 4.9.1. ■

Mit dem Endlichkeits-/Kompaktheitssatz lassen sich viele bemerkenswerte Resultate beweisen. Es folgen einige Beispiele davon.

Zusammenhang von gerichteten Graphen

Ein gerichteter Graph ist ein Paar bestehend aus einer Menge von Knoten und einer zweistelligen Relation über der Knotenmenge. Die Paare von Knoten in dieser Relation nennt man Kanten. Ein Pfad ist eine endliche Folge $N_0 \dots N_n$ von Knoten, so dass (N_{i-1}, N_i) für $1 \leq i \leq n$ eine Kante ist. Die Anzahl n der Kanten nennt man die Länge des Pfads. Ein gerichteter Graph heißt zusammenhängend, wenn es zu jedem Paar N, N' von Knoten einen Pfad gibt, der mit N beginnt und mit N' endet.

Satz 4.9.4. Der Zusammenhang von gerichteten Graphen ist in der Prädikatenlogik erster Stufe nicht ausdrückbar.

Beweis: Sei $\mathcal{L}$ ein Sprache der Prädikatenlogik erster Stufe mit Gleichheit mit zwei Konstanten a und b und einem zweistelligen Relationssymbol k. Jede normale $\mathcal{L}$-Interpretation M entspricht einem Graph mit dem Universum von M als Knotenmenge und k^M als Kantenrelation. Alle Terme, insbesondere a und b, repräsentieren also Knoten. Offensichtlich entspricht jeder Graph einer solchen Interpretation.

Angenommen, es gäbe eine Menge Z von Formeln der Prädikatenlogik erster Stufe, die den Zusammenhang ausdrückt. Genauer heißt das, dass eine normale $\mathcal{L}$-Interpretation genau dann ein Modell von Z ist, wenn sie einem Graph entspricht, in dem es einen Pfad gibt von dem Knoten, den a repräsentiert, zu dem Knoten, den b repräsentiert.

Nun betrachten wir folgende Formeln.

$$
\begin{array}{lll}
F_0 & := & \neg(a \doteq b) \\
F_1 & := & \neg k(a,b) \\
F_2 & := & \neg\exists x_1 \bigwedge^\star(k(a,x_1),\, k(x_1,b)) \\
F_3 & := & \neg\exists x_1 \exists x_2 \bigwedge^\star(k(a,x_1),\, k(x_1,x_2),\, k(x_2,b)) \\
F_4 & := & \neg\exists x_1 \exists x_2 \exists x_3 \bigwedge^\star(k(a,x_1),\, k(x_1,x_2),\, k(x_2,x_3),\, k(x_3,b)) \\
\vdots & & \\
F_n & := & \neg\exists x_1 \dots \exists x_{n-1} \bigwedge^\star(k(a,x_1),\, k(x_1,x_2), \dots, k(x_{n-1},b)) \\
\vdots & &
\end{array}
$$

Sei $S_0 := \{F_0\}$ und $S_{n+1} := S_n \cup \{F_{n+1}\}$ für alle $n \in \mathbb{N}$ und $S^\star := \bigcup_{n\in\mathbb{N}} S_n$.

Erfüllt eine normale $\mathcal{L}$-Interpretation die Formel F_n, entspricht sie einem Graphen, in dem es keinen Pfad der Länge n zwischen den durch a und b repräsentierten Knoten gibt. Erfüllt sie die Formelmenge S_n, gibt es zwischen diesen Knoten keinen Pfad einer Länge $\leq n$. Erfüllt sie $S^\star$, gibt es keinen Pfad zwischen diesen Knoten.

Für jedes $n \in \mathbb{N}$ ist $Z \cup S_n$ erfüllbar: Jeder Graph, in dem es zwischen den betrachteten Knoten einen Pfad einer Länge $> n$ gibt, entspricht einem normalen Modell. Damit hat auch jede endliche Teilmenge von $Z \cup S_n$ ein normales Modell, und somit auch jede endliche Teilmenge von $Z \cup S^\star$.

Nach dem Endlichkeitssatz gibt es ein normales Modell von $Z \cup S^\star$. In dem Graphen, der dieser Interpretation entspricht, gibt es aber keinen Pfad von dem Knoten, den a repräsentiert, zu dem Knoten, den b repräsentiert, im Widerspruch zu der Annahme über Z. ∎

Endlichkeit von Modellen

Satz 4.9.5. Die Endlichkeit ist in der Prädikatenlogik erster Stufe ohne Gleichheit nicht ausdrückbar.

Beweis: Sei $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$. Sei $\mathcal{L}$ eine Sprache der Prädikatenlogik erster Stufe ohne Gleichheit mit abzählbar vielen einstelligen Relationssymbolen $\{p_i \mid i \in \mathbb{N}^+\}$.

Angenommen, es gäbe eine Menge E von $\mathcal{L}$-Formeln, die die Endlichkeit ausdrückt. Das heißt, dass E von einer $\mathcal{L}$-Interpretation genau dann erfüllt wird, wenn diese Interpretation ein endliches Universum hat. Wir nehmen o.B.d.A. an, dass die Relationssymbole p_i nicht in E vorkommen.

1. Für alle $n \in \mathbb{N}^+$ und für $i \in \{1, \ldots, n\}$ sei

$$\begin{aligned} F_n &:= \exists x_1 \ldots \exists x_n \wedge^\star(G_1, G_2, \ldots, G_n) \\ G_i &:= \wedge^\star(\neg p_1(x_i), \ldots, \neg p_{i-1}(x_i),\ p_i(x_i),\ \neg p_{i+1}(x_i), \ldots, \neg p_n(x_i)) \end{aligned}$$

   Für jedes $n \in \mathbb{N}^+$ wird F_n offensichtlich nur von Interpretationen erfüllt, deren Universen mindestens n Elemente enthalten.

2. $S_1 := \{F_1\}$ und $S_{n+1} := S_n \cup \{F_{n+1}\}$ für $n \in \mathbb{N}^+$ und $S^\star := \bigcup_{n \in \mathbb{N}^+} S_n$.

   Offensichtlich wird $S^\star$ nur von Interpretationen erfüllt, deren Universen unendlich sind.

Für jedes $n \in \mathbb{N}$ ist $E \cup S_n$ erfüllbar: Nach Annahme über E ist jede $\mathcal{L}$-Interpretation, deren Universum endlich ist und mindestens n Elemente hat, ein Modell von $E \cup S_n$. Damit hat auch jede endliche Teilmenge von $E \cup S_n$ ein Modell, und somit auch jede endliche Teilmenge von $E \cup S^\star$.

Nach dem Endlichkeitssatz gibt es ein Modell von $E \cup S^\star$, was zusammen mit 2. der Annahme über E widerspricht. ■

Der Beweis überträgt sich wortwörtlich auf normale Interpretationen.

Satz 4.9.6. Die Endlichkeit ist in der Prädikatenlogik erster Stufe mit Gleichheit nicht ausdrückbar. ■

In der Hoffnung, Satz 4.9.6 zu widerlegen, könnte man versuchen, folgendermaßen zu argumentieren: Für $n \in \mathbb{N}^+$ sei

$$F_n = \forall x_1 \ldots \forall x_{n+1} \vee^\star(x_{n+1} \doteq x_1,\ x_{n+1} \doteq x_2,\ \ldots,\ x_{n+1} \doteq x_n)$$

Für jedes $n \in \mathbb{N}^+$ wird F_n offensichtlich nur von Interpretationen erfüllt, deren Universen höchstens n Elemente enthalten.

Die unendliche Disjunktion aller F_n für $n \in \mathbb{N}^+$ würde die Endlichkeit ausdrücken, wenn sie eine Formel wäre. Aber das ist sie nicht.

Hier zeigt sich eine Asymmetrie in der Prädikatenlogik erster Stufe: Eine unendliche Menge von Formeln ist ein Ersatz für die (nicht zulässige) unendliche Konjunktion aller dieser Formeln. Es gibt aber kein ähnliches Gegenstück zu einer unendlichen Disjunktion. Der Existenzquantor entspricht lediglich Disjunktionen unbestimmter, endlicher oder unendlicher, Länge.

4.10 Exkurs: Folgerung im Endlichen

Für viele Anwendungen der Informatik sind Formalisierungen in Sprachen der Prädikatenlogik erster Stufe passend und günstig, jedoch nur Modelle mit endlichen Universen sinnvoll. Das gilt zum Beispiel für die Integritätsbedingungen einer relationalen Datenbank oder für die möglichen Pannenursachen in komplexen Systemen, die in manchen Diagnose-Ansätzen Modellen einer Formelmenge entsprechen.

Anstelle der gewöhnlichen Folgerungsbeziehung $\models$ ist für solche Anwendungen eine Folgerungsbeziehung angebracht, die nur die Interpretationen mit endlichen Universen berücksichtigt. Diese Folgerung wird „Folgerung im Endlichen“ genannt und $\models_{\mathit{fin}}$ notiert.

Definition 4.10.1 (semantische Begriffe im Endlichen). Seien F und G Formeln einer Sprache $\mathcal{L}$ der Prädikatenlogik erster Stufe und S eine Menge von Formeln der selben Sprache $\mathcal{L}$.

- Eine $\mathcal{L}$-Interpretation heißt endlich, wenn ihr Universum endlich ist. Ein Modell einer Formel oder Formelmenge heißt endlich, wenn es eine endliche Interpretation ist.
- Aus F *folgt* G im Endlichen (notiert $F \models_{\mathit{fin}} G$) genau dann, wenn jedes endliche Modell von F ein Modell von G ist.
- Aus S *folgt* G im Endlichen (notiert $S \models_{\mathit{fin}} G$) genau dann, wenn jedes endliche Modell von S ein Modell von G ist.
- F heißt *erfüllbar* im Endlichen, wenn F ein endliches Modell hat. ∎

Man beachte, dass nicht etwa verlangt ist, dass jede beteiligte Formelmenge S endlich sei, also nur endlich viele Formeln enthalte.

Die Automatisierung der Folgerung im Endlichen ist für viele Anwendungen nützlich. Folgt zum Beispiel im Endlichen eine Integritätsbedingung I aus der Menge $\mathcal{C}$ der sonstigen Integritätsbedingungen einer relationalen Datenbank, gilt also $\mathcal{C} \models_{\mathit{fin}} I$, so kann I als redundant angesehen und ignoriert werden. Wenn dagegen nur die unendlichen Modelle von $\mathcal{C}$ auch I erfüllen, ist I relevant für die Konsistenz, weil das intendierte Modell einer relationalen Datenbank (normalerweise) endlich ist.

Wegen Satz 4.9.5 und Satz 4.9.6 ist es unmöglich, die Folgerung im Endlichen durch eine Axiomatisierung in der Prädikatenlogik erster Stufe auf die gewöhnliche Folgerungsbeziehung zurückzuführen. Es ist also nicht möglich, gewöhnliche Beweismethoden für die Prädikatenlogik erster Stufe zur Feststellung der Folgerung im Endlichen einzusetzen. Statt dessen werden sogenannte „Modellgenerierungsmethoden“ verwendet, die ähnlich wie die PUHR-Tableau-Methode Modelle für wachsende Universen von Grundtermen aufbauen.

Satz 4.10.2. Die Erfüllbarkeit einer Formel F im Endlichen ist semi-entscheidbar.

Beweis: Es ist entscheidbar, ob F von einer Interpretation erfüllt wird, deren Universum eine vorgegebene endliche Kardinalität k hat. Der Grund ist, dass nur endlich viele Relations- und Funktionssymbole in F vorkommen, und dass es über den k Elementen des Universums zu jeder Stelligkeit nur endlich viele Relationen und endliche viele Funktionen dieser Stelligkeit gibt, mit denen die Symbole interpretiert werden können (Es gibt zum Beispiel genau $2^{(k^3)}$ Relationen der Stelligkeit 3). Man kann also systematisch alle Interpretationen daraufhin testen, ob sie F erfüllen.

Diesen Test kann man in einer Schleife für wachsende Kardinalitäten durchführen. Ist F erfüllbar im Endlichen, so besitzt F ein Modell, dessen Universum eine endliche Kardinalität k hat. Nach endlicher Zeit berücksichtigt das Verfahren diese Kardinalität k. ∎

Die Erfüllbarkeit im Endlichen ist nicht entscheidbar. Dieses Resultat wurde von Boris A. Trachtenbrot in einem Satz nachgewiesen, dessen Beweis wesentlich schwieriger ist als der Beweis von Satz 4.10.2.

4.11 Nichtausdrückbarkeit des Induktionsaxioms in der PL1S

In Abschnitt 3.10 wurde das Peano'sche Axiomensystem eingeführt, das aus folgenden Formeln einer Sprache $\mathcal{L}_{o,s}$ der Prädikatenlogik besteht:

$P1$: $\forall x\, \neg(s(x) \doteq o)$
(Kein Wert der Nachfolgerfunktion 1+ ist 0.)

$P2$: $\forall x \forall y\, (s(x) \doteq s(y) \;\Rightarrow\; x \doteq y)$
(Die Nachfolgerfunktion 1+ ist injektiv.)

$P3$: $\forall X \left[\Big(X(o) \;\wedge\; \forall y\, [\, X(y) \Rightarrow X(s(y))\,] \Big) \quad \Rightarrow \quad \forall z\, X(z) \right]$
(Induktionsaxiom: Für jede Teilmenge X der Menge der natürlichen Zahlen gilt: Enthält X die Null und mit jedem Element auch dessen Nachfolger, dann ist X die gesamte Menge der natürlichen Zahlen.)

$P1$ und $P2$ sind Formeln der Prädikatenlogik erster Stufe. Das Induktionsaxiom $P3$ ist eine Formel der Prädikatenlogik zweiter Stufe. Nach dem Satz von Dedekind (Satz 3.10.6) ist jedes normale Modell des Peano'schen Axiomensystems isomorph zu $(\mathbb{N}, 0, 1{+})$.

Wir zeigen jetzt, dass keine Menge von Formeln der Prädikatenlogik erster Stufe diese Eigenschaft hat. Der Beweis ist eine weitere Anwendung des Endlichkeitssatzes.

Satz 4.11.1 (Satz von Skolem). Keine erfüllbare Menge von $\mathcal{L}_{o,s}$-Formeln der Prädikatenlogik erster Stufe besitzt bis auf Isomorphie nur $(\mathbb{N}, 0, 1{+})$ als normales Modell.

Beweis: Angenommen, es gäbe eine erfüllbare Menge N von $\mathcal{L}_{o,s}$-Formeln, so dass jede normale $\mathcal{L}_{o,s}$-Interpretation, die N erfüllt, mit $(\mathbb{N}, 0, 1{+})$ isomorph ist. Dann ist insbesondere auch $(\mathbb{N}, 0, 1{+})$ ein Modell von N.

Sei $\mathcal{L}_c$ die Erweiterung der Sprache $\mathcal{L}_{o,s}$ um eine Konstante c. Eine c-Erweiterung einer $\mathcal{L}_{o,s}$-Interpretation M sei eine $\mathcal{L}_c$-Interpretation, die der Konstanten c ein Element aus $Univ(M)$ zuordnet und ansonsten mit M übereinstimmt. Insbesondere c-Erweiterungen von $(\mathbb{N}, 0, 1{+})$ werden im folgenden eine Rolle spielen. Nun betrachten wir folgende $\mathcal{L}_c$-Formeln.

$$\begin{array}{lll}
F_0 & := & \neg(\, c \doteq o\,) \\
F_1 & := & \neg(\, c \doteq s(o)\,) \\
F_2 & := & \neg(\, c \doteq s(s(o))\,) \\
\vdots & & \\
F_n & := & \neg(\, c \doteq \underbrace{s(\ldots s(}_{n-\text{mal}} o) \ldots)\,) \\
\vdots & &
\end{array}$$

Sei $S_0 := \{F_0\}$ und $S_{n+1} := S_n \cup \{F_{n+1}\}$ für alle $n \in \mathbb{N}$ und $S^\star := \bigcup\limits_{n \in \mathbb{N}} S_n$.

Erfüllt eine c-Erweiterung von $(\mathbb{N}, 0, 1+)$ die Formel F_n, kann sie der Konstanten c nicht die natürliche Zahl n zuordnen. Erfüllt sie die Formelmenge S_n, kann sie der Konstanten c keine natürliche Zahl $\leq n$ zuordnen. Die Formelmenge $S^\star$ wird von keiner c-Erweiterung von $(\mathbb{N}, 0, 1+)$ erfüllt.

Für jedes $n \in \mathbb{N}$ ist $N \cup S_n$ erfüllbar: Nach Annahme über N ist jede c-Erweiterung von $(\mathbb{N}, 0, 1+)$, die der Konstanten c eine natürliche Zahl $> n$ zuordnet, ein normales Modell von $N \cup S_n$. Damit hat auch jede endliche Teilmenge von $N \cup S_n$ ein normales Modell, und somit auch jede endliche Teilmenge von $N \cup S^\star$.

Nach dem Endlichkeitssatz gibt es ein normales Modell M von $N \cup S^\star$. Wenn diese $\mathcal{L}_c$-Interpretation isomorph zu einer c-Erweiterung von $(\mathbb{N}, 0, 1+)$ wäre, könnte sie $S^\star$ nicht erfüllen. Also ist die Restriktion von M auf $\mathcal{L}_{o,s}$ nicht isomorph zu $(\mathbb{N}, 0, 1+)$, andererseits ist sie, da c in N nicht vorkommt, ein Modell von N, im Widerspruch zur Annahme. ■

Da $P1$ und $P2$ Formeln der Prädikatenlogik erster Stufe sind, ergibt sich sofort:

Folgesatz 4.11.2. Das Induktionsaxiom $P3$ ist in der Prädikatenlogik erster Stufe nicht ausdrückbar. ■

Nichtstandardmodelle

Die normalen Modelle von $\{P1, P2\}$, die mit $(\mathbb{N}, 0, 1+)$ nicht isomorph sind, heißen „Nichtstandardmodelle" des Peano'schen Axiomensystems. Die Bezeichnung „Nichtstandard" ist allgemein üblich für Modelle von Axiomensystemen die nicht mit den intendierten Modellen isomorph sind.

Für jedes normale Modell M von $\{P1, P2\}$ gilt, dass die Elemente

$$o^M, s(o)^M, s(s(o))^M, s(s(s(o)))^M, \ldots$$

des Universums von M alle verschieden sind. Das Universum muss also mindestens so viele Elemente haben wie $\mathbb{N}$.

Wenn M zusätzlich das Induktionsaxiom $P3$ erfüllt, kann das Universum keine anderen als diese Elemente enthalten. Nichtstandardmodelle von $\{P1, P2\}$, die also das Induktionsaxiom $P3$ nicht erfüllen, enthalten „zusätzliche" Elemente – zusätzlich nur bezüglich des intendierten Modells.

Welche Möglichkeiten hat nun ein Nichtstandardmodell M, zusätzliche Elemente im Universum zu enthalten? Die Funktion s^M muss ja auch auf diesen zusätzlichen Elementen definiert sein und sie muss weiterhin injektiv sein und darf keines der neuen Elemente auf o^M abbilden. Es ergeben sich drei prinzipielle Möglichkeiten:

1. Falls ein Element hinzukommt, das kein Bild von s^M ist, muss von diesem Element ein zweiter Strang ausgehen, analog zu dem Strang von o^M aus.

   Bsp.: $D = \mathbb{N} \cup \{0.5,\ 1.5,\ 2.5,\ 3.5,\ \ldots\}$ mit $s^M(d) = 1 + d$ für alle $d \in D$.

2. Falls jedes hinzukommende Element ein Bild von s^M ist, können endlich viele Elemente bezüglich s^M einen Zyklus bilden.

   Bsp.: $D = \mathbb{N} \cup \{0.5,\ 1.5,\ 2.5\}$ mit $s^M(d) = 1 + d$ für alle $d \in D$, außer $s^M(2.5) = 0.5$

3. Falls jedes hinzukommende Element ein Bild von s^M ist, aber kein endlicher Zyklus vorliegt, kommen unendlich viele Elemente hinzu, von denen jedes einen Vorgänger und einen Nachfolger unter s^M hat.

   Bsp.: $D = \mathbb{N} \cup \{\ldots, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5, \ldots\}$ mit $s^M(d) = 1 + d$ für alle $d \in D$.

Solche Zyklen und Ketten können auch gemeinsam vorkommen. Zudem kann es beliebig viele davon geben. Wegen $P2$ sind alle Zyklen oder Ketten paarweise disjunkt.

Man kann durch Hinzunahme von Formeln der Prädikatenlogik erster Stufe Fälle der ersten beiden Arten ausschließen. Den dritten Fall aber kann man nicht mit Formeln der Prädikatenlogik erster Stufe verhindern.

Erweitert man das Peano'sche Axiomensystem, so dass auch die Addition, die Multiplikation und die Kleinerbeziehung definiert werden, bleiben nur noch Nichtstandardmodelle übrig, in denen nach $(\mathbb{N}, <)$ eine Kopie von $(\mathbb{Z}, <)$ folgt („nach“ im Sinne der Kleinerbeziehung), danach weitere Kopien von $(\mathbb{Z}, <)$, so dass zwischen je zwei Kopien von $(\mathbb{Z}, <)$ eine weitere liegt.

Für eine Temporallogik repräsentiert $(\mathbb{N}, <)$ ein lineares diskretes Zeitmodell mit Ursprung. Aus dem Satz von Skolem folgt, dass das Zeitmodell einer linearen Temporallogik in der Prädikatenlogik erster Stufe nicht axiomatisierbar ist. Ein ähnliches Ergebnis gilt für verzweigte Zeitmodelle.

Index