

---

# Logik für Studierende der Informatik

---

Markus Junker (Mathematisches Institut, Universität Freiburg)

Wintersemester 2021/22 – Version von 14. Februar 2022

---

## Inhaltsverzeichnis

<b>0</b>	<b>Einleitung</b>	<b>3</b>
<b>1</b>	<b>Aussagenlogik</b>	<b>5</b>
1.1	Alphabet der Aussagenlogik . . . . .	5
1.2	Syntax der Aussagenlogik . . . . .	5
1.3	Darstellungen aussagenlogischer Formeln . . . . .	7
1.4	Semantik der klassischen zweiwertigen Aussagenlogik . . . . .	11
1.5	Disjunktive und konjunktive Normalform . . . . .	15
1.6	Gesetze der klassischen zweiwertigen Aussagenlogik . . . . .	20
1.7	Erfüllbarkeit . . . . .	25
1.7.1	Formeln in disjunktiver Normalform . . . . .	25
1.7.2	Baumkalkül / Tableau-Methode . . . . .	25
1.7.3	Die Methode von Quine . . . . .	28
1.7.4	Resolution (*) . . . . .	29
1.7.5	Der Kompaktheitssatz . . . . .	34
1.8	Boole'sche Algebren . . . . .	35
1.9	Semantik der intuitionistischen Aussagenlogik (*) . . . . .	43
<b>2</b>	<b>Prädikatenlogik</b>	<b>48</b>
2.1	Alphabet der Prädikatenlogik . . . . .	48
2.2	Syntax der Prädikatenlogik . . . . .	49
2.3	Semantik der Prädikatenlogik . . . . .	54
2.4	Gesetze der Prädikatenlogik . . . . .	59
2.4.1	Substitutionsregeln . . . . .	59

2.4.2	Gleichheitsgesetze und Quantorengesetze . . . . .	62
2.4.3	Normalformen und syntaktische Reduktionen . . . . .	65
2.5	Der Gödel’sche Vollständigkeitssatz . . . . .	67
2.6	Erfüllbarkeit und Allgemeingültigkeit . . . . .	74
2.6.1	Semantische Beweise und Beispiele . . . . .	74
2.6.2	Kalküle . . . . .	75
2.6.3	Syntaktische Reduktionen durch Spracherweiterungen . . . . .	79
2.6.4	Der Satz von Herbrand und Unifikation . . . . .	82
<b>3</b>	<b>Berechenbarkeit und Entscheidbarkeit</b>	<b>89</b>
3.1	Turing-Maschinen . . . . .	91
3.2	Das Halteproblem und die Unentscheidbarkeit der Prädikatenlogik . . . . .	94
3.3	NP-Vollständigkeit und der Satz von Cook . . . . .	100
3.4	Rekursive Funktionen . . . . .	104
3.5	Gödelisierung und die Arithmetik . . . . .	109
<b>4</b>	<b>Anhang</b>	<b>113</b>
4.1	Ein Sequenzenkalkül für die Aussagenlogik . . . . .	113

(\*) Im Wintersemester 2021/22 nicht Teil der Vorlesung

## 0 Kurze Einleitung

Grob gesprochen ist die Wissenschaftsdisziplin *Logik* die Theorie von Regelsystemen in formalen Sprachen. Es gibt die verschiedensten Systeme, die sich in der intendierten Anwendung, der verwendeten Sprache, ihrer Aussagestärke und den zugrundeliegenden Regeln unterscheiden können. Solch ein System wird auch kurz *eine Logik* genannt und oft in drei Schritten beschrieben:

- *Alphabet*: Welche *Zeichen* sind für die Konstruktion der formalen Sprache zugelassen?
- *Syntax*: Welche Gebilde aus diesen Zeichen bilden *Sätze* der formalen Sprache? (Häufig auch *wff's* – *well formed formulae* – genannt.)
- *Semantik*: Wie lassen sich diese Sätze in Strukturen/Kontexten ... *auswerten*?  
Aus diesen Auswertungen ergeben sich dann Definitionen der für die Logik relevanten Begriffe wie zum Beispiel *logische Folgerung* und *logische Äquivalenz*. Alternativ zu solch einer Semantik können diese Begriffe auch über Umformungs- oder Ableitungsregeln eingeführt werden. (Was damit gemeint sein kann, wird noch an Beispielen deutlich werden.)

Die grundlegendsten Logiken sind *Aussagenlogiken*. Eine Aussagenlogik ist dadurch charakterisiert, dass ihren Sätzen zur Auswertung *Wahrheitswerte* zugewiesen werden, und dass die Zeichen der formalen Sprache im Wesentlichen für Funktionen auf diesen Wahrheitswerten stehen. Die einfachste Aussagenlogik ist die *klassische zweiwertige Aussagenlogik*, die jedem Satz in jedem Kontext *genau einen* der beiden Wahrheitswerte 0 oder 1 (bzw. „falsch“ oder „wahr“) zuordnet und damit den Prinzipien des ausgeschlossenen Widerspruchs und des ausgeschlossenen Dritten unterliegt.

Man kann auch mehrwertige Logiken betrachten (wie im Extremfall die einst in Mode gewesene *fuzzy logic* mit einem Kontinuum an Wahrheitswerten) und nicht-klassische Logiken (wie z. B. die *intuitionistische Aussagenlogik*, die in der Informatik eine Rolle spielt, weil sie in gewisser Weise das Laufverhalten von Computerprogrammen adäquater beschreibt als die klassische).

Erweiterungen der Aussagenlogiken sind *Modallogiken* (wofür es im B.Sc.-Studiengang Informatik eine eigene Vorlesung gibt) und die *Prädikatenlogiken*. Nach der klassischen zweiwertigen Aussagenlogik wird in dieser Vorlesung die (klassische zweiwertige) *Prädikatenlogik erster Stufe* betrachtet, die geeignet ist, um beliebige mathematische Strukturen zu beschreiben.

Traditionell wird *Logik* häufig als die *Lehre vom korrekten Schließen* bezeichnet. Untersuchungsobjekte der Logik sind dann Argumente, die typischerweise die Form

$$\begin{array}{l} \text{Prämisse 1} \\ \vdots \\ \text{Prämisse } n \\ \hline \text{Schlussfolgerung} \end{array}$$

annehmen. Untersucht wird, ob sich die Schlussfolgerung „logisch korrekt“ aus den Prämissen ergibt. Dabei bezieht sich logische Korrektheit üblicherweise nur auf die *Form* der Sätze unter Abstraktion von ihrem konkreten Inhalt. Logische Schlüsse lassen sich daher als in einer formalen Sprache beschriebene Regeln wiedergeben, weshalb Logik auch eine Theorie formaler Sprachen begründet. Logik ist daher aus mehreren Gründen für die Informatik interessant:

- Logik ist Grundlage aller Wissenschaft, insofern wissenschaftliche Argumentation logischen

Standards genügen muss („Logik als Propädeutik“).

- Der Logik als eigener Disziplin lag auch immer schon der Gedanke der Berechenbarkeit logischer Schlüsse nahe, weshalb aus der Logik zu Beginn des 20. Jahrhunderts mit einer Theorie der Berechenbarkeit die theoretische Informatik entstanden ist, deutlich vor der Konstruktion der ersten Computer („Logik als historische Grundlage“).
- Die Sprache der Logik ist dazu geeignet, die Arbeitsweise von Programmen zu beschreiben, da eine Berechnung aus Eingaben einer Schlussfolgerung aus Prämissen entspricht.
- Die formalen Sprachen und die Methoden der Logik können helfen, Probleme aus der Informatik zu beschreiben oder zu lösen.

Anwendungen der Logik finden sich vor allem in den Vorlesungen zur Künstlichen Intelligenz und zur Programmverifikation, aber auch die Konstruktion einiger Programmiersprachen baut direkt auf Logik auf.

Die folgenden Namen aus der Geschichte der Logik sollten vielleicht vertraut sein:

- Aristoteles (384–322): Begründer der formalen Logik (*Syllogistik*)
- Ramon Llull (1232–1316): erste Idee einer „logischen Maschine“
- Gottfried Wilhelm Leibniz (1646–1716): Binärsystem, binäre Rechenmaschine, Idee einer „universellen Maschine“
- George Boole (1815–1864), Charles Sanders Peirce (1839–1914), Gottlob Frege (1848–1925): moderne symbolische/formalisierte Logik
- Emil Post (1897–1954), Alonzo Church (1903–1995), Stephen Kleene (1909–1994), Alan Turing (1912–1954): Begründer der Berechenbarkeitstheorie/theoretischen Informatik
- Kurt Gödel (1906–1978): bedeutende Ergebnisse mit dem Vollständigkeitssatz und den Unvollständigkeitssätzen
- Saul Kripke (\* 1940): Semantik der Modallogik

## Hinweise zum Gebrauch des Skripts

Wichtige Begriffe sind *farblich hervorgehoben*; in der Regel ist das englische Pendant dazu angegeben. *Kursive Schrift* (oder gerade Schrift *innerhalb von kursivem Text*) markiert Sprechweisen, Betonungen und nur vorübergehend gebrauchte oder weniger wichtige Begriffe. Abkürzungen und besondere mathematische Symbole sind dort, wo sie eingeführt werden, am Rand wiederholt, damit man diese Definitionen besser findet. Der Gebrauch von Farben in mathematischen Formeln wird jeweils bei Einführung erklärt.

# 1 Aussagenlogik

Genauer geht es in diesem Kapitel um die *klassische zweiwertige Aussagenlogik* [classical bivalent propositional logic]. Sätzen dieser Aussagenlogik wird genau einer der beiden Wahrheitswerte 0 und 1 zugewiesen, und die Ausdrucksstärke der Logik liegt darin, genau die Vielfalt möglicher Wahrheitswertfunktionen ausdrücken zu können.

Anders als in früheren Versionen dieses Skripts werde ich Sätze dieser Logik als *Bäume* einführen und die verbreiteteren Zeichenfolgen als mögliche Darstellungen dieser Bäume. Aber auch abgesehen davon ist die hier präsentierte Syntax der Aussagenlogik *eine mögliche Variante* einer aussagenlogischen Sprache. In der Literatur findet man viele andere Varianten, die sich in der Wahl der Symbole, in der Auswahl der Junktoren oder (in der Darstellung als Zeichenkette) in der Klammerweise unterscheiden.

## 1.1 Alphabet der Aussagenlogik

Zur Konstruktion aussagenlogischer Sätze werden die folgenden *Zeichen / Symbole* [symbols] benutzt. Der besseren Kenntlichkeit halber werde ich diese Zeichen blau schreiben, wenn sie als einzelne Zeichen angesprochen sind.

<i>Aussagenvariablen</i> [propositional variables]	<i>Junktoren</i> [connectives]
$A_0$	nullstellig: $\top$ und $\perp$
$A_1$	einstellig: $\neg$
$A_2$	zweistellig: $\wedge$ , $\vee$ , $\rightarrow$ und $\leftrightarrow$
$\vdots$	

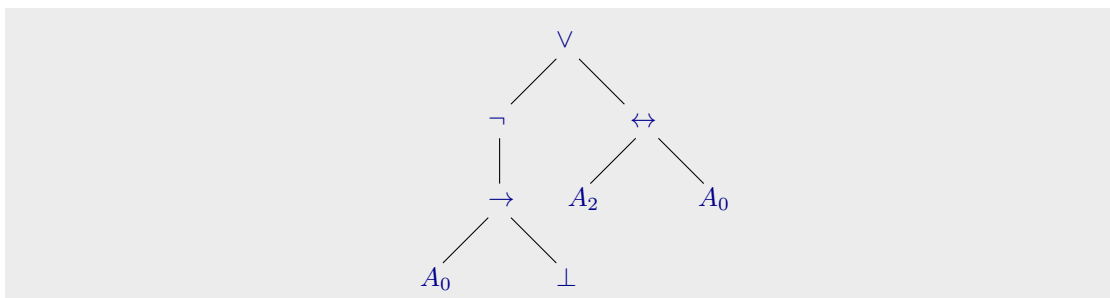
$A_i$   
 $\top$   
 $\perp$   
 $\neg$   
 $\wedge$   
 $\vee$   
 $\rightarrow$   
 $\leftrightarrow$

Es gibt unendlich viele Aussagenvariablen. Jede soll dabei als ein einziges Zeichen gelten, also z. B.  $A_{398652}$  als *ein* Symbol. Ein Ausdruck wie  $A_i$  wird als Variable für Aussagenvariablen verwendet, ohne selbst eine Aussagenvariable zu sein (ich schreibe ihn dennoch auch in blau).

Die Junktoren  $\top$ ,  $\perp$ ,  $\neg$ ,  $\wedge$ ,  $\vee$ ,  $\rightarrow$  und  $\leftrightarrow$  heißen der Reihe nach: *Verum*, *Falsum*, *Negations-*, *Konjunktions-*, *Disjunktions-*, *Implikations-* und *Äquivalenzjunktork*. In einer Formel werden die letzten fünf Junktoren oft gelesen als: *nicht / und / oder / wenn ... dann / genau dann, wenn*.

## 1.2 Syntax der Aussagenlogik

**Definition 1.2.1** *Ein aussagenlogischer Satz oder eine aussagenlogische Formel* [propositional formula] *ist ein endlicher etikettierter, orientierter Wurzelbaum wie zum Beispiel:*



der nach den unten folgenden Regeln gebildet werden kann ... (wird fortgesetzt)

Bevor die Definition fortgesetzt wird, sei zunächst die präzise Definition von „etikettierter, orientierter Wurzelbaum“ eingeschoben.

### Einschub: Begriffe aus der Graphentheorie

Ein *Graph* [graph] besteht aus einer Menge  $E$ , deren Elemente *Ecken* oder *Knoten* [vertices] genannt werden, und einer symmetrischen und irreflexiven Relation  $K \subseteq E^2$ , der *Kantenrelation*. Falls  $(e_1, e_2) \in K$ , so sagt man, dass die beiden Ecken  $e_1, e_2 \in E$  durch eine *Kante* [edge] verbunden sind oder dass  $e_1$  und  $e_2$  *Nachbarn* [neighbours] voneinander sind. Ein *Weg* oder *Pfad* [path] – und zwar der Länge  $n$  – zwischen Ecken  $e$  und  $e'$  ist eine Folge paarweise verschiedener Ecken  $e_0, e_1, \dots, e_n$  mit  $e_0 = e$  und  $e_n = e'$  und so, dass es jeweils eine Kante zwischen  $e_i$  und  $e_{i+1}$  gibt. Der *Abstand* [distance] zweier Ecken ist  $\infty$ , falls es keinen Weg zwischen den beiden Ecken gibt, und ansonsten die minimale Länge eines solchen Weges. Der Graph heißt endlich, wenn  $E$  endlich ist.

Ein *Baum* [tree] ist ein zusammenhängender, zyklfreier Graph: *zusammenhängend* [connected] bedeutet, dass es zwischen je zwei Ecken einen Weg gibt, und *zyklfrei* [without cycles] bedeutet, dass ein Weg zwischen zwei Ecken eindeutig bestimmt ist (sofern es einen gibt).

Ein *Wurzelbaum* [rooted tree] ist ein Baum mit einem ausgezeichneten Element, das *Wurzel* [root] heißt. Ein Baum heißt *etikettiert* [labelled] – und zwar mit Etiketten aus einer Menge  $M$  – wenn eine Abbildung  $E \rightarrow M$  festgelegt ist, die jeder Ecke ein Element aus  $M$  als *Etikett* [label] zuweist. Schließlich heißt ein Baum *orientiert* [oriented], wenn für jede Ecke  $e$  auf der Menge der *Nachfolger* [successors] von  $e$  – das sind die Nachbarn von  $e$ , die einen größeren Abstand von der Wurzel als  $e$  haben – eine Anordnung festgelegt ist.

Ein orientierter Wurzelbaum wird in der Regel zeilenweise dargestellt: In der 0-ten Zeile die Wurzel, in der  $i$ -ten Zeile die Ecken mit Abstand  $i$  zur Wurzel. Benachbarte Ecken werden mit einer Kante (also einem Strich) verbunden. Die vorgegebene Anordnung wird von links nach rechts abgetragen, und man macht das Ganze so, dass sich die Kanten nicht überschneiden. Ist es ein etikettierter Baum, schreibt man an die Stelle der Ecken ihre Etiketten und spricht gerne kurz von „der Ecke  $m$ “ statt von „der Ecke mit dem Etikett  $m$ “.

#### Definition 1.2.1 (Fortsetzung)

Ein aussagenlogischer Satz bzw. eine aussagenlogische Formel ist ein endlicher etikettierter, orientierter Wurzelbaum, der nach folgenden Regeln gebildet werden kann:

- Jeder einelementige Baum mit einer Aussagenvariable, Verum oder Falsum als Etikett bildet eine aussagenlogische Formel.
- Wenn  $F$  eine aussagenlogische Formel ist, ist auch

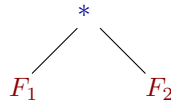
$$\neg$$

$$|$$

$$F$$

eine aussagenlogische Formel, d. h. der Baum, der aus  $F$  dadurch entsteht, dass eine neue Wurzel mit  $\neg$  als Etikett hinzugenommen wird, die allein mit der Wurzel von  $F$  in Kantenrelation steht.

- Wenn  $F_1$  und  $F_2$  aussagenlogische Formeln sind und  $*$  einer der vier zweistelligen Junktoren, ist auch



eine aussagenlogische Formel, d. h. der Baum, der aus  $F_1$  und  $F_2$  dadurch entsteht, dass eine neue Wurzel mit  $*$  als Etikett hinzugenommen wird, die allein mit den Wurzeln von  $F_1$  und  $F_2$  in Kantenrelation steht, und diese werden den Indizes entsprechend angeordnet (d. h. in der üblichen Darstellung steht die Wurzel von  $F_1$  links von der Wurzel von  $F_2$ ).

Mögliche Etiketten sind hier also die Aussagenvariablen und die Junktoren. Die Bäume sind *binär* [binary], d. h. jede Ecke hat höchstens zwei Nachfolger. Genauer ist die Anzahl der Nachfolger von  $e$  gerade die Stelligkeit des Junktors, der das Etikett von  $e$  ist. Die *Blätter* [leaves], d. h. die Ecken ohne Nachfolger, haben stets Aussagenvariablen oder nullstellige Junktoren (Verum/Falsum) als Etikett.

Aussagenlogische Formeln und ihre diversen Darstellungen schreibe ich **rot** (außer wenn ich erkläre, wie sie aus Einzelteilen zusammengesetzt sind). Als Variablen für aussagenlogische Formeln benutze ich meist  $F, G, H$  mit Varianten wie  $F_i, F', \dots$

$F, G, H, \dots$

Eine Definition wie die eben gegebene Definition aussagenlogischer Formeln heißt *induktive* oder *rekursive Definition* [inductive / recursive definition], weil sie nicht alle aussagenlogischen Formeln auf einen Schlag beschreibt, sondern einen Prozess: Aus einfachsten aussagenlogischen Formeln werden schrittweise („induktiv“) komplexere aufgebaut, bzw. eine komplexere Formel wird durch Zurückgreifen („Rekursion“) auf einfachere beschrieben.

Eine solche Definition ist stets so zu verstehen, dass einerseits alle Bäume, die durch endliches Anwenden der Regeln konstruiert werden können, aussagenlogische Formeln sind, und dass andererseits alles, was man nicht durch endliches Anwenden der Regeln konstruieren kann, auch keine aussagenlogische Formel ist.

Definitionen und Beweise von Sätzen über aussagenlogische Formeln müssen typischerweise (wenn sie sauber sein sollen) diesen Induktionsprozess nachvollziehen. Man spricht dann von einer Definition bzw. einem Beweis „über den Aufbau der Formeln“.

### 1.3 Darstellungen aussagenlogischer Formeln

Für das Verständnis aussagenlogischer Formeln ist es sinnvoll, sie als Bäume aufzufassen (weil ich sie so definiert habe); für den Gebrauch (und auch die Programmierung) ist eine eindimensionale Darstellung als Zeichenkette [string of symbols], d. h. als endliche Folge von Zeichen, nützlicher. Es gibt (mit Varianten) drei gebräuchliche Darstellungsweisen, die man alle per Induktion über den Aufbau der Formeln definieren kann:

#### Die Infix-Notation

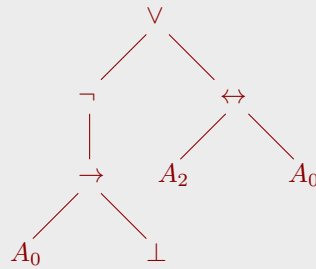
Jeder Formel  $F$  wird als *Infix-Notation* eine Zeichenkette  $\overline{F}$  zugeordnet, die sich an der klassischen Schreibweise mathematischer Operatoren wie  $+$  orientiert und für Menschen gut lesbar ist. Zusätzlich zu den Symbolen der aussagenlogischen Sprache braucht man allerdings die Klammern ( und ) oder andere Zusatzzeichen, die eine Priorisierung ermöglichen. Die Infix-Notation ist induktiv über den Aufgabe der Formeln wie folgt definiert:

$\overline{F}$

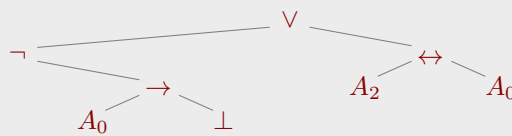
- Formeln, die einelementige Bäume sind, wird die einelementige Zeichenkette zugeordnet, die aus dem Etikett der Wurzel besteht, d. h.  $\overline{A_i} := A_i$ ,  $\overline{\top} := \top$  und  $\overline{\perp} := \perp$ .
- Einem Baum  $\overline{F}$  wird die Zeichenkette  $\neg \overline{F}$  zugeordnet.
- Einem Baum  $\overline{F_1 / \setminus^* F_2}$  wird die Zeichenkette  $(\overline{F_1} * \overline{F_2})$  zugeordnet.

Man kann sich einigermaßen leicht davon überzeugen, dass die Infix-Notation *eindeutig lesbar* ist. Das bedeutet, dass man den Baum aus der Infix-Notation rekonstruieren kann. <sup>1</sup>

**Beispiel:** Der Formel



wird die Zeichenkette  $(\neg(A_0 \rightarrow \perp) \vee (A_2 \leftrightarrow A_0))$  zugeordnet. Bildlich gesehen wird der Baum von oben nach unten zusammengedrückt (wobei Negationszeichen nach links „wegrutschen“):



Jede Zeichenkette  $Z$  hat eine **Länge [length]**, nämlich die Anzahl der Zeichen (mit Wiederholungen gezählt), die die Zeichenkette bilden. Für die Länge von  $Z$  schreibe ich kurz  $\text{lg}(Z)$ .

$\text{lg}$

Zum Beispiel ist  $\text{lg}((\neg(A_0 \rightarrow \perp) \vee (A_2 \leftrightarrow A_0))) = 14$ . <sup>2</sup>

Die Länge  $\text{lg}_i(F)$  der Infix-Darstellung einer Formel  $F$  lässt sich auch rekursiv definieren als

$\text{lg}_i$

- $\text{lg}_i(A_i) = \text{lg}_i(\top) = \text{lg}_i(\perp) := 1$
- $\text{lg}_i(\overline{F}) := 1 + \text{lg}_i(F)$
- $\text{lg}_i(\overline{F_1 / \setminus^* F_2}) := 3 + \text{lg}_i(F_1) + \text{lg}_i(F_2)$

## Die Polnische Notation

Jeder Formel  $F$  wird als **Polnische Notation** eine Zeichenkette  $\widehat{F}$  zugeordnet, die sich an der mathematischen Funktionsschreibweise orientiert und für Beweise und Programmierungen nützlich ist, für Menschen aber schwerer lesbar. Sie ist über den Aufbau der Formeln definiert durch:

$\widehat{F}$

<sup>1</sup>Für die eindeutige Lesbarkeit in Infix-Notation sind Klammern oder analoge Konstrukte notwendig. Allerdings könnte man sich durch kompliziertere Regeln einige Klammern sparen, etwa analog zur „Punkt vor Strich“-Regel der Arithmetik durch die Einführung unterschiedlicher Bindungsstärken.

In Abschnitt 1.5 werden noch gewisse abkürzende Schreibweisen für die Infix-Notation eingeführt.

<sup>2</sup>Erinnerung:  $A_i$  zählt immer nur als ein Zeichen, unabhängig von  $i$ .



- Formeln, die einelementige Bäume sind, wird wieder die einelementige Zeichenkette zugeordnet, die aus dem Etikett der Wurzel besteht, d. h.  $\widehat{A}_i := A_i$ ,  $\widehat{\top} := \top$  und  $\widehat{\perp} := \perp$ .
- Einem Baum  $\overline{\lceil}_F$  wird die Zeichenkette  $\neg \widehat{F}$  zugeordnet.
- Einem Baum  $\overline{\wedge^*}_{F_1 F_2}$  wird die Zeichenkette  $* \widehat{F}_1 \widehat{F}_2$  zugeordnet.

Man kann sich mit etwas mehr Mühe als bei der Infix-Notation davon überzeugen, dass die Polnische Notation ebenfalls eindeutig lesbar ist, dass man also den Baum aus der Polnischen Notation rekonstruieren kann.

**Beispiel:** Der Formel

wird die Zeichenkette  $V \neg \rightarrow A_0 \perp \leftrightarrow A_2 A_0$  zugeordnet. Bildlich gesehen wird der Baum nach links heruntergeklappt:

Die Länge  $lg_p(F)$  der Polnischen Notation ist die Anzahl der Knoten des Baums. Im Beispiel ist  $lg_p(F) = 8$ . Rekursiv wird sie definiert als:

$lg_p$

- $lg_p(A_i) = lg_p(\top) := lg_p(\perp) = 1$
- $lg_p(\overline{\lceil}_F) := 1 + lg_p(F)$
- $lg_p(\overline{\wedge^*}_{F_1 F_2}) := 1 + lg_p(F_1) + lg_p(F_2)$

### Die Umgekehrte Polnische Notation

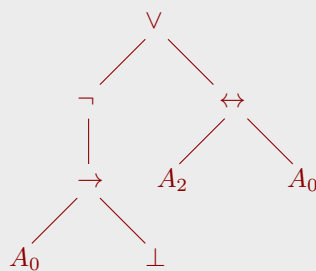
Jeder Formel  $F$  wird als **Umgekehrte Polnische Notation** eine Zeichenkette  $\check{F}$  zugeordnet:

$\check{F}$

- Formeln, die einelementige Bäume sind, wird erneut die einelementige Zeichenkette zugeordnet, die aus dem Etikett der Wurzel besteht, d. h.  $\check{A}_i := A_i$ ,  $\check{\top} := \top$  und  $\check{\perp} := \perp$ .
- Einem Baum  $\overline{\lceil}_F$  wird die Zeichenkette  $\check{F} \neg$  zugeordnet.
- Einem Baum  $\overline{\wedge^*}_{F_1 F_2}$  wird die Zeichenkette  $\check{F}_1 \check{F}_2 *$  zugeordnet.

Auch die Umgekehrte Polnische Notation ist eindeutig lesbar. Sie klappt bildlich gesehen einen Baum nach rechts herunter.

**Beispiel:** Der Formel



wird die Zeichenkette  $A_0 \perp \rightarrow \neg A_2 A_0 \leftrightarrow \vee$  zugeordnet.

Die Länge von Polnischer und Umgekehrter Polnischen Notation ist offensichtlich gleich.

### Höhe und Schachtelungstiefe einer Formel

Zwei Größen können noch nützlicherweise einer aussagenlogischen Formel zugeordnet werden:

- Die **Höhe** [height]  $ht(F)$  einer Formel  $F$  ist die Höhe des Baums, d. h. die maximale Länge eines Weges von der Wurzel zu einem Blatt (im Beispiel Höhe 3). ht
- Die **(Schachtelungs-)Tiefe** [depth]  $dp(F)$  einer Formel  $F$  ist die maximale Anzahl von Junktoren auf einem Weg von der Wurzel zu einem Blatt (im Beispiel Tiefe 4). dp

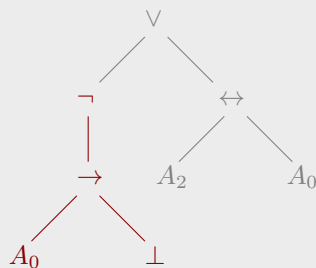
Es ist also  $ht(A_i) = dp(A_i) = 0$ , aber  $ht(\perp) = 0$  und  $dp(\perp) = 1$ .

Die Begriffe *Höhe* und *Tiefe* einer Formel und die Notationen  $\overline{F}$ ,  $\widehat{F}$  und  $\check{F}$  sind kein Standard und werden auch in dieser Vorlesung nur vorübergehend gebraucht!

Eine Induktion über den Aufbau der Formeln kann man nun als eine Induktion über die verschiedenen definierten Größen auffassen: wahlweise als Induktion über die Länge der Infix-Notation, über die Länge der Polnischen Notation, über die Höhe oder über die Tiefe.

**Definition 1.3.1** Eine **Teilformel** [subformula] einer aussagenlogischen Formel ist eine aussagenlogische Formel, die im Aufbauprozess der Formel vorkommt – einschließlich der Formel selbst, sonst spricht man von echten Teilformeln [proper subformulae].

In der Baumdarstellung kann man die Teilformeln folgendermaßen sehen: Man schneidet eine beliebige Ecke mit allem, was darunter hängt, vom Rest des Baumes ab; diese Ecke ist dann die Wurzel dieser Teilformel (im folgenden Beispiel die Ecke mit Etikett  $\neg$ ).



Aus der (in Infix-Notation geschriebenen) Formel  $(\neg(A_0 \rightarrow \perp) \vee (A_2 \leftrightarrow A_0))$  kann man also zunächst an jeder der acht Ecken eine Teilformel gewinnen, wobei aber  $A_0$  doppelt vorkommt.

Es gibt also sieben verschiedene Teilformeln und somit sechs echte Teilformeln. Diese sind:

$$\neg(A_0 \rightarrow \perp), (A_2 \leftrightarrow A_0), (A_0 \rightarrow \perp), A_2, A_0 \text{ und } \perp$$

### 1.4 Semantik der klassischen zweiwertigen Aussagenlogik

Die Menge der **Wahrheitswerte** [truth values] für die klassische zweiwertige Aussagenlogik ist  $\{0, 1\}$ . Der Wahrheitswert 0 heißt auch *falsch* [false], der Wahrheitswert 1 *wahr* [true]. Es ist praktisch, einerseits die beiden Wahrheitswerte durch  $0 < 1$  angeordnet zu denken und sie andererseits mit dem Körper  $\mathbb{F}_2$  zu identifizieren, so dass die arithmetischen Operationen  $+$ ,  $-$  und  $\cdot$  benutzt werden können. <sup>3</sup>

Eine **Belegung der Aussagenvariablen mit Wahrheitswerten**, kurz: Belegung [assignment], ist eine Abbildung  $\beta : \{A_i \mid i \in \mathbb{N}\} \rightarrow \{0, 1\}$ .

Jede Belegung  $\beta$  kann mittels der nächsten Festlegung induktiv zu einer Funktion fortgesetzt werden, die jeder aussagenlogischen Formel  $F$  einen Wahrheitswert  $\beta(F)$  zuordnet. Dazu wird zunächst jedem  $n$ -stelligen Junktor  $*$  folgendermaßen eine  $n$ -stellige **Wahrheitswertfunktion** [truth function]  $\tilde{*} : \{0, 1\}^n \rightarrow \{0, 1\}$  zugeordnet: <sup>4</sup>

$\tilde{\top} := 1$	$\tilde{\neg}(w) := 1 - w$	$\tilde{\wedge}(v, w) := \min\{v, w\}$	$\tilde{\rightarrow}(v, w) := \begin{cases} 1 & \text{falls } v \leq w \\ 0 & \text{falls } v > w \end{cases}$
$\tilde{\perp} := 0$	$\tilde{\vee}(v, w) := \max\{v, w\}$	$\tilde{\leftrightarrow}(v, w) := \begin{cases} 1 & \text{falls } v = w \\ 0 & \text{falls } v \neq w \end{cases}$	

Oder als Wertetabellen dargestellt:

$\tilde{\top}$	$\tilde{\perp}$	$x$	$\tilde{\neg}(x)$	$x$	$y$	$\tilde{\wedge}(x, y)$	$\tilde{\vee}(x, y)$	$\tilde{\rightarrow}(x, y)$	$\tilde{\leftrightarrow}(x, y)$
1	0	0	1	0	0	0	0	1	1
		1	0	0	1	0	1	1	0
				1	0	0	1	0	0
				1	1	1	1	1	1

**Definition 1.4.1** Jede Belegung  $\beta$  wird zu einer Funktion fortgesetzt, die jeder aussagenlogischen Formel  $F$  einen Wahrheitswert zuordnet, den Wahrheitswert von  $F$  unter der Belegung  $\beta$  [the truth value of  $F$  under the assignment  $\beta$ ]. Diesen Wahrheitswert schreibe ich  $\beta(F)$ , und er ist folgendermaßen über den Aufbau der Formeln definiert:

- $\beta(A_i)$  ist durch die Definition einer Belegung festgelegt.
- Wenn  $F$  mithilfe eines  $n$ -stelligen Junktors  $*$  aus Formeln  $F_1, \dots, F_n$  zusammengesetzt ist, wenn also in Polnischer Notation  $\widehat{F} = * \widehat{F}_1 \dots \widehat{F}_n$ , so setzt man

$$\beta(F) := \tilde{*}(\beta(F_1), \dots, \beta(F_n)).$$

<sup>3</sup>Richtiger ist es,  $\{0, 1\}$  als Boole'sche Algebra zu betrachten, was aber erst in Abschnitt 1.8 eingeführt wird.  
<sup>4</sup>Die Funktion  $\tilde{\rightarrow}$  ist die *charakteristische Funktion*  $\chi_{\leq}$  der  $\leq$ -Relation auf den Wahrheitswerten; die Funktion  $\tilde{\leftrightarrow}(x, y)$  wird in der Mathematik in anderem Kontext auch *Kronecker-Delta*  $\delta_{x,y}$  genannt.

Sowohl den Prozess, aus einer gegebenen Belegung  $\beta$  den Wahrheitswert  $\beta(F)$  auszurechnen, als auch diesen Wahrheitswert selbst, nenne ich **Auswertung** [evaluation] von  $F$  (in dem von  $\beta$  gegebenen Kontext).

Man sieht aus dieser Definition, dass sich der Wahrheitswert einer zusammengesetzten Formel  $F$  unabhängig von der konkreten Gestalt der Teilformeln nur aus dem *führenden Junktor* (d. h. dem Etikett der Wurzel) und den Wahrheitswerten der Teilformeln berechnet. Dieses Prinzip heißt *Frege-Prinzip* oder *Kompositionalitätsprinzip* [principle of compositionality]. Daraus ergibt sich auch die *Kontextfreiheit* [context-freeness], die bedeutet, dass die Berechnung unabhängig davon ist, in welcher umfassenderen Formel  $F$  als Teilformel vorkommen mag.

Die Infix-Notation nutzend haben die Junktoren also die folgende Wahrheitswertfunktionalität:

$\top$	$\perp$	$F$	$\neg F$	$F$	$G$	$(F \wedge G)$	$(F \vee G)$	$(F \rightarrow G)$	$(F \leftrightarrow G)$
1	0	0	1	0	0	0	0	1	1
		1	0	0	1	0	1	1	0
				1	0	0	1	0	0
				1	1	1	1	1	1

**Lemma 1.4.2** *Der Wahrheitswert einer Formel  $F$  unter einer Belegung hängt nur von den in  $F$  vorkommenden Aussagenvariablen ab, d. h. wenn  $\beta, \beta'$  zwei Belegungen sind mit  $\beta(A_i) = \beta'(A_i)$  für alle Aussagenvariablen  $A_i$ , die Etikett einer Ecke von  $F$  sind, dann gilt  $\beta(F) = \beta'(F)$ .*

BEWEIS: Das ist einerseits offensichtlich, soll aber andererseits doch einmal ordentlich bewiesen werden, und zwar per Induktion über die Tiefe von  $F$ .<sup>5</sup>

*Induktionsanfang:* Ist  $\text{dp}(F) = 0$ , dann ist  $F$  ein einelementiger Baum mit einer Aussagenvariable  $A_i$  als Etikett.  $A_i$  kommt also in  $F$  vor, und nach Voraussetzung ist dann

$$\beta(F) = \beta(A_i) = \beta'(A_i) = \beta'(F)$$

*Induktionsschritt:* Ist  $\text{dp}(F) = n + 1$ , so hat die Wurzel einen  $k$ -stelligen Junktor  $*$  als Etikett und ihre Nachfolger sind Teilformeln  $F_1, \dots, F_k$  (mit  $k \in \{0, 1, 2\}$ ). Da dann  $\text{dp}(F_i) \leq n$  für alle  $i = 1, \dots, k$ , ist nach Induktionsannahme  $\beta(F_i) = \beta'(F_i)$  und somit (jetzt Polnische Notation nutzend, ohne dies mit  $\hat{\phantom{x}}$  zu markieren)

$$\begin{aligned} \beta(F) &= \beta(*F_1 \dots F_k) = \tilde{*}(\beta(F_1), \dots, \beta(F_k)) = \\ &= \tilde{*}(\beta'(F_1), \dots, \beta'(F_k)) = \beta'(*F_1 \dots F_k) = \beta'(F) \quad \square \end{aligned}$$

Für die Berechnung von  $\beta(F)$  ist also nur die *partielle Belegung der in  $F$  vorkommenden Aussagenvariablen* relevant, d. h. die Einschränkung von  $\beta$  auf die in  $F$  vorkommenden Aussagenvariablen. Dafür benutze ich synonym die beiden abkürzenden Ausdrücke „partielle Belegung“ und „Belegung der (in  $F$ ) vorkommenden Aussagenvariablen“ (und spreche manchmal missbräuchlich nur von „Belegung“).

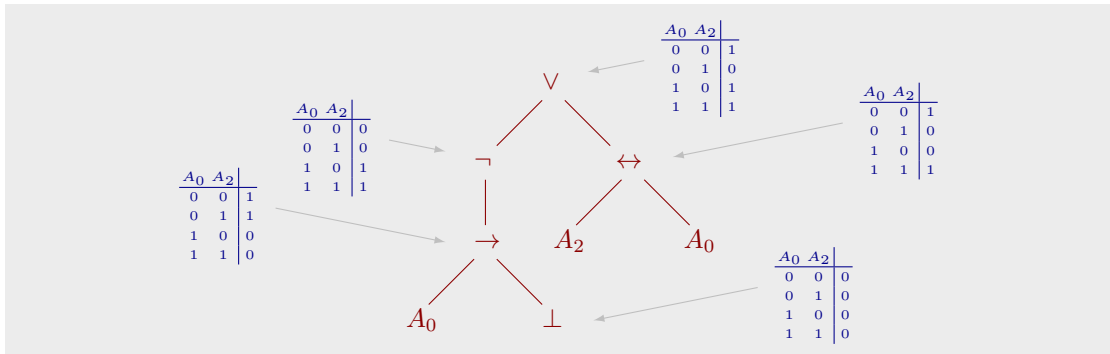
Als nützlich hat sich folgende (verbreitete) Notation erwiesen: Man schreibt  $F(A_{i_1}, \dots, A_{i_m})$  um auszudrücken, dass  $F$  eine aussagenlogische Formel ist, dass die Aussagenvariablen  $A_{i_1}, \dots, A_{i_m}$

<sup>5</sup>Der Beweis zeigt zudem die Nützlichkeit der Tiefe. Ebensogut könnte man die Induktion über die Höhe oder die Länge machen, allerdings bräuchte man dann neben den Aussagenvariablen auch Verum und Falsum als Induktionsanfang!

paarweise verschieden sind und dass sich jede in  $F$  vorkommende Aussagenvariable darunter befindet. Es müssen aber nicht unbedingt alle  $A_{i_1}, \dots, A_{i_m}$  tatsächlich in  $F$  vorkommen.

Es gibt  $2^m$  verschiedene Abbildungen  $\{A_{i_1}, \dots, A_{i_m}\} \rightarrow \{0, 1\}$ . Kommen in  $F(A_{i_1}, \dots, A_{i_m})$  die  $m$  Aussagenvariablen  $A_{i_1}, \dots, A_{i_m}$  tatsächlich alle vor, so gibt es also  $2^m$  relevante partielle Belegungen (der in  $F$  vorkommenden Aussagenvariablen). Die Abbildung, die jeder solchen partiellen Belegung  $\beta$  den Wahrheitswert  $\beta(F)$  zuordnet, wird **Wahrheitswertverlauf** von  $F$  oder *die von  $F$  induzierte Wahrheitswertfunktion* genannt.<sup>6</sup>

Den Wahrheitswertverlauf einer Formel  $F$  kann man induktiv von den Blättern des Baumes bis hoch zu der Wurzel berechnen:



Einfacher ist es, dies in einer Tabelle darzustellen, die alle Teilformeln in einer geeigneten Reihenfolge enthält:

$A_0$	$A_2$	$\perp$	$(A_0 \rightarrow \perp)$	$\neg(A_0 \rightarrow \perp)$	$(A_2 \leftrightarrow A_0)$	$(\neg(A_0 \rightarrow \perp) \vee (A_2 \leftrightarrow A_0))$
0	0	0	1	0	1	1
0	1	0	1	0	0	0
1	0	0	0	1	0	1
1	1	0	0	1	1	1

Man kann dies viel kompakter aufschreiben, indem man in einer der sequenziellen Darstellungsweisen die Wahrheitswertverläufe der Teilformeln unter deren führenden Junktoren notiert. In Infix-Notation (linke Tabelle), in der man als Mensch allerdings leicht den Überblick verliert, muss man zusätzlich angeben, in welcher Spalte der gesuchte Wahrheitswertverlauf steht.

$(\neg(A_0 \rightarrow \perp) \vee (A_2 \leftrightarrow A_0))$							$A_0 \perp \rightarrow \neg A_2 \ A_0 \leftrightarrow \vee$							
0	0	1	0	1	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	1	0	0	0	1	0	1	0	0	0
1	1	0	0	1	0	0	1	0	0	1	0	1	0	1
1	1	0	0	1	1	1	1	0	0	1	1	1	1	1

Geeigneter hierfür ist die Umgekehrte Polnische Notation, in der man die Spalten von links nach rechts auffüllen kann und in der der gesuchte Wahrheitswertverlauf stets in der Spalte ganz rechts steht (rechte Tabelle).

<sup>6</sup>Eine partielle Belegung  $\beta$  der Aussagenvariablen  $A_{i_1}, \dots, A_{i_m}$  kann man identifizieren mit dem  $m$ -Tupel  $(\beta(A_{i_1}), \dots, \beta(A_{i_m})) \in \{0, 1\}^m$ , sofern man weiß, welche Aussagenvariablen in welcher Reihenfolge vorkommen. Die Reihenfolge ist hier durch die Indizes vorgegeben.

**Bemerkung 1.4.3** Die Auswertung einer Formel unter einer Belegung definiert eine „Paarung“

$$\begin{aligned} \{F \mid F \text{ aussagenlogische Formel}\} &\times \{\beta \mid \beta \text{ Belegung}\} \rightarrow \{0, 1\} \\ (F, \beta) &\mapsto \beta(F) \end{aligned}$$

Für festes  $\beta$  erhält man daraus eine Abbildung von den aussagenlogischen Formeln nach  $\{0, 1\}$ ,  $F \mapsto \beta(F)$ . Diese Abbildung ist eine Fortsetzung von  $\beta$  und motiviert die Notation „ $\beta(F)$ “.

Andererseits erhält man für festes  $F$  eine Abbildung von den Belegungen nach  $\{0, 1\}$ ,  $\beta \mapsto \beta(F)$ . Diese Abbildung hängt nur von den Werten von  $\beta$  auf den  $m$  in  $F$  vorkommenden Aussagenvariablen  $A_{i_1}, \dots, A_{i_m}$  ab. Die Wahrheitstafel von  $F$  kann man daher als eine Wertetabelle dieser Funktion auffassen:

$A_{i_1}$	$\dots$	$A_{i_m}$	$F$
$\beta_0(A_{i_1})$	$\dots$	$\beta_0(A_{i_m})$	$\beta_1(F)$
$\beta_1(A_{i_1})$	$\dots$	$\beta_1(A_{i_m})$	$\beta_2(F)$
$\vdots$	$\dots$	$\vdots$	$\vdots$
$\beta_{2^k-1}(A_{i_1})$	$\dots$	$\beta_{2^k-1}(A_{i_m})$	$\beta_{2^k}(F)$

wobei  $\beta_0, \beta_1, \dots, \beta_{2^k-1}$  eine Folge von Belegungen ist, die auf den in  $F$  vorkommenden Aussagenvariablen alle möglichen partiellen Belegungen durchläuft.<sup>7</sup> Dieser Betrachtungsweise entspricht eine andere verbreitete Notation für die Auswertung einer Formel  $F$  unter einer Belegung  $\beta$ , nämlich  $F[\beta]$ .

Mit Hilfe der Semantik, also der Auswertung von Formeln, kann man nun die für die Logik zentralen Begriffe *logische Folgerung*, *logische Äquivalenz* und *Tautologie* definieren:

**Definition 1.4.4**

(a) Eine aussagenlogische Formel  $F$  heißt **Tautologie** [tautology] oder allgemeingültig [universally valid], falls  $\beta(F) = 1$  für alle Belegungen  $\beta$ . Dafür schreibt man  $\vdash F$ .

$F$  heißt **erfüllbar** [satisfiable] oder konsistent [consistent], falls es eine Belegung  $\beta$  gibt mit  $\beta(F) = 1$ .

(b) Zwei aussagenlogische Formeln  $F$  und  $G$  heißen **logisch äquivalent** [logically equivalent] zueinander, falls  $\beta(F) = \beta(G)$  für alle Belegungen  $\beta$ . Dafür schreibe ich  $F \sim G$ .

(c) Eine aussagenlogische Formel  $G$  wird von einer Menge von Formeln  $\{F_i \mid i \in I\}$  **impliziert** [is implied by] oder **folgt logisch aus** [follows from] dieser Menge, falls  $\beta(G) = 1$  für jede Belegung  $\beta$ , die für alle  $i \in I$  die Bedingung  $\beta(F_i) = 1$  erfüllt. Dafür schreibt man  $\{F_i \mid i \in I\} \vdash G$  bzw. im Fall einer endlichen Menge  $\{F_1, \dots, F_n\}$  auch  $F_1, \dots, F_n \vdash G$ .

(d) Eine Menge von Formeln  $\{F_i \mid i \in I\}$  heißt **erfüllbar** [satisfiable] oder widerspruchsfrei [consistent], falls es eine Belegung  $\beta$  gibt mit  $\beta(F_i) = 1$  für alle  $i \in I$ , und andernfalls widersprüchlich [contradictory].

Statt „logisch äquivalent“ und „folgt logisch“ sage ich oft kurz „äquivalent“ und „folgt“.

Es ist offensichtlich, dass die logische Äquivalenz eine Äquivalenzrelation (d. h. eine reflexive, symmetrische und transitive Relation) auf der Menge der aussagenlogischen Formeln ist.

<sup>7</sup>Das macht man am besten so, dass die  $\{0, 1\}$ -Folge  $\beta_i(A_{i_1}) \dots \beta_i(A_{i_m})$  als Binärzahl gelesen gerade  $i$  ist.

Aus der Definition ergibt sich unmittelbar das *ex-falso-quodlibet*-Prinzip: Jede beliebige Formel  $F$  folgt aus einer widersprüchlichen Formelmenge. Insbesondere gilt  $\perp \vdash F$ . „Dual“ dazu folgt das Verum aus jeder beliebigen Formel:  $F \vdash \top$  („*verum ex quodlibet*“).<sup>8</sup>

Ich werde von nun an frei zwischen den verschiedenen Darstellungsarten von Formeln wechseln und meistens die Infix-Notation nutzen, bei Beweisen aber gerne die Polnische Notation.

Die verschiedenen gerade definierten Begriffe wie *logische Äquivalenz*, *Tautologie* lassen sich mit Hilfe einiger der Junktoren ineinander übersetzen:

**Satz 1.4.5**

- (a)  $\vdash F \iff \emptyset \vdash F \iff \top \vdash F \iff F \sim \top \iff \neg F \text{ ist nicht erfüllbar.}$
- (b)  $F \sim G \iff \vdash (F \leftrightarrow G) \iff (F \vdash G \text{ und } G \vdash F)$
- (c)  $F_1, \dots, F_n \vdash G \iff ((\cdot (F_1 \wedge F_2) \wedge \dots) \wedge F_n) \vdash G$   
 $\iff \vdash (((\cdot (F_1 \wedge F_2) \wedge \dots) \wedge F_n) \rightarrow G)$
- (d)  $\{F_1, \dots, F_n\} \text{ ist erfüllbar} \iff ((\cdot (F_1 \wedge F_2) \wedge \dots) \wedge F_n) \text{ ist erfüllbar}$   
 $\{F_1, \dots, F_n\} \text{ ist nicht erfüllbar} \iff ((\cdot (F_1 \wedge F_2) \wedge \dots) \wedge F_n) \sim \perp$   
 $\iff F_1, \dots, F_n \vdash \perp$

BEWEIS: Übung – alles folgt unmittelbar aus den Definitionen! □

Im übernächsten Abschnitt folgen wichtige Beispiele. Vorher soll noch gezeigt werden, dass die Konstruktion der Aussagenlogik keine weiteren Junktoren „vergessen“ hat in dem Sinne, dass jeder Wahrheitswertverlauf durch eine aussagenlogische Formel erreicht werden kann.

**1.5 Disjunktive und konjunktive Normalform**

Im weiteren Verlauf werde ich folgende abkürzenden Schreibweisen verwenden:

$$(F_1 \wedge F_2 \wedge \dots \wedge F_n) \text{ oder } \bigwedge_{i=1}^n F_i \text{ für die Formel } ((\cdot (F_1 \wedge F_2) \wedge \dots) \wedge F_n) \quad \bigwedge_{i=1}^n$$

$$(F_1 \vee F_2 \vee \dots \vee F_n) \text{ oder } \bigvee_{i=1}^n F_i \text{ für die Formel } ((\cdot (F_1 \vee F_2) \vee \dots) \vee F_n) \quad \bigvee_{i=1}^n$$

Für  $n = 1$  sei dabei  $\bigwedge_{i=1}^1 F_i = \bigvee_{i=1}^1 F_i = F_1$ ; für  $n = 0$  sei  $\bigwedge_{i=1}^0 F_i = \top$  und  $\bigvee_{i=1}^0 F_i = \perp$ .

Diese Sonderfälle sind sinnvoll, weil nun z. B. für alle  $l = 0, \dots, n$  die Äquivalenz

$$\bigwedge_{i=1}^n F_i \sim \left( \bigwedge_{i=1}^l F_i \wedge \bigwedge_{i=l+1}^n F_i \right)$$

gilt, analog für die Disjunktion.<sup>9</sup>

---

<sup>8</sup>Näheres zum Konzept der Dualität siehe Abschnitt 1.8.  
<sup>9</sup>Interessiert man sich für aussagenlogische Formeln nur bis auf logische Äquivalenz, spielt die Reihenfolge und die Klammerung innerhalb einer Konjunktion bzw. Disjunktion keine Rolle, da sich bei anderer Reihenfolge

**Definition 1.5.1** Ein **Literal** [literal] ist eine aussagenlogische Formel, die entweder eine Aussagenvariable  $A_i$  oder eine negierte Aussagenvariable  $\neg A_i$  ist.

Eine Formel ist in **disjunktiver Normalform (DNF)** [disjunctive normal form], wenn sie eine Disjunktion sogenannter **Konjunktionsterme** ist, d. h. von der Form

DNF

$$((L_{11} \wedge \dots \wedge L_{1n_1}) \vee \dots \vee (L_{k1} \wedge \dots \wedge L_{kn_k})) \quad \text{bzw.} \quad \bigvee_{i=1}^k \bigwedge_{j=1}^{n_i} L_{ij},$$

eine Formel ist in **konjunktiver Normalform (KNF)** [conjunctive normal form, CNF], wenn sie eine Konjunktion sogenannter **Disjunktionsterme** oder **Klauseln** [clauses] ist, d. h. von der Form

KNF

$$((L_{11} \vee \dots \vee L_{1n_1}) \wedge \dots \wedge (L_{k1} \vee \dots \vee L_{kn_k})) \quad \text{bzw.} \quad \bigwedge_{i=1}^k \bigvee_{j=1}^{n_i} L_{ij},$$

wobei die  $L_{ij}$  jeweils Literale sind und  $k, n_1, \dots, n_k \in \mathbb{N}$ .

Es lohnt sich wieder, einen Blick auf die Sonderfälle zu werfen:

	DNF	KNF
$k = 0$	$\perp$	$\top$
$k = 1, n_1 = 0$	$\top$	$\perp$
$k = 1, n_1 = 1$	$L_{11}$	$L_{11}$
$k = 1, n_1$ beliebig	$(L_{11} \wedge \dots \wedge L_{1n_1})$	$(L_{11} \vee \dots \vee L_{1n_1})$
$k$ beliebig, $n_1 = \dots = n_k = 1$	$(L_{11} \vee \dots \vee L_{k1})$	$(L_{11} \wedge \dots \wedge L_{k1})$

Also sind  $\top$  und  $\perp$ , einzelne Literale sowie reine Konjunktionen und reine Disjunktionen von Literalen sowohl Formeln in DNF als auch Formeln in KNF.

**Satz 1.5.2** Jede Formel ist logisch äquivalent zu einer Formel in disjunktiver Normalform und logisch äquivalent zu einer Formel in konjunktiver Normalform.

Allgemeiner: Für gegebene endlich viele Aussagenvariablen lässt sich jeder beliebige Wahrheitswertverlauf durch eine Formel (in disjunktiver Normalform oder konjunktiver Normalform) erreichen.

Der Beweis soll zunächst an einem Beispiel veranschaulicht werden. Für eine Formel  $F$  mit dem in der Tabelle vorgegebenen Wahrheitswertverlauf soll eine äquivalente Formel in DNF gefunden werden. Zunächst wird für jeden Wahrheitswert 1 im Wahrheitswertverlauf von  $F$  eine Formel konstruiert, die genau an dieser Stelle den Wahrheitswert 1 hat und sonst überall 0. Dazu modifiziert man die Konjunktion der vorkommenden Aussagenvariablen durch Negationen an den notwendigen Stellen, nämlich bei den Aussagenvariablen, denen unter der betrachteten

logisch äquivalente Formeln ergeben. Man kann dann auch Mengennotationen wie  $\bigwedge \{F_i \mid i \in I\}$  etc. zulassen, und allgemeiner gilt für Indexmengen  $I$  und  $I'$  dann

$$\bigwedge_{i \in I \cup I'} F_i \sim \left( \bigwedge_{i \in I} F_i \wedge \bigwedge_{i \in I'} F_i \right).$$



partiellen Belegung der Wahrheitswert 0 zugewiesen wird:

$A_0$	$A_1$	$A_2$	$F$	$(A_0 \wedge A_1 \wedge A_2)$	$(A_0 \wedge \neg A_1 \wedge A_2)$	$(\neg A_0 \wedge \neg A_1 \wedge A_2)$
0	0	0	0	0	0	0
0	0	1	1	0	0	1
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	0	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	0
1	1	1	1	1	0	0

Nun ist  $F$  äquivalent zur Disjunktion dieser Formeln, also

$$F \sim ((A_0 \wedge A_1 \wedge A_2) \vee (A_0 \wedge \neg A_1 \wedge A_2) \vee (\neg A_0 \wedge \neg A_1 \wedge A_2)).$$

Für die KNF geht man die Wahrheitswerte 0 im vorgegebenen Wahrheitswertverlauf durch:

$A_0$	$A_1$	$A_2$	$F$	$(A_0 \vee \neg A_1 \vee A_2)$		$(\neg A_0 \vee A_1 \vee A_2)$	
				$(A_0 \vee A_1 \vee A_2)$	$(A_0 \vee \neg A_1 \vee \neg A_2)$	$(\neg A_0 \vee A_1 \vee A_2)$	$(\neg A_0 \vee \neg A_1 \vee A_2)$
0	0	0	0	0	1	1	1
0	0	1	1	1	1	1	1
0	1	0	0	1	0	1	1
0	1	1	0	1	1	0	1
1	0	0	0	1	1	1	0
1	0	1	1	1	1	1	1
1	1	0	0	1	1	1	0
1	1	1	1	1	1	1	1

und erhält eine zu  $F$  logisch äquivalente Formel als Konjunktion der Disjunktionsterme:

$$F \sim ((A_0 \vee A_1 \vee A_2) \wedge (A_0 \vee \neg A_1 \vee A_2) \wedge (A_0 \vee \neg A_1 \vee \neg A_2) \wedge (\neg A_0 \vee A_1 \vee A_2) \wedge (\neg A_0 \vee \neg A_1 \vee A_2))$$

BEWEIS: durch Kodieren der Wahrheitstafel wie im Beispiel, zunächst für die DNF.

Ohne Einschränkung kommen in  $F$  genau die  $n$  Aussagenvariablen  $A_0, \dots, A_{n-1}$  vor (sonst benennt man sie um). Für jede partielle Belegung der vorkommenden Aussagenvariablen  $\beta$  soll das Symbol  $[\neg_\beta A_i]$  für folgende Formel stehen:

$$[\neg_\beta A_i] := \begin{cases} A_i & \text{falls } \beta(A_i) = 1 \\ \neg A_i & \text{falls } \beta(A_i) = 0 \end{cases}$$

Man sieht nun problemlos, dass für eine Belegung  $\beta'$  genau dann  $\beta' \left( \bigwedge_{i=0}^{n-1} [\neg_\beta A_i] \right) = 1$  gilt, wenn  $\beta'$  und  $\beta$  auf  $A_0, \dots, A_{n-1}$  übereinstimmen. Es folgt:

$$F \sim \bigvee_{\beta(F)=1} \bigwedge_{i=0}^{n-1} [\neg_\beta A_i]$$

(weil die rechte Seite von einer Belegung  $\beta'$  genau dann wahr gemacht wird, wenn sie eine

der Teilformeln  $\bigwedge_{i=0}^{n-1} [\neg_{\beta} A_i]$  wahr macht, also genau dann, wenn sie auf den vorkommenden Aussagenvariablen mit einem  $\beta$  übereinstimmt, das  $F$  wahr macht).

Durch die „duale Konstruktion“ bekommt man die KNF als  $F \sim \bigwedge_{\beta(F)=0} \bigvee_{i=0}^{n-1} \neg[\neg_{\beta} A_i]$ .  $\square$

Die im diesem Beweis konstruierte disjunktive Normalform hat die Eigenschaft, dass in jedem Konjunktionsterm jede der in  $F$  vorkommenden Aussagenvariablen genau einmal vorkommt. Eine DNF mit dieser Eigenschaft ist eindeutig festgelegt bis auf die Reihenfolge der Konjunktionsterme, die Reihenfolge der Literale innerhalb der Konjunktionsterme und die Klammerungsreihenfolge. Legt man dafür Konventionen fest (z. B. Linksklammerung, Aussagenvariable innerhalb der Konjunktionsterme nach Indizes aufsteigend geordnet und partielle Belegungen lexikographisch), so erhält man eine eindeutige DNF, die *kanonische disjunktive Normalform* [canonical disjunctive normal form] (analog für die KNF).

### Beispiele:

(a)  $(A_1 \rightarrow A_0)$  ist logisch äquivalent zu  $(A_0 \vee \neg A_1)$ , was bereits eine Formel in DNF und in kanonischer KNF ist. Die kanonische DNF ist  $((\neg A_0 \wedge \neg A_1) \vee (A_0 \wedge \neg A_1) \vee (A_0 \wedge A_1))$ .

(b) Für die dreistellige Paritätsformel  $F = \neg((A_0 \leftrightarrow A_1) \leftrightarrow A_2)$  mit Wahrheitstafel

$A_0$	$A_1$	$A_2$	$F$	
0	0	0	1	ist die kanonische DNF:  $((\neg A_0 \wedge \neg A_1 \wedge \neg A_2) \vee (\neg A_0 \wedge A_1 \wedge A_2) \vee (A_0 \wedge \neg A_1 \wedge A_2) \vee (A_0 \wedge A_1 \wedge \neg A_2))$
0	0	1	0	
0	1	0	0	
0	1	1	1	
1	0	0	0	und die kanonische KNF:  $((A_0 \vee A_1) \vee \neg A_2) \wedge (A_0 \vee \neg A_1 \vee A_2) \wedge (\neg A_0 \vee A_1 \vee A_2) \wedge (\neg A_0 \vee \neg A_1 \vee \neg A_2)$
1	0	1	1	
1	1	0	1	
1	1	1	0	

**Definition 1.5.3** Eine Menge von Junktoren  $J$  heißt *vollständiges Junktorensystem* [complete system of connectives], falls es zu jeder aussagenlogischen Formel eine logisch äquivalente Formel gibt, in der keine anderen als Junktoren aus  $J$  vorkommen.

**Satz 1.5.4**  $\{\neg, \wedge\}$  und  $\{\neg, \vee\}$  bilden vollständige Junktorensysteme.

BEWEIS: Satz 1.5.2 zeigt, dass  $\{\neg, \vee, \wedge, \perp, \top\}$  ein vollständiges Junktorensystem ist, da man nur diese Junktoren für die DNF braucht. Man kann nun aufgrund von logischen Gesetzen aus Satz 1.6.1 und des Prinzips der äquivalenten Substitution 1.6.4 manche dieser Junktoren durch andere ausdrücken: Wegen  $\perp \sim \neg \top$  und  $\top \sim (A_0 \vee \neg A_0)$  braucht man Verum und Falsum nicht, und wegen  $(F \wedge G) \sim \neg(\neg F \vee \neg G)$  und  $(F \vee G) \sim \neg(\neg F \wedge \neg G)$  kann man auch auf Konjunktion oder Disjunktion verzichten.  $\square$

Man kann die Anforderung an ein vollständiges Junktorensystem  $J$  dahingehend verstärken, dass jede aussagenlogische Formel  $F$  zu einer Formel äquivalent sein soll, in der nur Junktoren aus  $J$  vorkommen und keine weiteren Aussagenvariablen als die in  $F$  vorkommenden. Dann braucht man zusätzlich zu  $\{\neg, \wedge\}$  bzw.  $\{\neg, \vee\}$  noch jeweils  $\top$  oder  $\perp$ .

**Beispiel:**  $\{\rightarrow, \perp\}$  ist ein weiteres vollständiges Junktorensystem im stärkeren Sinne.

Man kann das Konzept auch auf „neue“ Junktoren ausdehnen, die man zur Sprache hinzunehmen könnte. Zwei verbreitete solche Junktoren sind der *NOR-Junktor*  $\downarrow$  und der *NAND-Junktor*  $\uparrow$ , deren Wahrheitswertfunktionalitäten so sind, dass  $(F \downarrow G) \sim \neg(F \vee G)$  bzw.  $(F \uparrow G) \sim \neg(F \wedge G)$  gilt.<sup>10</sup> Alternativ könnte man solche neuen Junktoren auch als Abkürzungen einführen, in diesem Beispiel also dadurch, dass die in den Äquivalenzen links stehende Formel als Abkürzung für die jeweils rechts stehende Formel definiert wird.

Es sind nun  $\{\downarrow\}$  und  $\{\uparrow\}$  vollständige Junktorensysteme gemäß Definition 1.5.3 (und es sind die beiden einzigen vollständigen Junktorensysteme, die aus einem einzelnen Junktor der Stelligkeit höchstens 2 bestehen).

Welche Junktoren man für die Konstruktion der Aussagenlogik auswählt, ist in gewisser Weise Geschmackssache. Man kann Satz 1.4.5 und einige Regeln aus dem nächsten Abschnitt aber so verstehen, dass die Junktoren  $\top, \perp, \neg, \wedge, \vee, \rightarrow$  und  $\leftrightarrow$  auf natürliche Weise in der Aussagenlogik eine Rolle spielen, wenn man sich für die dort behandelten Konzepte wie z. B. logische Äquivalenz und Erfüllbarkeit interessiert. Sie kommen aber auch aus den Konzepten, die man für das mathematische Beweisen braucht, und teilweise haben sie auch Entsprechungen in der natürlichen Sprache (vor allem  $\neg, \wedge$  und  $\vee$ ) und ergeben sich daher auch, wenn man Argumentationen in natürlicher Sprache in formaler Sprache modellieren möchte.

Je nach Anwendung kann es sinnvoll sein, sich auf wenige Junktoren zu beschränken oder auch neue Junktoren hinzuzunehmen. Insbesondere werden Beweise über den Aufbau der Formeln oft kürzer, wenn man im Induktionsschritt nur wenige Junktoren zu behandeln hat. Gerne nimmt man dann  $\neg$  und  $\wedge$ .<sup>11</sup>

Die Sätze 1.5.2 und 1.5.4 lassen sich in eine Aussage über Wahrheitswertfunktionen übersetzen. Dazu braucht es die zwei (hier speziell für Wahrheitswertfunktionen definierten) Begriffe:

Eine *Projektion* ist eine Abbildung  $\pi_i^n : \{0, 1\}^n \rightarrow \{0, 1\}$ ,  $(x_1, \dots, x_n) \mapsto x_i$ . Eine *verallgemeinerte Komposition* von Abbildungen  $g : \{0, 1\}^m \rightarrow \{0, 1\}$  und  $h_1, \dots, h_m : \{0, 1\}^n \rightarrow \{0, 1\}$  ist die Abbildung

$$g \circ (h_1, \dots, h_m) : \{0, 1\}^n \rightarrow \{0, 1\}, (x_1, \dots, x_n) \mapsto g(h_1(x_1, \dots, x_n), \dots, h_m(x_1, \dots, x_n))$$

**Satz 1.5.5** Man erhält jede Wahrheitswertfunktion  $\{0, 1\}^n \rightarrow \{0, 1\}$  für  $n \geq 1$  aus den beiden Funktionen  $x \mapsto 1 - x$  und  $(x, y) \mapsto \min\{x, y\}$  und allen Projektionen  $\pi_i^n$  durch iterierte verallgemeinerte Komposition.<sup>12</sup>

BEWEIS: Identifiziert man die Argumente der Wahrheitswertfunktion  $\{0, 1\}^n \rightarrow \{0, 1\}$  mit den Aussagenvariablen  $A_1, \dots, A_n$ , so gibt die Funktion einen Wahrheitswertverlauf an, der sich durch eine Formel beschreiben lässt, in der neben den Aussagenvariablen nur die Junktoren  $\neg$

<sup>10</sup>NAND wird auch  $|$  geschrieben und *Sheffer-Strich* genannt; NOR heißt auch *Peirce-Funktion* oder *-Operator*.

<sup>11</sup>NAND und NOR, die die Beweise womöglich weiter verkürzen würden, haben sich nicht durchgesetzt, vermutlich weil Negation und Konjunktion für menschen intuitivere Konzepte sind und daher leichter zu verstehen.

<sup>12</sup>Will man jede Wahrheitswertfunktion  $\{0, 1\}^n \rightarrow \{0, 1\}$  für  $n \geq 0$ , muss man eine der beiden (Verum und Falsum entsprechenden) nullstelligen Funktionen hinzunehmen, da man bei verallgemeinerten Kompositionen keine Argumente verlieren kann.

und  $\wedge$  vorkommen. Der Wahrheitswertverlauf ist nach Definition 1.4.1 eine iterierte Komposition der Funktionen  $x \mapsto 1 - x$  und  $(x, y) \mapsto \min\{x, y\}$ , denen man mit Hilfe der Projektionen die richtigen Argumente zuweist.  $\square$

## 1.6 Gesetze der klassischen zweiwertigen Aussagenlogik

**Logische Gesetze** [laws of logic] besagen, dass gewisse Formeln logisch äquivalent zueinander sind, dass gewisse Formeln Tautologien sind oder dass zwischen gewissen Formeln eine Beziehung logischer Folgerung vorliegt. Logische Gesetze werden auch *logische Regeln* genannt und manche traditionell *Prinzipien*, weil sie grundlegend für die klassische Aussagenlogik sind.

Da es unendlich viele Formeln in  $n$  fest gewählten Aussagenvariablen gibt, dafür aber nur  $2^n$  mögliche Wahrheitswertverläufe, gibt es „viele“ Äquivalenzen zwischen Formeln, insbesondere unendlich viele logische Gesetze. Von besonderem Interesse sind *elementare* logische Gesetze, die man als Umformungs- oder Vereinfachungsregeln auffassen kann. Man könnte versuchen, solche Regeln in verschiedene Gruppen aufzuteilen, etwa:

- Vereinfachungsregeln, z. B.  $(F \wedge \top) \sim F$  oder  $\neg\neg F \sim F$
- Kommutativgesetze, z. B.  $(F \wedge G) \sim (G \wedge F)$
- Ersetzungsregeln, z. B.  $(F \rightarrow G) \sim (\neg F \vee G)$
- Vertauschungsregeln für das Aufeinandertreffen von zwei Junktoren, z. B.  $\neg(F \wedge G) \sim (\neg F \vee \neg G)$  und  $((F \wedge G) \wedge H) \sim (F \wedge (G \wedge H))$

**Satz 1.6.1** *Es gelten die folgenden elementaren logischen Gesetze für alle aussagenlogischen Formeln  $F, G, H$ :*

$\top$ -/ $\perp$ -Gesetze:		
$\neg\top \sim \perp$	$\neg\perp \sim \top$	<i>Verum-Falsum-Dualität</i>
$(F \wedge \top) \sim F$	$(F \vee \perp) \sim F$	<i>Neutrale Elemente</i> [neutral elements]
$(F \wedge \perp) \sim \perp$	$(F \vee \top) \sim \top$	<i>Absorbierende Elemente</i> [absorbing elements]
$\neg$ -Gesetze:		
$\neg\neg F \sim F$	<i>Doppelnegationsgesetz</i> [double negation]	
$(F \vee \neg F) \sim \top$	<i>Prinzip des ausgeschlossenen Dritten</i> [law of excluded middle]	
$(F \wedge \neg F) \sim \perp$	<i>Prinzip des ausgeschlossenen Widerspruchs</i> [law of non-contradiction]	
$\wedge$ -/ $\vee$ -Gesetze:		
$(F \wedge F) \sim F$	$(F \vee F) \sim F$	<i>Idem.</i>
$(F \wedge G) \sim (G \wedge F)$	$(F \vee G) \sim (G \vee F)$	<i>Kom.</i>
$((F \wedge G) \wedge H) \sim (F \wedge (G \wedge H))$	$((F \vee G) \vee H) \sim (F \vee (G \vee H))$	<i>Ass.</i>
$((F \wedge G) \vee H) \sim ((F \vee H) \wedge (G \vee H))$	$((F \vee G) \wedge H) \sim ((F \wedge H) \vee (G \wedge H))$	<i>Distr.</i>
Gesetze von <i>de Morgan</i> :		
$\neg(F \wedge G) \sim (\neg F \vee \neg G)$	$\neg(F \vee G) \sim (\neg F \wedge \neg G)$	
Definition von $\rightarrow$ bzw. $\leftrightarrow$ :		
$(F \rightarrow G) \sim (\neg F \vee G)$	$(F \leftrightarrow G) \sim ((F \rightarrow G) \wedge (G \rightarrow F))$	

Die  $\wedge$ -/ $\vee$ -Gesetze heißen der Reihe nach: *Idempotenz* [idempotence], *Kommutativität* [commutativity] und *Assoziativität* [associativity] von  $\wedge$  bzw.  $\vee$  sowie *Distributivität* [distributivity] von  $\vee$  über  $\wedge$  bzw. umgekehrt.

BEWEIS: Nachprüfen mit Wahrheitstafeln! □

Diese elementaren logischen Gesetze reichen (zusammen mit dem Prinzip der äquivalenten Substitution, das gleich noch eingeführt wird) aus, um alle andern logischen Gesetze herzuleiten. Trotzdem ist es nützlich, einige weitere Gesetze zu kennen:

**Satz 1.6.2** *Es gelten für alle aussagenlogischen Formeln  $F, G, H$ :*

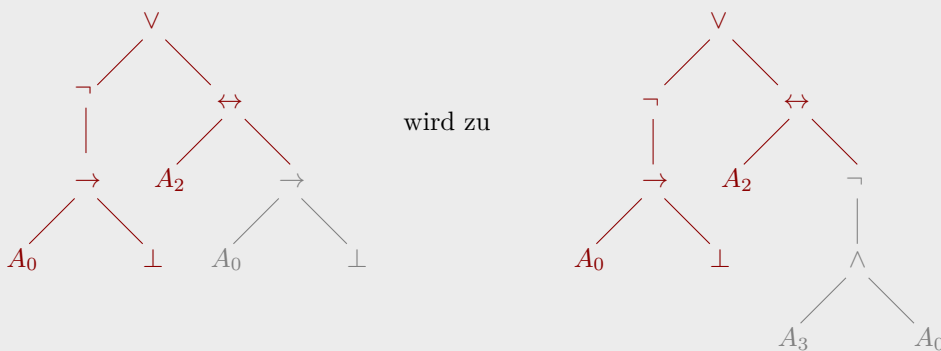
Definition von $\neg$ :	$(F \rightarrow \perp) \sim \neg F$	<i>und dual:</i> $(\top \rightarrow F) \sim F$
Absorptionsgesetze:	$((F \wedge G) \vee F) \sim F$	$((F \vee G) \wedge F) \sim F$
Kontraposition:	$(F \rightarrow G) \sim (\neg G \rightarrow \neg F)$	$(F \leftrightarrow G) \sim (\neg G \leftrightarrow \neg F)$
„Currying“:	$((F \wedge G) \rightarrow H) \sim (F \rightarrow (G \rightarrow H))$	
Transitivität von $\rightarrow$ :	$(F \rightarrow G), (G \rightarrow H) \vdash (F \rightarrow H)$	
( <i>daraus</i> modus ponens	$G, (G \rightarrow H) \vdash H$	<i>mit <math>F = \top</math></i>
<i>und</i> modus tollens	$(F \rightarrow G), \neg G \vdash \neg F$	<i>mit <math>H = \perp</math></i>
Links-Distributivität	$(F \rightarrow (G \wedge H)) \sim ((F \rightarrow G) \wedge (F \rightarrow H))$	
von $\rightarrow$ über $\wedge$ bzw. $\vee$ :	$(F \rightarrow (G \vee H)) \sim ((F \rightarrow G) \vee (F \rightarrow H))$	
Rechts-Antidistributivität	$((F \wedge G) \rightarrow H) \sim ((F \rightarrow H) \vee (G \rightarrow H))$	
von $\rightarrow$ über $\wedge$ bzw. $\vee$ :	$((F \vee G) \rightarrow H) \sim ((F \rightarrow H) \wedge (G \rightarrow H))$	

BEWEIS: Ebenfalls Nachprüfen mit Wahrheitstafeln! □

Um die logischen Gesetze anwenden zu können, braucht man noch das *Prinzip der äquivalenten Substitution*, das im Wesentlichen aussagt, dass man ein Gesetz wie  $\neg\neg F \sim F$  auch innerhalb einer umfassenderen Formel anwenden kann. In diesen Zusammenhang ordnet sich auch das zweite Substitutionsprinzip, das der *uniformen Substitution* ein, welches im Beispiel aussagt, dass es ausreicht, das Doppelnegationsgesetz für eine Aussagenvariable als  $\neg\neg A_0 \sim A_0$  zu formulieren.

**Lemma 1.6.3** *Wenn man in einer aussagenlogischen Formel  $F$  ein Vorkommen einer Teilformel  $F'$  durch eine Formel  $G$  ersetzt, bekommt man wieder eine aussagenlogische Formel.*

**Beispiel:** In der linken Formel wird das rechte Vorkommen der Teilformel  $(A_0 \rightarrow \perp)$  durch die Formel  $\neg(A_3 \wedge A_0)$  ersetzt:



Aus  $(\neg(A_0 \rightarrow \perp) \vee (A_2 \leftrightarrow (A_0 \rightarrow \perp)))$  wird  $(\neg(A_0 \rightarrow \perp) \vee (A_2 \leftrightarrow \neg(A_3 \wedge A_0)))$

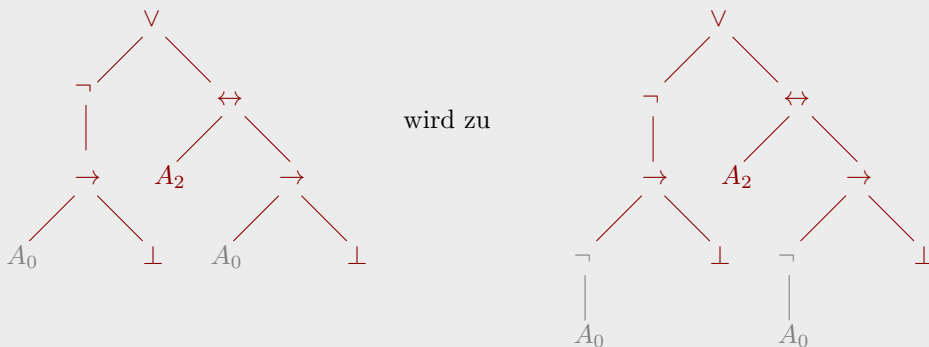
Beim Ersetzen werden alle Ecken der Teilformel  $F'$  aus  $F$  entfernt und dafür alle Ecken von  $G$  hinzugenommen. Die neue Kantenrelation besteht aus allen Kanten von  $F$ , in denen keine Ecke von  $F'$  vorkommt, aus allen Kanten von  $G$  und aus einer Kante von der Wurzel von  $G$  zu dem Vorgänger der Wurzel von  $F'$  (also zu der Ecke in  $F$ , deren Nachfolger die Wurzel von  $F'$  ist).

BEWEIS DES LEMMAS: Das wird am besten geglaubt – in jedem Einzelfall ist es offensichtlich; einen allgemeinen präzisen Beweis aufzuschreiben, ist aber mühsam und wenig instruktiv.  $\square$

Für das Ergebnis solch einer Substitution gibt es keine Notation, da man nicht auf das spezielle Vorkommen der Teilformel  $F'$ , falls es mehrere gibt, Bezug nehmen kann. Ein Spezialfall liegt vor, wenn sämtliche Vorkommen einer Teilformel  $F'$  in  $F$  durch  $G$  ersetzt werden: Dies nennt man eine *simultane Substitution* und schreibt dafür  $F[\frac{G}{F'}]$ . Achtung: Die Formel  $G$  könnte selbst wieder  $F'$  als Teilformel enthalten; diese neuen Vorkommen dürfen bei der simultanen Substitution nicht ersetzt werden!

$F[-]$

**Beispiel:** In der linken Formel  $F$  wird jedes Vorkommen der Teilformel  $A_0$  durch die Formel  $\neg A_0$  ersetzt, d. h. es wird  $F[\frac{\neg A_0}{A_0}]$  gebildet:



Aus  $(\neg(\boxed{A_0} \rightarrow \perp) \vee (A_2 \leftrightarrow (\boxed{A_0} \rightarrow \perp)))$  wird  $(\neg(\neg A_0 \rightarrow \perp) \vee (A_2 \leftrightarrow (\neg A_0 \rightarrow \perp)))$ .

**Satz 1.6.4 (Aussagenlogische Substitutionsprinzipien)**

Prinzip der äquivalenten Substitution:

Wenn  $F'$  Teilformel einer aussagenlogischen Formel  $F$  ist,  $F'$  logisch äquivalent zu  $G'$  ist und  $G$  aus  $F$  dadurch entsteht, dass ein Vorkommen von  $F'$  in  $F$  durch  $G'$  ersetzt wird, dann ist  $F$  logisch äquivalent zu  $G$ .

Prinzip der uniformen Substitution:

Wenn  $F$  und  $G$  zueinander logisch äquivalente Formeln sind, dann sind auch  $F[\frac{H}{A_i}]$  und  $G[\frac{H}{A_i}]$  zueinander logisch äquivalent, d. h. die Formeln, die durch simultane Substitution aller Vorkommen einer Aussagenvariablen  $A_i$  durch eine Formel  $H$  entstehen.

Insbesondere werden Tautologien durch äquivalente oder uniforme Substitution in Tautologien überführt. Das Prinzip der uniformen Substitution gilt auch für logische Folgerung anstelle von logischer Äquivalenz, d. h. aus  $F \vdash G$  folgt  $F[\frac{H}{A_i}] \vdash G[\frac{H}{A_i}]$ .

BEWEIS: Beide Prinzipien folgen sofort aus der Kompositionalität, da es nur auf die Wahrheitswerte von  $A_i$  bzw.  $F'$  und  $G'$  ankommt.  $\square$

**Beispiel:** Aus  $(A_0 \vee \neg A_0) \sim \top$  folgt mit  $A_0 \sim \neg \neg A_0$  und äquivalenter Substitution des zweiten Vorkommens von  $A_0$  die Äquivalenz  $(A_0 \vee \neg \neg A_0) \sim \top$ , und mit uniformer Substitution von  $A_0$  durch  $(A_1 \rightarrow A_2)$  folgt  $((A_1 \rightarrow A_2) \vee \neg(A_1 \rightarrow A_2)) \sim \top$ .

Achtung: Uniforme Substitution geht nicht mit beliebigen Teilformeln (die nur einen Wahrheitswert annehmen). Zum Beispiel gilt  $\top \sim (\top \rightarrow \top)$ , aber  $\top[\frac{A_0}{\top}] = A_0 \not\sim (A_0 \rightarrow A_0) = (\top \rightarrow \top)[\frac{A_0}{\top}]$ .

Mit dem Prinzip der uniformen Substitution hätte es ausgereicht, die logischen Gesetze aus den Sätzen 1.6.1 und 1.6.2 für Aussagenvariablen statt für beliebige aussagenlogische Formeln zu formulieren!

**Satz 1.6.5** *Alle logischen Äquivalenzen lassen sich durch sukzessive Anwendung der elementaren Gesetze aus Satz 1.6.1, des Prinzips der äquivalenten Substitution und der Symmetrie der logischen Äquivalenz („wenn  $F \sim G$ , dann auch  $G \sim F$ “) herleiten.*

Aus Satz 1.4.5 folgt zudem, dass sich jedes logische Gesetz, in dem nur endlich viele aussagenlogische Formeln vorkommen, in Form einer logischen Äquivalenz ausdrücken lässt.

BEWEIS: Es reicht zu zeigen, dass man mit den angegebenen Gesetzen jede Formel in ihre kanonische disjunktive Normalform umwandeln kann, und zwar bezüglich einer ggf. größeren Menge an Aussagenvariablen, weil zwei Formeln mit den gleichen vorkommenden Aussagenvariablen genau dann logisch äquivalent zueinander sind, wenn sie die gleiche kanonische disjunktive Normalform haben.<sup>13</sup> Die kanonische DNF wird mit folgendem Verfahren erreicht:

1. Ersetze alle Junktoren  $\leftrightarrow$  und dann alle Junktoren  $\rightarrow$  durch ihre Definition.
2. Vereinfache alle Vorkommen von  $\top$  und  $\perp$  mit Hilfe der  $\top$ -/ $\perp$ -Gesetze, bis nur noch eine Formel  $\top$  oder  $\perp$  übrig bleibt oder kein  $\top$  oder  $\perp$  mehr vorkommt.
3. Ziehe mit den *de Morgan*'schen Gesetzen alle Negationen „nach innen“ bis unmittelbar vor Aussagenvariablen oder weitere Negationszeichen. Eliminiere ggf. vorkommende Doppelnegationen mit der Doppelnegationsregel.
4. Wende solange das Distributivgesetz (von  $\wedge$  über  $\vee$ ) an, bis die Formel eine Disjunktion von Konjunktionen von Literalen ist.
5. Durch Kommutativität und Assoziativität können Konjunktionsterme beliebig sortiert werden, analog für die Disjunktionen, und dadurch DNF erreicht werden.
6. Ersetze doppelte Vorkommen von Literalen in einem Konjunktionsterm durch Idempotenz.
7. Ersetze gleichzeitiges Vorkommen von  $A_i$  und  $\neg A_i$  in einem Konjunktionsterm durch  $\perp$  (Prinzip des ausgeschlossenen Widerspruchs) und führe wieder den 2. Schritt durch. Dabei bleibt die Formel in DNF.
8. Ersetze ggf. einen Konjunktionsterm  $K$  durch  $(K \wedge \top) \sim (K \wedge (A_i \vee \neg A_i))$  und verwende Distributivität, Kommutativität und Assoziativität, um wieder DNF zu erreichen, bis in allen Konjunktionstermen alle Aussagenvariablen beider Formeln vorkommen.

<sup>13</sup> $(A_0 \vee \neg A_0)$  und  $(A_1 \vee \neg A_1)$  sind logisch äquivalente Formeln in kanonischer DNF, aber unterschiedlich, weil in den beiden Formeln nicht die gleichen Aussagenvariablen vorkommen.

Man muss sich nun noch davon überzeugen, dass das Verfahren stoppt und das gewünschte Ergebnis liefert.  $\square$

**Beispiel:** Beweis von  $(A_0 \wedge (A_1 \vee A_0)) \sim A_0$ , einem der Absorptionsgesetze, durch Umformung in kanonische DNF in den Aussagenvariablen  $A_0$  und  $A_1$  mit Hilfe der elementaren Umformungen, des Prinzips der äquivalenten Substitution (ÄS) und der Symmetrie von  $\sim$  (Sym):

$$\begin{array}{ll}
 (A_0 \wedge (A_1 \vee A_0)) & \\
 \sim ((A_1 \vee A_0) \wedge A_0) & \text{Kommutativität von } \wedge \\
 \sim ((A_1 \wedge A_0) \vee (A_0 \wedge A_0)) & \text{Distributivität von } \wedge \text{ über } \vee \\
 \sim ((A_1 \wedge A_0) \vee A_0) & \text{Idempotenz von } \wedge + \text{ÄS} \\
 \sim ((A_1 \wedge A_0) \vee (A_0 \wedge \top)) & \text{Gesetz für } \top + \text{ÄS} + \text{Sym} \\
 \sim ((A_1 \wedge A_0) \vee (A_0 \wedge (A_1 \vee \neg A_1))) & \text{ausgeschlossenes Drittes} + \text{ÄS} + \text{Sym} \\
 \sim ((A_1 \wedge A_0) \vee ((A_1 \vee \neg A_1) \wedge A_0)) & \text{Kommutativität von } \wedge + \text{ÄS} \\
 \sim ((A_1 \wedge A_0) \vee ((A_1 \wedge A_0) \vee (\neg A_1 \wedge A_0))) & \text{Distributivität von } \wedge \text{ über } \vee + \text{ÄS} \\
 \sim (((A_1 \wedge A_0) \vee (A_1 \wedge A_0)) \vee (\neg A_1 \wedge A_0)) & \text{Assoziativität von } \vee + \text{Sym} \\
 \sim ((A_1 \wedge A_0) \vee (\neg A_1 \wedge A_0)) & \text{Idempotenz von } \vee + \text{ÄS} \\
 \sim ((A_0 \wedge A_1) \vee (A_0 \wedge \neg A_1)) & \text{Kommutativität von } \wedge + \text{ÄS} \\
 \sim \underbrace{((A_0 \wedge \neg A_1) \vee (A_0 \wedge A_1))}_{\text{kanonische DNF}} & \text{Kommutativität von } \vee \\
 A_0 & \\
 \sim (A_0 \wedge \top) & \text{Gesetz für } \top + \text{Sym} \\
 \sim (A_0 \wedge (A_1 \vee \neg A_1)) & \text{ausgeschlossenes Drittes} + \text{ÄS} + \text{Sym} \\
 \sim ((A_1 \vee \neg A_1) \wedge A_0) & \text{Kommutativität von } \wedge \\
 \sim ((A_1 \wedge A_0) \vee (\neg A_1 \wedge A_0)) & \text{Distributivität von } \wedge \text{ über } \vee \\
 \sim ((A_0 \wedge A_1) \vee (A_0 \wedge \neg A_1)) & \text{Kommutativität von } \wedge + \text{ÄS} \\
 \sim \underbrace{((A_0 \wedge \neg A_1) \vee (A_0 \wedge A_1))}_{\text{kanonische DNF in } A_0 \text{ und } A_1} & \text{Kommutativität von } \vee
 \end{array}$$

Alternativ dazu, den Begriff der logischen Äquivalenz (oder der Tautologie, der Erfüllbarkeit, der logischen Folgerung) über die Semantik zu definieren, also über die Auswertung von Formeln, könnte man ihn in Folge von Satz 1.6.5 auch rein syntaktisch definieren durch ein System von Regeln. Solch ein Regelsystem besteht typischerweise aus *Axiomen* [axioms] (z. B.  $\neg\neg A_0 \sim A_0$ ) oder *Axiomschemata* [axiom schemes] (z. B.  $\neg\neg F \sim F$  für jede aussagenlogische Formel  $F$ ) und *Ableitungsregeln* [rules of inference, transformation rules] (z. B. wenn  $F \sim G$ , dann auch  $G \sim F$ ).<sup>14</sup>

Diese Herangehensweise ist sehr verbreitet und für manche Logiksysteme die einzig mögliche, nämlich wenn es kein adäquates Konzept der Auswertung einer Formel gibt. Hat man beides – den semantischen Zugang und ein Regelsystem – muss man deren Übereinstimmung beweisen. Dazu zeigt man, dass das Regelsystem *korrekt* [sound] und *vollständig* [complete] ist: Korrektheit bedeutet, dass alle Regeln tatsächlich logische Äquivalenzen liefern (hier Satz 1.6.1 und die entsprechenden Bemerkungen für die Ableitungsregeln); Vollständigkeit bedeutet, dass alle logischen Äquivalenzen erreicht werden können (hier Satz 1.6.5).

<sup>14</sup>Die beiden Substitutionsprinzipien sind typische Ableitungsregeln: Den Begriff „Prinzip“ benutze ich aus Gewohnheit und Tradition für Gesetze und Regeln, deren Gültigkeit konstitutiv für die betrachtete Logik ist.



## 1.7 Erfüllbarkeit

Typische Fragen zu aussagenlogischen Formeln lassen sich auch als *Erfüllbarkeitsprobleme* [satisfiability problems] formulieren: Zum Beispiel sind  $F$  und  $G$  genau dann äquivalent, wenn  $\neg(F \leftrightarrow G)$  nicht erfüllbar ist. Es gibt zwei Versionen des Erfüllbarkeitsproblems:

- Das *Entscheidungsproblem* [decision problem]: Ist eine gegebene Formel erfüllbar?
- Das *Such- oder Konstruktionsproblem* [function problem]:  
Finde für eine gegebene (erfüllbare) Formel eine erfüllende Belegung.

Beide Probleme sind lösbar durch das Aufstellen von Wahrheitstafeln. Allerdings muss man bei  $n$  Aussagenvariablen im schlimmsten Fall  $2^n$  Belegungen ausrechnen, hat also ein exponentielles Wachstum, was bereits bei etwa 100 Aussagenvariablen praktisch nicht mehr durchführbar ist. Es stellt sich daher die Frage, ob es bessere Algorithmen gibt als Wahrheitstafeln auszurechnen. Nach dem Satz von Cook (siehe Satz 3.3.3) ist das Entscheidungsproblem ein sogenanntes *NP-vollständiges Problem* [NP-complete problem]: Das bedeutet vereinfacht ausgedrückt, dass man zum einen keinen schnellen Algorithmus kennt, und dass zum anderen ein schneller Algorithmus ein großes offenes Problem der theoretischen Informatik (*ist  $P = NP$ ?*) lösen würde, und zwar in der von den meisten Experten nicht erwarteten Art. Das Erfüllbarkeitsproblem der Aussagenlogik (in der Entscheidungsvariante) spielt in der *Komplexitätstheorie*, die sich mit diesen Fragen beschäftigt, eine bedeutende Rolle, und wird dort meist mit dem Kürzel SAT bezeichnet.

Auf der anderen Seite gibt es gute Algorithmen für Formeln in besonderer Gestalt (z. B. Formeln in DNF, Horn-Formeln), und es gibt Algorithmen, die häufig besser als Wahrheitstafeln sind (Baum- oder Tableauverfahren, Resolution). In den nächsten Unterabschnitten werden einige Verfahren vorgestellt. Nur der letzte Abschnitt 1.7.5 behandelt einen anderen Aspekt des Erfüllbarkeitsthemas.

Zunächst sollte noch bemerkt werden, dass man für das Erfüllbarkeitsproblem Wahrheitstafeln besser „horizontal“ statt „vertikal“ ausfüllt, also jede partielle Belegung zunächst komplett durchrechnet. Sobald man eine erfüllende Belegung gefunden hat, kann man aufhören!

### 1.7.1 Formeln in disjunktiver Normalform

Einfach sind die Erfüllbarkeitsprobleme für Formeln  $F = \bigvee_i K_i$  in disjunktiver Normalform: Solch eine Formel ist genau dann erfüllbar, wenn  $F = \top$  oder wenn ein Konjunktionsterm  $K_i = (L_1 \wedge \dots \wedge L_m)$  existiert, der nicht  $\perp$  ist und zu keinem Literal  $L_j = A_k$  auch das Literal  $\neg A_k$  enthält. Eine erfüllende Belegung erhält man dann z. B. durch  $\beta(A_i) = 0$ , falls  $\neg A_i$  eines der Literale in  $K_i$  ist, und  $\beta(A_i) = 1$  sonst.

Eine gegebene Formel zunächst in DNF umzuformen, ist allerdings im Allgemeinen zeitaufwendig, da die DNF exponentiell größere Länge als die Ausgangsformel haben kann (Beispiel Paritätsformel).

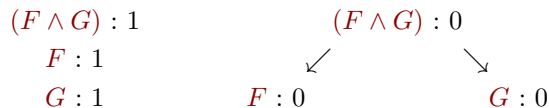
### 1.7.2 Baumkalkül / Tableau-Methode

In Wahrheitstafeln wird der Wahrheitswertverlauf einer Formel „von unten“ ausgerechnet, d. h. ausgehend von den Aussagenvariablen entlang des induktiven Aufbaus der Formel zu immer komplexeren Teilformeln. Alternativ kann man ihn „von oben“ bestimmen, d. h. unter Betrachtung des führenden Junktors untersuchen, unter welchen Bedingungen die Gesamtformel einen

bestimmten Wahrheitswert bekommen kann, und sich so sukzessive zu immer einfacheren Teilformeln durcharbeiten. Es gibt verschiedene Varianten in der Darstellung dieser Methode, die als *Tableau-Methoden*<sup>15</sup> oder *Baumkalküle* [tableau calculi / truth tree calculi] bekannt sind.

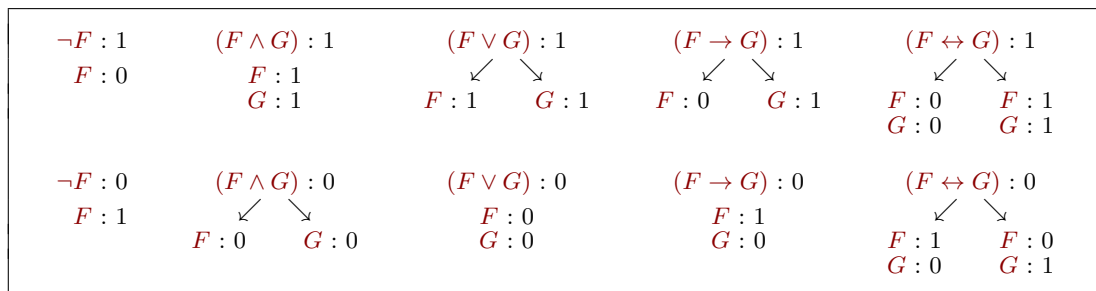
Am Beispiel der Konjunktion sei die Grundidee aufgezeigt:  $(F \wedge G)$  ist genau dann *wahr*, wenn  $F$  und  $G$  *wahr* sind. Der Fall, dass  $(F \wedge G)$  *wahr* ist, also den Wahrheitswert 1 bekommt – kurz  $(F \wedge G) : 1$  geschrieben –, löst sich daher in  $F : 1$  und  $G : 1$  auf. Der Fall, dass  $(F \wedge G)$  *falsch* ist – kurz:  $(F \wedge G) : 0$  – entspricht *drei* Möglichkeiten, die aber durch die folgenden *zwei* nicht-ausschließenden Fälle abgedeckt sind:  $F : 0$  oder  $G : 0$ . Schematisch stellt man dies dar durch:

$F:1$   
 $F:0$



wobei die Reihenfolge auf der rechten Seite keine Rolle spielt, man also rechts und links vertauschen kann.

Man kann sich nun leicht davon überzeugen, dass man jeden Junktore analog auflösen kann in entweder einen oder zwei Fälle:



Indem man alle Junktoren der Formel auf diese Weise sukzessive auflöst, bekommt man einen binären Baum mit der Ausgangsformel als Wurzel und Aussagenvariablen oder Verum/Falsum als Blätter. Die Reihenfolge, in der man die Formeln abarbeitet, ist beliebig. Geschickt ist es, wie im Beispiel zunächst die nicht-verzweigenden Fälle zu behandeln, um einen schmalen Baum zu bekommen.

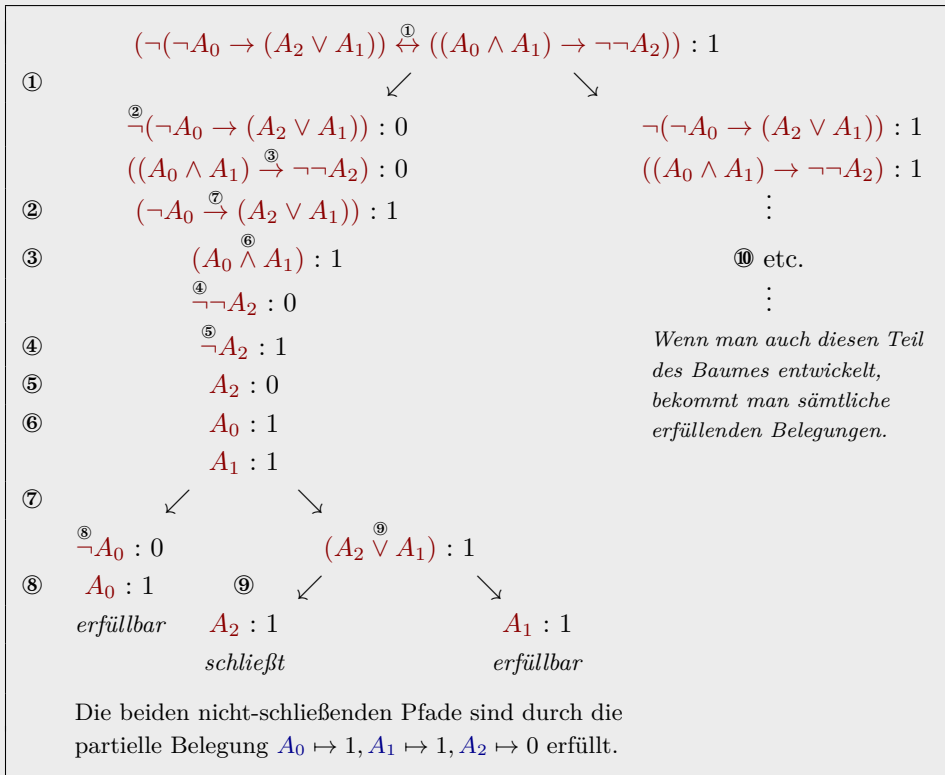
Man betrachtet nun die Pfade von der Wurzel zu den Blättern (die eigentlich *Zweige* [branches] heißen, die man in der Tableau-Methode aber meistens kurz „Pfade“ nennt). Man sagt, dass *ein Pfad schließt* [the branch closes], wenn er widersprüchliche Information enthält. Dafür gibt es drei (sich nicht ausschließende) Möglichkeiten:  $\top : 0$  oder  $\perp : 1$  oder für eine Aussagenvariable  $A_i$  sowohl  $A_i : 1$  als auch  $A_i : 0$ .

Die nicht-schließenden Pfade liefern dann genau die Belegungen, welche die Ausgangsbedingung  $F : 1$  oder  $F : 0$  erfüllen. Falls darin nicht allen Aussagenvariablen ein Wahrheitswert zugewiesen wird, kann man die anderen beliebig belegen. Beim Entscheidungsproblem der Erfüllbarkeit startet man also am besten mit  $F : 1$  und kann abbrechen, sobald man einen nicht-schließenden Pfad gefunden hat.

<sup>15</sup> *tableau* (Plural: *tableaux*) ist das französische Wort für Tabelle

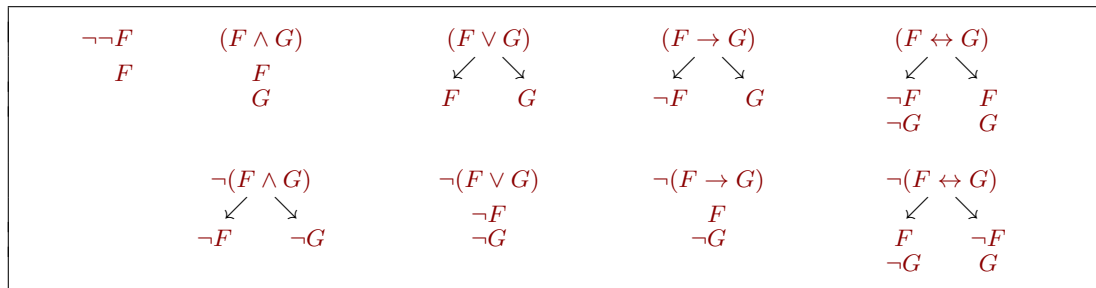
**Beispiel für die Tableau-Methode:**

Ist die Formel  $(\neg(\neg A_0 \rightarrow (A_2 \vee A_1)) \leftrightarrow ((A_0 \wedge A_1) \rightarrow \neg\neg A_2))$  erfüllbar?



Achtung: Auch bei einer Tautologie  $F$  können beim Start mit  $F : 1$  schließende Pfade vorkommen. Ein schließender Pfad bedeutet nicht, dass es Belegungen gibt, die  $F$  falsch machen!

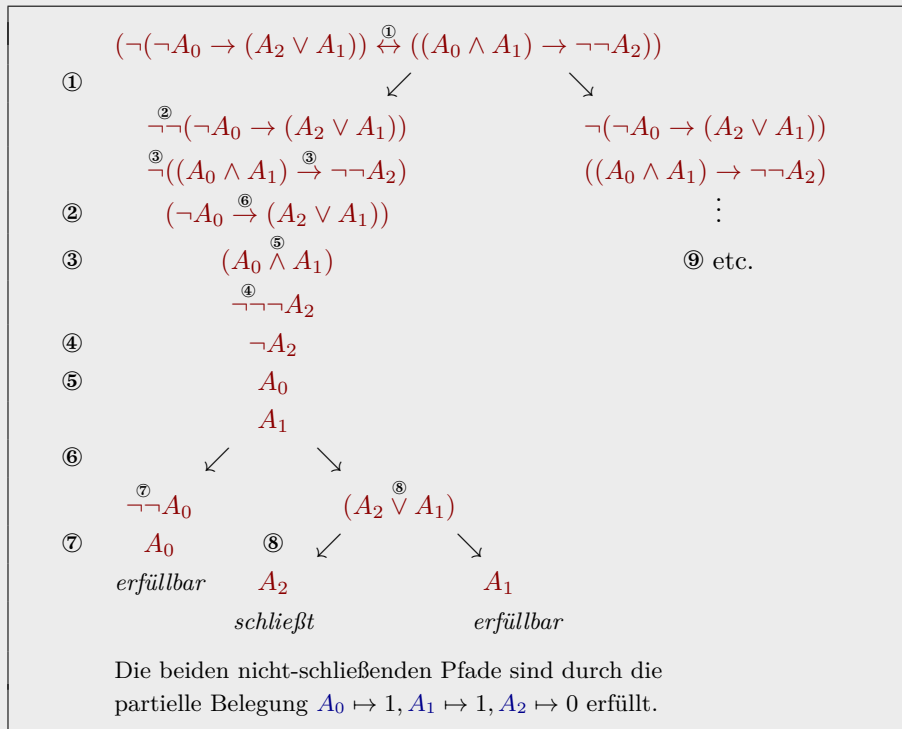
**Varianten:** In der Regel benutzt man viel kompaktere Darstellungen, bei denen man mit nur einem Wahrheitswert arbeitet. In der Variante für den Wahrheitswert 1 schreibt man kurz  $F$  in einen Baum/Tableau für das, was oben mit  $F : 1$  ausgedrückt ist. Man muss dann die Regeln entsprechend anpassen und braucht, weil man eine einfache Negation nicht mehr auflösen kann, spezielle Regeln für negierte Formeln:



In dieser Variante schließt ein Pfad, wenn er  $\perp$  oder  $\neg\top$  oder für eine Aussagenvariable  $A_i$  die beiden Formeln  $A_i$  und  $\neg A_i$  enthält. Aus einem nicht-schließenden Pfad erhält man eine erfüllende partielle Belegung, indem man eine Aussagenvariable  $A_i$  mit dem Wahrheitswert 1 belegt, wenn die Formel  $A_i$  im Pfad vorkommt, und mit dem Wahrheitswert 0, wenn die Formel  $\neg A_i$  im Pfad vorkommt.

**Beispiel für die kompakte Tableau-Methode in der „1-Variante“:**

Ist die Formel  $(\neg(\neg A_0 \rightarrow (A_2 \vee A_1)) \leftrightarrow ((A_0 \wedge A_1) \rightarrow \neg\neg A_2))$  erfüllbar?



Die kompakte Variante spart etwas Schreibarbeit und manchmal den Schritt, einzelne Negationen vor Aussagenvariablen aufzulösen. Dafür sind die Regeln etwas komplizierter und das Verfahren ist nicht selbsterklärend, sondern man muss wissen, was man tut. Insbesondere muss man wissen, welche Variante angewandt wird, denn man kann ebenso eine Kompaktvariante mit dem Wahrheitswert 0 aufstellen (bei der man kurz  $F$  in einen Baum/Tableau schreibt für das, was oben mit  $F : 0$  ausgedrückt ist). Dafür muss man erneut die Regeln anpassen. Je nachdem, für welche Fragestellung man die Tableau-Methode nutzen will, kann das eine oder das andere sinnvoller sein.

**1.7.3 Die Methode von Quine**

Die Methode von Quine [Quine's method] ist eine weitere Möglichkeit, den Wahrheitsverlauf einer Formel zu bestimmen bzw. das Erfüllbarkeitsproblem zu lösen. Dazu legt man zunächst den Wahrheitswert einer (ggf. geeignet gewählten) Aussagenvariable  $A_i$  fest und ersetzt  $A_i$  in der Formel durch  $\perp$  für den Fall  $A_i : 0$  und durch  $\top$  für den Fall  $A_i : 1$ . Die um die fehlenden Junktoren ergänzten  $\top$ -/ $\perp$ -Regeln lassen dann eine schnelle Vereinfachung der Formel zu:

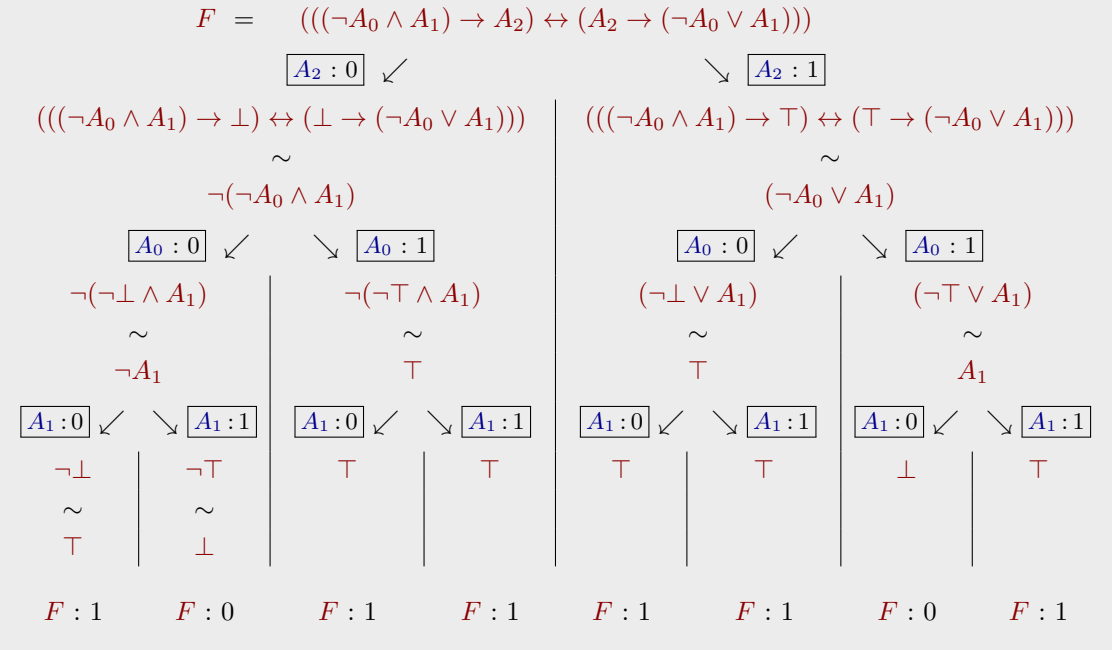
$\neg\top \sim \perp$	$(A \wedge \top) \sim A$	$(A \vee \top) \sim \top$	$(\top \rightarrow A) \sim A$	$(A \leftrightarrow \top) \sim A$
$\neg\perp \sim \top$	$(A \wedge \perp) \sim \perp$	$(A \vee \perp) \sim A$	$(A \rightarrow \perp) \sim \neg A$	$(A \leftrightarrow \perp) \sim \neg A$
			$(\perp \rightarrow A) \sim \top$	

In beiden Fällen fährt man dann mit der nächsten Aussagenvariable fort und erhält so einen

binären Baum, dessen Zweige den partiellen Belegungen entsprechen und an deren Blättern die Formel  $\top$  bzw.  $\perp$  steht, falls die Formel  $F$  unter der entsprechenden partiellen Belegung wahr bzw. falsch wird. Auch bei dieser Methode kann man natürlich, wenn es nur um Erfüllbarkeit geht, aufhören, sobald man eine erfüllende partielle Belegung gefunden hat.

**Beispiel für die Methode von Quine:**

Die Verkürzungen der Formeln nach den Einsetzungen von  $\top$  bzw.  $\perp$  sind hier nicht schrittweise ausgeführt, sondern es ist nur das Endergebnis angegeben.



**1.7.4 Resolution** Im WS 2021/22 voraussichtlich nicht Teil der Vorlesung.

Einer Formel in KNF sieht man die Erfüllbarkeit nicht so leicht an wie einer Formel in DNF, allerdings gibt es für Formeln in KNF eine spezielle Methode, die weiter unten erläuterte *Resolutionmethode*. Auch für die Umwandlung einer Formel in konjunktive Normalform gilt zunächst, dass die KNF exponentiell größere Länge als die Ausgangsformel haben kann (Beispiel Paritätsformel). Allerdings gibt es für die Frage der Erfüllbarkeit einen Trick, durch zusätzliche Aussagenvariablen diese Problematik zu umgehen:

**Lemma 1.7.1** *Zu jeder Formel  $F$  gibt es eine Formel  $F^+$  mit folgenden Eigenschaften:*

- $F$  und  $F^+$  sind erfüllbarkeitsäquivalent, d. h.  $F$  ist genau dann erfüllbar, wenn  $F^+$  es ist;
- $F^+$  ist in KNF mit maximal drei Literalen pro Klausel;
- es gibt eine von  $F$  unabhängige Konstante  $C \in \mathbb{N}$  mit  $\lg(F^+) \leq C \cdot \lg(F)$ .

BEWEIS: „Nicht-triviale Teilformeln“ sollen in diesem Beweis Teilformeln von  $F$  heißen, die nicht nur aus einer Aussagenvariablen bestehen. Für jede nicht-triviale Teilformel  $F'$  wählt man eine jeweils „neue“ Aussagenvariable  $A_{F'}$ , also ein  $A_i$ , das in  $F$  nicht vorkommt und verschieden ist von allen  $A_{F''}$  für  $F' \neq F''$ .

Wenn in Polnischer Notation  $F' = *F_1 \dots F_n$  für einen  $n$ -stelligen Junktor  $*$ , setzt man

$$E_{F'} := \leftrightarrow A_{F'} * A_{F_1} \dots A_{F_n}.$$

In Infix-Notation ist beispielsweise für  $F' = (G \wedge H)$  also  $\mathbf{E}_{F'} = (A_{F'} \leftrightarrow (A_G \wedge A_H))$ ; für  $F' = \neg G$  ist  $\mathbf{E}_{F'} = (A_{F'} \leftrightarrow \neg A_G)$  und für  $F' = \perp$  ist  $\mathbf{E}_{F'} = (A_{F'} \leftrightarrow \perp)$ .

Eine partielle Belegung  $\beta$  der in  $F$  vorkommenden Aussagenvariablen setzt sich nun durch  $\beta'(A_{F'}) := \beta(F')$  auf eindeutige Weise so zu einer Belegung der neuen Aussagenvariablen fort, dass alle Äquivalenzen  $\mathbf{E}_{F'}$  wahr werden. Insbesondere erfüllt  $\beta$  genau dann  $F$ , wenn  $\beta'(A_F) = 1$ . Also ist  $F$  genau dann erfüllbar, wenn die Menge

$$\{A_F\} \cup \{\mathbf{E}_{F'} \mid F' \text{ ist nicht-triviale Teilformel von } F\}$$

erfüllbar ist bzw. die Konjunktion aller Formeln in dieser Menge.

Jede Formel  $\mathbf{E}_{F'}$  lässt sich in KNF schreiben, wobei maximal drei Literale pro Klausel vorkommen können, da jede der Formeln maximal drei Aussagenvariablen enthält. Als die gesuchte Formel  $F^+$  nimmt man nun die Konjunktion von  $A_F$  und aller in KNF gebrachten Formeln  $\mathbf{E}_{F'}$ . Die Anzahl der nicht-trivialen Teilformeln von  $F$  ist genau die Anzahl von Vorkommen von Junktoren in  $F$  und daher durch die Länge  $\lg_p F$  beschränkt. Die Länge der Formeln in drei Aussagenvariablen und KNF lässt sich ebenfalls beschränken. Damit ergibt sich insgesamt, dass  $F^+$  gegenüber  $F$  in der Länge maximal um einen konstanten Faktor zunimmt (genauere Analyse zeigt, dass für Polnische Notation  $C = 20$  funktioniert.)  $\square$

**Beispiel:** Die Formel  $F = (A_0 \vee \neg(\neg A_1 \rightarrow A_0))$  mit Teilformeln  $F' = \neg(\neg A_1 \rightarrow A_0)$  und  $F'' = (\neg A_1 \rightarrow A_0)$  und  $F''' = \neg A_1$  wird ersetzt durch die erfüllbarkeitsäquivalente Formel

$$(A_F \wedge (A_F \leftrightarrow (A_0 \vee A_{F'})) \wedge (A_{F'} \leftrightarrow \neg A_{F''}) \wedge (A_{F''} \leftrightarrow (A_{F'''} \rightarrow A_0)) \wedge (A_{F'''} \leftrightarrow \neg A_1))$$

Beispielhaft sei angegeben, wie die Teilformel  $\mathbf{E}_{F''}$  in KNF aussieht:

$$\begin{aligned} \mathbf{E}_{F''} = (A_{F''} \leftrightarrow (A_{F'''} \rightarrow A_0)) &\sim ((A_{F''} \rightarrow (A_{F'''} \rightarrow A_0)) \wedge ((A_{F'''} \rightarrow A_0) \rightarrow A_{F''})) \\ &\sim ((\neg A_{F''} \vee \neg A_{F'''} \vee A_0) \wedge (\neg(\neg A_{F'''} \vee A_0) \vee A_{F''})) \\ &\sim ((\neg A_{F''} \vee \neg A_{F'''} \vee A_0) \wedge (A_{F''} \vee A_{F'''})) \wedge (A_{F''} \vee \neg A_0) \end{aligned}$$

Da es für die Formeln  $\mathbf{E}_{F'}$  aus dem Beweis nur endlich viele Möglichkeiten gibt, braucht man diese Umformungen natürlich nicht jedesmal vorzunehmen, sondern kann den Übergang von der Form der Formel zur Form der KNF fest einprogrammieren.

Man sieht auch, dass man sich bei der Konstruktion der erfüllbarkeitsäquivalenten Formel in KNF systematisch zwei Arten von Schritte ersparen kann, nämlich die Einführung von eigenen Aussagenvariablen für die gesamte Formel und für negierte Aussagenvariablen. Im Beispiel kann man  $(A_F \wedge (A_F \leftrightarrow (A_0 \vee A_{F'})))$  zu  $(A_0 \vee A_{F'})$  verkürzen und die negierte Aussagenvariable  $\neg A_1$  anstelle von  $A_{F'''}$  in den vorletzten Term der Konjunktion einsetzen. Dadurch erhält man eine ebenso leicht in KNF umformbare, erfüllbarkeitsäquivalente Formel

$$((A_0 \vee A_{F'}) \wedge (A_{F'} \leftrightarrow \neg A_{F''}) \wedge (A_{F''} \leftrightarrow (\neg A_1 \rightarrow A_0))).$$

Der besseren Übersichtlichkeit halber und um sich Schreibarbeit zu sparen, identifiziert man im Folgenden die Disjunktionsterme oder *Klauseln*  $(L_1 \vee \dots \vee L_k)$  in einer Formel in KNF (siehe Definition 1.5.1) mit der Menge  $\{L_1, \dots, L_k\}$  ihrer Literale.<sup>16</sup> Eine solche Menge von Literalen als Klausel aufgefasst wird also durch eine Belegung  $\beta$  erfüllt, wenn  $\beta$  die Disjunktion

<sup>16</sup>Allerdings kann man daraus die Klausel nur bis auf logische Äquivalenz zurückgewinnen, da Reihenfolgen und doppelte Vorkommen von Literalen verloren gehen.

der Literale wahr macht. Da dies allerdings der eigentlichen Definition der Erfüllbarkeit von Formelmengen widerspricht, schreibe ich, um Missverständnisse zu vermeiden, eine als Menge aufgefasste Klauseln mit „eingestülpten Mengenklammern“ als  $\{L_1, \dots, L_k\}$ .

{...}

Für eine Menge von Klauseln  $\{C_1, \dots, C_n\}$  hingegen soll Erfüllbarkeit durch  $\beta$  wie früher bedeuten, dass  $\beta$  jede der Klauseln erfüllt. Damit steht eine *Klauselmenge* hinsichtlich Erfüllbarkeit also für eine Formel in KNF.

**Beispiel:** Die Formel  $((\neg A_2 \vee \neg A_1 \vee A_0) \wedge (A_2 \vee A_1) \wedge (A_2 \vee \neg A_0))$  wird zur Klauselmenge

$$\{\{\neg A_2, \neg A_1, A_0\}, \{A_2, A_1\}, \{A_2, \neg A_0\}\}$$

Die *leere Klausel* [empty clause]  $\{\}$  enthält kein Literal, entspricht also der *leeren Disjunktion* im Sinne der auf Seite 15 eingeführten Notation, d. h. der Formel  $\perp$ , und ist somit nicht erfüllbar. Eine Menge an Klauseln, die die leere Klausel enthält – insbesondere auch die Menge  $\{\{\}\}$ , die nur die leere Klausel enthält – ist also ebenfalls nicht erfüllbar. Die *leere Menge an Klauseln*  $\{\}$  entspricht dagegen der *leeren Konjunktion*, d. h. der Formel  $\top$ , und ist erfüllbar.

**Definition 1.7.2** Sind  $C_1 = \{L_1, \dots, L_k, A_i\}$  und  $C_2 = \{L'_1, \dots, L'_l, \neg A_i\}$  Klauseln, dann heißt die Klausel  $R = \{L_1, \dots, L_k, L'_1, \dots, L'_l\}$  eine *Resolvente* [resolvent] von  $C_1$  und  $C_2$ . Man sagt, dass  $R$  durch *Resolution* [by resolution] aus  $C_1$  und  $C_2$  entsteht.

**Beispiele:**

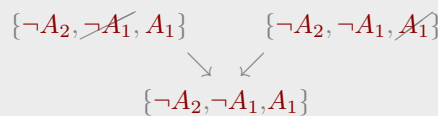
(a) Die Klauseln  $C_1 = \{\neg A_2, A_0, \neg A_1\}$  und  $C_2 = \{A_0, A_1, A_2, A_3\}$  haben zwei Resolventen: zum einen  $\{\neg A_2, A_0, A_2, A_3\}$  und zum andern  $\{\neg A_1, A_0, A_1, A_3\}$ .



Dagegen ist  $\{A_0, A_3\}$  keine Resolvente von  $C_1$  und  $C_2$ , da immer nur eine Aussagenvariable gleichzeitig resolviert werden darf.

(b) Die Klauseln  $C_3 = \{\neg A_2, A_1, A_0\}$  und  $C_4 = \{A_3, \neg A_2\}$  haben keine Resolvente miteinander.

(c) Die Klausel  $C_5 = \{\neg A_2, \neg A_1, A_1\}$  bildet mit sich selbst eine Resolvente, die aber nichts Neues ergibt, weswegen dieser Fall nicht betrachtet werden muss:



Eine endliche Klauselmenge enthält nur endlich viele Aussagenvariablen, aus denen sich nur endlich viele verschiedene Klauseln ergeben können. Daher kann man eine endliche Klauselmenge in endlich vielen Schritten unter Resolution abschließen, bis man die kleinste Ober-Klauselmenge erhält, aus der sich keine neuen Resolventen mehr ergeben.

**Satz 1.7.3 (Resolutionsmethode)** Eine endliche<sup>17</sup> Klauselmenge ist genau dann nicht erfüllbar, wenn sich durch sukzessive Resolution die leere Klausel ergibt.

BEWEIS: „ $\Leftarrow$ “: Eine Belegung  $\beta$  erfüllt mit  $C_1 = \{L_1, \dots, L_k, A_i\}$  und  $C_2 = \{L'_1, \dots, L'_l, \neg A_i\}$  auch jede Resolvente  $R = \{L_1, \dots, L_k, L'_1, \dots, L'_l\}$ . Denn entweder  $\beta(A_i) = 1$  und  $\beta$  muss eines der Literale  $L'_1, \dots, L'_l$  wahr machen, um  $C_2$  zu erfüllen, oder  $\beta(A_i) = 0$  und  $\beta$  muss eines der Literale  $L_1, \dots, L_k$  wahr machen, um  $C_1$  zu erfüllen. In jedem Fall ist  $R$  erfüllt. Daraus folgt, dass eine Klauselmenge, die erfüllbar ist, auch einen Abschluss unter Resolution hat, der erfüllbar ist, und somit nicht die leere Klausel enthalten kann.

„ $\Rightarrow$ “: Angenommen  $\mathcal{C}$  ist eine unter Resolution abgeschlossene Klauselmenge, die nicht die leere Klausel enthält. Ohne Einschränkung seien  $A_0, \dots, A_{n-1}$  die in  $\mathcal{C}$  vorkommenden Aussagenvariablen. Per Induktion über  $n$  wird gezeigt, dass  $\mathcal{C}$  erfüllbar ist.

Für  $n = 0$  gibt es nur die leere Klauselmenge  $\mathcal{C} = \{\}$ , die nicht die leere Klausel enthält.  $\mathcal{C}$  wird durch alle Belegungen erfüllt.

Im Induktionsschritt  $n \rightarrow n + 1$  betrachtet man die Aussagenvariable  $A_n$ . Es können nicht beide Klauseln  $\{A_n\}$  und  $\{\neg A_n\}$  in  $\mathcal{C}$  vorkommen, denn sonst bekäme man die leere Klausel als Resolvente. Angenommen also  $\{\neg A_n\} \notin \mathcal{C}$  (der andere Fall ist analog). Dann modifiziert man  $\mathcal{C}$  folgendermaßen zu einer Klauselmenge  $\mathcal{C}'$  in den Aussagenvariablen  $A_0, \dots, A_{n-1}$ :

- Wenn die Klausel  $C \in \mathcal{C}$  das Literal  $A_n$  enthält, entfällt  $C$ .
- Wenn die Klausel  $C \in \mathcal{C}$  das Literal  $\neg A_n$  enthält, wird dieses aus  $C$  entfernt und  $C' := C \setminus \{\neg A_n\}$  zu einer Klausel in  $\mathcal{C}'$ .
- Alle anderen Klauseln werden unverändert von  $\mathcal{C}$  nach  $\mathcal{C}'$  übernommen.

Man vergewissert sich unschwer, dass  $\mathcal{C}'$  eine unter Resolution abgeschlossene Klauselmenge ist, die zudem nicht die leere Klausel enthält, da  $\{\neg A_n\} \notin \mathcal{C}$ . Per Induktion ist  $\mathcal{C}'$  durch ein  $\beta'$  erfüllbar. Belegungen  $\beta$  mit  $\beta(A_n) = 1$ , die auf  $A_0, \dots, A_{n-1}$  mit  $\beta'$  übereinstimmen, erfüllen dann  $\mathcal{C}$ : Denn zum einen werden durch  $\beta(A_n) = 1$  alle weggelassenen Klauseln erfüllt; zum andern ändert sich nicht durch Hinzunahme von  $\neg A_n$  nicht, dass die modifizierten Klauseln durch  $\beta'$  (und damit jede Fortsetzung von  $\beta'$ ) erfüllt werden.  $\square$

Der Beweis zeigt auch, wie man aus dem Verfahren eine erfüllende Belegung gewinnen kann. Das Resolutionsverfahren wird in der Praxis benutzt und liefert häufig gute Ergebnisse; der Abschluss unter Resolution kann aber auch zu exponentiell vielen Resolventen führen.

**Beispiel 1:** Die Menge folgender Klauseln soll auf Erfüllbarkeit getestet werden:

$$C_1 = \{\neg A_0, A_1\}, C_2 = \{A_1, A_2\}, C_3 = \{A_0, \neg A_2\}, C_4 = \{\neg A_0, \neg A_1, \neg A_2\}, C_5 = \{\neg A_0, A_3\}$$

Im ersten Schritt bekommt man durch Resolution von  $C_1$  und  $C_3$  die neue Klausel  $C_6 = \{A_1, \neg A_2\}$  und durch Resolution von  $C_1$  und  $C_4$  die neue Klausel  $C_7 = \{\neg A_0, \neg A_2\}$ . Die Klauseln  $C_1$  und  $C_2$  bzw.  $C_5$  lassen keine Resolution zu. Aus  $C_2$  und  $C_3$  erhält man  $C_8 = \{A_0, A_1\}$ ; aus  $C_2$  und  $C_4$  die beiden Klauseln  $C_9 = \{\neg A_0, A_2, \neg A_2\}$  und  $C_{10} = \{\neg A_0, A_1, \neg A_1\}$ . Aus  $C_3$  und  $C_4$  bekommt man die Klausel  $C_{11} = \{\neg A_1, \neg A_2\}$  und aus  $C_3$  und  $C_5$  die Klausel  $C_{12} = \{\neg A_2, A_3\}$ ;  $C_2$  bzw.  $C_4$  und  $C_5$  lassen wieder keine Resolution zu.

<sup>17</sup>Aus dem Kompaktheitssatz 1.7.4 folgt, dass der Satz ebenso für unendliche Klauselmengen gilt. Der Abschluss unter Resolution lässt sich dann aber i. Allg. nicht mehr in endlich vielen Schritten erreichen.



Klauseln wie  $C_9$  und  $C_{10}$ , die eine Aussagenvariable und ihr Negat erhalten, sind immer erfüllbar und können nie dazu beitragen, dass die leere Klausel entsteht. Man kann die daher in der Betrachtung weglassen.

Im zweiten Schritt ergeben sich neue Resolutionen: aus  $C_1$  und  $C_8$  oder auch aus  $C_2$  und  $C_6$  erhält man  $C_{13} = \{A_1\}$ .  $C_1$  und  $C_{11}$  lassen ebenfalls eine Resolution zu, diese ergibt aber wieder  $C_7$ . An neuen, relevanten Klauseln ergibt sich noch  $\{\neg A_0, A_1\}$ ,  $\{A_1, A_3\}$  und  $\{\neg A_2\}$ .

Im dritten Schritt findet man als neue relevante Klausel  $\{\neg A_0, \neg A_2, \neg A_3\}$  und im vierten Schritt  $\{A_1, \neg A_2, \neg A_3\}$ . Danach erhält man durch Resolution keine neuen Klauseln mehr, die nicht eine Aussagenvariable und ihr Negat erhalten. Im Wesentlichen hat man damit also den Abschluss der Klauselmenge unter Resolution gefunden (bis auf für Erfüllbarkeit irrelevante Klauseln). Da die leere Klausel nicht dabei ist, ist die Ausgangsklauselmenge erfüllbar.

Man kann auch alle erfüllenden Belegungen gewinnen: Aus der Einerklausel  $\{A_1\}$  sieht man, dass  $A_1$  wahr sein muss. Löscht man alle Klauseln mit einem positiven Vorkommen von  $A_1$  und löscht alle Vorkommen von  $\neg A_1$  aus den verbleibenden Klauseln, erhält man:

$$\{A_0, \neg A_2\}, \{\neg A_0, \neg A_2\}, \{\neg A_0, A_3\}, \{\neg A_2\}, \{\neg A_0, \neg A_2, \neg A_3\}$$

Man sieht erneut an der Einerklausel  $\{\neg A_2\}$ , dass  $A_2$  falsch ein muss. Löscht man alle Klauseln mit einem Vorkommen von  $\neg A_2$  und löscht alle Vorkommen von  $A_2$  aus den verbleibenden Klauseln, erhält man:

$$\{\neg A_0, A_3\}$$

$A_0$  kann nun sowohl wahr als auch falsch sein. Setzt man  $A_0$  wahr, bleibt nach dem Reduktionsverfahren die Klausel  $\{A_3\}$  und  $A_3$  muss ebenfalls wahr sein. Setzt man  $A_0$  falsch, bleibt nach dem Reduktionsverfahren keine Klausel übrig und  $A_3$  kann einen beliebigen Wahrheitswert annehmen.

**Beispiel 2:** Aus der Menge der Klauseln

$$\{\neg A_0, A_1\}, \{\neg A_0, A_2\}, \{A_0, A_3\}, \{A_0, A_1\}, \{\neg A_1, \neg A_2\}, \{\neg A_1, \neg A_3\}$$

bekommt man durch Resolution aus der 1. und 4. Klausel  $\{A_1\}$ , aus der 3. und 6.  $\{A_0, \neg A_1\}$  und aus der 2. und 5.  $\{\neg A_0, \neg A_1\}$ . Diese beiden letzten lassen sich zu  $\{\neg A_1\}$  resolvieren; mit  $\{A_1\}$  ergibt sich dann die leere Klausel. Also ist die Klauselmenge nicht erfüllbar.

Es gibt viele Varianten und Verfeinerungen der Resolutionsmethode. Man kann zum Beispiel zunächst unter Resolution mit allen einelementigen Klauseln abschließen (was von der Komplexität her überschaubar bleibt) und dann die Größe steigern.

Einen Spezialfall stellen *Horn-Formeln* [Horn formulae] da: Eine Horn-Formel ist eine Formel in KNF in der jede Klausel eine *Horn-Klausel* [Horn clause] ist, d. h. maximal ein positives Literal enthält. Für Horn-Formeln gibt es lineare Algorithmen zum Testen von Erfüllbarkeit und zur Konstruktion von erfüllenden Belegungen, z. B. Varianten der Resolutionsmethode oder den *Markierungsalgorithmus*.

### Vollständiges Beispiel für Umformung in KNF und Resolution

Mit der in diesem Abschnitt präsentierten Methode soll die Gültigkeit des *modus tollens*

$$(A_0 \rightarrow A_1), \neg A_1 \vdash \neg A_0$$

überprüft werden, d. h. es soll die Unerfüllbarkeit gezeigt werden von der Formel

$$\neg(((A_0 \rightarrow A_1) \wedge \neg A_1) \rightarrow \neg A_0).$$

Für die „schnelle Umformung in KNF“ werden zunächst zusätzliche Aussagenvariablen für Teilformeln eingeführt. Dies ergibt im ersten Schritt:

$$(\neg A_4 \wedge (A_4 \leftrightarrow (A_3 \rightarrow \neg A_0))) \wedge (A_3 \leftrightarrow (A_2 \wedge \neg A_1)) \wedge (A_2 \leftrightarrow (A_0 \rightarrow A_1))$$

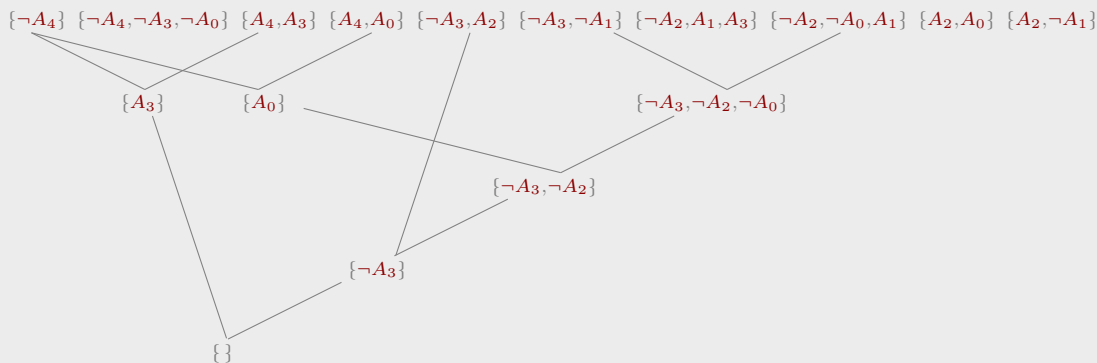
und in KNF umgeformt schließlich die Formel

$$\begin{aligned} &(\neg A_4 \wedge (\neg A_4 \vee \neg A_3 \vee \neg A_0) \wedge (A_4 \vee A_3) \wedge (A_4 \vee A_0) \\ &\wedge (\neg A_3 \vee A_2) \wedge (\neg A_3 \vee \neg A_1) \wedge (\neg A_2 \vee A_1 \vee A_3) \\ &\wedge (\neg A_2 \vee \neg A_0 \vee A_1) \wedge (A_2 \vee A_0) \wedge (A_2 \vee \neg A_1)), \end{aligned}$$

bzw. die Klauselmenge

$$\begin{aligned} &\{ \{\neg A_4\}, \{\neg A_4, \neg A_3, \neg A_0\}, \{A_4, A_3\}, \{A_4, A_0\}, \{\neg A_3, A_2\}, \\ &\{\neg A_3, \neg A_1\}, \{\neg A_2, A_1, A_3\}, \{\neg A_2, \neg A_0, A_1\}, \{A_2, A_0\}, \{A_2, \neg A_1\} \}. \end{aligned}$$

Mit systematischer Resolution bekommt man daraus die leere Klausel. Hier ist nur ein Auszug zielführender Resolutionen gezeigt:



#### 1.7.5 Der Kompaktheitssatz

Für eine endliche Formelmenge  $\{F_1, \dots, F_n\}$  gilt, dass sie genau dann erfüllbar ist, wenn die Formel  $(F_1 \wedge \dots \wedge F_n)$  erfüllbar ist. Unendliche Formelmengen lassen sich dagegen nicht in einer einzelnen Formel zusammenfassen. Dennoch kann man die Erfüllbarkeit unendlicher Formelmengen mit dem folgenden *Kompaktheitssatz* [compactness theorem] letztendlich auf die Erfüllbarkeit einzelner Formeln zurückführen:

**Satz 1.7.4 (Kompaktheitssatz)**

Sei  $T$  eine unendliche Formelmeng. <sup>18</sup>  $T$  ist genau dann erfüllbar, wenn  $T$  endlich erfüllbar [finitely satisfiable] ist, d. h. wenn jede endliche Teilmenge von  $T$  erfüllbar ist.

BEWEIS: Es ist offensichtlich, dass aus Erfüllbarkeit endliche Erfüllbarkeit folgt. Sei also  $T$  endlich erfüllbar. Per Induktion konstruiert man endlich erfüllbare Formelmengen  $T_0 \subseteq \dots \subseteq T_n \subseteq T_{n+1} \subseteq \dots$ , die sukzessive Literale zu allen Aussagenvariablen enthalten:

$$T_0 := T \quad \text{und} \quad T_{n+1} := \begin{cases} T_n \cup \{A_n\} & \text{falls endlich erfüllbar,} \\ T_n \cup \{\neg A_n\} & \text{sonst.} \end{cases}$$

Behauptung:  $T_{n+1}$  ist stets endlich erfüllbar.

Denn angenommen sowohl  $T_n \cup \{A_n\}$  als auch  $T_n \cup \{\neg A_n\}$  sind nicht endlich erfüllbar. Dann gibt es endliche Teilmengen  $S^+$  und  $S^-$  von  $T_n$ , so dass weder  $S^+ \cup \{A_n\}$  noch  $S^- \cup \{\neg A_n\}$  erfüllbar ist. Wegen der endlichen Erfüllbarkeit von  $T_n$  gibt es aber eine Belegung  $\beta$ , welche  $S^+ \cup S^-$  erfüllt. Wenn nun  $\beta(A_n) = 1$ , erfüllt  $\beta$  die Menge  $S^+ \cup S^- \cup \{A_n\}$  und damit auch  $S^+ \cup \{A_n\}$ , wenn  $\beta(A_n) = 0$ , erfüllt  $\beta$  entsprechend die Menge  $S^- \cup \{\neg A_n\}$ : Widerspruch.

Es folgt dann, dass auch  $T_\infty := \bigcup_{n \in \mathbb{N}} T_n$  endlich erfüllbar ist, da jede endliche Teilmenge von  $T_\infty$  bereits in einem  $T_n$  liegt.

Nun definiert man eine Belegung  $\beta$  und Literale  $L_i \in T_\infty$  durch

$$\beta(A_i) := \begin{cases} 1, & \text{falls } L_i := A_i \in T_\infty \\ 0, & \text{falls } L_i := \neg A_i \in T_\infty \end{cases}$$

Behauptung:  $\beta$  erfüllt  $T_\infty$ .

Denn sei  $F = F(A_0, \dots, A_m) \in T_\infty$ . Dann gibt es eine Belegung  $\beta'$ , die  $\{F, L_0, \dots, L_m\} \subseteq T_\infty$  erfüllt. Für  $i = 0, \dots, m$  gilt dann  $\beta'(A_i) = 1 \iff A_i = L_i$ , d. h.  $\beta'$  stimmt auf den in  $F$  vorkommenden Aussagenvariablen mit  $\beta$  überein. Da  $\beta'$   $F$  erfüllt, muss daher auch  $\beta$   $F$  erfüllen.  $\square$

**1.8 Boole'sche Algebren**

Wenn man etwa mit ganzen Zahlen rechnet, gelten gewisse Rechenregeln wie die Kommutativität der Addition  $n+m = m+n$ . Gerne würde man logische Gesetze wie z. B. die Kommutativität der Konjunktion  $(F \wedge G) \sim (G \wedge F)$  auch als eine solche Rechenregel auffassen und in diesem Sinne mit Formeln einfach „rechnen“. Nun ist  $(F \wedge G)$  als Formel – also als orientierter Baum oder auch eine diesen Baum darstellende Symbolfolge – nicht das gleiche wie die Formel  $(G \wedge F)$ , sondern eben nur logisch äquivalent dazu. <sup>19</sup> Um dieses Problem zu umgehen arbeitet man mit Äquivalenzklassen logischer Formeln. Die Äquivalenzklasse einer Formel  $F$  bezeichne ich dabei mit  $F/\sim$  und die Menge der Äquivalenzklassen aussagenlogischer Formeln mit  $\mathcal{F}_\infty$ .

$F/\sim$   
 $\mathcal{F}_\infty$

<sup>18</sup>In der Definition der aussagenlogischen Sprache wurden nur abzählbar unendlich viele Aussagenvariablen zugelassen; daher können Formelmengen bestenfalls abzählbar unendlich werden. Man kann die Aussagenlogik problemlos mit einer größeren Anzahl an Aussagenvariablen gestalten. Der Kompaktheitssatz gilt dann auch für überabzählbare Mengen – im Wesentlichen mit dem gleichen Beweis: Man braucht dazu allerdings ein wenig Handwerkszeug aus der Mengenlehre zum transfiniten Aufzählen überabzählbarer Mengen.

<sup>19</sup>Ein ähnliches Phänomen hat man bei arithmetischen Ausdrücken bzw. Gleichungen dazwischen. „ $2 + 3 = 3 + 2$ “ bedeutet nicht, dass die Ausdrücke „ $2 + 3$ “ und „ $3 + 2$ “ gleich wären (sie sind verschieden!), sondern nur, dass die Auswertungen der Ausdrücke, also die Ergebnisse der Berechnungen, gleich sind.

Wenn  $F \sim F'$  und  $G \sim G'$ , folgt aus dem Prinzip der äquivalenten Substitution, dass  $(F \wedge G) \sim (F' \wedge G')$ . Also kann man auf  $\mathcal{F}_\infty$  durch

$$F/\sim \wedge G/\sim := (F \wedge G)/\sim$$

eine Operation definieren, die der Einfachheit halber ebenfalls  $\wedge$  geschrieben wird. Entsprechendes gilt für  $\vee$  und  $\neg$ <sup>20</sup>. Für diese drei Operationen und für die Äquivalenzklassen von  $\top$  und  $\perp$  als besonderen Elementen gelten nun Rechenregeln, die die folgende Definition motivieren.

**Definition 1.8.1** Eine Boole'sche Algebra [Boolean algebra]  $\mathcal{B} = (B; \sqcap, \sqcup, ^c, 1, 0)$  ist eine Struktur mit zwei zweistelligen Verknüpfungen  $\sqcap$  und  $\sqcup$ , einer einstelligen Verknüpfung  $^c$  und zwei Konstanten 1 und 0, für die gelten:

Idemp., Komm., Ass.:	$\sqcap$ und $\sqcup$ sind idempotent, kommutativ und assoziativ	
Distributivgesetze:	$a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c)$	$a \sqcap (b \sqcup c) = (a \sqcap b) \sqcup (a \sqcap c)$
Absorptionsgesetze:	$a \sqcup (a \sqcap b) = a$	$a \sqcap (a \sqcup b) = a$
de Morgan'sche Regeln:	$(a \sqcap b)^c = a^c \sqcup b^c$	$(a \sqcup b)^c = a^c \sqcap b^c$
Doppelnegation:	$a^{cc} = a$	
Extremalgesetze:	$a \sqcap 1 = a$	$a \sqcup 0 = a$
	$a \sqcap 0 = 0$	$a \sqcup 1 = 1$
Komplementgesetze:	$a \sqcap a^c = 0$	$a \sqcup a^c = 1$
	<i>alles für alle <math>a, b, c \in B</math></i>	

$\sqcap$   
 $\sqcup$   
 $^c$   
1  
0

Aus den Komplement- und Extremalgesetzen zusammen mit der Kommutativität von  $\sqcap$  folgt  $1^c = 1^c \sqcap 1 = 1 \sqcap 1^c = 0$ ; analog  $0^c = 1$ .

Die Definition gibt eine traditionelle Liste von Axiomen wieder, die aber nicht minimal ist. Zum Beispiel kann man die de Morgan'schen Regeln und die Absorptionsgesetze aus den restlichen Axiomen herleiten.

**Beispiel:** Herleitung des ersten Absorptionsgesetzes aus den anderen Axiomen:

$a \sqcup (a \sqcap b) = (a \sqcap 1) \sqcup (a \sqcap b)$	Extremalgesetz
$= (a \sqcap (b \sqcup b^c)) \sqcup (a \sqcap b)$	Komplementgesetz
$= ((a \sqcap b) \sqcup (a \sqcap b^c)) \sqcup (a \sqcap b)$	Distributivgesetz
$= ((a \sqcap b) \sqcup (a \sqcap b)) \sqcup (a \sqcap b^c)$	Kommutativ- und Assoziativgesetze
$= (a \sqcap b) \sqcup (a \sqcap b^c)$	Idempotenzgesetz
$= a \sqcap (b \sqcup b^c)$	Distributivgesetz
$= a \sqcap 1$	Extremalgesetz
$= a$	Extremalgesetz

Einige der Axiome Boole'scher Algebren erinnern an die Ringaxiome. Tatsächlich kann man eine Boole'sche Algebra durch  $a \cdot b := a \sqcap b$  und  $a + b := (a \sqcap b^c) \sqcup (a^c \sqcap b)$  zu einem Ring mit 0 als neutralem Element der Addition und 1 als neutralem Element der Multiplikation machen. Ringe, die auf diese Weise von Boole'schen Algebren kommen, heißen *Boole'sche Ringe*. Sie sind dadurch charakterisiert, dass jedes Element *idempotent* ist, also dass stets  $b \cdot b = b$  gilt. Aus einem Boole'schen Ring bekommt man die Boole'sche Algebra zurück durch  $a \sqcap b := a \cdot b$ ,  $a \sqcup b := a + b + (a \cdot b)$  und  $a^c := a + 1$ .

<sup>20</sup>Und auch für  $\rightarrow$  und  $\leftrightarrow$ , die aber für das, was folgt, weniger interessant sind.

Kurz zur Einordnung von Definition 1.8.1 in eine umfassendere Terminologie: Eine Struktur mit zwei idempotenten, kommutativen und assoziativen Verknüpfungen  $\sqcap$  und  $\sqcup$ , für die die Absorptionsgesetze gelten, heißt **Verband [lattice]**. Verbände kann man durch sogenannte *Hasse-Diagramme [Hasse diagrams]* darstellen, wie z. B. auf Seite 38. Gelten die Distributivgesetze, heißt der Verband *distributiver Verband [distributive lattice]*. Gibt es Elemente 0 und 1, die die Extremalgesetze erfüllen, spricht man von einem *beschränkten Verband [bounded lattice]*; gibt es zusätzlich zu jedem Element  $a$  ein *Komplement [complement]  $a^c$* , so dass die Komplementgesetze erfüllt sind, nennt man den Verband *komplementären Verband [complemented lattice]*. Daher kann man Boole'sche Algebren auch als distributive komplementäre Verbände einführen.

**Beispiele Boole'scher Algebren:**

(a) Die Menge  $\mathcal{F}_\infty$  der Äquivalenzklassen aussagenlogischer Formeln zusammen mit den von den Junktoren  $\wedge, \vee$  und  $\neg$  kommenden Operationen sowie den Äquivalenzklassen von  $\top$  und  $\perp$

$$(\mathcal{F}_\infty; \wedge, \vee, \neg, \top/\sim, \perp/\sim)$$

ist nach den Sätzen 1.6.1 und 1.6.2 eine Boole'sche Algebra, die **Tarski-Lindenbaum-Algebra** oder **Lindenbaum-Algebra [Lindenbaum algebra]**. Genauer handelt es sich um die „Lindenbaum-Algebra der klassischen zweiwertigen Aussagenlogik“. <sup>21</sup>

(b) Andere Beispiele sind die **Potenzmengenalgebren [power set algebras]**. Zu einer Menge  $M$  ist  $\mathfrak{P}(M)$  die Potenzmenge von  $M$ , also die Menge aller Teilmengen von  $M$ . Für  $X \subseteq M$  sei nun  $X^c := M \setminus X$  das mengentheoretische Komplement in  $M$ . Dann ist

$$(\mathfrak{P}(M); \cap, \cup, ^c, M, \emptyset)$$

eine Boole'sche Algebra. Für endliches  $M$  ist auch  $\mathfrak{P}(M)$  endlich mit  $|\mathfrak{P}(M)| = 2^{|M|}$ .

(c) Die natürlichen Zahlen  $\mathbb{N}$  mit kgV und ggT bilden einen distributiven beschränkten Verband, der nicht komplementär ist, also keine Boole'sche Algebra.

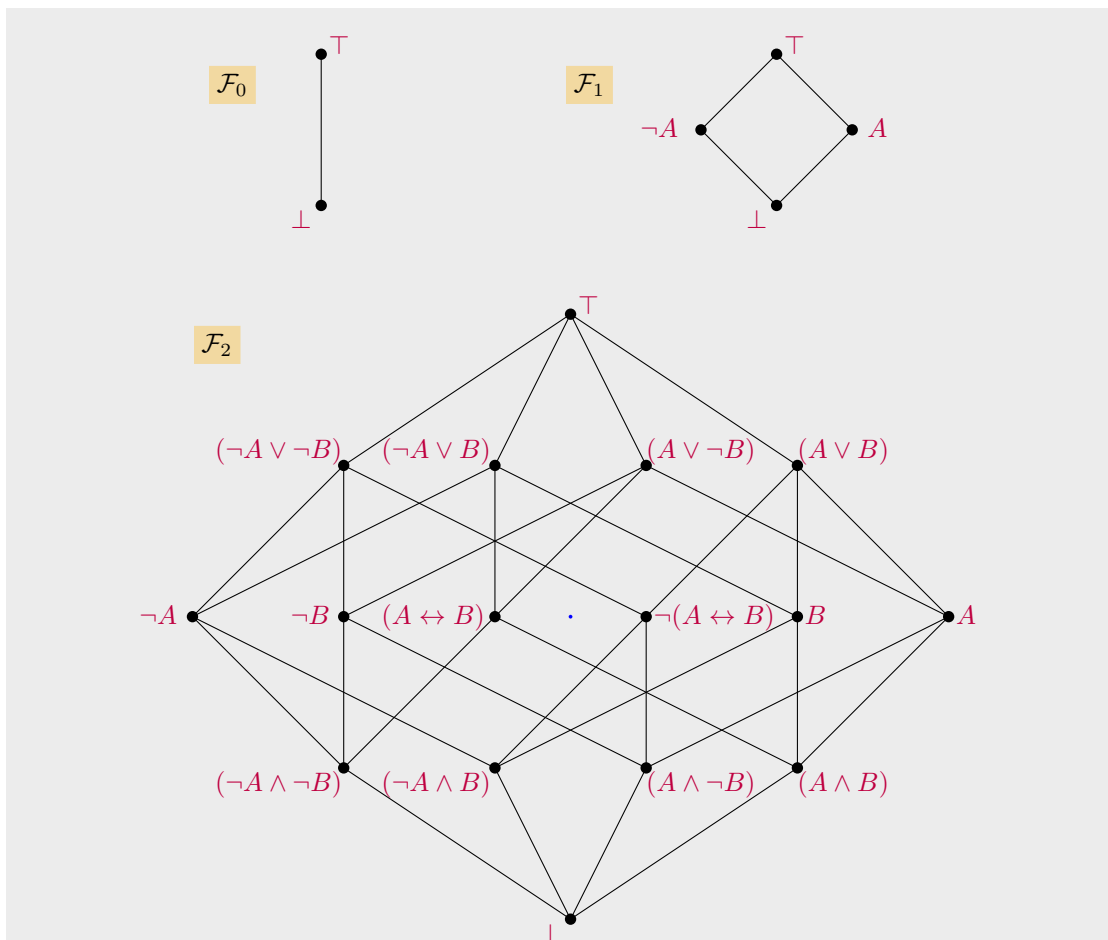
Eine **Unteralgebra [subalgebra]** einer Boole'schen Algebra ist eine Teilmenge  $U$ , die 0 und 1 enthält und unter den Operationen  $\sqcap, \sqcup$  und  $^c$  abgeschlossen ist.  $U$  ist dann mit den eingeschränkten Operationen selbst eine Boole'sche Algebra.

(d) Schränkt man im Beispiel (a) die Grundmenge ein auf die Äquivalenzklassen von Formeln, in denen nur die Aussagenvariablen  $A_0, \dots, A_{n-1}$  vorkommen, erhält man als Unteralgebra von  $\mathcal{F}_\infty$  die Tarski-Lindenbaum-Algebra  $\mathcal{F}_n$ .

Es gilt  $|\mathcal{F}_n| = 2^{2^n}$ , denn für die  $n$  Aussagenvariablen gibt es  $2^n$  partielle Belegungen, also  $2^{2^n}$  mögliche Wahrheitswertverläufe, von denen jeder genau einer Äquivalenzklasse entspricht.

Zur Illustration seien die Hasse-Diagramme der Tarski-Lindenbaum-Algebren  $\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2$  mit  $2^{2^0} = 2, 2^{2^1} = 4$  bzw.  $2^{2^2} = 16$  Elementen angegeben (mit ausgewählten Repräsentanten und der besseren Lesbarkeit willen mit  $A$  und  $B$  anstelle von  $A_0$  und  $A_1$ ).

<sup>21</sup>Auch für andere Aussagenlogiken kann man die Äquivalenzklassen von Formeln als Struktur auffassen; dies ist dann nicht mehr unbedingt eine Boole'sche Algebra. Zum Beispiel ergibt sich für die intuitionistische Aussagenlogik eine sogenannte *Heyting-Algebra*



(e) Es gibt noch viele andere Boole’sche Algebren. Für eine unendliche Menge  $M$  gibt es zum Beispiel die Unteralgebra von  $\mathfrak{P}(M)$ , die aus den *endlichen* und den *ko-endlichen* Teilmengen von  $M$  besteht. Letzteres sind die Komplemente endlicher Teilmengen in  $M$ . Diese Unteralgebra ist offensichtlich keine Potenzmengenalgebra.

Ein *Homomorphismus* [homomorphism] zwischen Boole’schen Algebren  $\mathcal{B}_1$  und  $\mathcal{B}_2$  ist eine Abbildung  $h : \mathcal{B}_1 \rightarrow \mathcal{B}_2$ , die mit allen Konstanten und Operationen verträglich ist, für die also gilt:<sup>22</sup>

$h(a \sqcap^{\mathcal{B}_1} b)$	$=$	$h(a) \sqcap^{\mathcal{B}_2} h(b)$
$h(a \sqcup^{\mathcal{B}_1} b)$	$=$	$h(a) \sqcup^{\mathcal{B}_2} h(b)$
$h(a^{c^{\mathcal{B}_1}})$	$=$	$h(a)^{c^{\mathcal{B}_2}}$
$h(0^{\mathcal{B}_1})$	$=$	$0^{\mathcal{B}_2}$
$h(1^{\mathcal{B}_1})$	$=$	$1^{\mathcal{B}_2}$

Ein *Isomorphismus* [isomorphism] zwischen Boole’schen Algebren ist ein bijektiver Homomorphismus (dessen Umkehrabbildung ebenfalls ein Homomorphismus ist, aber dies ist automatisch der Fall). Wenn es einen Isomorphismus zwischen zwei Boole’schen Algebren gibt, heißen sie zueinander *isomorph* [isomorphic].

<sup>22</sup>Der hochgestellte Index an den Operationen zeigt jeweils an, in welcher Struktur sie berechnet wird.

Die Menge der Wahrheitswerte  $\{0, 1\}$  mit den  $\wedge, \vee, \neg, \top$  und  $\perp$  entsprechenden Wahrheitswertfunktionen ist eine Boole'sche Algebra, die isomorph zu  $\mathcal{F}_0 = \{\perp/\sim, \top/\sim\}$  und zu  $\mathfrak{P}(\emptyset) = \{\emptyset, \{\emptyset\}\}$  ist.

Zueinander logisch äquivalente Formeln haben unter Belegungen den gleichen Wahrheitswert, d. h. man kann die Auswertung von Formeln unter einer Belegung  $\beta$  auf den Äquivalenzklassen definieren und erhält eine Abbildung  $\beta : \mathcal{F}_\infty \rightarrow \{0, 1\}$ ,  $F/\sim \mapsto \beta(F)$ . Die Auswertung von Formeln ist nun gerade so definiert, dass diese Abbildung ein Homomorphismus Boole'scher Algebren ist; z. B. hat man mit den Notationen der Wahrheitswertfunktionen von Seite 11

$$\beta((F \wedge G)/\sim) = \tilde{\wedge}(\beta(F/\sim), \beta(G/\sim)) = \tilde{\wedge}(\beta(F), \beta(G)).$$

Umgekehrt kommt jeder Homomorphismus  $h : \mathcal{F}_\infty \rightarrow \{0, 1\}$  auf diese Weise von einer Belegung, weil  $h$  insbesondere jeder Aussagenvariablen einen Wahrheitswert zuweist. Man kann also Homomorphismen  $\mathcal{F}_\infty \rightarrow \{0, 1\}$  mit Belegungen gleichsetzen.

Da eine Formel  $F$  bis auf logische Äquivalenz durch ihren Wahrheitwertverlauf bestimmt ist, kann man die Äquivalenzklasse  $F/\sim$  mit der Menge der Belegungen, die  $F$  wahr machen, identifizieren. Dies funktioniert sehr viel allgemeiner:

**Satz 1.8.2 (Darstellungssatz von Stone [Stone's representation theorem], 1936)**

*Jede Boole'sche Algebra ist isomorph zu einer Unteralgebra einer Potenzmengenalgebra.*

BEWEISSKIZZE: Sei  $\mathcal{B}$  eine beliebige Boole'sche Algebra. Man betrachtet

$$M := \{h : B \rightarrow \{0, 1\} \mid h \text{ ist Homomorphismus Boole'scher Algebren}\}.$$

Ein Element von  $B$  wird nun mit der Menge der Homomorphismen identifiziert, die das Element auf 1 abbilden:

$$S : B \rightarrow \mathfrak{P}(M), b \mapsto \{h : B \rightarrow \{0, 1\} \mid h \text{ Homomorphismus}, h(b) = 1\}$$

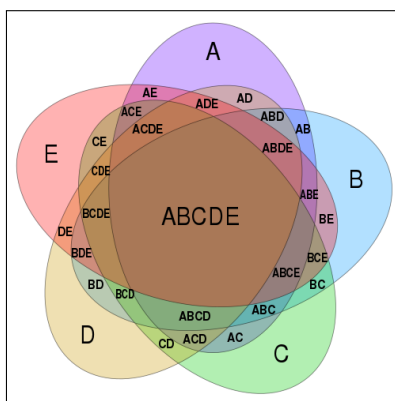
Die so definierte Abbildung  $S$  ist ein injektiver Homomorphismus Boole'scher Algebren und identifiziert damit  $\mathcal{B}$  mit einer Unteralgebra von  $\mathfrak{P}(M)$ , nämlich dem Bild von  $S$ :

Die Homomorphie-Eigenschaften sind einfach nachzuweisen. Man sieht sofort  $S(0) = \emptyset$  und  $S(1) = M$ . Wegen  $b \sqcup b^c = 1$  muss ein Homomorphismus eines von  $b, b^c$  auf 1 abbilden, und wegen  $b \cap b^c = 0$  nicht beide. Also folgt  $S(b^c) = M \setminus S(b)$ . Die Eigenschaft  $S(b \cap b') = S(b) \cap S(b')$  folgt daraus, dass  $h(b \cap b') = \tilde{\wedge}(h(b), h(b')) = \min\{h(b), h(b')\}$  genau dann 1 ist, wenn  $h(b) = 1$  und  $h(b') = 1$ . Für  $\sqcup$  argumentiert man analog. Die Injektivität erfordert für den allgemeinen Fall etwas Mathematik; für die Tarski-Lindenbaum-Algebren folgt sie dagegen sofort aus der Definition der logischen Äquivalenz, da sich zwei nicht äquivalente Formeln in mindestens einer Belegung unterscheiden müssen.  $\square$

Die endlichen Tarski-Lindenbaum-Algebren  $\mathcal{F}_n$  sind sogar isomorph zu Potenzmengenalgebren: Die Homomorphismen  $h$  im Beweis entsprechen in diesem Fall Belegungen der vorkommenden Aussagenvariablen  $A_0, \dots, A_{n-1}$ ; eine Äquivalenzklasse  $F/\sim$  wird unter  $S$  abgebildet auf die Menge dieser partiellen Belegungen, die  $F$  wahr machen. In Satz 1.5.2 über die disjunktive Normalform wurde gezeigt, dass diese Abbildung surjektiv ist, da jede Teilmenge (d. h. jeder

Wahrheitswertverlauf) vorkommt. Der injektive Homomorphismus  $S$  aus dem Beweis ist also bijektiv.

Aus dem Stone’schen Satz ergibt sich somit die Möglichkeit, die Gültigkeit logischer Gesetze durch Mengen-Diagramme zu veranschaulichen und auch nachzuweisen. Für Formeln mit  $n$  Aussagenvariablen braucht man dafür eine Menge mit  $2^n$  Elementen, die man geeignet mit den Belegungen der vorkommenden Aussagenvariablen identifiziert. Eine spezielle Art hiervon sind die *Karnaugh-Veitch-Diagramme* [Karnaugh maps], auf die hier aber nicht eingegangen werden soll. Häufig nimmt man für eine geometrische Darstellung nicht den Homomorphismus  $S : \mathcal{B} \rightarrow \mathfrak{P}(M)$  aus dem Beweis, sondern einen injektiven Homomorphismus  $\mathcal{B} \rightarrow \mathfrak{P}(\mathbb{R}^2)$ , bei dem die Bilder der Aussagenvariablen idealerweise beschränkte, einfach zusammenhängende Teile der Ebene sind, z. B. Kreise oder Ellipsen.<sup>23</sup> Solche Diagramme heißen *Venn-Diagramme* [Venn diagrams] und eignen sich gut für bis zu drei oder vier Aussagenvariablen. Auch für eine größere Anzahl von Aussagenvariablen gibt es schöne, symmetrische Lösungen; es wird aber zunehmend schwieriger wird, den Überblick zu behalten:



Quelle: [https://commons.wikimedia.org/wiki/File:Symmetrical\\_5-set\\_Venn\\_diagram.svg](https://commons.wikimedia.org/wiki/File:Symmetrical_5-set_Venn_diagram.svg)

Die fünf Ellipsen in diesem Venn-Diagramm entsprechen den fünf Aussagenvariablen  $A \dots E$ .

Eine Buchstabenfolge markiert das Gebiet, das der Konjunktion der darin vorkommenden Aussagenvariablen mit den Negationen der übrigen entspricht.

Die Konjunktion  $\wedge$  und Disjunktion  $\vee$  zweier Formeln entsprechen im Mengendiagramm dem Schnitt [intersection]  $\cap$  bzw. der Vereinigung [union]  $\cup$  der zugehörigen Teilmengen (beachten Sie, dass auch die gewählten Zeichen zueinander passen!). Die Negation entspricht dem mengentheoretischen Komplement [complement], das Falsum der leeren Menge [empty set] und das Verum der ganzen Menge [whole set]. Die dem Implikationsjunktors entsprechende mengentheoretische Operation hat keinen üblichen Namen; der Äquivalenzjunktors  $\leftrightarrow$  entspricht dem Komplement der symmetrischen Differenz [symmetric difference].

## Verbände als partielle Ordnungen

Einen anderen Zugang zur Tarski-Lindenbaum-Algebra und allgemein zu Boole’schen Algebren erhält man, wenn die Folgerungsbeziehung  $\vdash$  als grundlegende Relation betrachtet wird. Sie ist fast eine Ordnungsrelation auf der Menge der aussagenlogischen Formeln, denn sie ist *reflexiv*

<sup>23</sup>Mathematisch erhält man solch einen Homomorphismus aus  $S$ , indem man  $n$  Teilgebiete der Ebene so wählt, dass alle  $2^n$  Kombinationen aus Schnitten der Teilgebiete und ihrer Komplemente vorkommen. Dies definiert eine Surjektion  $\pi : \mathbb{R}^2 \rightarrow M$ , aus der man einen injektiven Homomorphismus  $\pi_* : \mathfrak{P}(M) \rightarrow \mathfrak{P}(\mathbb{R}^2)$ ,  $X \mapsto \pi^{-1}[X] = \{\bar{r} \in \mathbb{R}^2 \mid \pi(\bar{r}) \in X\}$  bekommt. Dann ist  $\pi_* \circ S$  der gesuchte Homomorphismus  $\mathcal{B} \rightarrow \mathfrak{P}(\mathbb{R}^2)$ .



[reflexive] und *transitiv* [transitive] und damit eine sogenannte *Prä- oder Quasiordnung* [preorder / quasiorder], aber nicht *antisymmetrisch* [antisymmetric], d. h. aus  $F \vdash G$  und  $G \vdash F$  folgt nicht die Gleichheit  $F = G$ . Es folgt aber, dass  $F$  und  $G$  logisch äquivalent zueinander sind.

Für jede Quasi-Ordnung  $(M, \sqsubseteq)$  definiert „ $x \sqsubseteq y$  und  $y \sqsubseteq x$ “ eine Äquivalenzrelation auf  $M$ , und die Quasi-Ordnung induziert eine *partielle Ordnung* [partial ordering] auf der Menge der Äquivalenzklassen, also eine reflexive, transitive und antisymmetrische Relation. Insbesondere bekommt man also aus der logischen Folgerung  $\vdash$  eine partielle Ordnung auf der Tarski-Lindenbaum-Algebra.

Allgemeiner ist jeder Verband (und damit jede Boole'sche Algebra) auf natürliche Weise partiell geordnet; die Ordnung erhält man durch

$$a \sqsubseteq b : \iff a \sqcup b = b \iff a \sqcap b = a$$

□

(für die Äquivalenz braucht man die Absorptionsgesetze!). In Potenzmengenalgebren ist dies gerade die Teilmengenbeziehung  $\subseteq$ . In den Hasse-Diagrammen sieht man die partielle Ordnung an den Verbindungslinien:  $a \sqsubseteq b$  gilt genau dann, wenn man von  $a$  aus entlang von nach oben führenden Verbindungslinien nach  $b$  kommt.

Angewandt auf die Tarski-Lindenbaum-Algebra bzw. auf aussagenlogische Formeln ergibt die Definition der partiellen Ordnung im Verband übrigens die bisher noch nicht als logischen Gesetze formulierten Äquivalenzen

$$F \vdash G \iff (F \wedge G) \sim F \iff (F \vee G) \sim G.$$

Umgekehrt findet man in der partiellen Ordnung die Verbandsoperationen auf die folgende Weise:  $a \sqcap b$  ist das *Infimum* [infimum] der beiden Elemente  $a$  und  $b$ , d. h. das größte Element  $x$ , für das  $x \sqsubseteq a$  und  $x \sqsubseteq b$  gilt, und  $a \sqcup b$  ist das *Supremum* [supremum] der beiden Elemente  $a$  und  $b$ , d. h. das kleinste Element  $x$ , für das  $a \sqsubseteq x$  und  $b \sqsubseteq x$  gilt. Man kann zeigen, dass jede partielle Ordnung, in der für je zwei Elemente Infimum und Supremum existieren, schon ein Verband ist. Die Elemente 1 und 0 eines beschränkten Verbandes findet man in der partiellen Ordnung wieder als das *größte Element* [greatest element / top] bzw. *kleinste Element* [least element / bottom]. Für die Tarski-Lindenbaum-Algebra entsprechen diese Eigenschaften den Gesetzen *verum ex quolibet* und *ex falso quodlibet*, also  $F \vdash \top$  bzw.  $\perp \vdash F$  für beliebiges  $F$ .<sup>24</sup>

## Dualität

**Definition 1.8.3** Zu einer Boole'schen Algebra  $\mathcal{B} = (B; \sqcap, \sqcup, ^c, 1, 0)$  ist die *duale Algebra* [dual algebra] definiert als  $\mathcal{B}^* = (B; \sqcup, \sqcap, ^c, 0, 1)$ . Man vertauscht also die Rollen von  $\sqcup$  und  $\sqcap$  und von 0 und 1. In der Betrachtungsweise als partielle Ordnung ist  $\mathcal{B}^* = (B; \sqsupseteq)$  die umgekehrte Ordnung zu  $\mathcal{B} = (B; \sqsubseteq)$ , also  $x \sqsubseteq_{\mathcal{B}^*} y \iff y \sqsubseteq_{\mathcal{B}} x$ .

□

**Satz 1.8.4**  $\mathcal{B}$  und  $\mathcal{B}^*$  sind isomorph zueinander via  $b \mapsto b^c$ .

BEWEIS: Aus der Doppelnegationsregel  $b^{cc} = b$  folgt, dass die Abbildung  $b \mapsto b^c$  selbst-invers und somit eine Bijektion ist. Die Homomorphieeigenschaft für  $\sqcap, \sqcup$  findet sich in den *de Morgan'schen* Regeln  $(a \sqcap b)^c = a^c \sqcup b^c$  und  $(a \sqcup b)^c = a^c \sqcap b^c$ , für die Konstanten 0, 1 in den Regeln  $0^c = 1$  und  $1^c = 0$ , und für das Komplement ist sie das trivialerweise gültige  $b^{cc} = b$ . □

<sup>24</sup>Komplemente sieht man nur über die Extremalgesetze. Will man Boole'sche Algebren als partielle Ordnungen definieren, wird man deswegen und wegen der Distributivität teilweise doch wieder bei den Axiomen von Definition 1.8.1 landen.

Für die Tarski-Lindenbaum-Algebra in der Sichtweise partieller Ordnungen ist dieser Satz nichts anderes als die Kontrapositionsregel:

$$F \vdash G \iff \vdash (F \rightarrow G) \iff \vdash (\neg G \rightarrow \neg F) \iff \neg G \vdash \neg F$$

In den Axiomen der Boole'schen Algebren sieht man die Dualität darin, dass sie (abgesehen von der Doppelnegationsregel) aus Paaren von zueinander dualen Axiomen bestehen. Zu jedem Axiom gibt es ein anderes, das durch die Vertauschung von  $\sqcap$  und  $\sqcup$  bzw. 1 und 0 entsteht. Dies kann man zu einem umfassenderen *Dualitätsprinzip* [duality principle] verallgemeinern:

**Definition 1.8.5** Sei  $F$  eine Formel, in der die Junktoren  $\rightarrow$  und  $\leftrightarrow$  nicht vorkommen. Die *duale Formel* [dual formula]  $F^*$  entsteht aus  $F$ , indem simultan alle Vorkommen von  $\wedge$  durch  $\vee$ , von  $\vee$  durch  $\wedge$ , von  $\top$  durch  $\perp$  und von  $\perp$  durch  $\top$  ersetzt werden.

$F^*$

**Beispiel:** Die duale Formel zu  $(\neg(A_0 \wedge \perp) \vee (\neg A_2 \vee A_0))$  ist  $(\neg(A_0 \vee \top) \wedge (\neg A_2 \wedge A_0))$ .

**Satz 1.8.6 (Dualitätsprinzip)**  $F \vdash G \iff G^* \vdash F^*$  und  $F \sim G \iff F^* \sim G^*$ .

BEWEIS: Falls  $F \vdash G$ , so folgt mit der Kontrapositionsregel  $\neg G \vdash \neg F$ . Kommen  $\rightarrow$  und  $\leftrightarrow$  nicht vor, kann man mittels der *de Morgan'schen* Regeln die Negation sukzessive nach innen ziehen; dabei wird  $F$  fast zu  $F^*$ , nur dass zusätzlich vor jeder Aussagenvariable ein Negationsjunktoren steht. Also hat man (wenn  $A_1, \dots, A_n$  die vorkommenden Aussagenvariablen sind)

$$G^* \left[ \frac{\neg A_0}{A_0} \dots \frac{\neg A_n}{A_n} \right] \vdash F^* \left[ \frac{\neg A_0}{A_0} \dots \frac{\neg A_n}{A_n} \right],$$

wobei diese Verallgemeinerung der Schreibweise aus Satz 1.6.4 für eine *simultane* uniforme Substitution der Aussagenvariablen durch ihre Negationen steht. Durch uniforme (Re-)Substitution ( $A_i$  wird erneut durch  $\neg A_i$  substituiert und die Doppelnegationsregel angewandt) erhält man schließlich  $G^* \vdash F^*$ . Die zweite Aussage über Äquivalenz folgt daraus.  $\square$

In dem (speziell dafür eingerichteten) Hasse-Diagramm von  $\mathcal{F}_2$  auf Seite 38 sieht man die Dualität, also die Isomorphie zwischen  $\mathcal{F}_2$  und  $\mathcal{F}_2^*$ , als Punktsymmetrie am Mittelpunkt. Die Substitution von Aussagenvariablen durch ihre Negationen ist fast eine Spiegelung an der Mittelsenkrechten (lediglich  $(A \leftrightarrow B)$  und  $\neg(A \leftrightarrow B)$  bleiben invariant). Den Übergang zur dualen Formel wie in Satz 1.8.6 sieht man daher als Spiegelung an der horizontalen Gerade durch den Mittelpunkt (wieder bis auf  $(A \leftrightarrow B)$  und  $\neg(A \leftrightarrow B)$ , die ineinander übergehen).

## 1.9 Semantik der intuitionistischen Aussagenlogik

**Im WS 2021/22 voraussichtlich nicht Teil der Vorlesung.**

Die *intuitionistische Aussagenlogik* [intuitionistic propositional logic] ist eine Logik, die die gleiche Syntax wie die klassische zweiwertige Aussagenlogik benutzt, aber eine andere Semantik. Entwickelt wurde der Intuitionismus von L. E. J. Brouwer zu Beginn des 20. Jahrhunderts als eine mathematik-philosophische Position im sogenannten Grundlagenstreit der Mathematik. Die intuitionistische Logik wurde von seinem Schüler Arend Heyting formalisiert.

Grob gesprochen werden in der Semantik die Wahrheitswerte *wahr* und *falsch* durch explizite Beweisbarkeit bzw. Widerlegbarkeit ersetzt. Dadurch entfällt das Prinzip des ausgeschlossenen Dritten, da es Aussagen geben kann, die weder beweisbar noch widerlegbar sind. Der Intuitionismus ist je nach Sichtweise schwächer oder stärker als die klassische Logik: Alle intuitionistischen Äquivalenzen sind auch klassisch gültig, aber nicht umgekehrt. Es gelten also *weniger* Gesetze, dadurch hat die intuitionistische Logik aber eine *größere* Differenzierungskraft.

Für die Informatik ist die intuitionistische Logik insofern von Bedeutung, als sie in einem gewissen Sinn das Berechenbarkeitsverhalten beschreibt: Die *Curry-Howard-Korrespondenz* [Curry-Howard correspondence] (auch: Curry-Howard-Isomorphismus) beschreibt eine Übersetzung zwischen Beweisen in intuitionistischer Logik und Programmen.

Ich beschränke mich in meiner kurzen Darstellung der intuitionistischen Logik hier auf das Junktorensystem  $\{\perp, \rightarrow, \wedge, \vee\}$ , da auch intuitionistisch die folgenden Äquivalenzen gelten, die daher als Definitionen für die fehlenden Junktoren  $\leftrightarrow$ ,  $\neg$  und  $\top$  genommen werden können:

$$\begin{aligned}(A_0 \leftrightarrow A_1) &\sim ((A_0 \rightarrow A_1) \wedge (A_1 \rightarrow A_0)) \\ \neg A_0 &\sim (A_0 \rightarrow \perp) \\ \top &\sim \neg \perp \sim (\perp \rightarrow \perp)\end{aligned}$$

Von diesen vier Junktoren kann allerdings kein weiterer mehr weggelassen werden. Die Regeln zur Formelbildung sind ansonsten die gleichen wie in der klassischen Aussagenlogik.

**Definition 1.9.1** Ein *Modell für die intuitionistische Logik* (in diesem Abschnitt kurz „Modell“ [model] genannt) ist eine *partiell geordnete Menge*  $(W, \leq)$ .<sup>25</sup>

Die Elemente von  $W$  nennt man *Zustände* [nodes] oder (*mögliche*) *Welten* [(possible) worlds], die Relation  $\leq$  heißt oft *Zugangsrelation* [accessibility relation].<sup>26</sup>

Für  $w_1 \leq w_2$  sagt man: „ $w_2$  kommt nach  $w_1$ “ oder „ $w_2$  ist später als  $w_1$ “ (bzw. „ $w_1$  sieht  $w_2$ “).

**Definition 1.9.2** Eine *Belegung* [assignment], genauer: eine *Belegung der Aussagenvariablen mit Wahrheitswerten in einem Modell für die intuitionistische Logik*, ist eine *monotone Abbildung*  $\beta : W \times \{A_i \mid i \in \mathbb{N}\} \rightarrow \{0, 1\}$ . *Monotonie* bedeutet hier, dass mit  $w_1 \leq w_2$  auch  $\beta(w_1, A_i) \leq \beta(w_2, A_i)$  gilt.  $\beta(w, A_i)$

<sup>25</sup>Zur Erinnerung: Dies bedeutet, dass  $\leq$  eine reflexive, transitive und antisymmetrische binäre Relation ist.

<sup>26</sup>Diese Terminologie stammt aus der Modallogik, in der man die intuitionistische Aussagenlogik interpretieren kann. Die Darstellung hier gibt diese Interpretation wieder, ohne über Modallogik zu sprechen.

Für diejenigen, die Modallogik kennen: Man interpretiert eine aussagenlogische Formel  $F$  durch eine modallogische Formel  $F^*$ , indem man zunächst alle Vorkommen von Aussagenvariablen  $A_i$  durch  $\Box A_i$  ersetzt und dann sukzessive über den Aufbau der Formel alle Vorkommen von Teilformeln  $(G \rightarrow H)$  durch  $\Box(G \rightarrow H)$ . Dann ist  $F$  genau dann eine intuitionistische Tautologie, wenn  $F^*$  eine Tautologie in der Modallogik S4 ist.

Solch eine Belegung liefert also insbesondere für jede Welt des Modells eine klassische Belegung. Konkret heißt die Monotonie, dass ein Wert 1 in allen späteren Zuständen erhalten bleibt, während sich ein Wert 0 in einem späteren Zustand zu 1 ändern kann. Vorstellen kann man sich ein  $w \in W$  mit  $\beta(w, A_i) = 1$  als einen Zeitpunkt, zu dem man einen Beweis (bzw. eine Berechnung) für  $A_i$  hat – der in späteren Zuständen nicht verloren geht – während  $\beta(w, A_i) = 0$  bedeutet, dass man (noch) keinen Beweis (bzw. keine Berechnung) für  $A_i$  hat. Im Gegensatz zur klassischen Logik bedeutet die Zuweisung des Wertes 0 zu  $A_i$  hier also nicht, dass  $A_i$  falsch ist, sondern nur, dass in diesem Zustand nicht bekannt ist, ob  $A_i$  gilt oder nicht. Die Formel  $A_i$  wird erst dann als falsch im Modell unter einer Belegung angesehen, wenn sie in jedem Zustand von der Belegung den Wert 0 zugewiesen bekommt.

Wie in der klassischen Logik wird die Belegung nun zu einer Funktion fortgesetzt, die in jedem Zustand  $w \in W$  allen aussagenlogischen Formeln  $F$  einen Wert  $\beta(w, F) \in \{0, 1\}$  zuordnet.

 $\beta(w, F)$ 

$$\begin{aligned} \beta(w, \perp) &:= 0 \\ \beta(w, (F \wedge G)) &:= \min\{\beta(w, F), \beta(w, G)\} \\ \beta(w, (F \vee G)) &:= \max\{\beta(w, F), \beta(w, G)\} \\ \beta(w, (F \rightarrow G)) &:= \begin{cases} 1 & \text{falls } \beta(w', F) \leq \beta(w', G) \text{ für alle } w' \in W \text{ mit } w \leq w' \\ 0 & \text{sonst} \end{cases} \end{aligned}$$

**Lemma 1.9.3** Die Fortsetzungen von Belegungen sind für alle Formeln monoton, d. h. für alle Formeln  $F$  folgt aus  $w_1 \leq w_2$  auch  $\beta(w_1, F) \leq \beta(w_2, F)$ .

BEWEIS: Per Induktion über den Aufbau der Formeln:

Monotonie gilt per Definition der Belegung für Aussagenvariablen und offensichtlich für  $\perp$ . Man weiß oder überlegt sich leicht, dass Maximum und Minimum monotoner Funktionen wieder monoton sind, also bleibt die Eigenschaft bei Konjunktionen und Disjunktionen erhalten.

Schließlich ist die Belegung für die Implikation so definiert, dass sie monoton wird: Wenn  $w \leq w''$  und  $\beta(w, (F \rightarrow G)) = 1$ , dann ist per Definition  $\beta(w', F) \leq \beta(w', G)$  für alle  $w'$  mit  $w \leq w'$ , insbesondere also auch für alle  $w'$  mit  $w'' \leq w'$ , da  $\leq$  transitiv ist.  $\square$

**Definition 1.9.4 (a)** Eine aussagenlogische Formel  $F$  heißt **intuitionistische Tautologie**, falls  $\beta(w, F) = 1$  für alle Modelle  $(W, \leq)$ , alle Belegungen  $\beta$  und alle  $w \in W$ .

**(b)** Zwei aussagenlogische Formeln  $F_1, F_2$  heißen **intuitionistisch äquivalent** zueinander, falls  $\beta(w, F_1) = \beta(w, F_2)$  für alle Modelle  $(W, \leq)$ , alle Belegungen  $\beta$  und alle  $w \in W$ .

**(c)** Eine aussagenlogische Formel  $F$  folgt **intuitionistisch** aus einer Menge  $\{F_i \mid i \in I\}$  von Formeln, falls für jedes Modell  $(W, \leq)$ , jede Belegung  $\beta$  und jedes  $w \in W$ , in dem für alle  $i \in I$  die Bedingung  $\beta(w, F_i) = 1$  erfüllt ist, auch  $\beta(w, F) = 1$  gilt.

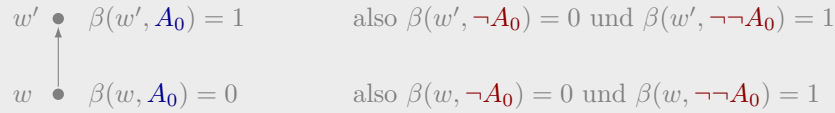
Man schreibt als Zeichen  $\vdash_{\text{int}}$  bzw.  $\sim_{\text{int}}$  und sieht leicht aus der Definition, dass entsprechende Äquivalenzen wie in der klassischen Logik gelten, also beispielsweise:

 $\vdash_{\text{int}}$  $\sim_{\text{int}}$ 

$$\begin{aligned} \vdash_{\text{int}} F &\iff F \sim_{\text{int}} \top \\ F_1 \sim_{\text{int}} F_2 &\iff \vdash_{\text{int}} (F_1 \leftrightarrow F_2) \\ F_1 \vdash_{\text{int}} F_2 &\iff \vdash_{\text{int}} (F_1 \rightarrow F_2) \end{aligned}$$

Endliche oder diskrete Modelle  $(W, \leq)$  zeichnet man gerne als gerichtete Graphen, wobei die Kanten in Richtung der Zugangsrelation weisen, also zu den späteren Zuständen hin, und nur Kanten zu den unmittelbaren echten Nachfolgern hin gezeichnet werden.

Das einfachste nicht-triviale Modell zeigt bereits zwei wesentliche Unterschiede zwischen der intuitionistischen und der klassischen Aussagenlogik:



Man sieht zum einen, dass wegen  $\beta(w, (A_0 \vee \neg A_0)) = 0$  das Prinzip des ausgeschlossenen Dritten verletzt ist. Zum andern sieht man ebenfalls im Zustand  $w$ , dass  $A_0$  und  $\neg\neg A_0$  nicht äquivalent sind, also die Doppelnegationsregel nicht gilt. Es gilt lediglich  $A_0 \vdash_{\text{int}} \neg\neg A_0$ . Auch die *de Morgan*'schen Regeln gelten nur teilweise.

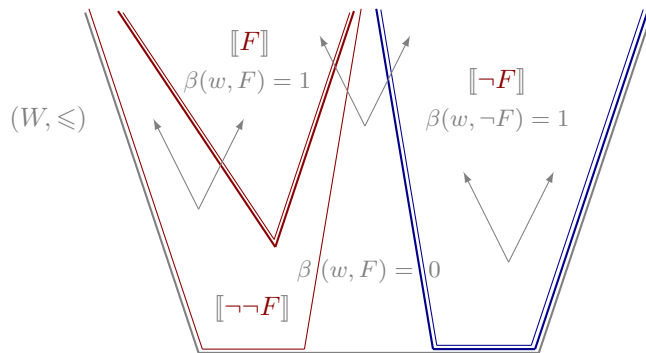
Man kann sich aber überlegen, dass die *Tripelnegationsregel* gilt, also  $\neg A_0 \sim_{\text{int}} \neg\neg\neg A_0$ .<sup>27</sup>

### Mengendarstellung und Heyting-Algebren

Eine Teilmenge von  $(W, \leq)$  nennt man  $\leq$ -abgeschlossen [ $\leq$ -closed], falls sie mit jedem Element auch alle größeren enthält. Lemma 1.9.3 zeigt, dass in einem Modell  $(W, \leq)$  mit Belegung  $\beta$  jede Formel  $F$  eine  $\leq$ -abgeschlossene Menge  $\llbracket F \rrbracket := \{w \in W \mid \beta(w, F) = 1\}$  bestimmt. Es gilt dann

- $\llbracket \perp \rrbracket = \emptyset$
- $\llbracket (F_1 \wedge F_2) \rrbracket = \llbracket F_1 \rrbracket \cap \llbracket F_2 \rrbracket$
- $\llbracket (F_1 \vee F_2) \rrbracket = \llbracket F_1 \rrbracket \cup \llbracket F_2 \rrbracket$
- $\llbracket (F_1 \rightarrow F_2) \rrbracket$  ist die maximale  $\leq$ -abgeschlossene Teilmenge von  $(W \setminus \llbracket F_1 \rrbracket) \cup \llbracket F_2 \rrbracket$

Damit ergibt sich, dass  $\llbracket \neg F \rrbracket$  die größte  $\leq$ -abgeschlossene Teilmenge im Komplement von  $\llbracket F \rrbracket$  ist, im folgenden Bild andeutungsweise illustriert (spätere Zustände stehen weiter oben, zum Teil durch Pfeile angedeutet):



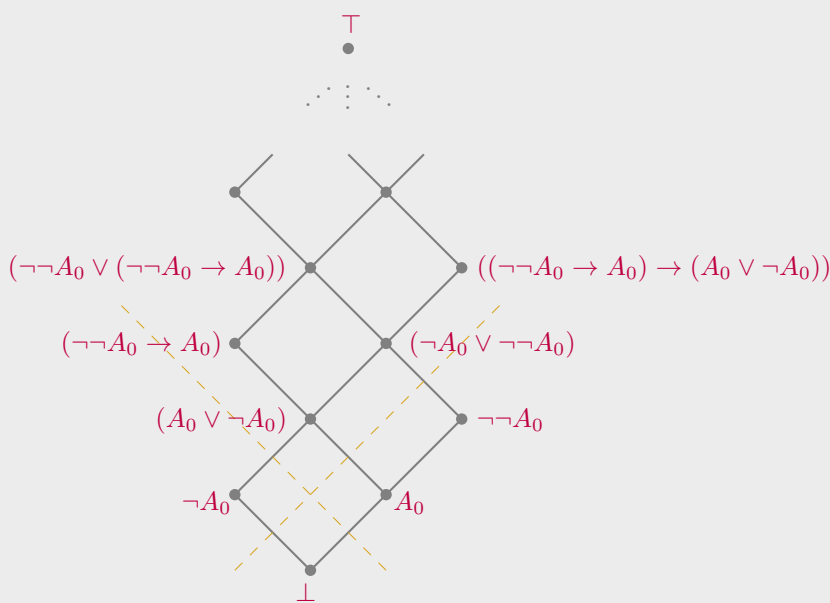
Mit dieser Überlegung kann man sich z. B. die Gültigkeit der Tripelnegationsregel (und andere

<sup>27</sup>Beweisidee: Aus  $A_0 \vdash_{\text{int}} \neg\neg A_0$  bekommt man zum einen durch Substitution  $\neg A_0 \vdash_{\text{int}} \neg\neg\neg A_0$ , zum andern durch Kontraposition  $\neg\neg\neg A_0 \vdash_{\text{int}} \neg A_0$ . Kontraposition funktioniert intuitionistisch aber nur in eine Richtung:  $(A_0 \rightarrow A_1) \vdash_{\text{int}} (\neg A_1 \rightarrow \neg A_0)$ .

Regeln) plausibel machen:  $\llbracket \neg F \rrbracket$  und  $\llbracket \neg\neg F \rrbracket$  sind bereits maximal komplementäre Teilmengen unter den  $\leq$ -abgeschlossenen Mengen, also kann  $\llbracket \neg\neg\neg F \rrbracket$  nicht noch größer als  $\llbracket \neg F \rrbracket$  werden.

Man kann nun Modelle  $(W, \leq)$  mit Belegungen  $\beta$  konstruieren, für die die Abbildung  $F \mapsto \llbracket F \rrbracket$  injektiv ist. Dadurch bekommt man eine Darstellung von Formeln durch Mengen analog zum Satz von Stone. Die Tarski-Lindenbaum-Algebra für die intuitionistische Aussagenlogik ist allerdings keine Boole'sche Algebra mehr, sondern eine sogenannte **Heyting-Algebra** [Heyting algebra]: ein distributiver beschränkter Verband mit Pseudokomplementen [pseudo complements] anstelle von Komplementen.<sup>28</sup>

Im Gegensatz zur klassischen Logik gibt es bereits mit einer Aussagenvariablen unendlich viele Formeln, die intuitionistisch paarweise nicht äquivalent zueinander sind. Das Heyting-Pendant  $\mathcal{H}_1$  zur Boole'schen Algebra  $\mathcal{F}_1$  ist also unendlich groß und sieht folgendermaßen aus:



In der klassischen Logik bleiben bis auf Äquivalenz nur  $\perp$ ,  $A_0$ ,  $\neg A_0$  und  $\top$  übrig:  $\neg\neg A_0$  ist logisch äquivalent zu  $A_0$  und alle Formeln oberhalb von  $(A_0 \vee \neg A_0)$  sind logisch äquivalent zu  $\top$ .

Die Heyting-Algebra  $\mathcal{H}_2$  mit zwei Aussagenvariablen ist bereits so kompliziert, dass man ihren Aufbau nicht mehr als Gesamtes versteht.

### Entscheidbarkeit

Man kann zeigen, dass auch für die intuitionistische Aussagenlogik die Fragen nach Erfüllbarkeit, logischer Äquivalenz oder logischer Folgerung entscheidbar sind, allerdings kann man nicht mehr einfach Wahrheitstabellen ausrechnen. Essentiell sind folgende Eigenschaften (die hier nicht bewiesen werden können), am Beispiel des Erfüllbarkeits- bzw. Tautologieproblems:

**Satz 1.9.5 (a)** *Es gibt ein endliches System von Regeln, aus denen sich (mit Substitution) alle intuitionistischen Tautologien (und nur Tautologien) ableiten lassen.*

<sup>28</sup> $c$  ist Pseudokomplement von  $a$ , wenn  $c$  das eindeutige größte Element mit  $a \sqcap c = 0$  ist. Ein Komplement erfüllt zusätzlich noch  $a \sqcup c = 1$ .

(b) Für jede Formel, die keine intuitionistische Tautologie ist, gibt es bereits ein endliches Modell, in dem sie falsch ist.

Das Entscheidungsverfahren für intuitionistische Tautologien funktioniert nun so: Man lässt zwei Maschinen gleichzeitig laufen. Die erste erzeugt aus dem endlichen Regelsystem systematisch alle Tautologien. Falls eine Formel gegebene Formel  $F$  eine Tautologie ist, wird diese Maschine irgendwann eine Ableitung dieser Tautologie aus den Regeln aufzeigen. Die zweite Maschine erzeugt systematisch alle endlichen Modelle (z. B. der Größe nach) und testet für alle Belegungen, ob  $F$  in allen Zuständen den Wert 1 bekommt. Falls  $F$  keine Tautologie ist, wird diese Maschine irgendwann ein Gegenbeispiel konstruiert haben.

Auch intuitionistisch gilt, dass eine Formel  $F$  genau dann eine Tautologie ist, wenn  $\neg F$  nicht erfüllbar ist (und  $\neg F$  ist genau dann eine Tautologie, wenn  $F$  nicht erfüllbar ist). Mit dem Tautologieproblem ist also auch das Erfüllbarkeitsproblem entschieden.

Im Gegensatz zur klassischen Aussagenlogik folgt die Entscheidbarkeit nicht allein aus Teil (a) des Satzes, da keine explizite kanonische Normalform für intuitionistische Formeln bekannt ist.

## 2 Prädikatenlogik

Die *Prädikatenlogik* [predicate calculus / predicate logic] ist eine Erweiterung der Aussagenlogik, in der es zum einen möglich ist, Aussagenvariablen durch genaue Aussagen über eine gegebene Struktur zu ersetzen, und in der man zum andern durch Quantifikationen eine zusätzliche Ausdrucksstärke erhält. Quantifikationen werden durch *Quantoren* [quantifiers] ausgedrückt und sagen etwas über die Anzahl der Elemente aus, auf die eine Formel zutrifft. Manche nennen die Prädikatenlogik daher auch *Quantorenlogik* [quantificational logic]. Bei der hier beschriebenen Logik handelt es sich genauer um die *Prädikatenlogik erster Stufe* [first-order logic / first-order predicate calculus], die nur Quantifikationen über Elemente zulässt. In der *Prädikatenlogik zweiter Stufe* [second-order logic / second-order predicate calculus] kann man auch über Teilmengen quantifizieren, in der dritten Stufe über Mengen von Teilmengen, etc.

### 2.1 Alphabet der Prädikatenlogik

Im Gegensatz zur Aussagenlogik gibt es in der Prädikatenlogik je nach ins Auge gefasster Anwendung verschiedene *Sprachen* [languages]. Der Gebrauch des Begriffs „Sprache“ ist in der Literatur nicht ganz einheitlich: Während ich ihn für den variablen Anteil des Alphabets benutze (der bei anderen *Signatur* [signature] oder *similarity type* heißt), nehmen ihn andere Autoren für die Menge aller Formeln. Als Notation für Sprachen schreibe ich  $\mathcal{L}$ .

Der feste, von der gewählten Sprache  $\mathcal{L}$  unabhängige Anteil des Alphabets ist:

<i>Individuenvariablen</i> [individual variables]	<i>Quantoren</i> [quantifiers]	<i>Junktoren</i> [connectives]
$v_0$	$\forall$	$\top$
$v_1$	$\exists$	$\perp$
$v_2$		$\neg$
$\vdots$		$\wedge$
	<i>Gleichheitszeichen</i> [equality sign]	$\vee$
	$\doteq$	$\rightarrow$
		$\leftrightarrow$

Das Zeichen  $\forall$  heißt *Allquantor* oder *universeller Quantor* [universal quantifier], das Zeichen  $\exists$  heißt *Existenzquantor* [existential quantifier]. Das Gleichheitszeichen wird mit einem Punkt notiert (*Ziegler'sche Konvention*), um es von dem metasprachlichen Gleichheitszeichen zu unterscheiden. Formeln werden auch in der Prädikatenlogik Bäume sein. Für die übliche Darstellung der Formeln als Zeichenketten (Symbolfolgen) braucht man wieder zusätzlich Klammern.

Die Sprache  $\mathcal{L}$ , also der variable Anteil des Alphabets, besteht aus:

<i>Funktionszeichen</i> [function symbols]	<i>Relationszeichen</i> [relation symbols]
$f_i$ für $i \in I$	$R_j$ für $j \in J$
Jedes Funktions- und jedes Relationszeichen hat in $\mathcal{L}$ eine feste Stelligkeit $\in \mathbb{N}$	

Genauer ist damit gemeint, dass es zwei Indexmengen  $I$  und  $J$  gibt (die endlich oder unendlich sein und insbesondere auch leer sein können), für jedes  $i \in I$  ein Funktionszeichen  $f_i$  und für



jedes  $j \in J$  ein Relationszeichen  $R_j$ . Falls z. B.  $I = \{0, 13, 35\}$  und  $J = \emptyset$ , soll  $\mathcal{L}$  also genau die Funktionszeichen  $f_0, f_{13}$  und  $f_{35}$  enthalten und keine Relationszeichen. In Anwendungen werde ich häufig andere (kürzere oder traditionelle) Namen für die Funktions- und Relationszeichen wählen, etwa  $f, g, h$  oder auch  $+, \circ, ^{-1}$  für Funktionszeichen,  $P, R, S$  oder auch  $\leq, \subseteq$  für Relationszeichen.

Die **Stelligkeit** [arity] ist eine Funktion  $\mathcal{L} \rightarrow \mathbb{N}$ , ordnet also jedem Funktions- und jedem Relationszeichen eine natürliche Zahl zu, die aus dem Zeichen heraus nicht erkennbar ist (sondern nur aus seiner Verwendung) und bei der Angabe einer Sprache  $\mathcal{L}$  festgelegt werden muss. Man spricht kurz von „ $n$ -stelligen Funktions- bzw. Relationszeichen“ (was sauberer „Zeichen für eine  $n$ -stelligen Funktion bzw. Relation“ heißen müsste.) Nullstellige Funktionszeichen können mit **Konstantenzeichen** oder kurz **Konstanten** [constants] identifiziert werden und werden dann gerne  $c_0, c_1, \dots$  geschrieben, in konkreten Beispielen auch  $0, 1, \dots$  (Erläuterung siehe Seite 54). Nullstellige Relationszeichen können mit **Aussagenvariablen** identifiziert werden und werden dann, wie in der Aussagenlogik, auch  $A_0, A_1, \dots$  geschrieben.<sup>29</sup> Einstellige Relationszeichen nenne ich auch **Prädikate**<sup>30</sup> [predicates], woher die Prädikatenlogik ihren Namen hat.

**Beispiele:** Eine für die Beschreibung der natürlichen Zahlen  $\mathbb{N}$  geeignete Sprache ist  $\mathcal{L}_{\mathbb{N}} = \{<, +, \cdot, 0, 1\}$ , wobei  $<$  ein zweistelliges Relationszeichen ist,  $+$  und  $\cdot$  zweistellige Funktionszeichen und  $0$  und  $1$  nullstellige Funktionszeichen (Konstanten).

Zur Beschreibung von Graphen nutzt man die Sprache  $\mathcal{L} = \{E\}$ , wobei  $E$  ein zweistelliges Relationszeichen ist.

Für Gruppen kann man die Sprache  $\mathcal{L} = \{\circ, ^{-1}, e\}$  verwenden, in der  $\circ$  zweistelliges,  $^{-1}$  einstelliges und  $e$  nullstelliges Funktionszeichen ist.

## 2.2 Syntax der Prädikatenlogik

Die Konstruktion der Formeln ist in der Prädikatenlogik komplizierter als in der Aussagenlogik: Zunächst muss man *Terme* und *atomare Formeln* der prädikatenlogischen Sprache  $\mathcal{L}$  definieren, aus denen man dann beliebige Formeln erhält. Gut verständlich wird ihre Konstruktion erst mit der Auswertung der Formeln, die ebenfalls komplizierter als in der Aussagenlogik ist und in passenden *Strukturen* vorgenommen wird.

Terme sind Ausdrücke, die für Elemente einer Struktur stehen werden, und sie bekommen, wenn sie nicht Teil einer Formel sind, eine eigene **violette Farbe**. Formeln wird man als Aussagen über Strukturen interpretieren können; sie behalten die **rote Formelfarbe** der Aussagenlogik. Sowohl Terme als auch Formeln sind induktiv definierte Bäume, für die es Darstellungen als Zeichenketten gibt. Definitionen und Beweise müssen in der Regel wieder induktiv *über den Aufbau der Terme* bzw. *über den Aufbau der Formeln* erfolgen, was wieder als Induktion über Länge, Höhe oder Schachtelungstiefe aufgefasst werden kann. Man kann die Vorgehensweise aus der Aussagenlogik übertragen; ich werde daher nicht mehr näher darauf eingehen.

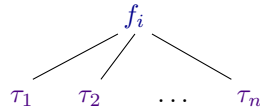
<sup>29</sup>Die meisten Autoren lassen keine 0-stelligen Relationszeichen zu, was den Nachteil hat, dass die Prädikatenlogik dann keine Erweiterung der Aussagenlogik ist.

<sup>30</sup>Nicht in der in der deutschen Grammatik üblichen Bedeutung (vgl. *Subjekt, Prädikat, Objekt*), sondern in der Bedeutung „(auszeichnende) Eigenschaft“ wie in *Prädikatswein, Prädikatsexamen* bzw. in der Bedeutung der klassischen Grammatik als *Satzaussage*, die eine Eigenschaft des Nomens ausdrückt. Genauer müsste es *Prädikatszeichen* heißen. Einige Autoren verwenden *Prädikat(szeichen)* auch mit beliebigen Stelligkeiten anstelle von *Relationszeichen*.

Im Folgenden ist  $\mathcal{L}$  stets eine feste prädikatenlogische Sprache.

**Definition 2.2.1** Ein  $\mathcal{L}$ -Term [ $\mathcal{L}$ -term] ist ein endlicher etikettierter, orientierter Wurzelbaum, der nach folgenden Regeln gebildet werden kann:

- Jede Individuenvariable ist ein  $\mathcal{L}$ -Term  $v_i$  (d. h. der einelementige Baum mit Etikett  $v_i$ ).
- Wenn  $f_i$  ein  $n$ -stelliges Funktionszeichen  $f_i$  in  $\mathcal{L}$  ist und  $\tau_1, \dots, \tau_n$   $\mathcal{L}$ -Terme sind, ist auch



ein  $\mathcal{L}$ -Term. Ich stelle Terme als Zeichenketten in Polnischer Notation  $f_i\tau_1 \dots \tau_n$  dar. <sup>31</sup>

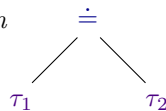
Als Variablen für Terme nehme ich üblicherweise kleine griechische Buchstaben wie  $\tau$  und  $\sigma$ .

$\tau, \sigma, \dots$

Analog zur Darstellung aussagenlogischer Formeln in Polnischer Notation sieht man auch für Terme die *eindeutige Lesbarkeit*, d. h. wenn  $f_i\tau_1 \dots \tau_n$  und  $f_j\tau'_1 \dots \tau'_m$   $\mathcal{L}$ -Terme sind, die als Zeichenketten gleich sind, gilt  $f_i = f_j$ ,  $n = m$  und  $\tau_i = \tau'_i$ .

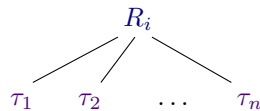
**Definition 2.2.2** Die folgenden etikettierten, orientierten Wurzelbäume sind *atomare  $\mathcal{L}$ -Formeln* [*atomic  $\mathcal{L}$ -formulae*]:

- Verum  $\top$  und Falsum  $\perp$  (als einelementige Bäume mit dem jeweiligen Etikett).
- Für alle  $\mathcal{L}$ -Terme  $\tau_1$  und  $\tau_2$  der Baum



Als Zeichenkette schreibe ich ihn in Infix-Notation, also  $\tau_1 \doteq \tau_2$ .

- Für alle  $n$ -stelligen Relationszeichen  $R_i$  in  $\mathcal{L}$  und alle  $\mathcal{L}$ -Terme  $\tau_1, \dots, \tau_n$  der Baum



Als Zeichenkette schreibe ich ihn in Polnischer Notation, also  $R_i\tau_1 \dots \tau_n$ . <sup>32</sup>

Die Terme sind in dieser Definition als Etiketten der Blätter gemeint. Es ändert im Grunde aber auch nichts, wenn man sie als Teilbäume ansieht.

**Bemerkung 2.2.3** Man kann das Gleichheitszeichen  $\doteq$  als ein „festes“ zweistelliges Relationszeichen auffassen (das im Unterschied zu den Relationszeichen der Sprache in jeder Struktur eine feste Interpretation hat, nämlich die Identität), und man kann Verum und Falsum als „feste“ nullstellige Relationszeichen auffassen (die ebenfalls in jeder Struktur eine feste Interpretation haben, nämlich im Wesentlichen den Wahrheitswert „wahr“ bzw. „falsch“). In diesem Sinne sind alle atomaren  $\mathcal{L}$ -Formeln von der Gestalt  $R\tau_1 \dots \tau_n$  für Terme  $\tau_i$  und ein Relationszeichen  $R$ , das entweder aus  $\mathcal{L}$  kommt oder eines der Zeichen  $\top$ ,  $\perp$ ,  $\doteq$  ist.

<sup>31</sup>In Anwendungen mit traditionellen, maximal zweistelligen Funktionszeichen wie  $+$ ,  $\circ$ ,  $^{-1}$  verwende ich die Infix-Notation, also  $(v_0 + v_1)$  statt  $+v_0 v_1$ , oder andere übliche Notationen wie  $(v_0^{-1})$  statt  $^{-1}v_0$ . Bei geschachtelten Termen braucht man dann Klammern, um die eindeutige Lesbarkeit sicherzustellen.

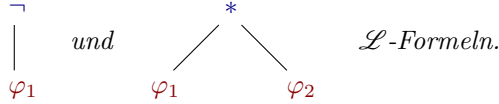
Manche Autoren schreiben Terme auch in der üblichen Funktionsschreibweise als  $f_i(\tau_1, \dots, \tau_n)$ .

<sup>32</sup>Analog zu Termen: In Anwendungen mit traditionellen, maximal zweistelligen Relationszeichen wie  $\leq$  verwende ich die Infix-Notation, also  $v_0 \leq v_1$  statt  $\leq v_0 v_1$ .

Manche Autoren schreiben auch bei Relationszeichen Klammern und Kommata:  $R_i(\tau_1, \dots, \tau_n)$ .

**Definition 2.2.4** Eine  $\mathcal{L}$ -Formel [ $\mathcal{L}$ -formula] ist ein endlicher etikettierter, orientierter Wurzelbaum, der nach folgenden Regeln gebildet werden kann:

- Jede atomare  $\mathcal{L}$ -Formel ist eine  $\mathcal{L}$ -Formel.
- Wenn  $\varphi_1$  und  $\varphi_2$   $\mathcal{L}$ -Formeln sind und  $*$  ein zweistelliger Junktor ist, sind auch



- Wenn  $\varphi$  eine  $\mathcal{L}$ -Formel ist, sind für jede Variable  $v_i$   $\forall v_i \varphi$  und  $\exists v_i \varphi$   $\mathcal{L}$ -Formeln. <sup>33</sup>

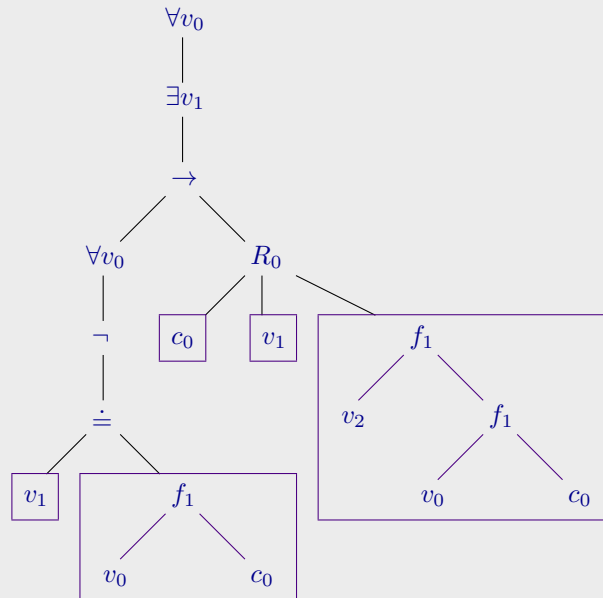
In der Darstellung als Zeichenkette nutze ich für die zweistelligen Junktoren die Infix-Notation mit Klammern:  $(\varphi_1 * \varphi_2)$ , ansonsten die Präfix-Stellung:  $\neg\varphi_1, \forall v_i \varphi$  bzw.  $\exists v_i \varphi$ .

Als Variablen für  $\mathcal{L}$ -Formeln nehme ich üblicherweise kleine griechische Buchstaben vom Ende des Alphabets wie  $\varphi, \chi, \psi$ . <sup>34</sup>

$\varphi, \chi, \psi \dots$

$\top$  und  $\perp$  sind in der Aussagenlogik nullstellige Junktoren und als solche hätten sie von der Systematik her eigentlich in den zweiten Punkt der gerade gegebenen Definition gehört. In der Prädikatenlogik scheint mir allerdings die Auffassung als nullstellige Relationszeichen und damit als atomare  $\mathcal{L}$ -Formeln passender, was auch dazu führt, dass jede  $\mathcal{L}$ -Formel ein atomare  $\mathcal{L}$ -Formel als Teilformel enthält.

**Beispiel:** Sei  $c_0$  ein nullstelliges und  $f_1$  ein einstelliges Funktionszeichen und  $R_0$  ein dreistelliges Relationszeichen in  $\mathcal{L}$ . Dann ist folgender Baum eine  $\mathcal{L}$ -Formel (Terme sind eingekästelt):



Als Zeichenkette lautet die Formel  $\forall v_0 \exists v_1 (\forall v_0 \neg v_1 \doteq f_1 v_0 c_0 \rightarrow R_0 c_0 v_1 f_1 v_2 f_1 v_0 c_0)$ .

<sup>33</sup>Das Etikett  $\forall v_i$  bzw.  $\exists v_i$  soll hierbei als eine Einheit verstanden werden.

Es gibt in der Literatur diverse Varianten der Quantorenschreibweise, häufig mit zusätzlichen Klammern.

<sup>34</sup>Griechische Buchstaben für prädikatenlogische Formeln sind weit verbreitet. Dass ich im Gegensatz dazu in der Aussagenlogik große lateinische Buchstaben genommen habe, hat keinen tieferen Grund; es dient lediglich der klareren Abgrenzung.

Man kann sich davon überzeugen, dass  $\mathcal{L}$ -Formeln, wenn sie in der angegebenen Weise als Zeichenketten geschrieben werden, eindeutig lesbar sind (trotz der Mischung aus Polnischer und Infix-Notation). Eine rein Polnische (wie auch eine rein Umgekehrt Polnische) Notation ist ebenfalls denkbar und für Beweise nützlich, aber schwer lesbar. Da die Stelligkeit von Funktions- und Relationszeichen auch größer als 2 sein kann, ist eine reine Infix-Notation nicht möglich.

Der Allquantor  $\forall$  und der Existenzquantor  $\exists$  kommen in einer  $\mathcal{L}$ -Formel nur in Verbindung mit einer Individuenvariablen  $v_i$  vor. Man nennt daher die Verbindung  $\forall v_i$  bzw.  $\exists v_i$  der Einfachheit halber ebenfalls einen *Quantor*.

**Definition 2.2.5** Eine *Teilformel* [subformula] einer  $\mathcal{L}$ -Formel  $\varphi$  ist eine  $\mathcal{L}$ -Formel, die im induktiven Aufbauprozess von  $\varphi$  vorkommt (analog zur Aussagenlogik).

Der *Wirkungsbereich* [scope] des Vorkommens eines Quantors  $\forall v_i$  bzw.  $\exists v_i$  in einer  $\mathcal{L}$ -Formel  $\varphi$  ist diejenige Teilformel  $\varphi'$ , die „unterhalb von diesem Vorkommen des Quantors hängt“.

$\varphi'$  ist also die  $\mathcal{L}$ -Formel, zu der im Aufbauprozess der Formel  $\varphi$  das Vorkommen des Quantors, um das es geht, als neue Wurzel hinzukommt. Anders ausgedrückt: Um  $\varphi'$  zu erhalten, nimmt man zunächst die Teilformel  $\varphi''$ , deren Wurzel das in Frage stehende Vorkommen des Quantors ist, und entfernt diese Wurzel.

Achtung: Wenn ein Quantor wie  $\forall v_0$  mehrfach in einer  $\mathcal{L}$ -Formel vorkommt, haben die verschiedenen Vorkommen verschiedene Wirkungsbereiche! Üblicherweise spricht man aber kurz vom „Wirkungsbereich eines Quantors“.

Der Wirkungsbereich des „obersten“ Vorkommens von  $\forall v_0$  in dem vorherigen Beispiel ist die Teilformel mit Wurzel  $\exists v_1$ . Der Wirkungsbereich des anderen Vorkommens von  $\forall v_0$  ist die Teilformel mit Wurzel  $\neg$ .

Der Wirkungsbereich des Quantor  $\exists v_1$  ist die Teilformel mit Wurzel  $\rightarrow$ .

Die Variable  $v_i$  in einem Quantor  $\forall v_i$  oder  $\exists v_i$  sollte man am besten als eine Art Zeiger verstehen, der angibt, auf welche Individuenvariablen in seinem Wirkungsbereich sich der Quantor bezieht. In der Symbolfolge  $\forall v_i$  oder  $\exists v_i$  soll das Zeichen  $v_i$  daher nicht als Vorkommen der Individuenvariable  $v_i$  in einer  $\mathcal{L}$ -Formel zählen. Individuenvariable kommen demgemäß in einer  $\mathcal{L}$ -Formel nur innerhalb von  $\mathcal{L}$ -Termen vor.

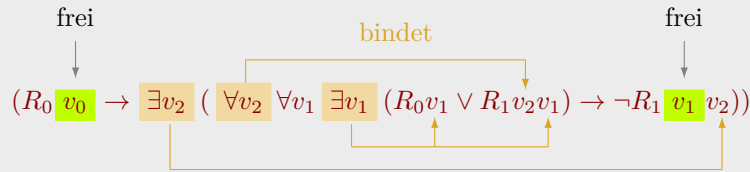
**Definition 2.2.6** Ein Vorkommen einer Individuenvariablen  $v_i$  in einer  $\mathcal{L}$ -Formel  $\varphi$  heißt *frei* [free], wenn es sich nicht im Wirkungsbereich eines Quantors  $\exists v_i$  bzw.  $\forall v_i$  befindet.

Ein Vorkommen einer Individuenvariablen  $v_i$  heißt *gebunden* [bounded] durch einen Quantor  $\exists v_i$  bzw.  $\forall v_i$ , wenn es sich im Wirkungsbereich  $\varphi'$  dieses Quantors befindet und in  $\varphi'$  frei ist.

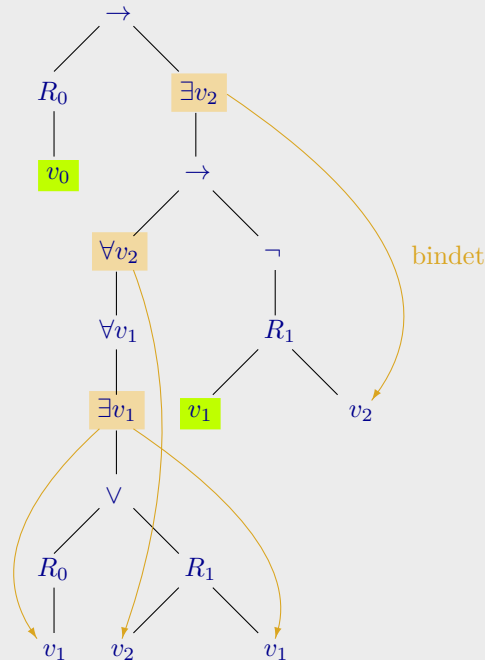
Die *freien Variablen* [free variables] einer Formel  $\varphi$  sind diejenigen Individuenvariablen, die ein freies Vorkommen in  $\varphi$  haben.

Ein  $\mathcal{L}$ -Term ohne Individuenvariablen heißt *geschlossener  $\mathcal{L}$ -Term* [closed  $\mathcal{L}$ -term], eine  $\mathcal{L}$ -Formel ohne freie Variable heißt  *$\mathcal{L}$ -Aussage*, geschlossene  $\mathcal{L}$ -Formel oder auch  *$\mathcal{L}$ -Satz* [ $\mathcal{L}$ -sentence / closed  $\mathcal{L}$ -formula].

**Beispiel:** Sei  $R_0$  einstelliges und  $R_1$  zweistelliges Relationszeichen.



Als Baum:



Das Vorkommen von  $v_0$  ist klarerweise frei, da es in der Formel überhaupt keine Quantoren  $\forall v_0$  oder  $\exists v_0$  gibt (und es auch nicht im Wirkungsbereich irgendeines Quantors liegt).

Der Quantor  $\forall v_1$  ist ein „unnötiger Quantor“, da er keine Variable bindet: Alle Vorkommen von  $v_1$  in seinem Wirkungsbereich liegen bereits im Wirkungsbereich von  $\exists v_1$ . Das „hinterste“ Vorkommen von  $v_i$  ist frei, da es nicht im Wirkungsbereich eines Quantors  $\forall v_i$  oder  $\exists v_i$  liegt.

Ein Vorkommen von  $v_2$  wird von  $\exists v_2$  und eines von  $\forall v_2$  gebunden, obwohl letzteres auch im Wirkungsbereich von  $\exists v_2$  liegt: Es ist aber bereits durch  $\forall v_2$  gebunden, bevor der Quantor  $\exists v_2$  im Aufbauprozess der Formel hinzukommt.

In Anwendungen sollte man sich darum bemühen, dass es keine „unnötigen Quantoren“ gibt, die keine Variablen binden, und dass die Variablen so benannt sind, dass alle Vorkommen von Variablen  $v_i$  im Wirkungsbereich eines Quantors  $\forall v_i$  oder  $\exists v_i$  auch von diesem Quantor gebunden werden. Mit anderen Worten: Man sollte Quantoren nicht so schachteln, dass sich die Wirkungsbereiche von zwei Quantoren  $\forall v_i$ ,  $\exists v_i$  überschneiden. Das kann man bis auf logische Äquivalenz stets erreichen.

Für  $n \geq 1$  und einen  $\mathcal{L}$ -Term  $\tau$  bzw. eine  $\mathcal{L}$ -Formel  $\varphi$  bedeuten die Notationen  $\tau(v_{i_1}, \dots, v_{i_n})$  und  $\varphi(v_{i_1}, \dots, v_{i_n})$ , dass mit  $v_{i_1}, \dots, v_{i_n}$  paarweise verschiedene Individuenvariablen benannt sind und dass alle in  $\tau$  vorkommenden bzw. alle in  $\varphi$  frei vorkommenden Individuenvariablen in der Menge  $\{v_{i_1}, \dots, v_{i_n}\}$  enthalten sind. Achtung: Es gibt keine spezielle Schreibweise dafür, dass ein  $\mathcal{L}$ -Term geschlossen ist oder eine  $\mathcal{L}$ -Formel eine  $\mathcal{L}$ -Aussage ist.

$\tau(v_{i_1}, \dots, v_{i_n})$   
 $\varphi(v_{i_1}, \dots, v_{i_n})$

## 2.3 Semantik der Prädikatenlogik

### $\mathcal{L}$ -Strukturen

Sei  $\mathcal{L}$  eine beliebige prädikatenlogische Sprache.

**Definition 2.3.1** Eine  $\mathcal{L}$ -Struktur  $\mathcal{M}$  besteht aus

- einer nicht-leeren Menge  $M$ , dem **Universum** (oder Träger oder Grundmenge) der Struktur;
- einer Funktion  $f_i^{\mathcal{M}} : M^n \rightarrow M$  für jedes  $n$ -stellige Funktionszeichen  $f_i \in \mathcal{L}$  und  $n \in \mathbb{N}$ ;
- einer Teilmenge  $R_j^{\mathcal{M}} \subseteq M^n$  für jedes  $n$ -stellige Relationszeichen  $R_j \in \mathcal{L}$  und  $n \in \mathbb{N}$ .

$f_i^{\mathcal{M}}$

$R_j^{\mathcal{M}}$

Man sagt, dass ein  $n$ -stelliges Funktionszeichen  $f_i$  durch die  $n$ -stellige Funktion  $f_i^{\mathcal{M}}$  **interpretiert** wird bzw. dass die Funktion  $f_i^{\mathcal{M}}$  **Interpretation** des Funktionszeichens  $f_i$  ist, und analog für Relationszeichen und Relationen. Eigentlich ist eine Relation  $R^{\mathcal{M}}$  eine Eigenschaft von  $n$ -Tupeln: Sie trifft auf ein  $n$ -Tupel zu oder nicht. Bisweilen schreibt man  $R^{\mathcal{M}} m_1 \dots m_n$  dafür, dass  $R^{\mathcal{M}}$  auf  $m_1 \dots m_n$  zutrifft. In der Mathematik werden Relationen in der Regel mit ihren Graphen identifiziert:

$R^{\mathcal{M}} m_1 \dots m_n$

$$R^{\mathcal{M}} = \{(m_1, \dots, m_n) \in M^n \mid R^{\mathcal{M}} \text{ trifft auf } m_1 \dots m_n \text{ zu}\}$$

so dass man statt  $R^{\mathcal{M}} m_1 \dots m_n$  auch  $(m_1, \dots, m_n) \in R^{\mathcal{M}}$  schreiben kann. Ich bevorzuge diese Schreibweise, u. a. weil damit optisch klarer zwischen der  $\mathcal{L}$ -Formel  $Rv_1 \dots v_n$  und ihrer Auswertung unterschieden werden kann.

Im Fall  $n = 0$  ist  $M^0 = \{\emptyset\}$  eine ein-elementige Menge. Eine 0-stellige Funktion  $f^{\mathcal{M}} : M^0 \rightarrow M$  ist also eine konstante Funktion und kann mit ihrem Bild  $f^{\mathcal{M}}(\emptyset) \in M$  identifiziert werden; ein 0-stelliges Funktionszeichen kann daher als *Konstantenzeichen* oder kurz *Konstante* verstanden werden. Eine 0-stellige Relation  $R^{\mathcal{M}}$  trifft entweder auf das einzige 0-Tupel  $\emptyset$  zu oder nicht, je nachdem ob der Graph von  $R^{\mathcal{M}}$  die ein-elementige Menge  $\{\emptyset\}$  oder die leere Menge  $\emptyset$  ist. Ein 0-stelliges Relationzeichen kann daher mit einer *Aussagenvariablen* gleichgesetzt werden: Trifft die interpretierende 0-stellige Relation auf  $\emptyset$  zu, entspricht dies der Zuordnung des Wahrheitswerts 1 oder *wahr*, andernfalls des Wahrheitswerts 0 oder *falsch*.

**Beispiel:** (a) Ein Graph  $(E, K)$  mit Eckenmenge  $E$  und Kantenrelation  $K$  wird für  $\mathcal{L}_{\text{Gr}} = \{R_0\}$  mit 2-stelligem Relationszeichen  $R_0$  zu einer  $\mathcal{L}_{\text{Gr}}$ -Struktur  $\mathcal{G}$ , indem man als Universum  $G = E$  die Menge der Ecken nimmt und das Relationszeichen durch die Kantenrelation interpretiert, also  $R_0^{\mathcal{G}} = K$  setzt.

(b) Sei  $\mathcal{L} = \{f_0, f_1, f_2, f_3, f_4, R_0\}$  mit 2-stelligen Funktionszeichen  $f_0, f_1$ , 1-stelligem Funktionszeichen  $f_2$ , 0-stelligen Funktionszeichen  $f_3, f_4$  und einem 2-stelligen Relationszeichen  $R_0$ .

Dann wird die Menge  $\mathbb{Z}$  der ganzen Zahlen zu einer  $\mathcal{L}$ -Struktur

$$\mathcal{Z} = (\mathbb{Z}; +^{\mathbb{Z}}, \cdot^{\mathbb{Z}}, -^{\mathbb{Z}}, 0^{\mathbb{Z}}, 1^{\mathbb{Z}}, \leq^{\mathbb{Z}}),$$

dem angeordneten Ringe der ganzen Zahlen. Es wird also  $f_0$  durch die Addition,  $f_1$  durch die Multiplikation,  $f_2$  durch die Abbildung  $z \mapsto -z$ ,  $f_3$  durch die Zahl 0 und  $f_4$  durch die Zahl 1 interpretiert, und das Relationszeichen  $R_0$  durch die Kleiner-Gleich-Relation.

Auch eine Boole'sche Algebra  $\mathcal{B}$  ist auf natürliche Weise eine  $\mathcal{L}$ -Struktur

$$\mathcal{B} = (B; \sqcap^{\mathcal{B}}, \sqcup^{\mathcal{B}}, c^{\mathcal{B}}, 1^{\mathcal{B}}, 0^{\mathcal{B}}, \sqsubseteq^{\mathcal{B}}),$$

indem man also  $R_0$  durch die partielle Ordnung der Boole'schen Algebra interpretiert,  $f_0$  und  $f_1$  durch Infimum bzw. Supremum,  $f_2$  durch die Komplementfunktion und  $f_3, f_4$  durch größtes bzw. kleinstes Element.

(Bei solchen „klassischen“ Beispielen möchte man sich aber die Umständlichkeit dieser Darstellung ersparen und die Standardnotationen übernehmen. Statt  $f_0, f_1, f_2$  schreibt man lieber  $+, \cdot, -$  bzw.  $\sqcap, \sqcup, ^c$ , die man dann durch die tatsächlichen Operationen interpretiert, die ich hier der Unterscheidung willen  $+^{\mathbb{Z}}, \cdot^{\mathbb{Z}}, -^{\mathbb{Z}}$  bzw.  $\sqcap^{\mathbb{B}}, \sqcup^{\mathbb{B}}, ^c^{\mathbb{B}}$  schreibe. In der normalen Mathematik macht man allerdings den Unterschied zwischen Zeichen und Interpretation nicht; das macht die Darstellung solcher Beispiele etwas schwierig, weil man nicht weiß, ob man  $+$  besser als das Funktionszeichen oder besser als die konkrete Funktion nimmt. Augenfällig wird dies bei den Konstanten: Man muss dann  $0$  als das Konstantenzeichen und  $0^{\mathbb{Z}}$  als seine Interpretation in  $\mathbb{Z}$ , also die konkrete Zahl Null nehmen.)

Man kann aber  $\mathbb{Z}$  auf ganz andere und willkürliche Art zu einer  $\mathcal{L}$ -Struktur machen; z. B. zu der  $\mathcal{L}$ -Struktur  $\mathcal{Z}'$  mit  $f_0^{\mathcal{Z}'}(x, y) = x \cdot y$ ,  $f_1^{\mathcal{Z}'}(x, y) = -27y$ ,  $f_2^{\mathcal{Z}'}(x) = 2^{|x|}$ ,  $f_3^{\mathcal{Z}'} = f_4^{\mathcal{Z}'} = -2875$  und  $R_0^{\mathcal{Z}'} = \emptyset$ .

(c) Jede Menge  $M \neq \emptyset$  kann für jede Sprache  $\mathcal{L}'$  zu einer  $\mathcal{L}'$ -Struktur gemacht werden, indem man die Funktions- und Relationszeichen irgendwie festlegt, z. B. können alle interpretierenden Funktionen konstant und alle interpretierenden Relationen die leere Menge sein.

### Auswertung von $\mathcal{L}$ -Formeln in $\mathcal{L}$ -Strukturen

**Definition 2.3.2** Eine Belegung der Individuenvariablen mit Werten in einer  $\mathcal{L}$ -Struktur  $\mathcal{M}$  – kurz: Belegung [assignment] – ist eine Abbildung  $\beta : \{v_i \mid i \in \mathbb{N}\} \rightarrow M$ .

Aus dem Kontext sollte immer klar sein, ob es sich bei einer Belegung um eine Belegung von Aussagenvariablen mit Wahrheitswerten oder um eine Belegung von Individuenvariablen mit Elementen einer Struktur handelt. Auch werde ich im Weiteren kurz „Variable“ für „Individuenvariable“ schreiben.

**Definition 2.3.3** Sei  $\mathcal{L}$  eine prädikatenlogische Sprache,  $\mathcal{M}$  eine  $\mathcal{L}$ -Struktur und  $\beta$  eine Belegung der Individuenvariablen in  $\mathcal{M}$ . Dann lässt sich  $\beta$  durch

$$\beta(f_i \tau_1 \dots \tau_n) := f_i^{\mathcal{M}}(\beta(\tau_1), \dots, \beta(\tau_n))$$

induktiv zu einer Abbildung fortsetzen, die jedem  $\mathcal{L}$ -Term  $\tau$  ein Element  $\beta(\tau) \in M$  zuordnet, die Interpretation oder die Auswertung des  $\mathcal{L}$ -Terms  $\tau$  in  $\mathcal{M}$  unter  $\beta$ .  $\beta(\tau)$

Eine andere übliche Schreibweise für  $\beta(\tau)$  ist  $\tau^{\mathcal{M}}[\beta]$ .

Wenn man im Beispiel des Rings  $\mathbb{Z}$  von Seite 54 den Term  $\tau = f_0 f_2 f_4 f_1 v_1 f_0 v_7 f_4$  hat, bzw. in klassischer Schreibweise  $(-1) + (v_1 \cdot (v_7 + 1))$ , ist also

$$\beta(\tau) = f_0^{\mathbb{Z}}(f_2^{\mathbb{Z}}(f_4^{\mathbb{Z}}), f_1^{\mathbb{Z}}(\beta(v_1), f_0^{\mathbb{Z}}(\beta(v_7), f_4^{\mathbb{Z}}))) = (-1^{\mathbb{Z}})^{\mathbb{Z}} +^{\mathbb{Z}} (\beta(v_1)^{\mathbb{Z}} \cdot^{\mathbb{Z}} (\beta(v_7)^{\mathbb{Z}} +^{\mathbb{Z}} 1^{\mathbb{Z}}))$$

oder, ohne die hochgestellten Indizes für die Interpretation,  $\beta(\tau) = (-1) + (\beta(v_1) \cdot (\beta(v_7) + 1))$ . Für eine Belegung  $\beta$  mit beispielsweise  $\beta(v_1) = 2$  und  $\beta(v_7) = -3$  ergibt sich daraus  $\beta(\tau) = (-1) + (2 \cdot ((-3) + 1)) = -5$ ; für eine andere Belegung  $\beta'$  mit  $\beta'(v_1) = \beta'(v_7) = 3$  ergibt sich  $\beta'(\tau) = (-1) + (3 \cdot (3 + 1)) = 11$ .

In dem Beispiel wurde implizit bereits das folgende offensichtliche Resultat benutzt, dass die Auswertung eines Terms  $\tau(v_{i_1}, \dots, v_{i_n})$  nur von  $\beta(v_{i_1}), \dots, \beta(v_{i_n})$  abhängt:

**Lemma 2.3.4** Falls zwei Belegungen  $\beta$  und  $\beta'$  auf allen in einem  $\mathcal{L}$ -Term  $\tau$  vorkommenden Variablen übereinstimmen, so ist  $\beta(\tau) = \beta'(\tau)$ .

BEWEIS: Klar (oder formaler Beweis per Induktion über den Aufbau der  $\mathcal{L}$ -Terme). □

**Definition 2.3.5** Sei  $\mathcal{L}$  eine prädikatenlogische Sprache,  $\mathcal{M}$  eine  $\mathcal{L}$ -Struktur und  $\beta$  eine Belegung in  $\mathcal{M}$ . Eine  $\mathcal{L}$ -Formel  $\varphi$  gilt in  $\mathcal{M}$  unter  $\beta$  [ $\varphi$  holds in  $\mathcal{M}$  under  $\beta$ ], dafür schreibt man  $(\mathcal{M}, \beta) \models \varphi$ , bzw. gilt nicht in  $\mathcal{M}$  unter  $\beta$ , dafür schreibt man  $(\mathcal{M}, \beta) \not\models \varphi$ , wenn es sich gemäß folgender induktiver Regeln ergibt:

$(\mathcal{M}, \beta) \models \varphi$

$(\mathcal{M}, \beta) \not\models \varphi$

$(\mathcal{M}, \beta) \models \top$	
$(\mathcal{M}, \beta) \not\models \perp$	
$(\mathcal{M}, \beta) \models \tau_1 \doteq \tau_2$	$\iff \beta(\tau_1) = \beta(\tau_2)$
$(\mathcal{M}, \beta) \models R_j \tau_1 \dots \tau_n$	$\iff (\beta(\tau_1), \dots, \beta(\tau_n)) \in R_j^{\mathcal{M}}$ für $n$ -stellige Relationszeichen $R_j$
$(\mathcal{M}, \beta) \models \neg \varphi$	$\iff (\mathcal{M}, \beta) \not\models \varphi$
$(\mathcal{M}, \beta) \models (\varphi_1 \wedge \varphi_2)$	$\iff (\mathcal{M}, \beta) \models \varphi_1$ und $(\mathcal{M}, \beta) \models \varphi_2$
$\vdots$	etc. für die weiteren Junktoren
$(\mathcal{M}, \beta) \models \exists v_i \varphi$	$\iff$ es gibt $m \in M$ mit $(\mathcal{M}, \beta \frac{m}{v_i}) \models \varphi$
$(\mathcal{M}, \beta) \models \forall v_i \varphi$	$\iff$ für alle $m \in M$ gilt $(\mathcal{M}, \beta \frac{m}{v_i}) \models \varphi$
	wobei $\beta \frac{m}{v_i}$ die Belegung $\beta'$ ist mit $\beta'(v_j) := \begin{cases} m & \text{falls } i = j \\ \beta(v_j) & \text{falls } i \neq j \end{cases}$

$\beta[-]$

Eine andere übliche Schreibweise für  $(\mathcal{M}, \beta) \models \varphi$  ist  $\mathcal{M} \models \varphi[\beta]$ . Ich nenne wieder *Auswertung* (von  $\varphi$  in  $\mathcal{M}$  unter  $\beta$ ) sowohl den Prozess zu bestimmen, ob  $(\mathcal{M}, \beta) \models \varphi$ , als auch manchmal das Ergebnis (also den Wahrheitswert der Aussage „ $(\mathcal{M}, \beta) \models \varphi$ “).

Man kann die ersten drei Punkte der Definition als Spezialfälle des vierten auffassen, wenn man das Gleichheitszeichen  $\doteq$  durch die Identität auf  $M$  interpretiert, also  $\doteq^{\mathcal{M}} = \{(m, m) \mid m \in M\}$  setzt, und den Überlegungen auf Seite 54 entsprechend  $\top^{\mathcal{M}} = M^0 = \{\emptyset\}$  und  $\perp^{\mathcal{M}} = \emptyset$ . Denn es gibt nur ein 0-Tupel über  $M$ , nämlich  $\emptyset$ , und für dieses gilt dann  $\emptyset \in \top^{\mathcal{M}}$  und  $\emptyset \notin \perp^{\mathcal{M}}$ .

Sei  $\mathcal{L} = \{f, R\}$  mit 2-stelligem Funktionszeichen  $f$  und 2-stelligem Relationszeichen  $R$ , und sei  $\mathcal{N}$  die  $\mathcal{L}$ -Struktur  $(\mathbb{N}; +^{\mathbb{N}}, \leq^{\mathbb{N}})$ , also die natürlichen Zahlen mit ihrer Addition und ihrer (schwachen) Ordnung. Schließlich soll  $\varphi(v_0, v_1)$  die  $\mathcal{L}$ -Formel  $Rv_0v_1$  sein. (Wenn man  $\leq$  statt  $R$  schreiben würde, wäre  $\varphi$  die Formel  $v_0 \leq v_1$ .)

Ich betrachte zunächst die  $\mathcal{L}$ -Formel  $\psi_1 = \exists v_1 \varphi$ . Für welche  $\beta$  gilt  $(\mathcal{N}, \beta) \models \psi_1$ ? Dazu muss es nach Definition ein  $n \in \mathbb{N}$  geben mit  $(\mathcal{N}, \beta \frac{n}{v_1}) \models \varphi$ , also so dass  $\beta(v_0) \leq^{\mathbb{N}} n$ . Dies ist z. B. mit  $n = \beta(v_0)$  der Fall, also gilt  $(\mathcal{N}, \beta) \models \exists v_1 \varphi$  für beliebige Belegungen  $\beta$ .

Daraus folgt, dass auch  $(\mathcal{N}, \beta) \models \forall v_0 \exists v_1 \varphi$  für jedes  $\beta$ , denn dafür muss man für jede Belegung  $\beta' = \beta \frac{n'}{v_0}$  für beliebiges  $n' \in \mathbb{N}$  zeigen, dass  $(\mathcal{N}, \beta \frac{n'}{v_0}) \models \exists v_1 \varphi$ .



Sei nun  $\psi_2 = \forall v_0 \varphi$ . Für welche  $\beta$  gilt  $(\mathcal{N}, \beta) \models \psi_2$ ? Dazu muss nach Definition für jedes  $n \in \mathbb{N}$  gelten, dass  $(\mathcal{N}, \beta \frac{n}{v_0}) \models \varphi$ , also  $n \leq^{\mathbb{N}} \beta(v_1)$ . Dies ist für kein  $\beta(v_1)$  der Fall, also gilt für jede Belegung  $(\mathcal{N}, \beta) \not\models \psi_2$ .

Ähnlich wie oben folgt daraus für jede Belegung  $(\mathcal{N}, \beta) \not\models \exists v_1 \forall v_0 \varphi$ .

Die Reihenfolge der Quantoren spielt also eine wichtige Rolle: Während bei  $\forall v_0 \exists v_1 \varphi$  für jede „Einsetzung“ für  $v_0$  die passende Einsetzung für  $v_1$  gewählt werden kann, müsste bei  $\exists v_1 \forall v_0 \varphi$  eine Einsetzung für  $v_1$  gefunden werden, die für alle Einsetzungen für  $v_0$  passt. Die Lesereihenfolge von links nach rechts (bzw. im Baum von oben nach unten) gibt also die Abhängigkeit der Quantoren wieder.

Auch die Benennung der Variablen ist insofern wichtig, als man wissen muss, welcher Quantor sich auf welches Vorkommen von Variablen bezieht: Sei  $\psi_3 = \forall v_1 \varphi$ . Damit  $(\mathcal{N}, \beta) \models \psi_3$ , muss nach Definition für jedes  $n \in \mathbb{N}$  gelten, dass  $(\mathcal{N}, \beta \frac{n}{v_1}) \models \varphi$ , also  $\beta(v_0) \leq^{\mathbb{N}} n$ . Dies ist nur für  $\beta(v_0) = 0$  der Fall. Für Belegungen  $\beta$  mit  $\beta(v_0) = 0$  gilt also  $(\mathcal{N}, \beta) \models \psi_3$ ; für alle anderen Belegungen ist  $(\mathcal{N}, \beta) \not\models \psi_3$ .

Es folgt daraus wieder, dass  $(\mathcal{N}, \beta) \models \exists v_0 \forall v_1 \varphi$ , aber  $(\mathcal{N}, \beta) \not\models \forall v_0 \forall v_1 \varphi$  (jeweils für alle  $\beta$ ).

Als letztes Beispiel soll die Formel  $\psi_4 = \forall v_0 \exists v_0 f v_0 v_0 \doteq v_0$  betrachtet werden, alternativ geschrieben als  $\forall v_0 \exists v_0 v_0 + v_0 \doteq v_0$ . Um zu überprüfen, ob  $(\mathcal{N}, \beta) \models \psi_4$ , muss man für jedes  $n \in \mathbb{N}$  testen, ob  $(\mathcal{N}, \beta') \models \exists v_0 f v_0 v_0 \doteq v_0$  mit  $\beta' = \beta \frac{n}{v_0}$  gilt. Dazu braucht man ein  $n' \in \mathbb{N}$  mit  $(\mathcal{N}, \beta'') \models f v_0 v_0 \doteq v_0$ , wobei  $\beta'' = \beta' \frac{n'}{v_0} = \beta \frac{n}{v_0} \frac{n'}{v_0}$ . Nun ist diese Abfolge von Ersetzungen von links nach rechts zu lesen; die zweite Ersetzung hebt also die erste auf und es ist  $\beta'' = \beta \frac{n'}{v_0}$ . Mit  $n' = 0$  funktioniert es, also gilt  $(\mathcal{N}, \beta) \models \psi_4$  (für jedes  $\beta$ ). Man sieht, dass im Auswertungsprozess zwar zunächst der Quantor  $\forall v_0$  betrachtet wird, seine Wirkung aber durch die nachfolgende Betrachtung des „inneren“ Quantors  $\exists v_0$  aufgehoben wird. Der Quantor  $\exists v_0$ , der die Vorkommen der Variablen bindet, ist also der entscheidende Quantor.

Auch in diesem Beispiel wurde implizit bereits vorausgesetzt, dass Variablen, die in den Formeln nicht vorkommen, bei der Auswertung keine Rolle spielen. Man sieht aber auch, dass in diesen Beispielen immer nur die *freien* Variablen eine Rolle spielen. Dies gilt allgemein:

**Satz 2.3.6** Falls zwei Belegungen  $\beta$  und  $\beta'$  auf allen in einer  $\mathcal{L}$ -Formel  $\varphi$  vorkommenden freien Variablen übereinstimmen, so gilt  $(\mathcal{M}, \beta) \models \varphi \iff (\mathcal{M}, \beta') \models \varphi$ .

BEWEIS: Beweis per Induktion über den Aufbau der Formeln, und zwar gleichzeitig für alle möglichen Belegungen (und nicht nur für feste  $\beta, \beta'$ ):

Für  $\top$  und  $\perp$  ist die Aussage trivial; für andere atomare  $\mathcal{L}$ -Formeln gilt es nach Lemma 2.3.4.

Die Junktorenschritte sind wiederum klar nach Definition 2.3.5, da die freien Variablen von  $\varphi = \neg\psi$  die gleichen wie die von  $\psi$  sind, und die freien Variablen von  $\psi_1$  und  $\psi_2$  jeweils enthalten sind in den freien Variablen von  $\varphi = (\psi_1 * \psi_2)$  für einen beliebigen zweistelligen Junktor  $*$ .

Sei nun  $\varphi = \exists v_i \psi$  und  $\beta$  stimme mit  $\beta'$  auf den freien Variablen von  $\varphi$  überein. Falls  $(\mathcal{M}, \beta) \models \varphi$ , dann gibt es ein  $m \in M$  mit  $(\mathcal{M}, \beta \frac{m}{v_i}) \models \psi$ . Die freien Variablen von  $\psi$  sind die freien Variablen von  $\varphi$  zusammen mit  $v_i$ : also stimmen  $\beta \frac{m}{v_i}$  und  $\beta' \frac{m}{v_i}$  auf ihnen überein. Nach Induktion gilt daher  $(\mathcal{M}, \beta' \frac{m}{v_i}) \models \psi$  und somit  $(\mathcal{M}, \beta') \models \varphi$ . Analog für den Allquantor.  $\square$

**Folgerung 2.3.7** Falls  $\varphi$  eine  $\mathcal{L}$ -Aussage ist, gilt  $(\mathcal{M}, \beta) \models \varphi$  entweder für alle oder für kein  $\beta$ . Im ersten Fall schreibt man  $\mathcal{M} \models \varphi$  und sagt: „ $\varphi$  gilt in  $\mathcal{M}$  / trifft in  $\mathcal{M}$  zu / ist wahr in  $\mathcal{M}$ “

$\mathcal{M} \models \varphi$

[holds in / is true in] oder „ $\mathcal{M}$  ist Modell von  $\varphi$  / erfüllt  $\varphi$ “ [is a model of / satisfies]. Andernfalls schreibt man  $\mathcal{M} \not\models \varphi$  und hat die verneinten Sprechweisen oder „ $\varphi$  ist falsch in  $\mathcal{M}$ “.

$\mathcal{M} \not\models \varphi$

**Definition 2.3.8** Seien  $\mathcal{M}$  eine  $\mathcal{L}$ -Struktur,  $m_1, \dots, m_n \in M$  und  $\beta_{m_1 \dots m_n}$  eine Belegung mit  $\beta_{m_1 \dots m_n}(v_i) = m_i$ . Für  $\mathcal{L}$ -Terme  $\tau(v_{i_1}, \dots, v_{i_n})$  bzw.  $\mathcal{L}$ -Formeln  $\varphi(v_{i_1}, \dots, v_{i_n})$  setzt man

$$\begin{aligned} \tau^{\mathcal{M}}[m_1, \dots, m_n] &:= \beta_{m_1 \dots m_n}(\tau) \\ \mathcal{M} \models \varphi[m_1, \dots, m_n] &:\iff (\mathcal{M}, \beta_{m_1 \dots m_n}) \models \varphi \end{aligned}$$

$\tau^{\mathcal{M}}[m_1, \dots, m_n]$   
 $\varphi[m_1, \dots, m_n]$

was nach Lemma 2.3.4 bzw. Satz 2.3.6 wohldefiniert ist.

**Bemerkung 2.3.9** Wenn  $v_{i_1}, \dots, v_{i_n}$  die in einem Term  $\tau(v_{i_1}, \dots, v_{i_n})$  vorkommenden bzw. in einer  $\mathcal{L}$ -Formel  $\varphi(v_{i_1}, \dots, v_{i_n})$  frei vorkommenden Variablen sind, definiert  $\tau$  in jeder  $\mathcal{L}$ -Struktur  $\mathcal{M}$  also eine  $n$ -stellige Abbildung  $\tau^{\mathcal{M}}$  und  $\varphi$  eine  $n$ -stellige Relation  $\varphi^{\mathcal{M}}$ :

$$\begin{aligned} \tau^{\mathcal{M}} : M^n &\rightarrow M, \quad (m_1, \dots, m_n) \mapsto \tau^{\mathcal{M}}[m_1, \dots, m_n] \\ \varphi^{\mathcal{M}} &:= \{(m_1, \dots, m_n) \in M^n \mid \mathcal{M} \models \varphi[m_1, \dots, m_n]\} \subseteq M^n \end{aligned}$$

$\varphi^{\mathcal{M}}$

Man kann die Ausdrücke  $\tau^{\mathcal{M}}[m_1, \dots, m_n]$  und  $\varphi[m_1, \dots, m_n]$  daher als Einsetzungen der Elemente  $m_i$  in die Variablen von  $\tau$  bzw. in die freien Variablen von  $\varphi$  verstehen. Es gilt insbesondere für beliebige Belegungen  $\beta$ :

$$\begin{aligned} \beta(\tau(v_1, \dots, v_n)) &= \tau^{\mathcal{M}}[\beta(v_1), \dots, \beta(v_n)] \\ (\mathcal{M}, \beta) \models \varphi(v_1, \dots, v_n) &\iff \mathcal{M} \models \varphi[\beta(v_1), \dots, \beta(v_n)] \end{aligned}$$

Die **Aussagenlogik** kann man folgendermaßen in den prädikatenlogischen Formalismus aufnehmen: Die Wahl einer Sprache  $\mathcal{L}$  entspricht einer Auswahl von Aussagenvariablen; jede aussagenlogische Formel  $F$  mit passenden Aussagenvariablen ist dann eine  $\mathcal{L}$ -Aussage. Eine Belegung  $\beta$  der Aussagenvariablen mit Wahrheitswerten kann man als Abstraktion von  $\mathcal{L}$ -Strukturen ansehen: Eine beliebige nicht-leere Menge  $M$  wird zu einer  $\mathcal{L}$ -Struktur  $\mathcal{M}_\beta$ , indem man für jede Aussagenvariable  $A_i \in \mathcal{L}$  als Interpretation  $A_i^{\mathcal{M}_\beta} = \{\emptyset\}$  setzt, wenn  $\beta(A_i) = 1$  ist, und  $A_i^{\mathcal{M}_\beta} = \emptyset$ , wenn  $\beta(A_i) = 0$ . Dann gilt  $\beta(F) = 1 \iff \mathcal{M}_\beta \models F$ .

### Allgemeingültigkeit, logische Folgerung und logische Äquivalenz von $\mathcal{L}$ -Formeln

#### Definition 2.3.10

(a) Eine  $\mathcal{L}$ -Formel  $\varphi$  heißt **allgemeingültig** [universally valid], falls sie in allen  $\mathcal{L}$ -Strukturen und unter allen Belegungen gilt – man schreibt dafür  $\vdash \varphi$  oder  $\models \varphi$  – und **erfüllbar** [satisfiable], wenn es eine  $\mathcal{L}$ -Struktur  $\mathcal{M}$  und eine Belegung  $\beta$  mit  $(\mathcal{M}, \beta) \models \varphi$  gibt.<sup>35</sup>

$\vdash \varphi, \models \varphi$

(b) Zwei  $\mathcal{L}$ -Formeln  $\varphi$  und  $\psi$  heißen **logisch äquivalent** [logically equivalent] zueinander, falls sie in allen  $\mathcal{L}$ -Strukturen und unter allen Belegungen gleichermaßen gelten, d. h. falls

$$(\mathcal{M}, \beta) \models \varphi \iff (\mathcal{M}, \beta) \models \psi$$

für alle  $\mathcal{M}$  und  $\beta$ . Dafür schreibt man wie in der Aussagenlogik  $\varphi \sim \psi$ .

$\varphi \sim \psi$

(c) Eine  $\mathcal{L}$ -Formel  $\psi$  folgt **logisch aus** [follows from] einer Menge  $\{\varphi_i \mid i \in I\}$  von  $\mathcal{L}$ -Formeln oder **wird von ihr impliziert** [is implied by], falls  $\psi$  in allen  $\mathcal{L}$ -Strukturen unter allen Belegungen gilt, in denen alle  $\varphi_i$  gelten. Man schreibt dafür  $\{\varphi_i \mid i \in I\} \vdash \psi$  oder  $\{\varphi_i \mid i \in I\} \models \psi$ .

$\{\varphi_i \mid i \in I\} \vdash \psi$   
 $\{\varphi_i \mid i \in I\} \models \psi$

<sup>35</sup>Traditionell nimmt man eher  $\models$  für eine semantisch begründete Folgerungsbeziehung und  $\vdash$  für eine syntaktisch begründete, wobei der Unterschied nach Beweis des entsprechenden Vollständigkeitssatzes entfällt. Ich schreibe aus Gewohnheit eher  $\vdash$  bei der Aussagenlogik und  $\models$  bei der Prädikatenlogik.

(d) Eine Menge  $\Phi$  von  $\mathcal{L}$ -Formeln ist erfüllbar [satisfiable] oder konsistent [consistent], falls es eine  $\mathcal{L}$ -Struktur  $\mathcal{M}$  und eine Belegung  $\beta$  gibt, die alle  $\mathcal{L}$ -Formeln in  $\Phi$  erfüllen. Ich schreibe dafür  $(\mathcal{M}, \beta) \models \Phi$ .

Eine Menge  $T$  von  $\mathcal{L}$ -Aussagen wird auch  $\mathcal{L}$ -Theorie [ $\mathcal{L}$ -theory] genannt. Die Erfüllbarkeit einer  $\mathcal{L}$ -Theorie ist unabhängig von einer Belegung; man schreibt  $\mathcal{M} \models T$  und nennt  $\mathcal{M}$  ein Modell von  $T$  [model of  $T$ ], falls alle  $\mathcal{L}$ -Aussagen in  $T$  in  $\mathcal{M}$  gelten.

Ich benutze für die Schreibweisen entsprechende Konventionen wie in der Aussagenlogik, und Satz 1.4.5 gilt entsprechend für die Prädikatenlogik. Eine Besonderheit ist bei Formeln mit freien Variablen zu beachten: Eine Formel  $\varphi(v_1, \dots, v_n)$  ist nach Definition genau dann allgemeingültig, wenn sie in allen  $\mathcal{L}$ -Strukturen unter allen Belegungen gilt, was genau dann der Fall ist, wenn die universell abquantifizierte Formel  $\forall v_1 \dots \forall v_n \varphi(v_1, \dots, v_n)$  allgemeingültig ist. Es gilt also

$$\models \varphi(v_1, \dots, v_n) \iff \models \forall v_1 \dots \forall v_n \varphi(v_1, \dots, v_n)$$

und daher auch

$$\begin{aligned} \varphi(v_1, \dots, v_n) \sim \psi(v_1, \dots, v_n) &\iff \models (\varphi \leftrightarrow \psi)(v_1, \dots, v_n) \\ &\iff \models \forall v_1 \dots \forall v_n (\varphi \leftrightarrow \psi). \end{aligned}$$

$$\text{aber im Allgemeinen: } \not\iff \forall v_1 \dots \forall v_n \varphi \sim \forall v_1 \dots \forall v_n \psi$$

Zum Beispiel ist  $\forall v_1 P v_1 \sim \forall v_2 P v_2$ , aber  $P v_1$  und  $P v_2$  sind nicht logisch äquivalent zueinander.

Die Definitionen der gerade eingeführten Begriffe hängen zunächst von der Sprache  $\mathcal{L}$  ab. Wenn  $\mathcal{L}' \supseteq \mathcal{L}$ , kann man jede  $\mathcal{L}$ -Formel  $\varphi$  auch als  $\mathcal{L}'$ -Formel auffassen. Man kann sich nun einigermaßen leicht überlegen, dass  $\varphi$  genau dann als  $\mathcal{L}$ -Formel allgemeingültig ist, wenn  $\varphi$  als  $\mathcal{L}'$ -Formel allgemeingültig ist: Denn jede  $\mathcal{L}'$ -Struktur  $\mathcal{M}'$  wird zu einer  $\mathcal{L}$ -Struktur, indem man die Interpretationen der zusätzlichen Zeichen weglässt, und umgekehrt kann man jede  $\mathcal{L}$ -Struktur  $\mathcal{M}$  zu einer  $\mathcal{L}'$ -Struktur machen, indem man die zusätzlichen Zeichen beliebig interpretiert. Da  $\varphi$  eine  $\mathcal{L}$ -Formel ist, hängt ihre Auswertung aber nicht von den Interpretationen der zusätzlichen Zeichen ab. Analog ist auch logische Äquivalenz und logische Folgerung unabhängig von der Sprache.

## 2.4 Gesetze der Prädikatenlogik

### 2.4.1 Substitutionsregeln

Substitutionsregeln geben an, wie man eine  $\mathcal{L}$ -Formel auswerten kann, die durch Ersetzen eines Teils aus einer anderen  $\mathcal{L}$ -Formel entsteht, ohne anhand der Definition induktiv über den Aufbau der Formel gehen zu müssen. Sie stellen in gewisser Weise eine Ausweitung des Kompositionalitätsprinzips dar und besagen, dass man im Auswertungsprozess syntaktisch sinnvolle Teile zusammenfassen kann.

Analog zur Aussagenlogik ist offensichtlich, dass man aus einer  $\mathcal{L}$ -Formel wieder eine  $\mathcal{L}$ -Formel erhält, wenn man eine Teilformel durch eine andere  $\mathcal{L}$ -Formel ersetzt. Ein Spezialfall hiervon ist es, Aussagenvariablen durch  $\mathcal{L}$ -Formeln zu ersetzen. Zunächst kann man die beiden Substitutionsprinzipien der Aussagenlogik übertragen:

**Satz 2.4.1****Äquivalente Substitution**

Wenn  $\psi$  aus der  $\mathcal{L}$ -Formel  $\varphi$  dadurch entsteht, dass eine Teilformel durch eine zu ihr logisch äquivalente Formel ersetzt wird, dann ist  $\psi$  logisch äquivalent zu  $\varphi$ .

**Uniforme Substitution**

Wenn eine aussagenlogische Formel  $F(A_1, \dots, A_n)$ ,  $\mathcal{L}$ -Formeln  $\varphi_1, \dots, \varphi_n$  und eine  $\mathcal{L}$ -Struktur  $\mathcal{M}$  mit Belegung  $\beta$  gegeben sind, dann gilt

$$(\mathcal{M}, \beta) \models F\left[\frac{\varphi_1}{A_1} \dots \frac{\varphi_n}{A_n}\right] \iff \beta'(F) = 1$$

wobei  $F\left[\frac{\varphi_1}{A_1} \dots \frac{\varphi_n}{A_n}\right]$  aus  $F$  durch simultanes Ersetzen aller Vorkommen von  $A_i$  durch  $\varphi_i$  entsteht und  $\beta'$  die Belegung der Aussagenvariablen  $A_1, \dots, A_n$  mit  $\beta'(A_i) = 1 \iff (\mathcal{M}, \beta) \models \varphi_i$  ist.

BEWEIS: Klar nach der Definition der logischen Äquivalenz in 2.3.10 und Definition 2.3.5.  $\square$

Man könnte eine allgemeinere Regel uniformer Substitution formulieren, indem man in einer allgemeinen  $\mathcal{L}$ -Formel Aussagenvariablen ersetzt. Dies ist aber von keinem großen Nutzen.

Aus den beiden Substitutionsprinzipien folgt, dass man aussagenlogische Umformungsschritte auch innerhalb der Prädikatenlogik vornehmen kann, dass also beispielsweise *de Morgan*:

$$\forall v_0 \neg (\exists v_1 R_1 v_0 v_1 \wedge \forall v_2 P v_2) \sim \forall v_0 (\neg \exists v_1 R_1 v_0 v_1 \vee \neg \forall v_2 P v_2)$$

Insbesondere folgt, dass man sich auch in der Prädikatenlogik auf ein vollständiges Junktorensystem zurückziehen kann. Häufig wird  $\{\neg, \wedge\}$  benutzt.

Aus der „Uniformen Substitution“ folgt zudem, dass jede  $\mathcal{L}$ -Formel, die aus einer aussagenlogischen Tautologie durch Ersetzen der Aussagenvariablen hervorgeht, allgemeingültig ist. Solche allgemeingültigen  $\mathcal{L}$ -Formeln nennt man auch  *$\mathcal{L}$ -Tautologien*. Beispielsweise kann man aus der Tautologie  $(A_0 \vee \neg A_0)$  die allgemeingültige  $\mathcal{L}$ -Formel  $(\exists v_1 R_1 v_0 v_1 \vee \neg \exists v_1 R_1 v_0 v_1)$  gewinnen.

Speziell prädikatenlogisch sind zwei Substitutionslemmata für den Fall, dass man eine Individuenvariable durch einen Term ersetzt – eines für Terme und eines für Formeln. Ähnlich wie bei der Ersetzung von Teilformeln muss man sich zunächst davon überzeugen, dass man aus einer  $\mathcal{L}$ -Formel wieder eine  $\mathcal{L}$ -Formel erhält, wenn man einen in der Formel vorkommenden  $\mathcal{L}$ -Term durch einen anderen  $\mathcal{L}$ -Term ersetzt. Für  $\mathcal{L}$ -Terme  $\tau$  und  $\sigma$  sei  $\tau\left[\frac{\sigma}{v_i}\right]$  der  $\mathcal{L}$ -Term, den man erhält, wenn man jedes Vorkommen von  $v_i$  in  $\tau$  durch  $\sigma$  ersetzt.

 $\tau[-]$ **Lemma 2.4.2 (Substitutionslemma für Terme)**

Seien  $\sigma$  und  $\tau$   $\mathcal{L}$ -Terme und  $\beta$  eine Belegung in einer  $\mathcal{L}$ -Struktur  $\mathcal{M}$ . Dann gilt

$$\beta\left(\tau\left[\frac{\sigma}{v_i}\right]\right) = \beta\left(\frac{\beta(\sigma)}{v_i}\right)(\tau)$$

BEWEIS: Klar durch Nachdenken darüber, was die Formel des Lemmas aussagt.  $\square$

Mit der Funktionsschreibweise aus Definition 2.3.8 bedeutet das Lemma für einen Term  $\tau(v_1)$  so viel wie  $\tau\left[\frac{\sigma}{v_1}\right]^{\mathcal{M}} = \tau^{\mathcal{M}} \circ \sigma^{\mathcal{M}}$  und allgemeiner für Terme  $\tau(v_1, \dots, v_n)$  und  $\sigma(v_1, \dots, v_n)$ :

$$\left(\tau\left[\frac{\sigma}{v_i}\right]\right)^{\mathcal{M}}(m_1, \dots, m_n) = \tau^{\mathcal{M}}(m_1, \dots, m_{i-1}, \sigma^{\mathcal{M}}(m_1, \dots, m_n), m_{i+1}, \dots, m_n).$$

Man sieht hieran deutlich, wie die Auswertung des Terms  $\tau[\frac{\sigma}{v_i}]$  „zerlegt“ wird in die Auswertung der Terme  $\tau$  und  $\sigma$ .

Für eine  $\mathcal{L}$ -Formel  $\varphi$  und einen  $\mathcal{L}$ -Term  $\sigma$  sei  $\varphi[\frac{\sigma}{v_i}]$  die  $\mathcal{L}$ -Formel, die man erhält, wenn man jedes freie Vorkommen von  $v_i$  in  $\varphi$  durch  $\sigma$  ersetzt. Bevor man das Substitutionslemma für Formeln analog zum Substitutionslemmas für Terme aufstellen kann, muss man noch ausschließen, dass bei der Substitution ungewollte Quantifizierungen eintreten. Dazu benötigt man die folgende Definition.

**Definition 2.4.3** Die Variable  $v_i$  ist frei für einen  $\mathcal{L}$ -Term  $\sigma$  in einer  $\mathcal{L}$ -Formel  $\varphi$ , wenn bei der Ersetzung  $\varphi[\frac{\sigma}{v_i}]$  kein Vorkommen einer Individuenvariablen  $v_j$  in  $\sigma$  durch einen Quantor von  $\varphi$  gebunden wird.

Formal über den Aufbau der Formeln:  $v_i$  ist frei für  $\sigma$  in  $\varphi$ , falls (a)  $v_i$  nicht frei in  $\varphi$  ist oder falls (b)  $v_i$  frei in  $\varphi$  ist und einer der folgenden Fälle gilt:

- $\varphi$  ist atomar
- $\varphi = \neg\psi$  und  $v_i$  ist frei für  $\sigma$  in  $\psi$
- $\varphi = (\psi * \psi')$  mit beliebigem zweistelligen Junktor  $*$  und  $v_i$  ist frei für  $\sigma$  in  $\psi$  und in  $\psi'$
- $\varphi = \exists v_j \psi$  oder  $\varphi = \forall v_j \psi$ ,  $v_i$  ist frei für  $\sigma$  in  $\psi$  und  $v_j$  kommt in  $\sigma$  nicht vor

#### Lemma 2.4.4 (Substitutionslemma für Formeln)

Seien  $\varphi$  eine  $\mathcal{L}$ -Formel,  $\sigma$  ein  $\mathcal{L}$ -Term und  $\beta$  eine Belegung in einer  $\mathcal{L}$ -Struktur  $\mathcal{M}$ . Dann gilt für jede Variable  $v_i$ , die frei für  $\sigma$  in  $\varphi$  ist:

$$(\mathcal{M}, \beta) \models \varphi[\frac{\sigma}{v_i}] \iff (\mathcal{M}, \beta^{\frac{\beta(\sigma)}{v_i}}) \models \varphi$$

BEWEIS: Beweis per Induktion über den Aufbau der Formeln. Für atomare  $\mathcal{L}$ -Formeln  $\varphi = R_j \tau_1 \dots \tau_n$  (inklusive  $\top$ ,  $\perp$  und  $\tau_1 \doteq \tau_2$ ) ist nach dem Substitutionslemma für Terme:

$$\begin{aligned} (\mathcal{M}, \beta) \models R_j \tau_1 \dots \tau_n [\frac{\sigma}{v_i}] &\iff (\mathcal{M}, \beta) \models R_j \tau_1[\frac{\sigma}{v_i}] \dots \tau_n[\frac{\sigma}{v_i}] \\ &\iff (\beta(\tau_1[\frac{\sigma}{v_i}]), \dots, \beta(\tau_n[\frac{\sigma}{v_i}])) \in R_j^{\mathcal{M}} \\ &\iff (\beta^{\frac{\beta(\sigma)}{v_i}}(\tau_1), \dots, \beta^{\frac{\beta(\sigma)}{v_i}}(\tau_n)) \in R_j^{\mathcal{M}} \\ &\iff (\mathcal{M}, \beta^{\frac{\beta(\sigma)}{v_i}}) \models R_j \tau_1 \dots \tau_n \end{aligned}$$

Die Junktorenschritte sind unproblematisch, wenn man sich klar macht, dass  $(\varphi \wedge \psi)[\frac{\sigma}{v_i}] = (\varphi[\frac{\sigma}{v_i}] \wedge \psi[\frac{\sigma}{v_i}])$  etc. gilt.

Sei nun also  $\varphi = \exists v_j \psi$  (für den Allquantor funktioniert die gleiche Argumentation). Wegen der Annahme, dass  $v_i$  frei für  $\sigma$  in  $\varphi$  ist, kommt  $v_j$  in  $\sigma$  nicht vor. Wenn  $v_i$  nicht frei in  $\varphi$  ist, ist das Substitutionslemma trivial. Sei also  $v_i$  frei in  $\varphi$ . Dann muss  $i \neq j$  sein. Nun gilt:

$$\begin{aligned} (\mathcal{M}, \beta) \models \exists v_j \psi[\frac{\sigma}{v_i}] &\iff \text{es gibt ein } m \in M \text{ mit } (\mathcal{M}, \beta^{\frac{m}{v_j}}) \models \psi[\frac{\sigma}{v_i}] \\ \text{nach Induktionsvoraussetzung} &\iff \text{--- " --- } (\mathcal{M}, \beta^{\frac{m}{v_j} \frac{\beta(\sigma)}{v_i}}) \models \psi \\ \text{da } v_j \text{ in } \sigma \text{ nicht vorkommt} &\iff \text{--- " --- } (\mathcal{M}, \beta^{\frac{m}{v_j} \frac{\beta(\sigma)}{v_i}}) \models \psi \\ \text{da } i \neq j &\iff \text{--- " --- } (\mathcal{M}, \beta^{\frac{\beta(\sigma)}{v_i} \frac{m}{v_j}}) \models \psi \\ &\iff (\mathcal{M}, \beta^{\frac{\beta(\sigma)}{v_i}}) \models \exists v_j \psi \quad \square \end{aligned}$$

Am Beispiel  $\mathcal{N} = (\mathbb{N}; +^{\mathbb{Z}}, 1^{\mathbb{Z}})$  sieht man die Notwendigkeit der Bedingung im Substitutionslemma: Es gilt genau dann  $(\mathcal{N}, \beta) \models \varphi = \exists v_j v_i + v_j \doteq 1$ , wenn  $\beta(v_i) \in \{0, 1\}$ . Ersetzt man in  $\varphi$  die freie Variable  $v_i$  durch den Term  $v_j$ , ergibt sich die in  $\mathcal{N}$  nicht mehr erfüllbare  $\{+, 0\}$ -Formel  $\exists v_j v_j + v_j \doteq 1$ . Das Substitutionslemma ist nicht anwendbar, da  $v_i$  nicht frei für  $v_j$  in  $\varphi$  ist.

## 2.4.2 Gleichheitsgesetze und Quantorengesetze

### Satz 2.4.5 ( $\mathcal{L}$ -Gleichheitsgesetze)

Die folgenden  $\mathcal{L}$ -Aussagen sind allgemeingültig (für beliebige  $n$ -stellige Funktionszeichen  $f$  und Relationszeichen  $R$  in  $\mathcal{L}$ ):

Reflexivität:	$\forall v_{i_0} v_{i_0} \doteq v_{i_0}$
Symmetrie:	$\forall v_{i_0} \forall v_{i_1} (v_{i_0} \doteq v_{i_1} \rightarrow v_{i_1} \doteq v_{i_0})$
Transitivität:	$\forall v_{i_0} \forall v_{i_1} \forall v_{i_2} ((v_{i_0} \doteq v_{i_1} \wedge v_{i_1} \doteq v_{i_2}) \rightarrow v_{i_0} \doteq v_{i_2})$
Kongruenz:	$\forall v_{i_1} \dots \forall v_{i_{2n}} ((v_{i_1} \doteq v_{i_{n+1}} \wedge \dots \wedge v_{i_n} \doteq v_{i_{2n}}) \rightarrow f v_{i_1} \dots v_{i_n} \doteq f v_{i_{n+1}} \dots v_{i_{2n}})$ $\forall v_{i_1} \dots \forall v_{i_{2n}} ((v_{i_1} \doteq v_{i_{n+1}} \wedge \dots \wedge v_{i_n} \doteq v_{i_{2n}}) \rightarrow (R v_{i_1} \dots v_{i_n} \leftrightarrow R v_{i_{n+1}} \dots v_{i_{2n}}))$

BEWEIS: Klar □

### Satz 2.4.6 (Logische Gesetze für Quantoren)

„Unnötige Quantoren“:

$$\text{Wenn } v_i \text{ nicht frei in } \varphi \text{ ist, dann } \quad \exists v_i \varphi \sim \forall v_i \varphi \sim \varphi$$

Umbenennung gebundener Variablen:

Wenn  $v_i$  frei für  $v_j$  in  $\varphi$  ist und  $v_j$  nicht frei in  $\varphi$  ist, dann

$$\exists v_i \varphi \sim \exists v_j \varphi \left[ \frac{v_j}{v_i} \right] \quad \forall v_i \varphi \sim \forall v_j \varphi \left[ \frac{v_j}{v_i} \right]$$

Verhältnis von Quantoren untereinander:

$$\begin{aligned} \forall v_i \varphi &\models \exists v_i \varphi & \exists v_i \forall v_j \varphi &\models \forall v_j \exists v_i \varphi \\ \exists v_i \exists v_j \varphi &\sim \exists v_j \exists v_i \varphi & \forall v_i \forall v_j \varphi &\sim \forall v_j \forall v_i \varphi \end{aligned}$$

Dualität / verallgemeinerte Regeln von *de Morgan*:

$$\exists v_i \neg \varphi \sim \neg \forall v_i \varphi \quad \forall v_i \neg \varphi \sim \neg \exists v_i \varphi$$

Vertauschungen von Quantoren mit Junktoren:

$$\begin{aligned} \exists v_i (\varphi \vee \psi) &\sim (\exists v_i \varphi \vee \exists v_i \psi) & \forall v_i (\varphi \wedge \psi) &\sim (\forall v_i \varphi \wedge \forall v_i \psi) \\ \exists v_i (\varphi \wedge \psi) &\models (\exists v_i \varphi \wedge \exists v_i \psi) & (\forall v_i \varphi \vee \forall v_i \psi) &\models \forall v_i (\varphi \vee \psi) \end{aligned}$$

Falls  $v_i$  nicht frei in  $\varphi$  ist, dann

Falls  $v_i$  nicht frei in  $\varphi$  ist, dann

$$\begin{aligned} \exists v_i (\varphi \wedge \psi) &\sim (\varphi \wedge \exists v_i \psi) & \forall v_i (\varphi \vee \psi) &\sim (\varphi \vee \forall v_i \psi) \\ \exists v_i (\varphi \vee \psi) &\sim (\varphi \vee \exists v_i \psi) & \forall v_i (\varphi \wedge \psi) &\sim (\varphi \wedge \forall v_i \psi) \end{aligned}$$

„Definition der Quantoren“: Falls  $v_i$  frei für  $\tau$  in  $\varphi$  ist, dann

$$\varphi \left[ \frac{\tau}{v_i} \right] \models \exists v_i \varphi \quad \forall v_i \varphi \models \varphi \left[ \frac{\tau}{v_i} \right]$$

Die Umkehrungen der sechs Implikationen  $\models$  gelten im Allgemeinen nicht!

Bei der Umbenennung gebundener Variablen muss man ähnlich wie beim Substitutionslemma vorsichtig sein, dass Variablen nicht irrtümlich in den Wirkungsbereich von Quantoren gelangen, was die beiden Bedingungen verhindern. In folgenden Beispielen darf  $v_0$  nicht durch  $v_1$  ersetzt werden:  $\varphi = \forall v_0 v_0 \dot{=} v_1$  ist nicht äquivalent zu  $\forall v_1 v_1 \dot{=} v_1$  ( $v_1$  ist frei in  $\varphi$ ), und  $\psi = \forall v_0 \forall v_1 v_0 \dot{=} v_1$  ist nicht äquivalent zu  $\forall v_1 \forall v_1 v_1 \dot{=} v_1$  ( $v_0$  ist nicht frei für  $v_1$  in  $\psi$ ).

Auch bei den „Quantorendefinitionen“ ist die Bedingung notwendig ist, denn beispielweise kann man nicht aus  $(\forall v_1 v_0 \dot{=} v_1) \left[ \frac{v_1}{v_0} \right] = \forall v_1 v_1 \dot{=} v_1$  auf  $\exists v_0 \forall v_1 v_0 \dot{=} v_1$  schließen.

Für jede  $\mathcal{L}$ -Struktur  $\mathcal{M}$  kann man die Sprache  $\mathcal{L}$  um Konstanten für Elemente aus  $M$  erweitern:  $\mathcal{L}_M$  besteht aus  $\mathcal{L}$  zusammen mit *neuen* Konstanten  $c_m$  für jedes Element  $m \in M$  („neu“ heißt  $c_m \notin \mathcal{L}$ ). Durch die offensichtliche Interpretation  $c_m^{\mathcal{M}_M} = m$  wird  $\mathcal{M}$  zu einer  $\mathcal{L}_M$ -Struktur  $\mathcal{M}_M$ . Ist  $M = \{m_1, \dots, m_n\}$  endlich, dann gilt für eine  $\mathcal{L}$ -Formel  $\varphi(v_0)$ :

$$\begin{aligned} \mathcal{M} \models \forall v_0 \varphi &\iff \mathcal{M}_M \models (\varphi \left[ \frac{c_{m_1}}{v_0} \right] \wedge \dots \wedge \varphi \left[ \frac{c_{m_n}}{v_0} \right]) \\ \text{und } \mathcal{M} \models \exists v_0 \varphi &\iff \mathcal{M}_M \models (\varphi \left[ \frac{c_{m_1}}{v_0} \right] \vee \dots \vee \varphi \left[ \frac{c_{m_n}}{v_0} \right]) \end{aligned}$$

In diesem Sinne ist also der Allquantor eine Art verallgemeinerte Konjunktion und der Existenzquantor eine Art verallgemeinerte Disjunktion.<sup>36</sup> Dies erklärt auch die verallgemeinerten *de Morgan*'schen Regeln und die Vertauschbarkeit des Allquantors mit der Konjunktion und des Existenzquantors mit der Disjunktion.

BEWEIS DER QUANTORENGESETZE AUS SATZ 2.4.6 :

- Das Gesetz für „unnötige Quantoren“ folgt unmittelbar aus Satz 2.3.6.
- Umbenennung gebundener Variablen:

$$\begin{aligned} (\mathcal{M}, \beta) \models \exists v_j \varphi \left[ \frac{v_j}{v_i} \right] &\stackrel{(Def.)}{\iff} \text{es gibt ein } m \in M \text{ mit } (\mathcal{M}, \beta \frac{m}{v_j}) \models \varphi \left[ \frac{v_j}{v_i} \right] \\ (v_i \text{ frei für } v_j \text{ in } \varphi + &\iff \text{es gibt ein } m \in M \text{ mit } (\mathcal{M}, \beta \frac{m}{v_j} \left[ \frac{\beta \frac{m}{v_j}(v_j)}{v_i} \right] = m) \models \varphi \\ \text{Substitutionslemma}) & \\ (v_j \text{ nicht frei in } \varphi) &\iff \text{es gibt ein } m \in M \text{ mit } (\mathcal{M}, \beta \frac{m}{v_i}) \models \varphi \\ (Definition) &\iff (\mathcal{M}, \beta) \models \exists v_i \varphi \end{aligned}$$

- Die Regeln für die Vertauschungen der Quantoren bzw. der Quantoren mit den Junktoren sieht man unmittelbar durch die Anwendung der Definitionen, weil die entsprechenden Gesetze auf der Meta-Ebene gelten. Ein Beispiel für den Fall, dass  $v_i$  nicht frei in  $\varphi$  ist:

$$\begin{aligned} (\mathcal{M}, \beta) \models \exists v_i (\varphi \wedge \psi) &\stackrel{(Def.)}{\iff} \text{es gibt } m \in M \text{ mit } (\mathcal{M}, \beta \frac{m}{v_i}) \models (\varphi \wedge \psi) \\ (Definition) &\iff \text{es gibt } m \in M \text{ mit } (\mathcal{M}, \beta \frac{m}{v_i}) \models \varphi \text{ und } (\mathcal{M}, \beta \frac{m}{v_i}) \models \psi \\ (Voraussetzung) &\iff \text{es gibt } m \in M \text{ mit } (\mathcal{M}, \beta) \models \varphi \text{ und } (\mathcal{M}, \beta \frac{m}{v_i}) \models \psi \\ (Meta-Gesetz) &\iff (\mathcal{M}, \beta) \models \varphi \text{ und es gibt } m \in M \text{ mit } (\mathcal{M}, \beta \frac{m}{v_i}) \models \psi \\ (Definition) &\iff (\mathcal{M}, \beta) \models \varphi \text{ und } (\mathcal{M}) \models \exists v_i \psi \\ (Definition) &\iff (\mathcal{M}, \beta) \models (\varphi \wedge \exists v_i \psi) \end{aligned}$$

Ein weiteres Beispiel ist auf Seite 75 ausgeführt. Bei  $\forall v_0 \varphi \models \exists v_0 \varphi$  ist die Konvention, dass Strukturen nicht leer sein dürfen, entscheidend.

<sup>36</sup>Manche Autoren schreiben daher  $\bigwedge$  für  $\forall$  und  $\bigvee$  für  $\exists$ .

- Dualität:  $(\mathcal{M}, \beta) \models \neg \exists v_0 \varphi \stackrel{(Def.)}{\iff} (\mathcal{M}, \beta) \not\models \exists v_0 \varphi \stackrel{(Def.)}{\iff}$  es gibt kein  $m \in M$  mit  $(\mathcal{M}, \beta \frac{m}{v_0}) \models \varphi$   
 (Meta-Gesetz)  $\iff$  für alle  $m \in M$  gilt  $(\mathcal{M}, \beta \frac{m}{v_0}) \not\models \varphi$   
 (Definition)  $\iff$  für alle  $m \in M$  gilt  $(\mathcal{M}, \beta \frac{m}{v_0}) \models \neg \varphi$   
 (Definition)  $\iff (\mathcal{M}, \beta) \models \forall v_0 \neg \varphi$

Das andere Dualitätsgesetz folgt daraus mit den Substitutionsprinzipien:

$$\neg \forall v_0 \varphi \sim \neg \forall v_0 \neg \neg \varphi \sim \neg \neg \exists v_0 \neg \varphi \sim \exists v_0 \neg \varphi$$

- „ $\exists$ -Definition“: Angenommen  $(\mathcal{M}, \beta) \models \varphi[\frac{\tau}{v_i}]$ . Wegen der Freiheitsannahme folgt mit dem Substitutionslemma  $(\mathcal{M}, \beta \frac{\beta(\tau)}{v_i}) \models \varphi$ . Also gibt es ein  $m$ , nämlich  $m = \beta(\tau)$ , mit  $(\mathcal{M}, \beta \frac{m}{v_i}) \models \varphi$ . Somit gilt  $(\mathcal{M}, \beta) \models \exists v_i \varphi$ .

„ $\forall$ -Definition“ lässt sich daraus durch Dualität und die Substitutionsprinzipien ableiten (wie generell die  $\forall$ -Version eines Gesetzes aus der  $\exists$ -Version und umgekehrt): Die Kontraposition der „ $\exists$ -Definition“ für  $\neg \varphi$  ergibt  $\forall v_i \varphi \sim \neg \exists v_i \neg \varphi \models \neg \neg \varphi[\frac{\tau}{v_i}] \sim \varphi[\frac{\tau}{v_i}]$ .

- Gegenbeispiele zu den fehlenden Umkehrungen: In  $\mathcal{Z} = (\mathbb{Z}; 0^{\mathbb{Z}}, <^{\mathbb{Z}})$  sieht man

$\mathcal{Z} \models \exists v_0 v_0 \doteq 0$	aber	$\mathcal{Z} \not\models \forall v_0 v_0 \doteq 0$
$\mathcal{Z} \models \forall v_1 \exists v_0 v_0 < v_1$	aber	$\mathcal{Z} \not\models \exists v_0 \forall v_1 v_0 < v_1$
$\mathcal{Z} \models \forall v_0 (v_0 \doteq 0 \vee \neg v_0 \doteq 0)$	aber	$\mathcal{Z} \not\models (\forall v_0 v_0 \doteq 0 \vee \forall v_0 \neg v_0 \doteq 0)$
$\mathcal{Z} \models (\exists v_0 v_0 < 0 \wedge \exists v_0 0 < v_0)$	aber	$\mathcal{Z} \not\models \exists v_0 (v_0 < 0 \wedge 0 < v_0)$

**Satz 2.4.7 (Ableitungsregeln)** Angenommen  $(\varphi \rightarrow \psi)$  ist eine allgemeingültige  $\mathcal{L}$ -Formel.

**Modus Ponens:** Wenn  $\varphi$  ebenfalls allgemeingültig ist, dann ist auch  $\psi$  allgemeingültig.

**$\exists$ -Einführungsregel:** Wenn  $v_i$  nicht frei in  $\psi$  ist, dann ist auch  $(\exists v_i \varphi \rightarrow \psi)$  allgemeingültig.

**$\forall$ -Einführungsregel:** Wenn  $v_i$  nicht frei in  $\varphi$  ist, dann ist auch  $(\varphi \rightarrow \forall v_i \psi)$  allgemeingültig.

$\exists$ - und  $\forall$ -Einführungsregel sind hier etwas allgemeiner formuliert, als ich sie im nächsten Abschnitt brauche. Wichtig ist der Spezialfall  $\varphi = \top$  mit  $(\varphi \rightarrow \psi) \sim \psi$  und  $(\varphi \rightarrow \forall v_i \psi) \sim \forall v_i \psi$ .

**BEWEIS:** Die Korrektheit des *modus ponens* ist klar, und die Korrektheit der  $\forall$ -Einführungsregel folgt mit Dualität aus der Korrektheit der  $\exists$ -Einführungsregel.

Sei nun  $(\varphi \rightarrow \psi)$  allgemeingültig und es gelte  $(\mathcal{M}, \beta) \models \exists v_i \varphi$ ; zu zeigen ist  $(\mathcal{M}, \beta) \models \psi$ . Nach Definition existiert ein  $m \in M$  mit  $(\mathcal{M}, \beta \frac{m}{v_i}) \models \varphi$ . Aus  $(\mathcal{M}, \beta \frac{m}{v_i}) \models (\varphi \rightarrow \psi)$  ergibt sich mit dem aussagenlogischen *modus ponens*  $(\mathcal{M}, \beta \frac{m}{v_i}) \models \psi$ . Da  $v_i$  nicht frei in  $\psi$  ist, folgt  $(\mathcal{M}, \beta) \models \psi$ .  $\square$

Aus einer Implikation wie  $\forall v_i \varphi \models \exists v_i \varphi$  bzw.  $\models (\forall v_i \varphi \rightarrow \exists v_i \varphi)$  ergibt sich mit *modus ponens* eine Ableitungsregel: Wenn  $\forall v_i \varphi$  allgemeingültig ist, dann auch  $\exists v_i \varphi$ . Die Umkehrung gilt aber im Allgemeinen nicht! Beispielsweise besagt die  $\forall$ -Einführungsregel *nicht*, dass  $((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \forall v_i \psi))$  allgemeingültig ist, wenn  $v_i$  nicht frei in  $\varphi$  ist. Man kann den *modus ponens* also nicht umkehren: Wenn aus der Allgemeingültigkeit von  $\varphi$  die Allgemeingültigkeit von  $\psi$  folgt, impliziert dies nicht die Allgemeingültigkeit von  $(\varphi \rightarrow \psi)$ .

Dass  $((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \forall v_i \psi))$  im Allgemeinen nicht allgemeingültig ist, sieht man am besten am Beispiel  $\varphi = \top$  und  $\psi = Pv_0$ . Dann ergibt sich  $(Pv_0 \rightarrow \forall v_i Pv_0)$ , was nicht allgemeingültig ist, weil  $\forall v_i (Pv_0 \rightarrow \forall v_i Pv_0) \sim (\forall v_i \neg Pv_0 \vee \forall v_i Pv_0)$  es offenbar nicht ist.

Dagegen ist  $\exists v_0 (\varphi \rightarrow \forall v_0 \varphi) \sim (\neg \forall v_0 \varphi \vee \forall v_0 \varphi)$  allgemeingültig, was etwa im Beispiel von „Krivines Hut“ kontra-intuitiv wirken kann: In jeder nicht-leeren Menschenmenge gibt es einen



*Menschen mit der Eigenschaft: Wenn dieser Mensch einen Hut trägt, dann tragen alle in der Menge einen Hut. (Wenn alle einen Hut tragen, kann jeder diese Rolle einnehmen; wenn nicht alle einen Hut tragen, wählt man jemanden, der keinen Hut trägt. Man darf die logische Implikation nicht mit einer kausalen Beziehung verwechseln.)*

### 2.4.3 Normalformen und syntaktische Reduktionen

In diesem Abschnitt geht es darum, in welcher Weise man  $\mathcal{L}$ -Formeln bis auf logische Äquivalenz einfacher oder (analog zu DNF und KNF der Aussagenlogik) in einer Normalform, also in einer standardisierten Form darstellen kann. Zunächst eine direkte Folgerung aus Satz 2.4.6 :

**Folgerung 2.4.8**  $\{\exists\}$  und  $\{\forall\}$  sind vollständige Quantorensysteme, d. h. jede  $\mathcal{L}$ -Formel ist logisch äquivalent zu einer  $\mathcal{L}$ -Formel, in der der jeweils andere Quantor nicht vorkommt.

$\{\neg, \wedge, \exists\}$  und  $\{\neg, \wedge, \forall\}$  sind Beispiele vollständiger Junktoren-Quantoren-Systeme, d. h. jede  $\mathcal{L}$ -Formel ist logisch äquivalent zu einer  $\mathcal{L}$ -Formel, in der jeweils keine anderen Junktoren und Quantoren vorkommen.

**Definition 2.4.9** Eine  $\mathcal{L}$ -Formel ist in **pränexer Normalform** [prenex normal form], falls sie die Form

$$Q_1 v_{i_1} \dots Q_n v_{i_n} \psi$$

hat mit  $Q_i \in \{\exists, \forall\}$  und quantorenfreiem  $\psi$  (d. h.  $\psi$  enthält keine Quantoren).

**Satz 2.4.10** Jede  $\mathcal{L}$ -Formel ist zu einer  $\mathcal{L}$ -Formel in pränexer Normalform äquivalent.

Dies geht natürlich nur, wenn man beide Quantoren  $\exists$  und  $\forall$  benutzt!

BEWEIS: Atomare  $\mathcal{L}$ -Formeln sind quantorenfrei und damit in pränexer Normalform. Es reicht nun zu zeigen, dass Zusammensetzungen von Formeln in pränexer Normalform mit dem vollständigen Junktoren-Quantoren-System  $\{\neg, \wedge, \vee, \exists, \forall\}$  wieder in pränexer Normalform gebracht werden können.

- $\exists v_i \varphi$  und  $\forall v_i \varphi$  sind bereits in pränexer Normalform, wenn  $\varphi$  es ist.
- $\neg Q_1 v_{i_1} \dots Q_n v_{i_n} \psi \sim \widetilde{Q}_1 v_{i_1} \dots \widetilde{Q}_n v_{i_n} \neg \psi$  mit  $\widetilde{\exists} = \forall$  und  $\widetilde{\forall} = \exists$ .
- Seien  $Q v_j \varphi$  und  $\chi$  in pränexer Normalform mit  $Q \in \{\exists, \forall\}$  und  $v_k$  eine Variable, die in  $\chi$  nicht frei vorkommt. Dann gilt

$$\begin{aligned} (Q v_j \varphi \wedge \chi) &\sim (Q v_k \varphi[\frac{v_k}{v_j}] \wedge \chi) \sim Q v_k (\varphi[\frac{v_k}{v_j}] \wedge \chi) \\ (Q v_j \varphi \vee \chi) &\sim (Q v_k \varphi[\frac{v_k}{v_j}] \vee \chi) \sim Q v_k (\varphi[\frac{v_k}{v_j}] \vee \chi) \end{aligned}$$

Wegen der Kommutativität der Konjunktion und Disjunktion gelten die analogen Regeln, wenn der Quantor vor  $\chi$  steht. Auf diese Weise kann man sukzessive alle Quantoren vor die Klammer ziehen.  $\square$

Für den Beweis würde es ausreichen, z. B. mit dem vollständigen Junktoren-Quantoren-System  $\{\neg, \wedge, \exists\}$  zu arbeiten. Die hier vorgestellte Variante ist in der Praxis gut verwendbar; für die Junktoren  $\rightarrow$  und  $\leftrightarrow$  gelten schwerer zu merkende Regeln.

**Beispiel:**  $(\exists v_0 \forall v_1 Rv_0v_1 \rightarrow \forall v_1 \exists v_0 Rv_0v_1) \sim (\neg \exists v_0 \forall v_1 Rv_0v_1 \vee \forall v_1 \exists v_0 Rv_0v_1)$   
 $\sim (\forall v_0 \exists v_1 \neg Rv_0v_1 \vee \forall v_1 \exists v_0 Rv_0v_1)$   
 $\sim (\forall v_0 \exists v_1 \neg Rv_0v_1 \vee \forall v_2 \exists v_3 Rv_3v_2)$   
 $\sim \forall v_0 \exists v_1 \forall v_2 \exists v_3 (\neg Rv_0v_1 \vee Rv_3v_2)$   
aber auch:  $\sim \forall v_2 \forall v_0 \exists v_1 \exists v_3 (\neg Rv_0v_1 \vee Rv_3v_2)$

Die pränex Normalform ist also nicht eindeutig, insbesondere kann die Reihenfolge mancher Quantoren vertauscht werden, anderer nicht, und es können gebundene Variable umbenannt werden.

Diese Phänomene können manchmal dazu führen, dass es Äquivalenzen gibt, die zunächst falsch aussehen. Es ist zum Beispiel

$$\begin{aligned} ((\forall v_0 Pv_0 \wedge \exists v_1 Pv_1) \rightarrow \forall v_2 Pv_2) &\sim (\neg \forall v_0 Pv_0 \vee \neg \exists v_1 Pv_1 \vee \forall v_2 Pv_2) \\ &\sim (\exists v_0 \neg Pv_0 \vee \forall v_1 \neg Pv_1 \vee \forall v_2 Pv_2) \\ &\sim \exists v_0 \forall v_1 \forall v_2 ((Pv_0 \wedge Pv_1) \rightarrow Pv_2) \end{aligned}$$

Es gilt hier aber auch (gewissermaßen zufälligerweise)

$$((\forall v_0 Pv_0 \wedge \exists v_1 Pv_1) \rightarrow \forall v_2 Pv_2) \sim \forall v_0 \exists v_1 \forall v_2 ((Pv_0 \wedge Pv_1) \rightarrow Pv_2)$$

weil man  $v_0$  und  $v_1$  umbenennen, die Quantoren in einer anderen Reihenfolge aus der Disjunktion ziehen und die Kommutativität von  $\wedge$  ausnutzen kann.

Man kann die pränex Normalform mit der Disjunktiven (oder der Konjunktiven) Normalform kombinieren, indem man den quantorenfreien Teil einer Formel in pränexer Normalform in DNF oder KNF bringt (wobei die atomaren Teilformeln wie Aussagenvariablen behandelt werden).

Anders als in der Aussagenlogik, wo man durch einfache Festlegungen eine eindeutige „kanonische DNF“ (oder auch KNF) erreichen kann, ist dies in der Prädikatenlogik nicht so einfach möglich, und es ist mir auch kein Versuch bekannt, eine kanonische pränex Normalform zu definieren, die eindeutig wäre.<sup>37</sup>

Eine weitere Möglichkeit,  $\mathcal{L}$ -Formeln bis auf logische Äquivalenz in eine möglichst standardisierte Form zu bringen und dabei ihre Komplexität zu verringern, besteht darin, geschachtelte Terme zu eliminieren.

**Definition 2.4.11** Eine  $\mathcal{L}$ -Formel heißt **term-reduziert** [term-reduced], falls jede ihrer atomaren Teilformeln höchstens ein Relations- oder Funktionszeichen enthält, also von der Form  $\top$ ,  $\perp$ ,  $Rv_{i_1} \dots v_{i_n}$ ,  $v_{i_0} \doteq fv_{i_1} \dots v_{i_n}$  oder  $fv_{i_1} \dots v_{i_n} \doteq v_{i_0}$  ist.

**Satz 2.4.12** Jede  $\mathcal{L}$ -Formel  $\varphi$  ist zu einer term-reduzierten  $\mathcal{L}$ -Formel  $\varphi'$  logisch äquivalent.

**BEWEISSKIZZE:** Man findet  $\varphi'$ , indem man sukzessive atomare Teilformeln  $R\sigma_{j_1} \dots \sigma_{j_m}$  mit

<sup>37</sup>Man kann natürlich immer das Alphabet geeignet abzählen und dann zu einer gegebenen  $\mathcal{L}$ -Formel  $\varphi$  unter allen zu  $\varphi$  logisch äquivalenten  $\mathcal{L}$ -Formeln diejenige suchen, die in der lexikographischen Ordnung zuerst kommt. Das ist dann zwar eindeutig, aber die Reihenfolge auf dem Alphabet ist willkürlich, und es könnte sein, dass es „zufällige“ logische Äquivalenzen gibt, die nicht durch einfache Umformungen erkennbar sind.

einem Term  $\sigma_{j_k} = f\tau_{i_1} \dots \tau_{i_n}$  ersetzt durch

$$\exists v_l (v_l \doteq f\tau_{i_1} \dots \tau_{i_n} \wedge R\sigma_{j_1} \dots \sigma_{j_{k-1}} v_l \sigma_{j_{k+1}} \dots \sigma_{j_m}),$$

wobei  $v_l$  eine noch nicht vorkommenden Variable ist und  $R$  auch  $\doteq$  sein darf. Die Formel wird durch diese Ersetzung zwar länger, die „Gesamtschachtelung“ aber geringer, so dass man per Induktion sieht, dass der Ersetzungsprozess terminiert und  $\varphi'$  liefert. <sup>38</sup>  $\square$

Weitere Reduktionen, die aber nicht zu logisch äquivalenten Formeln führen, sondern nur z. B. zu erfüllbarkeitsäquivalenten, finden sich in Abschnitt 2.6.

## 2.5 Der Gödel'sche Vollständigkeitssatz

Ziel dieses Abschnittes ist es, ein System von logischen Gesetzen und Ableitungsregeln für die Prädikatenlogik anzugeben, aus denen sich alle logischen Gesetze herleiten lassen. Anders als in der Aussagenlogik, wo es ausreichte, die Regeln zu formulieren, die die Umformung in kanonische DNF ermöglichen, ist es für die Prädikatenlogik nicht einfach zu sehen, ob ein Satz von Regeln ausreichend ist.

Im Hauptbeweis wird eine Induktion über den Aufbau der Formeln geführt, bei dem ich die Induktionsschritte auf  $\neg$ ,  $\wedge$  und  $\forall$  beschränken möchte. Ich will deshalb der Einfachheit halber in diesem Abschnitt den Existenzquantor  $\exists v_i$  als Abkürzung für  $\neg\forall v_i\neg$  ansehen und die Junktoren  $\vee$ ,  $\rightarrow$ ,  $\leftrightarrow$  ebenfalls in geeigneter Weise als Abkürzungen. Andernfalls muss man den Kalkül um Regeln für  $\exists$  erweitern und den Beweis um die fehlenden Junktorschritte.

**Definition 2.5.1** Ein Beweis [proof] von  $\varphi$  im Kalkül [deductive system]  $\mathbb{K}$  ist eine Folge von  $\mathcal{L}$ -Formeln  $\varphi_0, \dots, \varphi_n$ , wobei  $\varphi_n = \varphi$  und jedes  $\varphi_i$  entweder ein  $\mathbb{K}$ -Axiom ist oder sich durch eine  $\mathbb{K}$ -Regel aus  $\varphi_0, \dots, \varphi_{i-1}$  ergibt. Eine  $\mathcal{L}$ -Theorie  $T$  beweist [proves] eine  $\mathcal{L}$ -Formel  $\varphi$  in  $\mathbb{K}$ , oder auch:  $\varphi$  ist in  $T$   $\mathbb{K}$ -beweisbar [ $\mathbb{K}$ -provable], wenn es einen Beweis von  $\varphi$  in  $\mathbb{K}$  gibt. Dafür schreibe ich  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi$ .

$\mathbb{K}$

$\vdash_{\mathbb{K}}^{\mathcal{L}}$

<b><math>\mathbb{K}</math>-Axiome sind:</b>	
[AL]	Alle $\mathcal{L}$ -Tautologien („Aussagenlogik“).
[ $\doteq$ -Axiom]	Alle Gleichheitsgesetze aus Satz 2.4.5.
[ $\forall$ -Axiom]	Alle $\mathcal{L}$ -Formeln der Form $(\forall v_i \varphi \rightarrow \varphi_{\frac{\tau}{v_i}})$ , wobei $v_i$ frei für $\tau$ in $\varphi$ ist.
<b><math>\mathbb{K}</math>-Regeln sind:</b>	
[Prämisse]	Wenn $\varphi \in T$ , dann $T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi$ .
[ $\rightarrow$ -Einführung]	Wenn $T \cup \{\varphi\} \vdash_{\mathbb{K}}^{\mathcal{L}} \psi$ für eine $\mathcal{L}$ -Aussage $\varphi$ , dann $T \vdash_{\mathbb{K}}^{\mathcal{L}} (\varphi \rightarrow \psi)$ .
[MP]	Wenn $T \vdash_{\mathbb{K}}^{\mathcal{L}} (\varphi \rightarrow \psi)$ und $T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi$ , dann $T \vdash_{\mathbb{K}}^{\mathcal{L}} \psi$ („modus ponens“).
[ $\forall$ -Einführung]	Wenn $T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi$ , dann $T \vdash_{\mathbb{K}}^{\mathcal{L}} \forall v_i \varphi$ .

Eine  $\mathcal{L}$ -Theorie heißt  $\mathbb{K}$ -widersprüchlich [ $\mathbb{K}$ -contradictory], falls  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \perp$ , und andernfalls  $\mathbb{K}$ -widerspruchsfrei [ $\mathbb{K}$ -consistent].

<sup>38</sup>Für diese „Gesamtschachtelung“ summiert man über alle atomaren Teilformeln jeweils die Anzahl der Vorkommen von Relations- und Funktionszeichen in der Teilformel minus 1.

Der Begriff der  $\mathbb{K}$ -Beweisbarkeit bezieht sich in der Definition auf die Sprache  $\mathcal{L}$ ; in Wirklichkeit ist er aber in folgendem Sinn unabhängig von der Sprache: Falls  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi$  für eine  $\mathcal{L}$ -Theorie  $T$  und eine  $\mathcal{L}$ -Formel  $\varphi$ , dann bleibt ein  $\mathbb{K}$ -Beweis in  $\mathcal{L}$  auch ein  $\mathbb{K}$ -Beweis in größeren Sprachen  $\mathcal{L}' \supseteq \mathcal{L}$ . Die Umkehrung wird der Vollständigkeitssatz zeigen: Falls es einen  $\mathbb{K}$ -Beweis in  $\mathcal{L}' \supseteq \mathcal{L}$  gibt, dann gibt es bereits einen  $\mathbb{K}$ -Beweis in  $\mathcal{L}$ .

**Beispiel:** Ein  $\mathbb{K}$ -Beweis von  $(\forall v_0(\varphi \wedge \psi) \rightarrow (\forall v_0\varphi \wedge \forall v_0\psi))$ :

1. [Prämisse]:  $\forall v_0(\varphi \wedge \psi) \vdash_{\mathbb{K}} \forall v_0(\varphi \wedge \psi)$
2. [ $\forall$ -Axiom] mit  $\tau = v_0$ :  $\forall v_0(\varphi \wedge \psi) \vdash_{\mathbb{K}} (\forall v_0(\varphi \wedge \psi) \rightarrow (\varphi \wedge \psi))$
3. [MP] auf 1, 2:  $\forall v_0(\varphi \wedge \psi) \vdash_{\mathbb{K}} (\varphi \wedge \psi)$
4. [AL]:  $\forall v_0(\varphi \wedge \psi) \vdash_{\mathbb{K}} ((\varphi \wedge \psi) \rightarrow \varphi)$
5. [AL]:  $\forall v_0(\varphi \wedge \psi) \vdash_{\mathbb{K}} ((\varphi \wedge \psi) \rightarrow \psi)$
6. [MP] auf 3, 4:  $\forall v_0(\varphi \wedge \psi) \vdash_{\mathbb{K}} \varphi$
7. [MP] auf 3, 5:  $\forall v_0(\varphi \wedge \psi) \vdash_{\mathbb{K}} \psi$
8. [ $\forall$ -Einführung] auf 6:  $\forall v_0(\varphi \wedge \psi) \vdash_{\mathbb{K}} \forall v_0\varphi$
9. [ $\forall$ -Einführung] auf 7:  $\forall v_0(\varphi \wedge \psi) \vdash_{\mathbb{K}} \forall v_0\psi$
10. [AL]:  $\forall v_0(\varphi \wedge \psi) \vdash_{\mathbb{K}} (\forall v_0\varphi \rightarrow (\forall v_0\psi \rightarrow (\forall v_0\varphi \wedge \forall v_0\psi)))$
11. [MP] auf 8, 10:  $\forall v_0(\varphi \wedge \psi) \vdash_{\mathbb{K}} (\forall v_0\psi \rightarrow (\forall v_0\varphi \wedge \forall v_0\psi))$
12. [MP] auf 9, 11:  $\forall v_0(\varphi \wedge \psi) \vdash_{\mathbb{K}} (\forall v_0\varphi \wedge \forall v_0\psi)$
13. [ $\rightarrow$ -Einführung] auf 12:  $\vdash_{\mathbb{K}} (\forall v_0(\varphi \wedge \psi) \rightarrow (\forall v_0\varphi \wedge \forall v_0\psi))$

Man sieht, dass dies im Vergleich zu einem semantischen Beweis (siehe Seite 75) eher mühsam ist. Kalküle wie  $\mathbb{K}$  haben aber den Vorteil, maschinell ausführbar zu sein.

**Lemma 2.5.2 (a)** Wenn  $T \cup \{\neg\varphi\}$   $\mathbb{K}$ -widersprüchlich ist, gilt  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi$ .

(b) Wenn  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi$  und  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \neg\varphi$ , ist  $T$   $\mathbb{K}$ -widersprüchlich.

BEWEIS: (a) Wenn  $T \cup \{\neg\varphi\}$   $\mathbb{K}$ -widersprüchlich ist, also  $T \cup \{\neg\varphi\} \vdash_{\mathbb{K}}^{\mathcal{L}} \perp$ , folgt  $T \vdash_{\mathbb{K}}^{\mathcal{L}} (\neg\varphi \rightarrow \perp)$  mit [ $\rightarrow$ -Einführung]. Dann [AL]  $T \vdash_{\mathbb{K}}^{\mathcal{L}} ((\neg\varphi \rightarrow \perp) \rightarrow \varphi)$ , also  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi$  mit [MP].

(b)  $(\varphi \rightarrow (\neg\varphi \rightarrow \perp))$  ist eine  $\mathcal{L}$ -Tautologie. Aus den Voraussetzungen folgt also  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \perp$  durch zweimalige Anwendung von [MP].  $\square$

Ziel dieses Abschnittes ist der Beweis des Vollständigkeitssatzes:

**Satz 2.5.3 (Vollständigkeitssatz [completeness theorem] für  $\mathbb{K}$ , Gödel 1929<sup>39</sup>)**

Eine  $\mathcal{L}$ -Theorie  $T$  beweist genau dann eine  $\mathcal{L}$ -Formel  $\varphi$  in  $\mathbb{K}$ , wenn  $\varphi$  aus  $T$  logisch folgt:

$$T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi \iff T \models \varphi$$

Mit  $\varphi = \perp$  gilt insbesondere:  $T$  ist genau dann  $\mathbb{K}$ -widerspruchsfrei, wenn  $T$  konsistent ist (also ein Modell hat); und mit  $T = \emptyset$  gilt insbesondere:  $\varphi$  ist genau dann  $\mathbb{K}$ -beweisbar, wenn  $\varphi$  allgemeingültig ist.

Genau genommen sagt die (nicht triviale) Richtung „ $\Leftarrow$ “ die *Vollständigkeit* des Kalküls  $\mathbb{K}$  aus und die Richtung „ $\Rightarrow$ “ die *Korrektheit* oder *soundness* von  $\mathbb{K}$ . Der Beweis des Vollständigkeitsatzes ist nicht konstruktiv: Er liefert kein Verfahren dafür, einen  $\mathbb{K}$ -Beweis zu finden (das kann beliebig schwer werden). Der Vollständigkeitsatz liefert auch kein Entscheidungsverfahren für die Frage, ob eine  $\mathcal{L}$ -Formel  $\varphi$  aus einer  $\mathcal{L}$ -Theorie  $T$  folgt, oder für die Frage, ob eine  $\mathcal{L}$ -Formel  $\varphi$  allgemeingültig ist. Ein  $\mathbb{K}$ -Beweis beantwortet zwar positiv diese Fragen; sie werden aber nicht dadurch negativ beantwortet, dass man keinen  $\mathbb{K}$ -Beweis hat, sondern man müsste dafür beweisen, dass es keinen  $\mathbb{K}$ -Beweis gibt (was im Allgemeinen schwer ist).

Es gibt sehr viele Varianten von Kalkülen, für die man den Vollständigkeitsatz beweisen kann. Wichtig in Hinblick auf die Folgerungen ist jeweils, dass Axiome maschinell überprüft und Regeln maschinell angewandt werden können. Bekannte Arten sind *Baum-* oder *Tableaukalküle*, *Sequenzkalküle* und *Kalküle des natürlichen Schließens*. Oft wird zwischen *Axiomen-* oder *Hilbert-Kalkülen* mit vielen Axiomen und wenigen Regeln (meist nur *modus ponens*) und *Regel-* oder *Gentzen-Kalkülen* mit wenigen Axiomen und vielen Regeln unterschieden.  $\mathbb{K}$  ist eine pragmatische, für den Beweis des Vollständigkeitsatzes optimierte Mischung und im Wesentlichen von Martin Ziegler übernommen.

BEWEIS DER RICHTUNG „ $\Rightarrow$ “ VON SATZ 2.5.3: Dazu muss man zeigen, dass alle  $\mathbb{K}$ -Axiome allgemeingültig und alle  $\mathbb{K}$ -Regeln korrekt sind, also logische Folgerungen in logische Folgerungen überführen. Das wurde im Wesentlichen im vorherigen Abschnitt gezeigt: Man muss lediglich für die Ableitungsregeln die Argumentation im Beweis von Satz 2.4.7 auf Modelle von  $T$  einschränken.  $\square$

Der Beweis der anderen Richtung wird den restlichen Abschnitt in Anspruch nehmen.

**Lemma 2.5.4** Sei  $C$  eine zu  $\mathcal{L}$  disjunkte Menge „neuer“ Konstanten und  $\mathcal{L}_C = \mathcal{L} \cup C$ .

Sei  $T$  eine  $\mathcal{L}$ -Theorie,  $\varphi(v_1, \dots, v_n)$  eine  $\mathcal{L}$ -Formel und seien  $c_1, \dots, c_n \in C$  paarweise verschieden. Dann gilt:

$$T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi \iff T \vdash_{\mathbb{K}}^{\mathcal{L}} \forall v_1 \dots \forall v_n \varphi \iff T \vdash_{\mathbb{K}}^{\mathcal{L}_C} \varphi \left[ \frac{c_1}{v_1} \dots \frac{c_n}{v_n} \right]$$

Diese Äquivalenz gilt auch für logische Folgerung  $\models$  anstelle von  $\mathbb{K}$ -Beweisbarkeit  $\vdash_{\mathbb{K}}$ !

BEWEIS: Aus  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi$  folgt mit  $[\forall$ -Einführung]  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \forall v_n \varphi$  und sukzessive  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \forall v_1 \dots \forall v_n \varphi$ .

Ein  $\mathbb{K}$ -Beweis dafür ist auch ein Beweis in  $\mathcal{L}_C$ , also gilt auch  $T \vdash_{\mathbb{K}}^{\mathcal{L}_C} \forall v_1 \dots \forall v_n \varphi$ . Mit dem  $[\forall$ -Axiom]  $T \vdash_{\mathbb{K}}^{\mathcal{L}_C} (\forall v_1 \dots \forall v_n \varphi \rightarrow \forall v_2 \dots \forall v_n \varphi \left[ \frac{c_1}{v_1} \right])$  und *modus ponens* erhält man daraus  $T \vdash_{\mathbb{K}}^{\mathcal{L}_C} \forall v_2 \dots \forall v_n \varphi \left[ \frac{c_1}{v_1} \right]$ , und sukzessive  $T \vdash_{\mathbb{K}}^{\mathcal{L}_C} \varphi \left[ \frac{c_1}{v_1} \dots \frac{c_n}{v_n} \right]$ .

Ersetzt man nun in dem  $\mathbb{K}$ -Beweis für diese Ableitung jedes  $c_i$  durch eine bisher nicht vorkommende Variable, beispielsweise  $c_i$  durch  $v_{i+k}$  für ein sehr großes  $k$ , dann erhält man daraus einen  $\mathbb{K}$ -Beweis für  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi \left[ \frac{v_{1+k}}{v_1} \dots \frac{v_{n+k}}{v_n} \right]$ . Der gleiche Ableitungsweg wie eben erlaubt es nun, daraus zunächst auf  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \forall v_{1+k} \dots \forall v_{n+k} \varphi \left[ \frac{v_{1+k}}{v_1} \dots \frac{v_{n+k}}{v_n} \right]$  zu schließen, und schließlich durch „Re-Substitution“ von  $v_{i+k}$  durch  $v_i$  über die entsprechenden  $[\forall$ -Axiome] wieder auf  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi(v_1, \dots, v_n)$ . Im Ringschluss sind also die drei  $\mathbb{K}$ -Beweisbarkeiten äquivalent.  $\square$

**Folgerung 2.5.5** Man kann sich im Beweis des Vollständigkeitsatzes auf  $\mathcal{L}$ -Aussagen  $\varphi$  beschränken.

<sup>39</sup>Gödel hat 1929 die Vollständigkeit eines Kalküls gezeigt, der aber nicht genau wie  $\mathbb{K}$  aussieht. Die Vollständigkeit überhaupt eines Kalküls für die Prädikatenlogik zu zeigen, war eine bedeutende Leistung. Daher spricht man auch bei anderen Kalkülen gerne vom *Gödel'schen Vollständigkeitsatz*.

Zum Beweis des Vollständigkeitsatzes konstruiert man für eine  $\mathbb{K}$ -widerspruchsfreie  $\mathcal{L}$ -Theorie ein Modell. Das macht man am besten über die sogenannte *Henkin-Konstruktion*, deren Idee darin besteht, die Sprache  $\mathcal{L}$  durch Konstanten so anzureichern, dass im Wesentlichen die  $\mathcal{L}$ -Terme das Universum des Modells bilden. Dazu wird die Theorie in zwei Schritten erweitert: Zunächst zu einer sogenannten *Henkin-Theorie* und dann zu einer *vollständigen Theorie*, jedesmal ohne die  $\mathbb{K}$ -Widerspruchsfreiheit zu verlieren.

**Definition 2.5.6** Für jede  $\mathcal{L}$ -Formel  $\varphi(v_i)$  sei  $c_\varphi \notin \mathcal{L}$  eine neue Konstante. Dann sei

$$\mathcal{L}^* := \mathcal{L} \cup \{c_\varphi \mid \varphi(v_i) \text{ } \mathcal{L}\text{-Formel}\} \quad \text{und} \quad T^* := T \cup \{(\exists v_i \varphi \rightarrow \varphi[\frac{c_\varphi}{v_i}]) \mid \varphi(v_i) \text{ } \mathcal{L}\text{-Formel}\}.$$

Die Henkin-Theorie von  $T$  ist die  $\mathcal{L}_C$ -Theorie  $T_C$ , wobei

$$\mathcal{L}_C := \mathcal{L} \cup \mathcal{L}^* \cup \mathcal{L}^{**} \cup \dots \quad \text{und} \quad T_C := T \cup T^* \cup T^{**} \cup \dots$$

Die Konstanten  $c_\varphi$  heißen *Henkin-Konstanten*, die Formeln  $(\exists v_i \varphi \rightarrow \varphi[\frac{c_\varphi}{v_i}])$  *Henkin-Axiome*.

**Lemma 2.5.7**

Die Henkin-Theorie einer  $\mathbb{K}$ -widerspruchsfreien  $\mathcal{L}$ -Theorie ist  $\mathbb{K}$ -widerspruchsfrei.

Dieses und das folgende Lemma beweise ich (ähnlich wie den Kompaktheitssatz der Aussagenlogik) nur für den *abzählbaren Fall*, also für endlich viele oder höchstens abzählbar unendlich viele Funktions- und Relationszeichen. Dann ist auch auch die Menge aller  $\mathcal{L}$ -Formeln abzählbar.<sup>40</sup>

BEWEIS: Ich zeige zunächst, dass  $\mathbb{K}$ -Widerspruchsfreiheit bei Hinzunahme eines einzelnen Henkin-Axioms erhalten bleibt. Angenommen

$$T \cup \{(\exists v_i \varphi \rightarrow \varphi[\frac{c_\varphi}{v_i}])\} \vdash_{\mathbb{K}}^{\mathcal{L}_C} \perp$$

für eine  $\mathcal{L}$ -Formel  $\varphi(v_i)$  mit Henkin-Konstante  $c_\varphi \in C$ . Ähnlich wie im Beweis von Lemma 2.5.2 folgt mit  $[\rightarrow\text{-Einführung}]$ , [AL] und [MP] daraus

$$T \vdash_{\mathbb{K}}^{\mathcal{L}_C} \exists v_i \varphi \quad \text{und} \quad T \vdash_{\mathbb{K}}^{\mathcal{L}_C} \neg \varphi[\frac{c_\varphi}{v_i}]$$

denn  $((A_0 \rightarrow A_1) \rightarrow \perp) \rightarrow A_0$  und  $((A_0 \rightarrow A_1) \rightarrow \perp) \rightarrow \neg A_1$  sind Tautologien. Mit Lemma 2.5.4 kann man die Konstanten eliminieren und erhält  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \exists v_i \varphi = \neg \forall v_i \neg \varphi$  und  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \neg \varphi$ . Aus letzterem bekommt man mit  $[\forall\text{-Einführung}]$   $T \vdash_{\mathbb{K}}^{\mathcal{L}} \forall v_i \neg \varphi$ , also ist  $T$  nach Lemma 2.5.2 (b)  $\mathbb{K}$ -widersprüchlich: Widerspruch!

Nun ist  $T^*$  aufsteigende Vereinigung einer Kette  $T = T_0 \subseteq T_1 \subseteq T_2 \subseteq \dots$  von  $\mathbb{K}$ -widerspruchsfreien Theorien, die sukzessive durch Hinzunahme der abzählbar vielen Henkin-Axiome entstehen.  $T^*$  ist dann ebenfalls  $\mathbb{K}$ -widerspruchsfrei, weil in einem  $\mathbb{K}$ -Beweis von  $\perp$  nur endlich viele Formeln vorkämen, die bereits in einem  $T_n$  lägen: Dann wäre schon  $T_n$   $\mathbb{K}$ -widersprüchlich! Mit dem gleichen Argument ist dann auch die Henkin-Theorie  $T_C$   $\mathbb{K}$ -widerspruchsfrei als Vereinigung der Kette  $\mathbb{K}$ -widerspruchsfreier Theorien  $T \subseteq T^* \subseteq T^{**} \subseteq \dots$ .  $\square$

Man kann also für den Beweis des Vollständigkeitsatzes ohne Beschränkung der Allgemeinheit annehmen, dass man eine  $\mathbb{K}$ -widerspruchsfreie  $\mathcal{L}$ -Theorie hat, die eine Henkin-Theorie ist.

<sup>40</sup>Das Alphabet ist abzählbar, z. B. erst alle Junktoren, Quantoren, Gleichheitszeichen und Klammern, und dann abwechselnd Individuenvariablen, Funktions- und Relationszeichen:  $v_0, f_0, R_0, v_1, \dots$ . Die Menge aller endlichen Folgen dieser Symbole ist mit dem Cantor'schen Diagonalverfahren ebenfalls abzählbar, und damit auch die Menge aller Formeln als Teilmenge davon.

**Definition 2.5.8** Eine  $\mathcal{L}$ -Theorie heißt **vollständig** [complete], falls für jede  $\mathcal{L}$ -Aussage  $\varphi$  entweder  $\varphi \in T$  oder  $\neg\varphi \in T$ .

Falls  $\mathcal{M}$  eine  $\mathcal{L}$ -Struktur ist, dann ist  $\text{Th}(\mathcal{M}) := \{\varphi \text{ } \mathcal{L}\text{-Aussage} \mid \mathcal{M} \models \varphi\}$  eine vollständige  $\mathcal{L}$ -Theorie. (Umgekehrt kann man zeigen, dass jede vollständige  $\mathcal{L}$ -Theorie, die unter logischer Äquivalenz und Konjunktionen abgeschlossen ist und  $\top$  enthält, von dieser Form ist.)

**Lemma 2.5.9** Jede  $\mathbb{K}$ -widerspruchsfreie  $\mathcal{L}$ -Theorie lässt sich zu einer vollständigen  $\mathbb{K}$ -widerspruchsfreien  $\mathcal{L}$ -Theorie erweitern.

Der Beweis läuft ganz analog zum Beweis des Kompaktheitssatzes der Aussagenlogik!

BEWEIS: Sei  $\{\varphi_i \mid i \in \mathbb{N}\}$  eine Aufzählung aller  $\mathcal{L}$ -Formeln. Man definiert induktiv

$$T_0 := T \quad \text{und} \quad T_{n+1} := \begin{cases} T_n \cup \{\varphi_n\} & \text{falls } \mathbb{K}\text{-widerspruchsfrei,} \\ T_n \cup \{\neg\varphi_n\} & \text{andernfalls.} \end{cases}$$

Nun ist  $T_{n+1}$  in jedem Fall  $\mathbb{K}$ -widerspruchsfrei. Denn sonst hätte man sowohl  $T_n \cup \{\varphi_n\} \vdash_{\mathbb{K}}^{\mathcal{L}} \perp$  als auch  $T_n \cup \{\neg\varphi_n\} \vdash_{\mathbb{K}}^{\mathcal{L}} \perp$ , also mit  $[\rightarrow\text{-Einführung}]$   $T_n \vdash_{\mathbb{K}}^{\mathcal{L}} (\varphi_n \rightarrow \perp)$  und  $T_n \vdash_{\mathbb{K}}^{\mathcal{L}} (\neg\varphi_n \rightarrow \perp)$ . Mit der  $\mathcal{L}$ -Tautologie  $((\varphi_n \rightarrow \perp) \rightarrow ((\neg\varphi_n \rightarrow \perp) \rightarrow \perp))$  und zweimaligem *modus ponens* ergäbe dies  $T_n \vdash_{\mathbb{K}}^{\mathcal{L}} \perp$ : Widerspruch!

Dann ist, wie im Beweis von Lemma 2.5.7, auch  $T_\infty := \bigcup_{n \in \mathbb{N}} T_n$   $\mathbb{K}$ -widerspruchsfrei. Schließlich ist  $T_\infty$  auch vollständig, denn per Konstruktion gilt  $\varphi \in T_\infty$  oder  $\neg\varphi \in T_\infty$  für jede  $\mathcal{L}$ -Formel  $\varphi$ . Würde beides gelten, hätte man mit Lemma 2.5.2 (b) einen Widerspruch!  $\square$

Bei der Vervollständigung einer Henkin-Theorie bleibt die Eigenschaft erhalten, dass die Theorie für jedes  $\varphi(v_i)$  ein Henkin-Axiom enthält. Man kann also für den Beweis des Vollständigkeitssatzes ohne Beschränkung der Allgemeinheit annehmen, dass man eine vollständige  $\mathbb{K}$ -widerspruchsfreie  $\mathcal{L}$ -Theorie hat, die eine Henkin-Theorie ist.

**Satz 2.5.10** Jede vollständige,  $\mathbb{K}$ -widerspruchsfreie Henkin-Theorie hat ein Modell.

Daraus folgt die „ $\Leftarrow$ “-Richtung des Vollständigkeitssatzes 2.5.3:

Falls  $T \not\vdash_{\mathbb{K}}^{\mathcal{L}} \varphi$ , ist  $T \cup \{\neg\varphi\}$  nach Lemma 2.5.2 nicht  $\mathbb{K}$ -widersprüchlich und kann nach Lemma 2.5.7 und Lemma 2.5.9 zu einer vollständigen  $\mathbb{K}$ -widerspruchsfreien Henkin- $\mathcal{L}_C$ -Theorie  $T_\infty$  erweitert werden.  $T_\infty$  hat nach dem noch zu beweisenden Satz eine  $\mathcal{L}_C$ -Struktur  $\mathcal{M}$  als Modell. Indem man die Interpretationen der Henkin-Konstanten „vergisst“, wird daraus eine  $\mathcal{L}$ -Struktur  $\mathcal{M}^-$ , die Modell der  $\mathcal{L}$ -Theorie  $T \cup \{\neg\varphi\}$  bleibt und somit  $T \not\models \varphi$  zeigt.

BEWEIS VON SATZ 2.5.10: Sei  $T$  die vollständige,  $\mathbb{K}$ -widerspruchsfreie Henkin-Theorie in der Sprache  $\mathcal{L}$ . Auf der Menge der  $\mathcal{L}$ -Terme ist eine Relation definiert durch:

$$\tau_1 \approx \tau_2 : \iff T \vdash_{\mathbb{K}}^{\mathcal{L}} \tau_1 \doteq \tau_2$$

$\approx$

**Lemma 2.5.11**  $\approx$  ist ein Äquivalenzrelation.

BEWEIS: Das Lemma folgt daraus, dass die ersten drei Gleichheitsgesetze  $\mathbb{K}$ -Axiome sind.

Ich zeige nur beispielhaft, aber im Detail, die Transitivität. Sei dazu  $\tau_0 \approx \tau_1$  und  $\tau_1 \approx \tau_2$ , also  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \tau_0 \doteq \tau_1$  und  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \tau_1 \doteq \tau_2$ , und seien  $j$  und  $k$  so groß, dass  $v_j, v_k$  nicht in  $\tau_0, \tau_1$  vorkommen. Dann sind in  $T$   $\mathbb{K}$ -beweisbar:

$$\begin{aligned}
[\dot{=} \text{-Ax}] \quad & \forall v_i \forall v_j \forall v_k ((v_i \dot{=} v_j \wedge v_j \dot{=} v_k) \rightarrow v_i \dot{=} v_k) \\
[\forall \text{-Ax}] \quad & (\forall v_i \forall v_j \forall v_k ((v_i \dot{=} v_j \wedge v_j \dot{=} v_k) \rightarrow v_i \dot{=} v_k) \rightarrow \forall v_j \forall v_k ((\tau_0 \dot{=} v_j \wedge v_j \dot{=} v_k) \rightarrow \tau_0 \dot{=} v_k)) \\
[\text{MP}] \quad & \forall v_j \forall v_k ((\tau_0 \dot{=} v_j \wedge v_j \dot{=} v_k) \rightarrow \tau_0 \dot{=} v_k) \\
[\forall \text{-Ax}] \quad & (\forall v_j \forall v_k ((\tau_0 \dot{=} v_j \wedge v_j \dot{=} v_k) \rightarrow \tau_0 \dot{=} v_k) \rightarrow \forall v_k ((\tau_0 \dot{=} \tau_1 \wedge \tau_1 \dot{=} v_k) \rightarrow \tau_0 \dot{=} v_k)) \\
[\text{MP}] \quad & \forall v_k ((\tau_0 \dot{=} \tau_1 \wedge \tau_1 \dot{=} v_k) \rightarrow \tau_0 \dot{=} v_k) \\
[\forall \text{-Ax}] \quad & (\forall v_k ((\tau_0 \dot{=} \tau_1 \wedge \tau_1 \dot{=} v_k) \rightarrow \tau_0 \dot{=} v_k) \rightarrow ((\tau_0 \dot{=} \tau_1 \wedge \tau_1 \dot{=} \tau_2) \rightarrow \tau_0 \dot{=} \tau_2)) \\
[\text{MP}] \quad & ((\tau_0 \dot{=} \tau_1 \wedge \tau_1 \dot{=} \tau_2) \rightarrow \tau_0 \dot{=} \tau_2) \\
[\text{AL}] \quad & (\tau_0 \dot{=} \tau_1 \rightarrow (((\tau_0 \dot{=} \tau_1 \wedge \tau_1 \dot{=} \tau_2) \rightarrow \tau_0 \dot{=} \tau_2) \rightarrow (\tau_1 \dot{=} \tau_2 \rightarrow \tau_0 \dot{=} \tau_2))) \\
[\text{MP}] \quad & (((\tau_0 \dot{=} \tau_1 \wedge \tau_1 \dot{=} \tau_2) \rightarrow \tau_0 \dot{=} \tau_2) \rightarrow (\tau_1 \dot{=} \tau_2 \rightarrow \tau_0 \dot{=} \tau_2)) \\
[\text{MP}] \quad & (\tau_1 \dot{=} \tau_2 \rightarrow \tau_0 \dot{=} \tau_2) \\
[\text{MP}] \quad & \tau_0 \dot{=} \tau_2 \quad \square
\end{aligned}$$

Sei  $M$  nun die Menge der  $\approx$ -Äquivalenzklassen von  $\mathcal{L}$ -Termen. Ich schreibe  $\tau/\approx$  oder  $\tilde{\tau}$  für die Äquivalenzklasse von  $\tau$ . Auf  $M$  wird folgendermaßen eine  $\mathcal{L}$ -Struktur  $\mathcal{M}$  definiert:

$$\begin{aligned}
f^{\mathcal{M}}(\tilde{\tau}_1, \dots, \tilde{\tau}_n) & := f\tau_1 \dots \tau_n / \approx \\
(\tilde{\tau}_1, \dots, \tilde{\tau}_n) \in R^{\mathcal{M}} & : \iff T \vdash_{\mathbb{K}}^{\mathcal{L}} R\tau_1 \dots \tau_n
\end{aligned}$$

**Lemma 2.5.12** Die Struktur  $\mathcal{M}$  ist wohldefiniert.

BEWEIS: Zu zeigen ist also: Wenn  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \tau_1 \dot{=} \tau'_1, \dots, T \vdash_{\mathbb{K}}^{\mathcal{L}} \tau_n \dot{=} \tau'_n$ , dann gilt

$$T \vdash_{\mathbb{K}}^{\mathcal{L}} f\tau_1 \dots \tau_n \dot{=} f\tau'_1 \dots \tau'_n \quad \text{und} \quad T \vdash_{\mathbb{K}}^{\mathcal{L}} (R\tau_1 \dots \tau_n \leftrightarrow R\tau'_1 \dots \tau'_n).$$

Das folgt wie in Lemma 2.5.11 aus [ $\dot{=}$ -Axiom], nun allerdings aus „Kongruenz“.  $\square$

**Lemma 2.5.13** In der Äquivalenzklasse  $\tilde{\tau}$  jedes Terms  $\tau(v_0, \dots, v_{n-1})$  liegt eine Konstante.

BEWEIS: Man nimmt dafür die Henkin-Konstante  $c_\varphi$  der  $\mathcal{L}$ -Formel  $\varphi = \exists v_n \tau \dot{=} v_n$ . Das zugehörige Henkin-Axiom  $(\exists v_n \tau \dot{=} v_n \rightarrow \tau \dot{=} c_\varphi)$  liegt in  $T$ . Mit [MP] reicht es daher zu zeigen, dass  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \exists v_n \tau \dot{=} v_n = \neg \forall v_n \neg \tau \dot{=} v_n$ . Andernfalls ist wegen der Vollständigkeit  $\forall v_n \neg \tau \dot{=} v_n$  in  $T$ . Aus dem passenden [ $\forall$ -Axiom] und [MP] bekommt man daraus  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \neg \tau \dot{=} \tau$ ; aus [ $\dot{=}$ -Axiom]  $\forall v_n v_n \dot{=} v_n$  folgt aber auf gleiche Weise  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \tau \dot{=} \tau$ : Widerspruch!  $\square$

$$\text{Sei } \beta_{\approx} : \{v_i \mid i \in \mathbb{N}\} \rightarrow M \text{ die Belegung mit } \beta_{\approx}(v_i) = \tilde{v}_i$$

**Lemma 2.5.14** Für jeden  $\mathcal{L}$ -Term  $\tau$  gilt  $\beta_{\approx}(\tau) = \tilde{\tau}$  ( $= \tau/\approx$ ).

BEWEIS: Dazu reicht der Induktionsschritt

$$\beta_{\approx}(f\tau_1 \dots \tau_n) = f^{\mathcal{M}}(\beta_{\approx}(\tau_1), \dots, \beta_{\approx}(\tau_n)) = f^{\mathcal{M}}(\tilde{\tau}_1, \dots, \tilde{\tau}_n) = f\tau_1 \dots \tau_n / \approx \quad \square$$

**Satz 2.5.15**  $\mathcal{M}$  ist ein Modell von  $T$ . Allgemeiner gilt für jede  $\mathcal{L}$ -Formel  $\varphi$ :

$$(\mathcal{M}, \beta_{\approx}) \models \varphi \iff T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi$$



BEWEIS: Per Induktion über den Aufbau der  $\mathcal{L}$ -Formeln:

$T \vdash_{\mathbb{K}}^{\mathcal{L}} \top$  gilt wegen [AL] und  $T \not\vdash_{\mathbb{K}}^{\mathcal{L}} \perp$ , da  $T$  nach Voraussetzung  $\mathbb{K}$ -widerspruchsfrei ist. Für die weiteren atomaren  $\mathcal{L}$ -Formeln gilt die Aussage nach Definition von  $\approx$  bzw.  $\mathcal{M}$ , denn:

$$\begin{aligned} (\mathcal{M}, \beta_{\approx}) \models \tau_1 \doteq \tau_2 &\iff \beta_{\approx}(\tau_1) = \beta_{\approx}(\tau_2) \iff \tilde{\tau}_1 = \tilde{\tau}_2 \iff T \vdash_{\mathbb{K}}^{\mathcal{L}} \tau_1 \doteq \tau_2 \\ (\mathcal{M}, \beta_{\approx}) \models R\tau_1 \dots, \tau_n &\iff (\beta_{\approx}(\tau_1), \dots, \beta_{\approx}(\tau_n)) \in R^{\mathcal{M}} \\ &\iff (\tilde{\tau}_1, \dots, \tilde{\tau}_n) \in R^{\mathcal{M}} \iff T \vdash_{\mathbb{K}}^{\mathcal{L}} R\tau_1 \dots, \tau_n \end{aligned}$$

Für die Negation hat man per Induktion und da  $T$  vollständig ist:

$$(\mathcal{M}, \beta_{\approx}) \models \neg\varphi \iff (\mathcal{M}, \beta_{\approx}) \not\models \varphi \iff T \not\vdash_{\mathbb{K}}^{\mathcal{L}} \varphi \iff T \vdash_{\mathbb{K}}^{\mathcal{L}} \neg\varphi$$

Für die Konjunktion hat man per Induktion zunächst:

$$(\mathcal{M}, \beta_{\approx}) \models (\varphi \wedge \psi) \iff ((\mathcal{M}, \beta_{\approx}) \models \varphi \text{ und } (\mathcal{M}, \beta_{\approx}) \models \psi) \iff (T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi \text{ und } T \vdash_{\mathbb{K}}^{\mathcal{L}} \psi)$$

Daraus und aus der  $\mathcal{L}$ -Tautologie  $(\varphi \rightarrow (\psi \rightarrow (\varphi \wedge \psi)))$  erhält man  $T \vdash_{\mathbb{K}}^{\mathcal{L}} (\varphi \wedge \psi)$  durch zweimalige [MP]-Anwendung. Die Umkehrung folgt ebenfalls mit [MP] und den  $\mathcal{L}$ -Tautologien  $((\varphi \wedge \psi) \rightarrow \varphi)$  und  $((\varphi \wedge \psi) \rightarrow \psi)$ .

Bleibt der Quantorenschritt:

$$\begin{aligned} (\mathcal{M}, \beta_{\approx}) \models \forall v_i \varphi &\iff \text{für alle } \tilde{\tau} \in M \text{ gilt } (\mathcal{M}, \beta_{\approx} \frac{\tilde{\tau}}{v_i}) \models \varphi \\ \text{Lemma 2.5.13} &\iff \text{für alle Konstanten } c \in \mathcal{L} \text{ gilt } (\mathcal{M}, \beta_{\approx} \frac{c}{v_i}) \models \varphi \\ \text{Substitutionslemma} &\iff \text{für alle Konstanten } c \in \mathcal{L} \text{ gilt } (\mathcal{M}, \beta_{\approx}) \models \varphi \left[ \frac{c}{v_i} \right] \\ \text{Induktion} &\iff \text{für alle Konstanten } c \in \mathcal{L} \text{ gilt } T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi \left[ \frac{c}{v_i} \right] \end{aligned}$$

Aus  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \forall v_i \varphi$  folgt  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \varphi \left[ \frac{c}{v_i} \right]$  für jede Konstante  $c$  durch das passende [ $\forall$ -Axiom] und [MP], also nach der gerade gezeigten Äquivalenz  $(\mathcal{M}, \beta_{\approx}) \models \forall v_i \varphi$ .

Aus  $T \not\vdash_{\mathbb{K}}^{\mathcal{L}} \forall v_i \varphi = \neg \exists v_i \neg \varphi$  folgt wegen der Vollständigkeit  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \exists v_i \neg \varphi$ . Mit dem Henkin-Axiom  $(\exists v_i \neg \varphi \rightarrow \neg \varphi \left[ \frac{c \neg \varphi}{v_i} \right]) \in T$  und [MP] bekommt man daraus  $T \vdash_{\mathbb{K}}^{\mathcal{L}} \neg \varphi \left[ \frac{c \neg \varphi}{v_i} \right]$ , also nach der obigen Äquivalenz  $(\mathcal{M}, \beta_{\approx}) \not\models \forall v_i \varphi$ .  $\square \square$

### Folgerung 2.5.16 (Kompaktheitssatz für die Prädikatenlogik)

Eine unendliche Menge von  $\mathcal{L}$ -Formeln ist genau dann erfüllbar, wenn sie endlich erfüllbar ist.

BEWEIS: „ $\Rightarrow$ “ ist trivial, denn ein Modell einer unendlichen Formelmengens ist auch Modell jeder endlichen Teilmenge. Umgekehrt kann man sich zunächst auf den Fall von  $\mathcal{L}$ -Aussagen zurückziehen, indem man freie Variablen durch neue Konstanten ersetzt; dies ändert nichts an der (Nicht-)Erfüllbarkeit. Eine inkonsistente  $\mathcal{L}$ -Theorie  $T$  ist nach dem Vollständigkeitssatz  $\mathbb{K}$ -widersprüchlich; in einem  $\mathbb{K}$ -Beweis von  $\perp$  kommen aber nur endlich viele  $\mathcal{L}$ -Aussagen aus  $T$  vor. Die endliche Teilmenge von  $T$ , die aus diesen  $\mathcal{L}$ -Aussagen besteht, ist dann bereits  $\mathbb{K}$ -widersprüchlich und nach dem Vollständigkeitsatz auch nicht erfüllbar.  $\square$

Eine Anwendung des Kompaktheitssatzes findet sich im Beweis des Satzes von Herbrand 2.6.4.

## 2.6 Erfüllbarkeit und Allgemeingültigkeit

Entsprechend wie in der Aussagenlogik lassen sich typische Fragen der Prädikatenlogik als Erfüllbarkeitsprobleme bzw. Allgemeingültigkeitsprobleme formalisieren. Darunter fallen alle offenen Vermutungen der Mathematik/Theoretischen Informatik. Zum Beispiel kann man die Goldbach'sche Vermutung, dass jede gerade Zahl ab 4 Summe von zwei Primzahlen ist, als Erfüllbarkeitsproblem formalisieren: Man arbeitet in der Sprache  $\mathcal{L} = \{+, \cdot, 0, 1, \leq\}$ , formuliert in  $\mathcal{L}$ -Formeln  $\gamma(v_0)$  und  $\pi(v_0)$  die Eigenschaften, gerade bzw. Primzahl zu sein, und fragt, ob die  $\mathcal{L}$ -Aussage

$$\neg \forall v_0 ((\gamma(v_0) \wedge 1 + 1 + 1 + 1 \leq v_0) \rightarrow \exists v_1 \exists v_2 (\pi(v_1) \wedge \pi(v_2) \wedge v_0 \doteq v_1 + v_2))$$

zusammen mit  $\text{Th}(\mathcal{N})$  für die  $\mathcal{L}$ -Struktur  $\mathcal{N} = (\mathbb{N}; +^{\mathbb{N}}, \cdot^{\mathbb{N}}, 0^{\mathbb{N}}, 1^{\mathbb{N}}, \leq^{\mathbb{N}})$  – oder einer geeigneten Axiomatisierung davon – erfüllbar ist. Auch schwerer zu formulierende, aber möglicherweise relevantere Fragen wie  $P = NP$  lassen sich (allerdings mit hohem Aufwand) so formalisieren.

Zur Erinnerung:  $\varphi$  ist genau dann erfüllbar, wenn  $\neg\varphi$  nicht allgemeingültig ist. Eine  $\mathcal{L}$ -Formel  $\psi(v_{i_1}, \dots, v_{i_n})$  ist genau dann allgemeingültig, wenn die *universell* abquantifizierte  $\mathcal{L}$ -Aussage  $\forall v_{i_1} \dots \forall v_{i_n} \psi$  allgemeingültig ist. Also ist eine  $\mathcal{L}$ -Formel  $\varphi(v_{i_1}, \dots, v_{i_n})$  genau dann erfüllbar, wenn die *existentiell* abquantifizierte  $\mathcal{L}$ -Aussage  $\exists v_{i_1} \dots \exists v_{i_n} \varphi$  erfüllbar ist. Man kann sich also für beide Fragen immer auf  $\mathcal{L}$ -Aussagen zurückziehen!

In diesem Abschnitt sollen die bereits besprochenen Methoden, an solche Fragen heranzugehen, kurz wiederholt und einige weiteren Methoden vorgestellt werden. In den letzten Jahren haben sogenannte *automatische Beweiser* wie z. B. Lean, die zum Teil auf diesen Methoden beruhen, durch beachtliche Erfolge neue Aufmerksamkeit erregt.

### 2.6.1 Semantische Beweise und Beispiele

Die Erfüllbarkeit einer  $\mathcal{L}$ -Formel  $\varphi$  kann man versuchen dadurch zeigen, dass man ein Modell angibt, also eine  $\mathcal{L}$ -Struktur  $\mathcal{M}$  und eine Belegung  $\beta$  mit  $(\mathcal{M}, \beta) \models \varphi$ . Die Allgemeingültigkeit einer  $\mathcal{L}$ -Formel  $\varphi$  kann man versuchen dadurch zu beweisen, dass man anhand der Definition zeigt, dass jede  $\mathcal{L}$ -Struktur  $\mathcal{M}$  mit jeder Belegung  $\beta$  ein Modell von  $\varphi$  ist.

(a) Sei  $\mathcal{L} = \{R_0, R_1\}$  mit zwei Prädikaten. Um die Erfüllbarkeit der  $\mathcal{L}$ -Aussage

$$\varphi_1 = ((\exists v_0 R_0 v_0 \wedge \exists v_0 R_1 v_0) \rightarrow \exists v_0 (R_0 v_0 \wedge R_1 v_0))$$

zu zeigen, reicht es eine  $\mathcal{L}$ -Struktur anzugeben, in der  $\varphi_1$  gilt. Das ist z. B. eine einelementige Struktur  $\mathcal{M}$  mit  $R_0^{\mathcal{M}} = R_1^{\mathcal{M}} = M$ .

(b) Ein Modell zu finden, kann aber schwerer sein, als es zunächst aussieht. Sei zum Beispiel  $\varrho_n$  die  $\{E\}$ -Aussage, die die Existenz von Graphen mit mindestens  $n$  Knoten beschreibt, die keine *5-Clique* und keine *5-Anticlique* enthalten ( $E$  zweistelliges Relationszeichen):

$$\varrho_n = \left( \forall v_0 \forall v_1 (E v_0 v_1 \leftrightarrow E v_1 v_0) \wedge \forall v_0 \neg E v_0 v_0 \wedge \right. \\ \left. \neg \exists v_0 \dots \exists v_4 \left( \bigwedge_{0 \leq i < j \leq 5} E v_i v_j \vee \bigwedge_{0 \leq i < j \leq 5} \neg E v_i v_j \right) \wedge \exists v_0 \dots \exists v_{n-1} \bigwedge_{0 \leq i < j \leq n-1} \neg v_i \doteq v_j \right)$$

Stand 2021 weiß man, dass alle  $\varrho_n$  bis  $n = 42$  erfüllbar sind und alle  $\varrho_n$  ab  $n = 48$  nicht erfüllbar ist, aber nichts über die Werte dazwischen, obwohl es sich um überschaubare Zahlen handelt.<sup>41</sup> Für  $n = 43$  gibt es aber  $2^{\binom{43}{2}} = 2^{903} \approx 10^{301}$  Graphen, die man anschauen müsste.

(c) Sei  $\mathcal{L}$  wie oben. Um die Allgemeingültigkeit der  $\mathcal{L}$ -Aussage

$$\varphi_2 = (\forall v_0 (R_0 v_0 \wedge R_1 v_0) \rightarrow (\forall v_0 R_0 v_0 \wedge \forall v_0 R_1 v_0))$$

zu zeigen, nimmt man eine beliebige  $\mathcal{L}$ -Struktur  $\mathcal{M}$  und Belegung  $\beta$ . Dann gilt:

$$\begin{aligned} (\mathcal{M}, \beta) \models \forall v_0 (R_0 v_0 \wedge R_1 v_0) & \\ \iff \text{für alle } m \in M \text{ gilt } (\mathcal{M}, \beta \frac{m}{v_0}) \models (R_0 v_0 \wedge R_1 v_0) & \\ \iff \text{für alle } m \in M \text{ gilt } (\mathcal{M}, \beta \frac{m}{v_0}) \models R_0 v_0 \text{ und } (\mathcal{M}, \beta \frac{m}{v_0}) \models R_1 v_0 & \\ \iff \text{für alle } m \in M \text{ gilt } (\mathcal{M}, \beta \frac{m}{v_0}) \models R_0 v_0 \text{ und für alle } m \in M \text{ gilt } (\mathcal{M}, \beta \frac{m}{v_0}) \models R_1 v_0 & \\ \iff (\mathcal{M}, \beta) \models \forall v_0 R_0 v_0 \text{ und } (\mathcal{M}, \beta) \models \forall v_0 R_1 v_0 & \\ \iff (\mathcal{M}, \beta) \models (\forall v_0 R_0 v_0 \wedge \forall v_0 R_1 v_0) & \end{aligned}$$

Die Wahrheitswertfunktionalität von  $\rightarrow$  zeigt nun, dass  $\varphi_2$  allgemeingültig ist.

(d) Sei  $\circ$  ein zweistelliges Funktionszeichen. Die folgende  $\{\circ\}$ -Aussage ist allgemeingültig:

$$\begin{aligned} & ((\forall v_0 \forall v_1 \forall v_2 (v_0 \circ v_1) \circ v_2 \doteq v_0 \circ (v_1 \circ v_2) \wedge \exists v_0 \forall v_1 (v_0 \circ v_1 \doteq v_1 \wedge v_1 \circ v_0 \doteq v_1) \\ & \wedge \exists v_0 \forall v_1 (v_1 \circ v_1 \doteq v_0) \rightarrow \forall v_0 \forall v_1 (v_0 \circ v_1 \doteq v_1 \circ v_0)) \end{aligned}$$

Sie besagt, dass ein Monoid  $M$  mit neutralem Element  $e$  und vom Exponenten 2 (d. h.  $m^2 = e$  für alle  $m \in M$ ) notwendigerweise kommutativ ist. Um die Allgemeingültigkeit zu zeigen, muss man entweder alle solchen Monoide anschauen oder eine mathematische Argumentation über solche Monoide führen.

Die Beispiele legen eine Unterscheidung nahe zwischen „formalen“ Fragen, für deren Beantwortung man keine besonderen Eigenschaften der betrachteten  $\mathcal{L}$ -Strukturen braucht, und „inhaltlichen“ Fragen, die eine nähere Analyse der Strukturen erfordern. Die erste Art scheint sich nur auf die Konstruktion der Logik zu beziehen, die zweite Art ist das traditionelle Geschäft der Mathematiker (und theoretischen Informatiker). Es ist aber nicht klar, ob eine solche Unterscheidung tatsächlich möglich und sinnvoll ist.

### 2.6.2 Kalküle

Kalküle wie der in Definition 2.5.1 eingeführte Kalkül  $\mathbb{K}$  sind eine weitere Möglichkeit, die Allgemeingültigkeit (und damit auch die Erfüllbarkeit) von  $\mathcal{L}$ -Formeln zu zeigen. Es gibt sehr viele Varianten von Kalkülen, und man kann problemlos einen Kalkül durch weitere allgemeingültige  $\mathcal{L}$ -Formeln als Axiome des Kalküls und weitere korrekte Ableitungsregeln als Regeln ergänzen.

Für einen Beweis des Vollständigkeitssatzes angelegte Kalküle wie  $\mathbb{K}$  sind in der Regel nicht gut geeignet dafür, die Allgemeingültigkeit oder die Erfüllbarkeit interessanter  $\mathcal{L}$ -Formeln zu zeigen. Es fehlen in der Regel gute Heuristiken, um Ableitungen in solchen Kalkülen zu finden. Ein Beispiel, nämlich der Beweis der Allgemeingültigkeit der  $\mathcal{L}$ -Aussage  $\varphi_2$  aus Beispiel (c) von oben, findet sich auf Seite 68.

Ein in der Praxis brauchbarer Kalkül ist die Erweiterung des Baumkalküls/der Tableau-Methode der Aussagenlogik für die Prädikatenlogik. Es ist einfacher, dabei nur mit  $\mathcal{L}$ -Aussagen zu arbeiten. Daher benutzt man eine erweiterte Sprache  $\mathcal{L}_C \supseteq \mathcal{L}$ , die mit den ggf. schon in  $\mathcal{L}$

<sup>41</sup>  $\varrho_n$  ist genau dann erfüllbar, wenn die Ramsey-Zahl  $R(5, 5) > n$ .

vorhandenen Konstanten nun abzählbar viele Konstanten  $c_i$  für  $i \in \mathbb{N}$  enthält. Dadurch kann man beim Auflösen von Quantoren Konstanten einsetzen und zu  $\mathcal{L}_C$ -Aussagen übergehen. Ferner ist es deutlich angenehmer, wenn man sich nicht um  $\mathcal{L}$ -Terme kümmern muss. Daher soll  $\mathcal{L}$  zunächst *keine Funktionszeichen außer Konstanten* enthalten und es sollen zunächst nur  $\mathcal{L}$ -Aussagen *ohne Gleichheitszeichen* betrachtet werden.

Im ersten Schritt werden nun die Regeln der Tableau-Methode der Aussagenlogik dahingehend angepasst, dass die Junktoren  $\mathcal{L}_C$ -Aussagen verbinden können und nicht mehr nur aussagenlogische Formeln. Neu hinzu kommen Regeln für den All- und den Existenzquantor (der besseren Verständlichkeit halber hier mit horizontalen Trennstrichen notiert, die man aber in der Anwendung weglässt):

$\forall v_i \varphi : 1$	$\forall v_i \varphi : 0$	$\exists v_i \varphi : 1$	$\exists v_i \varphi : 0$
$\varphi[\frac{c_0}{v_i}] : 1$	$\varphi[\frac{c_j}{v_i}] : 0$	$\varphi[\frac{c_j}{v_i}] : 1$	$\varphi[\frac{c_0}{v_i}] : 0$
$\varphi[\frac{c_1}{v_i}] : 1$	↑	↑	$\varphi[\frac{c_1}{v_i}] : 0$
$\varphi[\frac{c_2}{v_i}] : 1$	dabei muss $c_j$ eine im		$\varphi[\frac{c_2}{v_i}] : 0$
⋮	Pfad neue Konstante sein!		⋮
↑			↑
Diese beiden Regeln dürfen beliebig (aber endlich) oft angewandt werden!			
Es dürfen neue als auch schon im Pfad vorhandene Konstanten eingesetzt werden.			

Atomare  $\mathcal{L}_C$ -Aussagen können durch Regeln nicht weiter aufgelöst werden. Ein Pfad schließt, wenn er sowohl  $\varphi : 1$  als auch  $\varphi : 0$  für eine  $\mathcal{L}_C$ -Aussage  $\varphi$  enthält oder  $\top : 0$  oder  $\perp : 1$ . Dafür schreibt man auch kurz  $\times$  unter den Pfad.

**Satz 2.6.1** Sei  $\varphi$  eine  $\mathcal{L}$ -Aussage, in der weder Gleichheitszeichen noch Funktionszeichen einer Stelligkeit  $\geq 1$  vorkommen. Dann gilt:

- $\varphi$  ist genau dann erfüllbar, wenn in jedem Tableau für  $\varphi : 1$  ein Pfad offen bleibt.
- $\varphi$  ist genau dann allgemeingültig, wenn es ein Tableau für  $\varphi : 0$  gibt, in dem alle Pfade schließen.

BEWEIS: Es reicht, die Behauptung für Erfüllbarkeit zu zeigen.

„ $\Rightarrow$ “ Zunächst stellt man fest, dass in einem Tableau nur  $\mathcal{L}_C$ -Aussagen vorkommen, keine  $\mathcal{L}$ -Formeln mit freien Variablen. Sei  $\mathcal{M}$  eine  $\mathcal{L}$ -Struktur, in der  $\varphi$  gilt.

Behauptung: Es gibt in jedem Tableau einen Pfad und – für die Menge  $C'$  der im diesem Pfad zusätzlich zu  $\mathcal{L}$  vorkommenden Konstanten – eine Expansion  $\mathcal{M}_{C'}$  von  $\mathcal{M}$  zu einer  $\mathcal{L}_{C'}$ -Struktur (d. h. zusätzlich zu der Struktur  $\mathcal{M}$  Interpretationen der Konstanten aus  $C'$ ), so dass  $\mathcal{M} \models \psi$ , falls  $\psi : 1$  im Pfad vorkommt, und  $\mathcal{M} \not\models \psi$ , falls  $\psi : 0$  im Pfad vorkommt.

Man beweist dies „per Induktion über den Abbau von  $\varphi$ “, d. h. man zeigt, dass diese Behauptung unter Anwendung aller Regeln erhalten bleibt. Für die Junktorenregeln ist das klar. Die Regel  $\forall v_i \varphi : 0$  führt auf dasselbe wie  $\exists v_i \neg \varphi : 1$  und die Regel  $\exists v_i \varphi : 0$  auf dasselbe wie  $\forall v_i \neg \varphi : 1$ , daher reicht es, die Quantorenregeln für den Wahrheitswert 1 zu betrachten.

Sei  $\mathcal{M}'$  die Expansion von  $\mathcal{M}$  um die im Pfad schon vorgekommenen neuen Konstanten. Falls  $\mathcal{M}' \models \exists v_i \varphi$ , also  $(\mathcal{M}', \beta) \models \exists v_i \varphi$  für eine beliebige Belegung  $\beta$ , dann gibt es ein  $m \in M$  mit  $(\mathcal{M}', \beta[\frac{m}{v_i}]) \models \varphi$ . Die Regel für  $\exists v_i \varphi : 1$  führt zu  $\varphi[\frac{c_j}{v_i}] : 1$  für eine Konstante  $c_j$ , die neu im Pfad ist. Setzt man also  $c_j^{\mathcal{M}''} = m$ , bekommt man eine Expansion  $\mathcal{M}''$  von  $\mathcal{M}'$  mit  $\mathcal{M}'' \models \varphi[\frac{c_j}{v_i}]$ .

Falls  $\mathcal{M}' \models \forall v_i \varphi$ , gilt  $\mathcal{M}' \models \varphi[\frac{c_j}{v_i}]$  für jede schon vorgekommene Konstante  $c_j$ , in Einklang mit der Regel für  $\forall : 1$ . Wird die Regel mit einer neuen Konstanten angewandt, expandiert man  $\mathcal{M}'$  wie oben zu  $\mathcal{M}''$ , indem man als Interpretation von  $c_j$  irgendein Element aus  $M$  wählt.

„ $\Leftarrow$ “ Angenommen man hat einen Pfad  $P$  in einem Tableau für  $\varphi : 1$ , der nicht geschlossen werden kann. Falls in  $P$  nur  $\top : 1$  oder  $\perp : 0$  als Einträge mit atomaren Formeln vorkommen, gibt es in  $\varphi$  bestenfalls „unnötige Quantoren“, d. h.  $\varphi$  ist im Wesentlichen eine aussagenlogische Formel und das Ergebnis folgt aus der aussagenlogischen Variante der Tableau-Methode.

Andernfalls kommt mindestens eine Konstante in  $P$  vor. Nun wendet man die Regeln für  $\forall : 1$  und  $\exists : 0$  mit allen bisher im Pfad vorkommenden Konstanten an und löst dann die entstehenden  $\mathcal{L}_C$ -Formeln wieder bis zur atomaren Ebene auf, wobei ggf. neue Konstanten eingeführt werden müssen. Dies wiederholt man unendlich oft (wobei evtl. nach endlich vielen Schritten nichts mehr passiert) und erhält dadurch einen möglicherweise unendlich langen „bzgl. Regelanwendung abgeschlossenen“ Pfad  $P^*$ . Dieser Pfad  $P^*$  schließt nicht, da man sonst  $P$  in endlich vielen Schritten schließen könnte.

Auf der Menge  $M = C''$  der in  $P^*$  vorkommenden Konstanten definiert man eine  $\mathcal{L}_{C''}$ -Struktur  $\mathcal{M}$ , indem man jede Konstante in  $C''$  durch sich selbst interpretiert und für in  $\varphi$  vorkommende Relationszeichen  $R$  setzt:

$$(c_{i_1}, \dots, c_{i_n}) \in R^{\mathcal{M}} \iff Rc_{i_1} \dots c_{i_n} : 1 \text{ kommt in } P^* \text{ vor}$$

Relationszeichen aus  $\mathcal{L}$ , die in  $\varphi$  nicht vorkommen, interpretiert man durch  $\emptyset$ .

Nun gilt: Wenn  $\psi : 1$  in  $P^*$  vorkommt, ist  $\mathcal{M} \models \psi$ , und wenn  $\psi : 0$  in  $P^*$  vorkommt, ist  $\mathcal{M} \not\models \psi$ . (Insbesondere ist also  $\mathcal{M}$  Modell von  $\varphi$ .)

Beweis über den Aufbau von  $\psi$ : Für  $\top$  und  $\perp$  ist es klar, und für atomares  $\psi = Rc_{i_1} \dots c_{i_n}$  gilt es im Fall von  $\psi : 1$  nach Definition von  $\mathcal{M}$ . Kommt dagegen  $\psi : 0$  in  $P^*$  vor, so gilt  $(c_{i_1}, \dots, c_{i_n}) \notin R^{\mathcal{M}}$ , weil  $P^*$  nicht schließt. Die Junktorenschritte sind wiederum klar. Da  $M$  nur aus den Interpretationen der Konstanten besteht, sieht man (formal mit dem Substitutionslemma)

$$\mathcal{M} \models \forall v_i \psi' \iff \text{für jede Konstante } c \in C'' \text{ gilt } \mathcal{M} \models \psi'[\frac{c}{v_i}]$$

Analog für den Existenzquantor. Das sind gerade die Quantorenschritte! □

Ist  $\exists v_0 (Pv_0 \rightarrow \forall v_1 Pv_1)$  allgemeingültig?

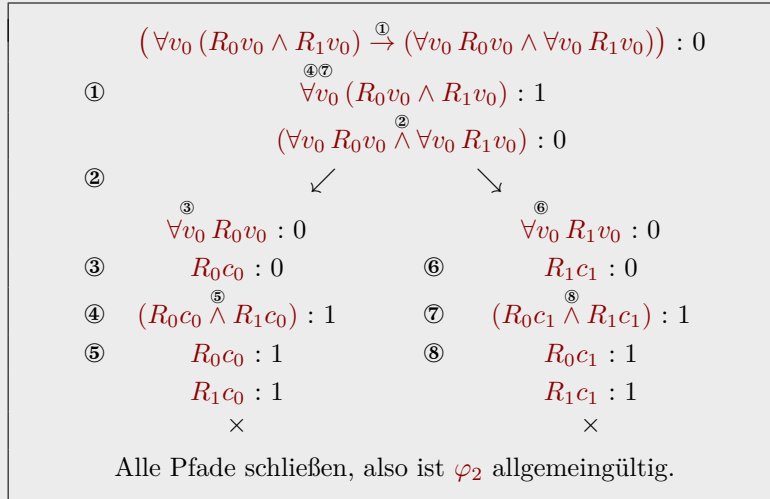
	<sup>①④</sup> $\exists v_0 (Pv_0 \rightarrow \forall v_1 Pv_1) : 0$	
①	<sup>②</sup> $(Pc_0 \rightarrow \forall v_1 Pv_1) : 0$	
②	$Pc_0 : 1$	
	<sup>③</sup> $\forall v_1 Pv_1 : 0$	
③	$Pc_1 : 0$	$c_1$ neu!
④	<sup>⑤</sup> $(Pc_1 \rightarrow \forall v_1 Pv_1) : 0$	
⑤	$Pc_1 : 1$	
	$\forall v_1 Pv_1 : 0$	
	×	
	Alle Pfade schließen, also allgemeingültig!	

Man sieht an diesem Beispiel, dass man die Regel für  $\exists : 0$  mehrfach aufrufen muss (Schritte ① und ④) und dass die Regel für  $\forall : 0$  eine im Pfad neue Konstante braucht (Schritt ③).

Als zweites Beispiel soll wieder die Allgemeingültigkeit der  $\mathcal{L}$ -Aussage  $\varphi_2$  von Seite 75, also

$$(\forall v_0 (R_0 v_0 \wedge R_1 v_0) \rightarrow (\forall v_0 R_0 v_0 \wedge \forall v_0 R_1 v_0))$$


gezeigt werden, damit man die verschiedenen Methoden im Vergleich sieht.



Auf der rechten Seite könnte man statt  $c_1$  auch  $c_0$  nehmen, da Schritt ⑥ nur eine *im Pfad neue* Konstante fordert.

Hier nun noch ein Beispiel, wie man aus einem nicht-schließenden Pfad ein Modell gewinnt. Sei  $\mathcal{L} = \{R\}$  mit zweistelligem Relationszeichen und  $\varphi$  die  $\mathcal{L}$ -Aussage

$$(\exists v_0 \exists v_1 \neg R v_0 v_1 \wedge \forall v_0 \exists v_1 R v_0 v_1)$$

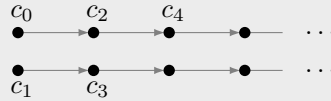
$\mathcal{L}$ -Strukturen kann man als gerichtete Graphen ansehen.  $\varphi$  sagt dann aus, dass von jedem Knoten eine gerichtete Kante zu einem anderen führt, aber nicht von jedem Knoten zu jedem. Ein einfaches Modell von  $\varphi$  ist . Die Tableau-Methode liefert nun zum Beispiel:

1.		$(\exists v_0 \exists v_1 \neg R v_0 v_1 \wedge \forall v_0 \exists v_1 R v_0 v_1) : 1$
2.	aus 1.	$\exists v_0 \exists v_1 \neg R v_0 v_1 : 1$
3.		$\forall v_0 \exists v_1 R v_0 v_1 : 1$
4.	aus 2.	$\exists v_1 \neg R c_0 v_1 : 1$
5.	aus 4.	$\neg R c_0 c_1 : 1$
6.	aus 5.	$R c_0 c_1 : 0$
7.	aus 3.	$\exists v_1 R c_0 v_1 : 1$
8.	aus 7.	$R c_0 c_2 : 1$
9.	aus 3.	$\exists v_1 R c_1 v_1 : 1$
10.	aus 9.	$R c_1 c_3 : 1$
11.	aus 3.	$\exists v_1 R c_2 v_1 : 1$
12.	aus 11.	$R c_2 c_4 : 1$
⋮		⋮

Man kann sich hier schnell davon überzeugen, dass das Tableau nicht schließen wird, weil kein neuer Eintrag von der Form  $R c_i c_j : 0$  entstehen kann und man mit keiner Regelanwendung

$Rc_0c_1 : 1$  bekommen kann. (Dies ist allerdings eine Argumentation *über* das Tableau und keine Anwendung der Tableau-Methode an sich. Ob ein Pfad noch schließen kann oder nicht, ist im Allgemeinen nicht entscheidbar.)

Man sieht aber auch, dass man immer wieder neue Konstanten einführen muss, die man wieder in Zeile 3. einsetzen kann. Im Beweis von Satz 2.6.1 werden aus nicht-schließenden Pfaden Modelle konstruiert, in denen nur solche Relationen bestehen, die durch die Axiome zwingend gegeben sind. Aus dem Tableau oben entsteht auf diese Weise folgendes Modell von  $\varphi$ :



Für  $\mathcal{L}$ -Aussagen mit Gleichheitszeichen braucht man zusätzliche Regeln, etwa:

	$c_j \doteq c_k : 1$	$c_j \doteq c_k : 1$
	$\varphi[\frac{c_j}{v_i}] : 1$	$\varphi[\frac{c_j}{v_i}] : 0$
<hr style="width: 50%; margin: 0 auto;"/>	<hr style="width: 50%; margin: 0 auto;"/>	<hr style="width: 50%; margin: 0 auto;"/>
$c_i \doteq c_i : 1$	$\varphi[\frac{c_k}{v_i}] : 1$	$\varphi[\frac{c_k}{v_i}] : 0$

Man darf also jederzeit in einen Pfad eine Zeile  $c_i \doteq c_i : 1$  einfügen, und sofern  $c_j \doteq c_k : 1$  in einem Pfad vorkommt, darf man in allen anderen  $\mathcal{L}_C$ -Formeln im Pfad beliebig viele Vorkommen von  $c_j$  durch  $c_k$  ersetzen.

(Insbesondere bekommt man, wenn man diese Regel auf  $c_j \doteq c_j : 1$  anwendet, mit  $c_j \doteq c_k : 1$  auch  $c_k \doteq c_j : 1$ , d. h. die Situation ist symmetrisch bzgl. beider Seiten des Gleichheitszeichens!)

Ist  $\forall v_0 \forall v_1 \forall v_2 ((v_0 \doteq v_1 \wedge v_1 \doteq v_2) \rightarrow v_0 \doteq v_2)$  allgemeingültig?

	<sup>①</sup> $\forall v_0 \forall v_1 \forall v_2 ((v_0 \doteq v_1 \wedge v_1 \doteq v_2) \rightarrow v_0 \doteq v_2) : 0$
①	<sup>②</sup> $\forall v_1 \forall v_2 ((c_0 \doteq v_1 \wedge v_1 \doteq v_2) \rightarrow c_0 \doteq v_2) : 0$
②	<sup>③</sup> $\forall v_2 ((c_0 \doteq c_1 \wedge c_1 \doteq v_2) \rightarrow c_0 \doteq v_2) : 0$
③	<sup>④</sup> $((c_0 \doteq c_1 \wedge c_1 \doteq c_2) \rightarrow c_0 \doteq c_2) : 0$
④	<sup>⑤</sup> $(c_0 \doteq c_1 \wedge c_1 \doteq c_2) : 1$
	⑥ $c_0 \doteq c_2 : 0$
⑤	⑦ $c_0 \doteq c_1 : 1$
	$c_1 \doteq c_2 : 1$
⑦ in ⑥	$c_1 \doteq c_2 : 0$
	×

Möchte man  $\mathcal{L}$ -Aussagen mit Funktionszeichen behandeln, arbeitet man am besten mit term-reduzierten  $\mathcal{L}$ -Formeln. Alternativ kann man auch Funktionszeichen durch Relationszeichen ersetzen, so wie im nächsten Abschnitt beschrieben.

### 2.6.3 Syntaktische Reduktionen durch Spracherweiterungen

Wenn es um die Erfüllbarkeit bzw. die Allgemeingültigkeit einer  $\mathcal{L}$ -Formel geht und nicht um logische Äquivalenz, kann man (vergleichbar zu Lemma 1.7.1) besondere syntaktische Formen

erreichen, indem man die Sprache durch zusätzliche Zeichen erweitert.

**$\mathcal{L}$ -Formeln ohne Funktionen** Man kann in der Prädikatenlogik ohne Einschränkung der Ausdrucksstärke auf Funktionszeichen verzichten. Dazu führt man für jedes Funktionszeichen  $f \in \mathcal{L}$  der Stelligkeit  $n$  zunächst ein neues  $(n + 1)$ -stelliges Relationszeichen  $R_f$  ein, dessen Interpretation der Graph der Interpretation von  $f$  sein wird, und setzt  $\mathcal{L}^+ = \mathcal{L} \cup \{R_f\}$  und  $\mathcal{L}_R = \mathcal{L}^+ \setminus \{f\}$ .

Jede  $\mathcal{L}$ -Struktur  $\mathcal{M}$  kann nun auf eindeutige Weise so zu einer  $\mathcal{L}^+$ -Struktur  $\mathcal{M}^+$  gemacht werden, dass die  $\mathcal{L}^+$ -Aussage

$$(*) \quad \forall v_0 \dots \forall v_n (v_0 \doteq f v_1 \dots v_n \leftrightarrow R v_0 \dots v_n)$$

gilt (man nennt dies eine *definitonische Erweiterung* [expansion by definition]). Indem man die Interpretation von  $f$  vergisst, wird aus  $\mathcal{M}^+$  eine  $\mathcal{L}_R$ -Struktur  $\mathcal{M}_R$ . In  $\mathcal{M}_R$  gilt somit die  $\mathcal{L}_R$ -Aussage

$$\gamma_R := \forall v_1 \dots \forall v_n \exists v_{n+1} (R v_0 \dots v_n \wedge \forall v_{n+1} (R v_{n+1} v_1 \dots v_n \rightarrow v_{n+1} \doteq v_0))$$

die ausdrückt, dass die Interpretation von  $R$  Graph einer  $n$ -stelligen Funktion ist.

Umgekehrt ist jede  $\mathcal{L}_R$ -Struktur  $\mathcal{S}$ , die  $\gamma_R$  erfüllt, von der Form  $\mathcal{M}_R$ , denn  $\mathcal{S}$  wird auf eindeutige Weise zu einer  $\mathcal{L}^+$ -Struktur  $\mathcal{S}^+$ , in der (\*) gilt, indem man nämlich  $f$  durch die Funktion interpretiert, deren Graph  $R^{\mathcal{S}}$  ist, und dann zu einer  $\mathcal{L}$ -Struktur  $\mathcal{S}_f$ , indem man die Interpretation von  $R$  vergisst. Offenbar ist dann  $\mathcal{M}_{Rf} = \mathcal{M}$  und  $\mathcal{S}_{fR} = \mathcal{S}$ , d. h.  $\mathcal{L}$ -Strukturen entsprechen ein-eindeutig  $\mathcal{L}_R$ -Strukturen, die Modelle von  $\gamma_R$  sind.

Daher kann man folgendermaßen auf Funktionszeichen verzichten: Zu jeder  $\mathcal{L}$ -Formel  $\varphi$  betrachtet man zunächst eine logisch äquivalente term-reduzierte  $\mathcal{L}$ -Formel  $\varphi'$  und ersetzt darin jedes Vorkommen einer Teilformel  $v_{i_0} \doteq f v_{i_1} \dots v_{i_n}$  oder  $f v_{i_1} \dots v_{i_n} \doteq v_{i_0}$  durch  $R v_{i_0} v_{i_1} \dots v_{i_n}$ , um eine  $\mathcal{L}_R$ -Formel  $\varphi''$  zu erhalten. Es folgt, dass  $\varphi$  genau dann erfüllbar ist, wenn  $(\gamma_R \wedge \varphi'')$  erfüllbar ist, und genau dann allgemeingültig, wenn  $(\gamma_R \rightarrow \varphi'')$  allgemeingültig ist.

Man kann dies nun simultan für alle in einer Formel vorkommenden Funktionszeichen tun.

## Skolem- und Herbrand-Normalform

**Definition 2.6.2** Eine  $\mathcal{L}$ -Formel in pränexer Normalform heißt **universell** [universal], wenn in ihr keine Existenzquantoren vorkommen, und **existenziell** [existential], wenn in ihr keine Allquantoren vorkommen. Quantorenfreie Formeln sind sowohl universell als auch existenziell.

**Satz 2.6.3** Eine  $\mathcal{L}$ -Aussage  $\varphi$  in pränexer Normalform kann auf algorithmische Weise

- (a) zu einer erfüllbarkeitsäquivalenten universellen  $\mathcal{L}^*$ -Aussage  $\varphi^*$  und  $\varphi^*$
- (b) zu einer existenziellen  $\mathcal{L}_*$ -Aussage  $\varphi_*$  gemacht werden, die genau dann allgemeingültig ist, wenn  $\varphi$  es ist, wobei  $\mathcal{L}^*$  und  $\mathcal{L}_*$  Erweiterungen von  $\mathcal{L}$  durch Funktionszeichen sind. <sup>42</sup>  $\varphi_*$

Die zusätzlichen Funktionszeichen heißen **Skolem-Funktionen** [Skolem functions]. Die Formel  $\varphi^*$  heißt **Skolem-Normalform** [Skolem normal form] oder **Skolemisierung** [Skolemization] von  $\varphi$ , und die Formel  $\varphi_*$  **Herbrand-Normalform** [Herbrand normal form] von  $\varphi$ .

<sup>42</sup>In der Regel enthält  $\mathcal{L}_*$  andere zusätzliche Funktionszeichen als  $\mathcal{L}^*$ .



Trivialerweise gibt es zu jeder  $\mathcal{L}$ -Aussage  $\varphi$  sogar eine quantorenfreie erfüllbarkeitsäquivalente Aussage, nämlich  $\top$ , falls  $\varphi$  erfüllbar ist, und  $\perp$  andernfalls. Wichtig ist in diesem Satz daher, dass man die erfüllbarkeitsäquivalente  $\mathcal{L}^*$ -Aussage algorithmisch finden kann, also ohne vorher entscheiden zu müssen, ob  $\varphi$  erfüllbar ist oder nicht (was, wie in Kapitel 3 gezeigt wird, im Allgemeinen nicht geht).

BEWEIS: (a) Sei  $\varphi$  in pränexer Normalform. Wenn  $\varphi$  bereits universell ist, ist nichts zu tun. Andernfalls ist  $\varphi$  ohne Einschränkung (d. h. nach eventueller Umbenennung der Variablen) von der Form

$$\forall v_1 \dots \forall v_k \exists v_{k+1} \chi$$

für  $k \in \mathbb{N}$  und eine  $\mathcal{L}$ -Formel  $\chi$ , die möglicherweise weitere All- und Existenzquantoren enthält. Nun erweitert man  $\mathcal{L}$  um ein neues  $k$ -stelliges Funktionszeichen  $f$  (im Falle von  $k = 0$  also um eine neue Konstante) und ersetzt die  $\mathcal{L}$ -Formel  $\varphi$  durch die  $\mathcal{L} \cup \{f\}$ -Formel

$$\forall v_1 \dots \forall v_k \chi \left[ \frac{fv_1 \dots v_k}{v_{k+1}} \right]$$

Auf diese Weise ersetzt man von außen nach innen (d. h. von links nach rechts) alle vorkommenden Existenzquantoren und erhält zum Schluss  $\varphi^*$ .

Falls  $\varphi$  erfüllbar ist, definiert man in einem Modell  $\mathcal{M}$  von  $\varphi$  die neuen Funktionen  $f^{\mathcal{M}}$  folgendermaßen: Für jedes  $(m_1, \dots, m_k) \in M^k$  wählt man für  $f^{\mathcal{M}}(m_1, \dots, m_k)$  ein Element  $m'$ , so dass  $\mathcal{M} \models \chi[m_1, \dots, m_n, m']$  gilt. Dadurch wird  $\mathcal{M}$  zu einem Modell  $\mathcal{M}^*$  von  $\varphi^*$  *expandiert* [expanded] und folglich ist  $\varphi^*$  erfüllbar.

Ist umgekehrt  $\mathcal{M}^*$  ein Modell von  $\varphi^*$  und  $\beta$  eine Belegung, dann ist  $m' = f^{\mathcal{M}^*}(\beta(v_1), \dots, \beta(v_k))$  ein Element, „das den Existenzquantor  $\exists v_{k+1}$  erfüllt“, für das also  $(\mathcal{M}, \beta \frac{m'}{v_{k+1}}) \models \chi$ .

(b)  $\varphi$  ist genau dann allgemeingültig, wenn  $\neg\varphi$  nicht erfüllbar ist, also nach (a) genau dann, wenn  $(\neg\varphi)^*$  nicht erfüllbar ist, d. h. wenn  $\neg(\neg\varphi)^*$  allgemeingültig ist. Zieht man in dieser Formel die äußere Negation nach innen, erhält man die existenzielle Formel  $\varphi_*$ . Die Sprache  $\mathcal{L}_*$  für  $\varphi$  ist also  $\mathcal{L}^*$  für  $\neg\varphi$ .  $\square$

(a) Die Skolem-Normalform von  $\varphi = \forall v_0 \exists v_1 \forall v_2 \exists v_3 (\neg Rv_0v_1 \vee Rv_3v_2)$  ist

$$\varphi^* = \forall v_0 \forall v_2 (\neg Rv_0 f v_0 \vee R g v_0 v_2 v_2)$$

mit neuem einstelligem Funktionszeichen  $f$  und neuem zweistelligem Funktionszeichen  $g$ .

Für die Herbrand-Normalform  $\varphi_*$  von  $\varphi$  führt man folgende Schritte durch:

$$\varphi \text{ negieren und pränex machen: } \exists v_0 \forall v_1 \exists v_2 \forall v_3 \neg(\neg Rv_0v_1 \vee Rv_3v_2)$$

$$\text{Skolemisieren: } \forall v_1 \forall v_3 \neg(\neg Rcv_1 \vee Rv_3hv_1)$$

$$\text{erneut negieren und pränex machen: } \exists v_1 \exists v_3 (\neg Rcv_1 \vee Rv_3hv_1) = \varphi_*$$

wobei  $c$  neue Konstante und  $h$  neues einstelliges Funktionszeichen. Im Endeffekt eliminiert man also die Allquantoren in gleicher Weise wie bei der Skolemisierung die Existenzquantoren.

(b) Bei mehreren Existenzquantoren hintereinander braucht man jeweils ein neues Funktionszeichen gleicher Stelligkeit.  $\forall v_2 \forall v_0 \exists v_1 \exists v_3 (\neg Rv_0v_1 \vee Rv_3v_2)$  wird bei der Skolemisierung zu  $\forall v_2 \forall v_0 (\neg Rv_0 f v_0 v_2 \vee R g v_0 v_2 v_2)$  mit neuen zweistelligen Funktionszeichen  $f$  und  $g$ .

### 2.6.4 Der Satz von Herbrand und Unifikation

Erinnerung: Ein Term heißt *geschlossen*, wenn er keine Individuenvariablen enthält. Damit es geschlossene  $\mathcal{L}$ -Terme gibt, muss  $\mathcal{L}$  mindestens eine Konstante enthalten. Ist dies nicht der Fall, kann man  $\mathcal{L}$  um eine Konstante erweitern, um den folgenden Satz anwenden zu können.

**Satz 2.6.4 (Satz von Herbrand, 1930)** Sei  $\mathcal{L}$  eine Sprache mit mindestens einer Konstante und  $\varphi = \exists v_1 \dots \exists v_n \psi$  eine existenzielle  $\mathcal{L}$ -Aussage mit quantorenfreiem  $\psi$ .

Dann ist  $\varphi$  genau dann allgemeingültig, wenn es ein  $N \in \mathbb{N}$  und geschlossene  $\mathcal{L}$ -Terme  $\tau_{i1}, \dots, \tau_{in}$  für  $i = 1, \dots, N$  gibt, so dass

$$\bigvee_{i=1}^N \psi(\tau_{i1}, \dots, \tau_{in}) := \left( \psi\left[\frac{\tau_{i1}}{v_1} \dots \frac{\tau_{in}}{v_n}\right] \vee \dots \vee \psi\left[\frac{\tau_{N1}}{v_1} \dots \frac{\tau_{Nn}}{v_n}\right] \right)$$

allgemeingültig ist.

Man kann für  $N$  keine Schranke angeben und sollte sich typischerweise eine große Zahl darunter vorstellen.

Für den Beweis braucht man noch den Begriff der *Unterstruktur* einer  $\mathcal{L}$ -Struktur  $\mathcal{M}$ . Dies ist eine  $\mathcal{L}$ -Struktur, die von  $\mathcal{M}$  auf einer Teilmenge von  $M$  induziert wird: Falls  $U$  eine unter allen Funktionen  $f_i^{\mathcal{M}}$  für  $f_i \in \mathcal{L}$  abgeschlossene und nicht-leere Teilmenge von  $M$  ist, so ist die  $\mathcal{L}$ -Unterstruktur [ $\mathcal{L}$ -substructure]  $\mathcal{U}$  mit Universum  $U$  definiert durch:

- $f_i^{\mathcal{U}} := f_i^{\mathcal{M}} \upharpoonright_{U^n}$  für jedes  $n$ -stellige Funktionszeichen  $f_i \in \mathcal{L}$ ;
- $R_j^{\mathcal{U}} := R_j^{\mathcal{M}} \cap U^n$  für jedes  $n$ -stellige Relationszeichen  $R_j \in \mathcal{L}$ .

BEWEIS: Klar ist  $\psi\left[\frac{\tau_{i1}}{v_1} \dots \frac{\tau_{in}}{v_n}\right] \models \exists v_1 \dots \exists v_n \psi$ , also auch  $\bigvee_{i=1}^N \psi(\tau_{i1}, \dots, \tau_{in}) \models \exists v_1 \dots \exists v_n \psi$ , was die Richtung „ $\Leftarrow$ “ beweist.

Für „ $\Rightarrow$ “ nimmt man an, dass  $\varphi$  allgemeingültig ist, aber keine der möglichen Disjunktionen von Term-Einsetzungen. Also ist jedes  $\neg \bigvee_{i=1}^N \psi(\tau_{i1}, \dots, \tau_{in}) \sim \bigwedge_{i=1}^N \neg \psi(\tau_{i1}, \dots, \tau_{in})$  erfüllbar. Dies bedeutet, dass die Menge

$$\{\neg \psi\left[\frac{\tau_1}{v_1} \dots \frac{\tau_n}{v_n}\right] \mid \tau_1, \dots, \tau_n \text{ geschlossene } \mathcal{L}\text{-Terme}\}$$

endlich erfüllbar ist und nach dem Kompaktheitssatz 2.5.16 also ein Modell  $\mathcal{M}$  hat.

Sei  $M_0 := \{\tau^{\mathcal{M}} \mid \tau \text{ geschlossener } \mathcal{L}\text{-Term}\}$  die Teilmenge von  $M$ , die aus den Interpretationen geschlossener Terme besteht. Per Definition ist  $M_0$  unter den Funktionen  $f^{\mathcal{M}}$  für  $f \in \mathcal{L}$  abgeschlossen, da

$$f^{\mathcal{M}}(\tau_1^{\mathcal{M}}, \dots, \tau_n^{\mathcal{M}}) = f\tau_1 \dots \tau_n^{\mathcal{M}}.$$

Also ist  $M_0$  Universum einer  $\mathcal{L}$ -Unterstruktur  $\mathcal{M}_0$  von  $\mathcal{M}$ . Unterstrukturen sind gerade so definiert, dass alle geschlossenen Terme die gleiche Interpretation in  $\mathcal{M}_0$  wie in  $\mathcal{M}$  haben und dass alle atomaren  $\mathcal{L}$ -Aussagen die gleiche Gültigkeit in  $\mathcal{M}_0$  wie in  $\mathcal{M}$  haben. Also gilt auch eine quantorenfreie  $\mathcal{L}$ -Aussagen in  $\mathcal{M}_0$  genau dann, wenn sie in  $\mathcal{M}$  gilt. Insbesondere folgt daraus  $\mathcal{M}_0 \models \neg \psi\left[\frac{\tau_1}{v_1} \dots \frac{\tau_n}{v_n}\right]$  für alle Terme  $\tau_1, \dots, \tau_n$ . Da  $M_0$  aber nur aus Interpretationen von Termen besteht, gilt  $\mathcal{M}_0 \models \forall v_1 \dots \forall v_n \neg \psi \sim \neg \exists v_1 \dots \exists v_n \psi \sim \neg \varphi$ , im Widerspruch zur Allgemeingültigkeit von  $\varphi$ !  $\square$

**Bemerkung 2.6.5 (a)** Wenn man  $\mathcal{L}$ -Aussagen ohne Gleichheitszeichen betrachtet, kommt auch in der Herbrand-Normalform kein Gleichheitszeichen vor. Durch die Term-Einsetzungen

erhält man eine Disjunktion quantorenfreier  $\mathcal{L}$ -Aussagen. Die Frage, ob solche eine Aussage allgemeingültig ist, lässt sich in ein rein aussagenlogisches Problem überführen: Eine quantorenfreie  $\mathcal{L}$ -Aussage  $\varphi$  ohne Gleichheitszeichen ist genau dann allgemeingültig, wenn ihr „aussagenlogisches Gerüst“  $F_\varphi$  eine Tautologie ist. Dabei entsteht  $F_\varphi$  aus  $\varphi$ , indem man die atomaren Teilformeln von  $\varphi$  durch Aussagenvariablen ersetzt (wobei zwei atomare Formeln genau dann durch die gleiche Aussagenvariable ersetzt werden, wenn sie gleich sind).

Jetzt soll der Satz von Herbrand benutzt werden, um die Allgemeingültigkeit zu zeigen von

$$\varphi_2 = (\forall v_0 (R_0 v_0 \wedge R_1 v_0) \rightarrow (\forall v_0 R_0 v_0 \wedge \forall v_0 R_1 v_0))$$

Als erstes wird  $\varphi_2$  in pränex Normalform gebracht, in Hinblick auf den nächsten Schritt am besten so, dass die Existenzquantoren möglichst weit innen stehen:

$$\forall v_1 \forall v_2 \exists v_0 (\neg(R_0 v_0 \wedge R_1 v_0) \vee (R_0 v_1 \wedge R_1 v_2))$$

Mit zwei neuen Konstanten  $c_1$  und  $c_2$  und  $\mathcal{L}_* = \{R_0, R_1, c_1, c_2\}$  bekommt man die Herbrand-Normalform

$$\varphi_{2*} = \exists v_0 (\neg(R_0 v_0 \wedge R_1 v_0) \vee (R_0 c_1 \wedge R_1 c_2))$$

Geschlossene  $\mathcal{L}_*$ -Terme, die man für  $v_0$  einsetzen kann, sind  $c_1$  und  $c_2$ . Setzt man beide wie im Satz von Herbrand ein, erhält man

$$\chi = ((\neg(R_0 c_1 \wedge R_1 c_1) \vee (R_0 c_1 \wedge R_1 c_2)) \vee (\neg(R_0 c_2 \wedge R_1 c_2) \vee (R_0 c_1 \wedge R_1 c_2)))$$

mit aussagenlogischen Gerüst

$$F_\chi = ((\neg(A_0 \wedge A_1) \vee (A_0 \wedge A_2)) \vee (\neg(A_3 \wedge A_2) \vee (A_0 \wedge A_2))).$$

Mit einer der Methoden aus der Aussagenlogik zeigt man nun, dass  $F_\chi$  eine Tautologie ist!

Um zu überprüfen, ob  $F_\varphi$  eine Tautologie ist, versucht man am besten mit der Resolutionsmethode die Nicht-Erfüllbarkeit von  $\neg F_\varphi$  zu zeigen. Indem man das Negationszeichen nach innen zieht, erhält man de facto folgende Variante der Resolutionsmethode:

Man bringt  $F_\varphi$  in DNF und betrachtet die einzelnen Disjunktionsglieder als „duale Klauseln“, d. h. jedes Disjunktionsglied wird zu einer Menge von Literalen, die als ihre Konjunktion verstanden wird. Auf diese dualen Klauseln wendet man nun Resolution an, und sieht, dass  $F_\varphi$  genau dann allgemeingültig ist, wenn sich die leere duale Klausel ergibt.

**(b)** Man kann zeigen, dass man sich immer auf eine  $\mathcal{L}$ -Aussage ohne Gleichheitszeichen zurückziehen kann: Sei  $E$  ein neues zweistelliges Relationszeichen. Für eine  $\mathcal{L}$ -Aussage  $\varphi$  sei  $\mathcal{L}^\varphi$  die Teilmenge der in  $\varphi$  vorkommenden Zeichen aus  $\mathcal{L}$ , und  $\mathcal{L}_E^\varphi := \mathcal{L}^\varphi \cup \{E\}$ . Man ersetzt nun jede Teilformel  $\tau_1 \doteq \tau_2$  in  $\varphi$  durch  $E\tau_1\tau_2$ , und erhält dadurch eine  $\mathcal{L}_E^\varphi$ -Formel  $\varphi_E$  ohne Gleichheitszeichen.

Sei  $K_E^\varphi := \{\gamma_E \mid \gamma \text{ } \mathcal{L}^\varphi\text{-Gleichheitsgesetz}\}$ . In jedem Modell  $\mathcal{M}$  von  $K_E^\varphi$  ist  $E^\mathcal{M}$  eine Äquivalenzrelation und sogar eine Kongruenzrelation bzgl.  $\mathcal{L}^\varphi$ . Auf der Menge  $\mathcal{M}/E^\mathcal{M}$  der Äquivalenzklassen kann man daher auf natürliche Weise eine  $\mathcal{L}_E^\varphi$ -Struktur  $\mathcal{M}/E$  definieren:

$$\begin{aligned} f^{\mathcal{M}/E}(m_1/E^\mathcal{M}, \dots, m_n/E^\mathcal{M}) &:= f^\mathcal{M}(m_1, \dots, m_n)/E^\mathcal{M} \\ (m_1/E^\mathcal{M}, \dots, m_n/E^\mathcal{M}) \in R^{\mathcal{M}/E} &: \iff (m_1, \dots, m_n) \in R^\mathcal{M} \end{aligned}$$

Insbesondere ist  $E^{M/E}$  die Identitätsrelation auf  $M/E^M$ . Nun kann man per Induktion über den Formalaufbau für alle  $\mathcal{L}^\varphi$ -Aussagen  $\psi$  zeigen:

$$\mathcal{M}/E \models \psi \iff \mathcal{M} \models \psi_E.$$

Außerdem ist jede  $\mathcal{L}^\varphi$ -Struktur  $\mathcal{N}$  von der Form  $\mathcal{M}/E$ , da man  $\mathcal{M} = \mathcal{N}$  wählen und  $E$  darin durch die Identität interpretieren kann. Aus beiden Überlegungen folgt, dass die  $\mathcal{L}$ -Aussage  $\varphi$  genau dann allgemeingültig ist, wenn die  $\mathcal{L}_E^\varphi$ -Aussage  $(\bigwedge K_E^\varphi \rightarrow \varphi_E)$  es ist, die kein Gleichheitszeichen enthält.

Anstelle von  $E$  könnte man natürlich auch Symbol wie  $\doteq$  wählen, das einem Gleichheitszeichen ähnlich ist. Da in  $(\bigwedge K_E^\varphi \rightarrow \varphi_E)$  kein Gleichheitszeichen vorkommt, könnte man es darin sogar wieder durch  $\doteq$  ersetzen. Letztendlich läuft die gesamte Überlegung also darauf hinaus, dass man für die Anwendung des Satzes von Herbrand die  $\mathcal{L}$ -Aussage  $\varphi$  durch die  $\mathcal{L}$ -Aussage  $(\bigwedge K^\varphi \rightarrow \varphi)$  ersetzen kann, wobei  $K^\varphi$  die Menge der  $\mathcal{L}^\varphi$ -Gleichheitsgesetze ist, und dann das Gleichheitszeichen wie die Relationszeichen in  $\mathcal{L}$  behandelt.

**Der Rest des Abschnittes ist im WS 2021/22 voraussichtlich nicht Teil der Vorlesung!**

Wenn man die Allgemeingültigkeit von z. B.  $(\exists v_0 \forall v_1 Rv_0v_1 \rightarrow \forall v_1 \exists v_0 Rv_0v_1)$  mit dem Satz von Herbrand zeigen möchte, muss man also geeignete Term-Einsetzungen in die Herbrand-Normalform  $\exists v_1 \exists v_3 (\neg Rcv_1 \vee Rv_3hv_1)$  betrachten. Eine Chance, die Allgemeingültigkeit einer Disjunktion von Term-Einsetzungen zu zeigen, hat man nur, wenn gleiche atomare Formeln auftreten. In diesem Fall möchte man also durch geeignete Einsetzungen erreichen, dass aus  $Rcv_1$  und  $Rv_3hv_1$  die gleiche Formel wird. Hier erreicht man dies offenbar, indem man zum einen  $c$  für  $v_1$  und für  $v_3$  einsetzt und dadurch  $(\neg Rcc \vee Rchc)$  erhält, und zum andern  $hc$  für  $v_1$  und z. B.  $c$  für  $v_3$  einsetzt und dadurch  $(\neg Rchc \vee Rchc)$  erhält. Die Disjunktion beider Formeln ist offenbar allgemeingültig, ihr „aussagenlogische Gerüst“ ist  $((\neg A_0 \vee A_1) \vee (\neg A_1 \vee A_2))$ .

## Unifikation

Die Frage, wie man durch Term-Einsetzungen gleiche atomare Formeln erreichen kann, wird durch das Verfahren der **Unifikation** [unification] geklärt.

**Definition 2.6.6** Sei  $P = \{(\tau_1, \tau'_1), \dots, (\tau_n, \tau'_n)\}$  eine Menge von Paaren von  $\mathcal{L}$ -Termen.  $P$  wird durch eine Folge von Substitutionen  $(\frac{\sigma_1}{v_{i_1}}, \dots, \frac{\sigma_k}{v_{i_k}})$  **unifiziert** [unified], falls

$$\tau_j \left[ \frac{\sigma_1}{v_{i_1}} \right] \left[ \frac{\sigma_2}{v_{i_2}} \right] \dots \left[ \frac{\sigma_k}{v_{i_k}} \right] = \tau'_j \left[ \frac{\sigma_1}{v_{i_1}} \right] \left[ \frac{\sigma_2}{v_{i_2}} \right] \dots \left[ \frac{\sigma_k}{v_{i_k}} \right]$$

für alle  $j = 1, \dots, n$ .<sup>43</sup> Die Menge  $P$  heißt dann **unifizierbar** [unifiable] und die Folge der Substitutionen **Unifikator** [unifier] von  $P$ .

## Satz 2.6.7 (J. A. Robinson, 1965)

Falls  $P$  unifizierbar ist, so gibt es einen minimalen Unifikator oder **Haupt-Unifikator** [most general unifier], aus dem jeder Unifikator durch weitere Substitutionen hervorgeht.

<sup>43</sup>Die Substitutionen werden hier nacheinander von links nach rechts ausgeführt! Man beachte den Unterschied: Bei gleichzeitiger Substitution ist z. B.  $gv_0v_1[\frac{v_1}{v_0} \frac{v_2}{v_1}] = gv_1v_2$ , bei nacheinander ausgeführter Substitution ist  $gv_0v_1[\frac{v_1}{v_0}][\frac{v_2}{v_1}] = gv_1v_1[\frac{v_2}{v_1}] = gv_2v_2$ .

(a) Wenn  $f$  einstellig,  $g$  und  $h$  zweistellig,  $k$  dreistellig und  $c$  eine Konstante ist, dann sind die beiden Terme  $gv_1hv_2v_3$  und  $gfv_3v_4$  durch den Hauptunifikator  $(\frac{fv_3}{v_1}, \frac{hv_2v_3}{v_4})$  unifizierbar und ergeben beide den Term  $gfv_3hv_2v_3$ . Aber auch  $gfv_1hfv_1fv_1$  ist mögliches Ergebnis einer Unifikation durch  $(\frac{ffv_1}{v_1}, \frac{fv_1}{v_2}, \frac{fv_1}{v_3}, \frac{hfv_1fv_1}{v_4})$ .

Dagegen sind  $kv_0gv_0v_1v_3$  und  $kfv_3v_3c$  nicht unifizierbar, da  $v_3$  sowohl mit  $c$  als auch mit  $gv_0v_1$  unifiziert werden müsste, was offenbar nicht geht.

(b) Der Hauptunifikator kann je nach Reihenfolge der Substitutionen verschiedene Gestalt annehmen und ist nur im Ergebnis eindeutig. Wenn z.B.  $P = \{(v_0, fc), (fv_0, v_1)\}$ , dann ist sowohl  $(\frac{fc}{v_0}, \frac{ffc}{v_1})$  als auch  $(\frac{fv_0}{v_1}, \frac{fc}{v_0})$  Hauptunifikator; beide liefern als Ergebnis die unifizierten Paare  $(fc, fc)$  und  $(ffc, ffc)$ .

Anderes Beispiel: Für  $P = \{(v_0, v_1)\}$  sind  $\frac{v_0}{v_1}$  und  $\frac{v_1}{v_0}$  Hauptunifikatoren, aber auch  $(\frac{v_3}{v_0}, \frac{v_3}{v_1})$ .

Folgendes Verfahren entscheidet, ob eine Menge von Paaren unifizierbar ist, und bestimmt ggf. den Hauptunifikator:

1. Sei  $P$  gegeben. Starte mit leerer Folge  $\Sigma$  an Substitutionen.
2. Betrachte ein beliebiges  $(\tau, \tau') \in P$ :
  - (a) Falls  $\tau = f\varrho_1 \dots \varrho_k$  und  $\tau' = g\sigma_1 \dots \sigma_l$  mit Funktionszeichen  $f \neq g$  beginnen (auch Konstanten), dann stoppe mit Ausgabe „ $P$  ist nicht unifizierbar“.
  - (b) Falls  $\tau = f\varrho_1 \dots \varrho_k$  und  $\tau' = f\varrho'_1 \dots \varrho'_k$ , dann ersetze  $P$  durch  $(P \setminus \{(\tau, \tau')\}) \cup \{(\varrho_1, \varrho'_1), \dots, (\varrho_k, \varrho'_k)\}$ .
  - (c) Falls  $\tau = v_i = \tau'$ , ersetze  $P$  durch  $P \setminus \{(\tau, \tau')\}$ .
  - (d) Falls  $\tau = v_i \neq \tau'$  und  $v_i$  in  $\tau'$  vorkommt, dann stoppe mit Ausgabe „ $P$  ist nicht unifizierbar“.
  - (e) Falls  $\tau = v_i$  und  $v_i$  in  $\tau'$  nicht vorkommt, dann hänge  $\frac{\tau'}{v_i}$  an  $\Sigma$  an und ersetze  $P$  durch  $\{(\sigma[\frac{\tau'}{v_i}], \sigma'[\frac{\tau'}{v_i}]) \mid (\sigma, \sigma') \in P, (\sigma, \sigma') \neq (\tau, \tau')\}$ .
  - (f) Die letzten beiden Fälle gelten ebenso mit vertauschten Rollen für  $\tau$  und  $\tau'$ .
3. Falls  $P \neq \emptyset$ : springe zu 2. Falls  $P = \emptyset$ : stoppe mit Ausgabe „Hauptunifikator ist  $\Sigma$ “.

BEWEIS VON SATZ 2.6.7 UND DER KORREKTHEIT DES VERFAHRENS:

Das *Unifikationsmaß* einer Menge  $P$  von Termpaaren sei  $(\#V, \#F, \#P)$ , wobei  $\#V$  die Anzahl der in  $P$  vorkommenden Individuenvariablen sei (jede Variable einmal gezählt, unabhängig wie oft sie vorkommt),  $\#F$  die Anzahl von Vorkommen von Funktionszeichen und Konstanten in  $P$  und  $\#$  die Anzahl von Paaren in  $P$ . Diese Tripel werden lexikografisch geordnet.

(2b) reduziert  $\#F$ , ohne  $\#V$  zu ändern; (2c) reduziert  $\#P$ , ohne  $\#F$  zu ändern und ohne  $\#V$  zu erhöhen; (2e) reduziert  $\#V$ : In jedem Verfahrensschritt wird das Unifikationsmaß also kleiner, und daher stoppt das Verfahren nach endlich vielen Schritten mit  $P = \emptyset$ .

Im Fall (2a) ist klar, dass eine Substitutionsfolge genau dann  $(\varrho_1, \varrho'_1), \dots, (\varrho_k, \varrho'_k)$  unifiziert, wenn sie  $(\tau, \tau')$  unifiziert. Im Fall (2c) wird lediglich ein bereits unifiziertes Termpaar entfernt. Diese beiden Schritte sind also neutral in dem Sinne, dass sie die Unifizierbarkeit bzw. Nicht-Unifizierbarkeit von  $P$  nicht ändern.

Im Fall (2e) muss eine Unifikation  $\tau = v_i$  mit  $\tau'$  identifizieren, also  $\tau'$  für  $v_i$  einsetzen. Somit ist eine ausgegebene Substitutionsfolge zum einen tatsächlich ein Unifikator, und zum andern minimal. Schließlich ist es im Fall (2a) offensichtlich und im Fall (2d) leicht einzusehen, dass keine Unifikation möglich ist. Also ist das Verfahren auch insgesamt korrekt.  $\square$

### Zusammenfassung der Herbrand'schen Methode:

Es gibt folgendes *Semi-Entscheidungsverfahren* (vgl. Definition 3.0.1) für die Allgemeingültigkeit einer  $\mathcal{L}$ -Formel  $\varphi$ :

1. Gegebenenfalls eliminiert man das Gleichheitszeichen, bringt dann  $\varphi$  in pränexer Normalform und den quantorenfreien Teil von  $\varphi$  in DNF.
2. Nun bestimmt man die Herbrand'sche Normalform  $\varphi_*$ .
3. Mithilfe von Unifikation sucht man  $N$  geeignete Term-Einsetzungen.
4. Mit der dualen Resolutionsmethode testet man die Disjunktion der Term-Einsetzungen auf Allgemeingültigkeit.
5. Man durchläuft Schritte 3. und 4. mit wachsendem  $N$ , bis sich eine allgemeingültige Disjunktion von Term-Einsetzungen ergibt. In diesem Fall ist  $\varphi$  allgemeingültig. Andernfalls erhält man keine Antwort und das Verfahren stoppt nicht.

Umgekehrt gilt: wenn  $\varphi$  allgemeingültig ist, gibt es ein  $N$  und geeignete Term-Einsetzungen, welche dies nachweisen. Wenn man die möglichen Term-Einsetzungen systematisch abarbeitet, gibt die Methode für allgemeingültige  $\mathcal{L}$ -Formeln irgendwann ein positives Ergebnis. Für nicht allgemeingültige  $\mathcal{L}$ -Formeln stoppt das Verfahren nicht und gibt daher kein Ergebnis. Daher ist es ein Semi-Entscheidungsverfahren.

Sei  $\mathcal{L} = \{\circ, e\}$  mit einem zweistelligen Funktionssymbol  $\circ$  und einer Konstanten  $e$ . Falls in einer  $\mathcal{L}$ -Struktur  $\mathcal{M}$  die Operation  $\circ^{\mathcal{M}}$  kommutativ ist,  $e^{\mathcal{M}}$  neutrales Element ist und jedes Element ein Linksinverses hat, dann hat natürlich auch jedes Element ein Rechtsinverses. Genaueres Hinschauen zeigt, dass bereits aus den beiden schwächeren Voraussetzungen  $\forall v_0 \forall v_1 (v_0 \circ v_1 \doteq e \leftrightarrow v_1 \circ v_0 \doteq e)$  und  $\forall v_0 \exists v_1 v_0 \circ v_1 \doteq e$  das Ergebnis  $\forall v_0 \exists v_1 v_1 \circ v_0 \doteq e$  folgt. Dies soll nun mit der Herbrand'schen Methode nachvollzogen werden, d. h. es soll gezeigt werden, dass die  $\mathcal{L}$ -Formel

$$\varphi = ((\forall v_0 \forall v_1 (v_0 \circ v_1 \doteq e \leftrightarrow v_1 \circ v_0 \doteq e) \wedge \forall v_0 \exists v_1 v_0 \circ v_1 \doteq e) \rightarrow \forall v_0 \exists v_1 v_1 \circ v_0 \doteq e)$$

allgemeingültig ist.

(1) Der erste Schritt würde nun darin bestehen, die Gleichheitsgesetze hinzuzunehmen und  $\varphi$  zu ersetzen durch

$$\begin{aligned} & ((\forall v_0 v_0 \doteq v_0 \wedge \forall v_0 \forall v_1 (v_0 \doteq v_1 \rightarrow v_1 \doteq v_0) \\ & \quad \wedge \forall v_0 \forall v_1 \forall v_2 ((v_0 \doteq v_1 \wedge v_1 \doteq v_2) \rightarrow v_0 \doteq v_2) \\ & \quad \wedge \forall v_0 \dots \forall v_3 ((v_0 \doteq v_2 \wedge v_1 \doteq v_3) \rightarrow v_0 \circ v_1 \doteq v_2 \circ v_3)) \rightarrow \varphi) \end{aligned}$$

Es stellt sich aber heraus, dass dies in diesem speziellen Fall nicht nötig ist. Damit es überschaubar bleibt, arbeite ich deshalb mit dem originalen  $\varphi$ .<sup>44</sup> Außerdem schreibe ich der besseren Lesbarkeit halber  $\neq$  für eine verneinte Gleichheit und  $u, \dots, z$  für die Individuenvariablen.

Als nächstes wird  $\varphi$  in pränex Normalform mit quantoremfreiem Teil in DNF gebracht. Dabei bringt man die Allquantoren am besten möglichst weit nach außen, also etwa:

$$\forall y \exists w \forall x \exists u \exists v \exists z ((u \circ v \doteq e \wedge v \circ u \not\doteq e) \vee (u \circ v \not\doteq e \wedge v \circ u \doteq e) \vee w \circ x \not\doteq e \vee z \circ y \doteq e)$$

(2) Die Herbrand'sche Normalform  $\varphi_*$  ist dann

$$\exists w \exists u \exists v \exists z ((u \circ v \doteq e \wedge v \circ u \not\doteq e) \vee (u \circ v \not\doteq e \wedge v \circ u \doteq e) \vee w \circ fw \not\doteq e \vee z \circ c \doteq e)$$

mit neuer Konstante  $c$  und neuem einstelligen Funktionszeichen  $f$ .

Man hat nun unendlich viele  $\mathcal{L}$ -Terme  $e, c, fe, fc, eoe, eoc, coe, coc, ffe, ffc, eofe \dots$ , die man in allen möglichen Kombinationen für  $u, v, w$  und  $z$  in  $\varphi_*$  einsetzen kann. Der Überschaubarkeit der Darstellung wegen wähle ich hier einige zielführende Einsetzungen aus und erläutere, wie man auf diese Einsetzungen kommt. Ein automatisiertes Vorgehen wird dagegen systematisch alle möglichen Einsetzungen vornehmen, die an einer Stelle eine atomare Teilformel  $\psi$  und an anderer Stelle (und ggf. durch eine andere Einsetzung) die negiert-atomare Teilformel  $\psi$  entstehen lassen.

(3a) Für eine erste Einsetzung sucht man eine atomare Teilformel und eine negierte atomare Teilformel „gleicher Gestalt“<sup>45</sup>, die durch geeignete Term-Einsetzungen in der anschließenden Resolution einen Resolutionsschritt ermöglichen. Man wählt etwa  $v \circ u \not\doteq e$  und  $z \circ c \doteq e$ , und unifiziert also  $\{(v, z), (u, c)\}$ , was durch den Hauptunifikator  $(\frac{c}{u}, \frac{z}{v})$  geschieht. Setzt man nach dieser Substitution einen geschlossenen  $\mathcal{L}$ -Term wie zum Beispiel  $c$  für  $w$  und  $z$  in  $\varphi_*$  ein, erhält man

$$((c \circ c \doteq e \wedge c \circ c \not\doteq e) \vee (c \circ c \not\doteq e \wedge c \circ c \doteq e) \vee c \circ fc \not\doteq e \vee c \circ c \doteq e)$$

(4a) Die atomaren Teilformeln werden nun durch Aussagenvariablen ersetzt und die Konjunktionsglieder als duale Klauseln aufgefasst. Durch  $A_0$  für  $c \circ c \doteq e$  und  $A_1$  für  $c \circ fc \doteq e$  erhält man die dualen Klauseln

$$\{A_0, \neg A_0\}, \{\neg A_0, A_0\}, \{\neg A_1\}, \{A_0\}.$$

Die Menge dieser dualen Klauseln ist bereits unter Resolution abgeschlossen und enthält nicht die leere Klausel. Also hat man noch keinen Beweis der Allgemeingültigkeit von  $\varphi$ .

(3b) Um durch die nächste Term-Einsetzung einen neuen Resolutionsschritt zu ermöglichen, möchte man eine Klausel mit positivem  $A_1$  erhalten, im Idealfall sogar  $\{A_1\}$ . Positive Einerklauseln bekommt man nur aus der Teilformel  $z \circ c \doteq e$ . Der Versuch, daraus  $A_1$ , also  $c \circ fc \doteq e$  zu erhalten, scheitert aber, da  $\{(z, c), (c, fc)\}$  nicht unifizierbar ist.

(3c) Im zweiten Versuch, eine Klausel mit positivem  $A_1$  zu erhalten, betrachtet man  $w \circ fw \not\doteq e$  und  $v \circ u \doteq e$  und unifiziert  $\{(v, w), (u, fw)\}$  durch den Hauptunifikator  $(\frac{w}{v}, \frac{fw}{u})$ . Setzt man nach dieser Substitution  $c$  für  $w$  (und auch für  $z$ ) in  $\varphi_*$  ein, erhält man

$$((fc \circ c \doteq e \wedge c \circ fc \not\doteq e) \vee (fc \circ c \not\doteq e \wedge c \circ fc \doteq e) \vee c \circ fc \not\doteq e \vee c \circ c \doteq e)$$

zusammen mit der Einsetzung aus (3a) also die Disjunktion

$$((c \circ c \doteq e \wedge c \circ c \not\doteq e) \vee (c \circ c \not\doteq e \wedge c \circ c \doteq e) \vee c \circ fc \not\doteq e \vee c \circ c \doteq e \vee \\ (fc \circ c \doteq e \wedge c \circ fc \not\doteq e) \vee (fc \circ c \not\doteq e \wedge c \circ fc \doteq e) \vee c \circ fc \not\doteq e \vee c \circ c \doteq e)$$

(was nun eine Disjunktion von Term-Einsetzungen ist, wie sie im Satz von Herbrand vorkommt).

(4c) Die neue atomare Formel  $fc \circ c \doteq e$  wird durch  $A_2$  ersetzt; zusammen mit der ersten Term-Einsetzung erhält man also die dualen Klauseln

$$\{A_0, \neg A_0\}, \{\neg A_0, A_0\}, \{\neg A_1\}, \{A_0\}, \{A_2, \neg A_1\}, \{\neg A_2, A_1\}, \{\neg A_1\}, \{A_0\}.$$

Der Abschluss der Menge der dualen Klauseln unter Resolution ist

$$\{\{A_0\}, \{\neg A_1\}, \{\neg A_2\}, \{A_1, \neg A_2\}, \{\neg A_1, A_2\}, \{A_0, \neg A_0\}, \{A_1, \neg A_1\}, \{A_2, \neg A_2\}\}$$

(3d) Man sieht, dass man zum Ziel kommt, wenn man die Klausel  $\{A_2\}$  erhält. Will man analog zu (3b)  $A_2$  aus  $z \circ c \doteq e$  erhalten, also  $fc \circ c \doteq e$ , muss man  $\{(z, fc), (c, c)\}$  unifizieren, was durch den Hauptunifikator  $\frac{fc}{z}$  geschieht. Damit in (3c) die  $A_2$  entsprechende Teilformel entsteht, hat man die Ersetzungen  $(\frac{w}{v}, \frac{fw}{u}, \frac{c}{w})$  durchgeführt. Zusammen erhält man

$$((fc \circ c \doteq e \wedge c \circ fc \not\equiv e) \vee (fc \circ c \not\equiv e \wedge c \circ fc \doteq e) \vee c \circ fc \not\equiv e \vee fc \circ c \doteq e)$$

(4d) Damit bekommt man eine neue duale Klausel  $\{A_2\}$ , im Abschluss unter Resolution also die leere Klausel und damit einen Beweis der Allgemeingültigkeit von  $\varphi$ .

(5) Wenn man genauer hinsieht, erkennt man, dass man bereits durch *eine* Einsetzung (also  $N = 1$ , nämlich  $fc$  für  $u$  und  $z$  und  $c$  für  $v$  und  $w$ ) zum Ziel kommt. Im Beispiel ist dargestellt, wie man sich durch ein ansatzweise systematisches Vorgehen über die Schritte  $N = 1, 2, 3$  dem Ziel annähert. Das funktioniert hier gut, weil man im Prinzip nur eine Möglichkeit hat, das Resolutionsverfahren weiterzutreiben. Bei umfangreicheren Formeln wird man nicht so zielgerichtet vorgehen können.

Die gerade angegebene Einsetzung, die bereits alleine ausreicht, kann man übrigens aus dem algebraischen Beweis bekommen: Dafür würde man sich zunächst ein beliebiges  $c \in M$  wählen. Nach Voraussetzung gibt es zu jedem  $m \in M$  ein Linksinverses. Dieses muss nicht eindeutig bestimmt sein, aber man kann sich eine Funktion  $f : M \rightarrow M$  wählen, welches für jedes  $m \in M$  ein Linksinverses angibt, also  $f(m) \circ m = e$ . (Diese Schritte entsprechen der Einführung der Skolem-Funktionen!) Nun gilt also  $f(c) \circ c = e$  und nach Voraussetzung auch  $f(c) \circ c = e \Leftrightarrow c \circ f(c) = e$ , also folgt  $c \circ f(c) = e$ . Somit hat man ein Rechtinverses gefunden.

<sup>44</sup>Würde man die volle Kommutativität  $\forall v_0 \forall v_1 (v_0 \circ v_1 \doteq v_1 \circ v_0)$  anstelle der ersten Voraussetzung nehmen, bräuchte man dagegen zumindest die Transitivität der Gleichheit.

<sup>45</sup>In diesem Beispiel haben alle atomaren Teilformeln die gleiche Gestalt  $\tau_1 \circ \tau_2 \doteq e$ .



### 3 Berechenbarkeit und Entscheidbarkeit

Ziel dieses Abschnitts ist es, die *Unentscheidbarkeit* [undecidability] der Prädikatenlogik zu zeigen: Es gibt unter gewissen Minimal-Voraussetzungen an die Sprache  $\mathcal{L}$  kein allgemeines Verfahren, mit dem man für beliebige  $\mathcal{L}$ -Formeln entscheiden oder berechnen kann, ob sie allgemeingültig sind oder nicht. Um dies beweisen zu können, muss zunächst spezifiziert werden, was mit „allgemeinem Verfahren“ bzw. mit „entscheidbar“ oder „berechenbar“ gemeint ist.

Es gibt viele Möglichkeiten, diese Begriffe zu präzisieren. Es haben sich aber bisher alle hinreichend starken Präzisierungen als äquivalent herausgestellt, so dass man davon ausgeht, dass man damit den intuitiven Berechenbarkeitsbegriff adäquat beschrieben hat (dies ist die sogenannte „Church’sche These“). Zwei dieser Präzisierungen sollen hier vorgestellt werden: Berechenbarkeit durch *Turing-Maschinen* und Berechenbarkeit im Sinne *rekursiver Funktionen*. Der Beweis der Äquivalenz dieser beiden Konzepte kann in dieser Vorlesung allerdings nicht geleistet werden: Es ist zwar klar, wie der Beweis aufgebaut sein muss, ihn durchzuführen ist aber eine langwierige Detailarbeit.

Als erster Überblick sollen die wichtigsten Begriffe der Berechenbarkeitstheorie und ihre Zusammenhänge zunächst auf einer intuitiven Ebene dargestellt werden. Die folgenden Definitionen sind informell; sie sprechen von *Algorithmen* oder *Verfahren*. Um sie zu präzisieren, muss man festlegen, was ein Algorithmus ist bzw. wie die zugelassenen Verfahren aussehen. Typischerweise werden es schrittweise Prozeduren sein, wobei die einzelnen Schritte aus einer festgelegten Liste endlich vieler Möglichkeiten stammen.

**Definition 3.0.1 (informell)** Sei  $M$  ein festgewählte Menge.

(a) Eine Teilmenge  $X \subseteq M$  heißt **entscheidbar** [decidable], wenn es einen Algorithmus gibt, der aus der Eingabe eines beliebigen Elements  $m \in M$  die Ausgabe „ja“ produziert, wenn  $m \in X$ , und „nein“, wenn  $m \notin X$ .

(b)  $X \subseteq M$  heißt **semi-entscheidbar** oder **aufzählbar** [semi-decidable / enumerable], wenn es einen Algorithmus gibt, der aus der Eingabe eines beliebigen Elements  $m \in M$  die Ausgabe „ja“ produziert, wenn  $m \in X$ , und nicht terminiert, wenn  $m \notin X$ .

(c) Eine Funktion  $f : M \rightarrow N$  heißt **berechenbar** [computable], wenn es einen Algorithmus gibt, der aus der Eingabe eines beliebigen Elements  $m \in M$  die Ausgabe  $f(m) \in N$  produziert.

Die Menge  $M$  ist für den Entscheidbarkeitsbegriff relevant, daher müsste man eigentlich genauer von „(semi-)entscheidbar in  $M$ “ sprechen. Denn jede Menge  $X$  ist als Teilmenge von sich selbst natürlich entscheidbar, da der Algorithmus in diesem Fall nur konstant „ja“ ausgeben muss. Es wird aber im Kontext immer festgelegt sein, welche umgebende Menge  $M$  gemeint ist.

Typischerweise arbeitet man mit einem endlichen Alphabet  $A$  und betrachtet als  $M$  die Menge  $A^*$  aller endlichen Wörter über  $A$ , d. h. die Menge aller endlichen Folgen von Elementen aus  $A$ . Eine Vergrößerung des Alphabets ist unschädlich, d. h. ändert nicht die Entscheidbarkeit: Wenn  $X \subseteq A^*$  entscheidbar ist und  $A \subseteq B$ , kann man zunächst bei Eingabe eines Wortes  $w \in B^*$  prüfen, ob ein Symbol aus  $B \setminus A$  in  $w$  vorkommt (dann Ausgabe „nein“), und sonst den Entscheidungsalgorithmus für  $A$  anwenden. Die Umkehrung ist etwas weniger klar, aber im Wesentlichen kann man einen für  $B$  funktionierenden Algorithmus auf  $A$  einschränken.

Für nicht-leeres  $A$  ist  $A^*$  eine abzählbar unendliche Menge und kann daher mit  $\mathbb{N}$  identifiziert werden. Man kann sich daher auch auf den Fall  $X \subseteq \mathbb{N}$  zurückziehen bzw. auf Funktionen

$\mathbb{N} \rightarrow \mathbb{N}$  oder auch  $\mathbb{N}^k \rightarrow \mathbb{N}$ , was insbesondere bei der Betrachtung der rekursiven Funktionen der Fall sein wird. Insbesondere kann man immer ohne Probleme  $0, 1 \in A^*$  annehmen.

Für alle gebräuchlichen Berechenbarkeitsbegriffe gelten die folgenden Zusammenhänge:

**Lemma 3.0.2** Eine Teilmenge  $X \subseteq M$  ist genau dann entscheidbar, wenn  $X$  und  $M \setminus X$  semi-entscheidbar sind, und genau dann, wenn die charakteristische Funktion  $\chi_X$  mit

 $\chi_X$ 

$$\chi_X(m) = \begin{cases} 1 & m \in X \\ 0 & m \notin X \end{cases}$$

berechenbar ist.

**BEWEIS FÜR DEN INTUITIVEN ENTSCHEIDBARKEITSBEGRIFF:** Wenn  $X$  entscheidbar ist, ist natürlich auch  $M \setminus X$  entscheidbar: Man muss nur die Ausgaben „ja“ und „nein“ vertauschen. Ein Entscheidungsverfahren für  $X$  (bzw.  $M \setminus X$ ) wird zu einem Semi-Entscheidungsverfahren, indem man die Ausgabe „nein“ durch eine unendliche Schleife ersetzt.

Hat man Semi-Entscheidungsverfahren für  $X$  und  $M \setminus X$ , baut man daraus ein Entscheidungsverfahren für  $X$ , indem man abwechselnd einen Schritt im Semi-Entscheidungsverfahren für  $X$  und einen Schritt im Semi-Entscheidungsverfahren für  $M \setminus X$  ansetzt. Eines der beiden Verfahren wird irgendwann eine positive Antwort liefern. Ist es das Verfahren für  $X$ , gibt man „ja“ aus, andernfalls „nein“.

Ein Entscheidungsverfahren für  $X$  und ein Berechnungsverfahren für  $\chi_X$  gehen ineinander über, indem man die Ausgaben „ja“/„nein“ mit „1“/„0“ tauscht.  $\square$

Wenn man in analoger Weise die Semi-Entscheidbarkeit durch die Berechenbarkeit einer Funktion beschreiben will, braucht man *partielle Funktionen*:

**Definition 3.0.3** Eine **partielle Funktion** [partial function]  $f : M \dashrightarrow N$  ist eine Funktion  $f' : D \rightarrow N$  für eine Teilmenge  $D \subseteq M$ , den Definitionsbereich von  $f$ .

 $M \dashrightarrow N$ 

Für  $m \in D$  setzt man  $f(m) := f'(m)$  und sagt, dass  $f$  in  $m$  definiert oder bestimmt ist.

Für  $m \notin D$  sagt man, dass  $f$  in  $m$  nicht definiert oder unbestimmt ist.

Kurz sagt man im zweiten Fall auch, dass  $f(m)$  unbestimmt ist, obwohl  $f(m)$  dann keine festgelegte Größe ist.

**Lemma 3.0.4** Eine Teilmenge  $X \subseteq M$  ist genau dann semi-entscheidbar, wenn die „partielle charakteristische Funktion“  $\overline{\chi}_X$ <sup>46</sup> mit

 $\overline{\chi}_X$ 

$$\overline{\chi}_X(m) = \begin{cases} 1 & m \in X \\ \text{unbestimmt} & m \notin X \end{cases}$$

berechenbar ist.

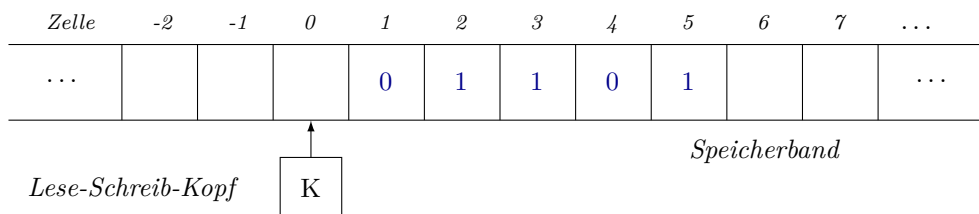
**BEWEIS FÜR DEN INTUITIVEN ENTSCHEIDBARKEITSBEGRIFF:** Analog zu Lemma 3.0.2.  $\square$

<sup>46</sup>Weder Name noch Bezeichnungsweise sind Standard-Notationen!

### 3.1 Turing-Maschinen

*Turing-Maschinen* [Turing machines] stellen ein von Alan Turing 1936 – also deutlich vor der Konstruktion der ersten Computer – entwickeltes, besonders einfaches abstraktes Rechnermodell dar. Es hat sich im Laufe der Zeit aber herausgestellt, dass der Begriff der Berechenbarkeit durch Turing-Maschinen trotz der Primitivität des Rechnermodells so umfassend wie komplexere Berechenbarkeitsbegriffe ist (vgl. die *Church’sche These* S. 94).

Die „*Hardware*“ einer Turing-Maschine aus einem beidseitig unendlichen *Speicherband* mit einzelnen, nebeneinander liegenden *Speicherzellen* (die man sich also als mit den ganzen Zahlen  $\mathbb{Z}$  indiziert vorstellen kann), und einem *Lese-Schreib-Kopf*, der in jedem Berechnungsschritt auf genau eine Speicherzelle zugreift.



Die „*Programmiersprache*“ einer Turing-Maschine arbeitet mit einem nicht-leeren endlichen *Alphabet*  $A$ , das nicht das Symbol  $\emptyset$  enthalten soll. In jeder Zelle des Speicherbandes steht entweder genau ein Zeichen aus  $A$  oder die Zelle ist leer. Kennzeichnet man leere Zellen durch  $\emptyset$ , steht also in jeder Zelle genau ein Symbol aus  $A^+ := A \cup \{\emptyset\}$ . (Bei einem Alphabet mit  $0 \in A$  muss man also zwischen einer leeren Zelle und einer Zelle mit dem Wert 0 unterscheiden.) Sodann gibt es eine endliche Menge  $Z$  an *Zuständen*, darunter den *Startzustand*  $z_0 \in Z$  und den *Endzustand*  $z_{\text{end}} \in Z$ . Zu jedem Zeitpunkt befindet sich die Maschine in genau einem Zustand. Der Zustand der Maschine ist also eine zusätzliche abstrakte Gegebenheit, die unabhängig von der Position des Lese-Schreib-Kopfes und der aktuellen Information auf dem Speicherband ist.

Das „*Programm*“ einer Turing-Maschine besteht aus einer *Übergangsfunktion*

$$U : Z \times A^+ \rightarrow Z \times A^+ \times \{-1, 0, 1\}$$

wobei  $U(z_{\text{end}}, a) = (z_{\text{end}}, a, 0)$  für alle  $a \in A^+$ . Die Übergangsfunktion kann man sich vorstellen als eine Liste von Zuweisungen  $(z, a) \mapsto (z', a', p')$ .

Der Programmlauf einer Turing-Maschine funktioniert dann folgendermaßen:

- In der Startkonfiguration steht der Lese-Schreib-Kopf in einer definierten Startzelle. Alle Zellen links bis einschließlich der Startzelle sind leer. Unmittelbar in den  $l$  Zellen rechts von der Startzelle stehen Symbole aus  $A$ , die ein Wort  $w \in A^*$  der Länge  $l$  als *Eingabe* des Programms bilden. Alle weiteren Zellen rechts davon sind leer (wie im Bild oben angedeutet). Die Turing-Maschine befindet sich außerdem im Startzustand  $z_0$ .
- In jedem Programmschritt befindet sich die Turing-Maschine in genau einem Zustand  $z$  und liest das Symbol  $a \in A^+$ , welches in der Zelle  $n$  steht, an der sich der Lese-Schreib-Kopf befindet. Das Programm wertet dann  $U(z, a) = (z', a', p')$  aus, die Maschine ersetzt das Symbol  $a$  in Zelle  $n$  durch  $a'$ , bewegt den Lese-Schreib-Kopf in die Zelle  $n + p'$  und geht in den Zustand  $z'$  über. Damit ist der Programmschritt beendet und der nächste folgt.

- Sobald sich weder der Zustand, noch die Position des Kopfes, noch die Symbole auf dem Band mehr ändern, sagt man *Die Maschine hält an* oder *Die Maschine stoppt*. Per Definition stoppt die Maschine also stets dann, wenn sie den Endzustand  $z_{\text{end}}$  erreicht hat. In diesem Fall sagt man auch, dass die Maschine die Eingabe *akzeptiert*.
- Falls die Maschine im Endzustand stoppt und der Bandinhalt dann aus einem Wort  $w' \in A^*$  besteht, das (analog zur Startkonfiguration) links und rechts von ausschließlich leeren Zellen begrenzt ist, nennt man  $w'$  die *Ausgabe*. Sie kann an einer beliebigen Stelle des Bandes stehen, nicht notwendigerweise rechts der Startzelle.

Es gibt verschiedenste Varianten von Turing-Maschinen (die sich aber alle gegenseitig simulieren lassen). Üblich sind etwa:

- Maschinen mit mehreren Bändern, z. B. für Eingabe, Ausgabe und Zwischenberechnungen.
- Maschinen mit halbseitig unendlichen Bändern, also mit durch  $\mathbb{N}$  statt  $\mathbb{Z}$  indizierten Zellen.
- Maschinen, die ein zusätzliches spezielles Symbol nutzen, das Beginn und Ende von Ein- und Ausgabe markiert.

Bei der hier vorgestellten Variante geht es mir vor allem um das grundlegende Prinzip. Andere Varianten sind für Beweise besser geeignet, verlangen aber eine etwas umfangreichere Definition.

**Definition 3.1.1**

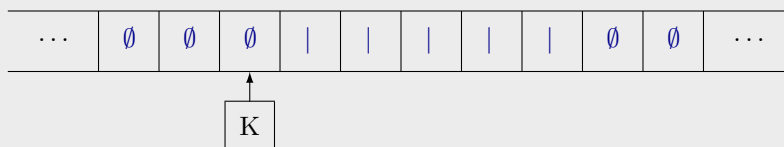
(a) Eine (partielle) Funktion  $f : A^* \rightarrow B^*$  heißt **Turing-berechenbar** [Turing computable], wenn es eine Turing-Maschine gibt, die bei Eingabe  $w \in A^*$  stoppt und die Ausgabe  $f(w)$  liefert, falls  $f$  in  $w$  definiert ist, und nicht stoppt, falls  $f$  in  $w$  unbestimmt ist.

(b)  $X \subseteq A^*$  heißt **Turing-entscheidbar** [Turing decidable], wenn es eine Turing-Maschine gibt, die bei Eingabe  $w \in X$  im Endzustand stoppt und bei Eingabe  $w \notin X$  in einem anderen Zustand stoppt.

(c)  $X \subseteq A^*$  heißt **Turing-semi-entscheidbar** oder **Turing-aufzählbar** [Turing semi-decidable / Turing enumerable] wenn es eine Turing-Maschine gibt, die bei Eingabe  $w \in X$  im Endzustand stoppt und bei Eingabe  $w \notin X$  nicht stoppt.

Wenn man die Turing-Berechenbarkeit einer charakteristischen Funktion betrachtet, müssen nicht unbedingt die beiden Symbole 0 und 1 im Alphabet  $A$  liegen, sondern es reicht aus, dass man sie durch geeignete Wörter in  $A^*$  kodiert, beispielsweise 0 durch das leere Wort und 1 durch irgendein Symbol  $a_1 \in A$ .

(a) Sei  $A$  das unäre Alphabet  $A = \{ | \}$ . Die Teilmenge  $X \subseteq A^*$  aller Wörter gerader Länge ist Turing-entscheidbar:

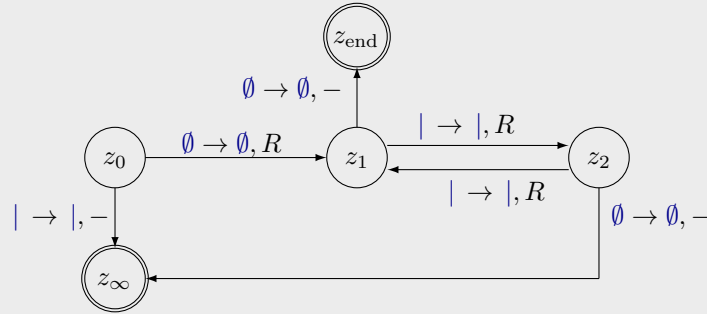


Die Turing-Maschine hat fünf Zustände:  $Z = \{z_0, z_1, z_2, z_\infty, z_{\text{end}}\}$  und folgende Übergangsfunktion:

$$\begin{array}{ll}
 (z_0, \emptyset) \mapsto (z_1, \emptyset, 1) & (z_0, |) \mapsto (z_\infty, |, 0) \\
 (z_1, \emptyset) \mapsto (z_{\text{end}}, \emptyset, 0) & (z_1, |) \mapsto (z_2, |, 1) \\
 (z_2, \emptyset) \mapsto (z_\infty, \emptyset, 0) & (z_2, |) \mapsto (z_1, |, 1) \\
 (z_\infty, \emptyset) \mapsto (z_\infty, \emptyset, 0) & (z_\infty, |) \mapsto (z_\infty, |, 0)
 \end{array}$$

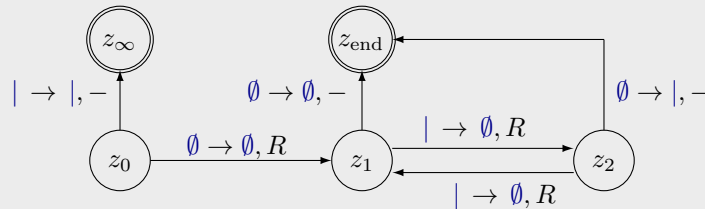
$z_\infty$  ist also ein Zustand, in dem die Maschine stoppt, ohne dass es der Endzustand ist.

Übersichtlicher stellt man die Übergangsfunktion bzw. die Funktionsweise der Turing-Maschine durch ein Flussdiagramm dar:

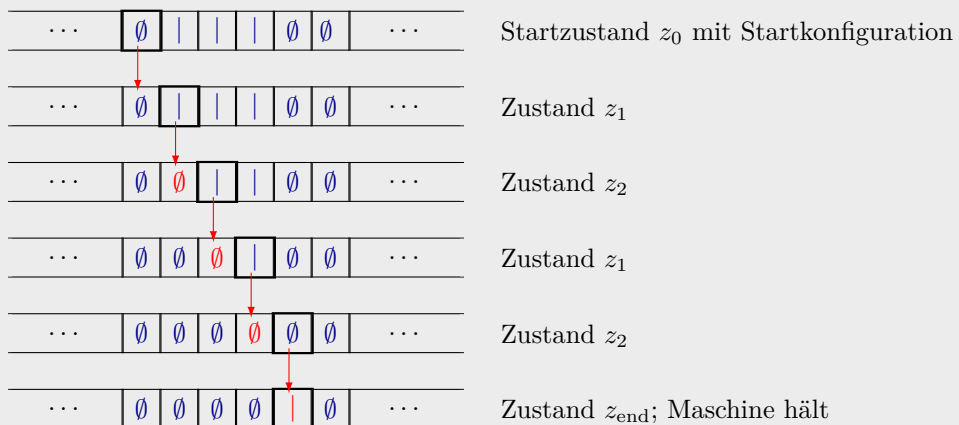


Zustände werden dabei in Kreisen dargestellt; doppelte Kreise kennzeichnen Zustände, in denen die Maschine stoppt. Der Übergang von einem Zustand in den anderen wird durch Pfeile gekennzeichnet; an dem Pfeil steht, welches Symbol für den Übergang verantwortlich ist und wie es überschrieben wird, und auch, wie die Maschine sich bewegt. Um die Funktionsweise intuitiver verständlich zu machen, schreibe ich für die Bewegungen des Lese-Schreib-Kopfes  $L, -, R$  statt  $-1, 0, 1$ .

(b) Eine Modifikation dieser Turing-Maschine berechnet den Rest einer unär geschriebenen Zahl modulo 2, gibt also einen Strich (= 1) aus, wenn die Eingabe aus einer ungeraden Anzahl an Strichen besteht, und das leere Band (= 0), wenn die Eingabe aus einer geraden Anzahl an Strichen besteht:



Der Lauf der Maschine ist dann wie folgt bei Eingabe  $3 = |||$ , wobei die Position des Lese-Schreib-Kopfes hier durch das dicker eingerahmte Kästchen markiert ist und das Ersetzen der Symbole durch den roten Pfeil (und neue Symbole rot geschrieben sind):



Damit kann man nun die **These von Church** [Church's thesis] für Turing-Maschinen formulieren:

*Jede im intuitiven Sinne berechenbare Funktion ist Turing-berechenbar.*

Diese These ist für viele Maschinenmodelle und Programmiersprachen verifiziert und wird allgemein akzeptiert. Sie wird in den folgenden Abschnitten in dem Sinne verwendet, dass nur die intuitive Berechenbarkeit dargelegt wird, wo man konkrete Turing-Maschinen angeben müsste (und könnte).

Beispiele sind die Beweise von Lemma 3.0.2 für Turing-Entscheidbarkeit und -Berechenbarkeit. Unter Benutzung der Church'schen These reichen die angestellten Überlegungen. Andernfalls müsste man konkret zeigen, wie die benötigten Turing-Maschinen aussehen:

Einfach ist es, aus einem Turing-Entscheidungsverfahren für  $X$  ein Semi-Entscheidungsverfahren zu machen: Wenn die Turing-Maschine in einem Zustand  $z' \neq z_{\text{end}}$  hält, ändert man die Übergangsfunktion so, dass sie im Zustand  $z'$  dauerhaft weiter nach rechts läuft und dadurch nicht stoppt. Um aus Turing-Maschinen  $T_X$  und  $T_{M \setminus X}$ , die Semi-Entscheidungsverfahren für  $X$  bzw.  $M \setminus X$  liefern, ein Entscheidungsverfahren für  $X$  zu machen, verdoppelt man zunächst eine Eingabe  $a_1 \dots a_n \in A^*$  zu  $a_1 a_1 \dots a_n a_n$  und simuliert auf den Zellen mit geradem Index die Maschine  $T_X$ , auf den Zellen mit ungeradem Index die Maschine  $T_{M \setminus X}$ , die abwechselnd laufen. Dies konkreter durchzuführen, ist sehr mühsam (deutlich einfacher geht es mit einer Mehrbänder-Maschine). Sobald eine der beiden Maschinen ihren eigentlichen Endzustand erreicht, lässt man die neue Turing-Maschine in einem geeigneten Zustand anhalten.

Aus einer Turing-Maschine, die die charakteristische Funktion  $\chi_X$  berechnet, erhält man relativ einfach eine Entscheidungsmaschine für  $X$ : Statt bei Ausgabe 0 in den Endzustand zu gehen, lässt man die Maschine in einem anderen Zustand halten. Hat man umgekehrt eine Maschine, die  $X$  entscheidet, muss man an die beiden Zustände, in denen die Maschine halten kann, ein Programm anschließen, das das Band löscht und in einem Fall **1**, im andern Fall **0** auf Band schreibt und dann jeweils in den Endzustand geht. (Schwierig ist hier, dass man eine Kontrolle darüber haben muss, bis wohin man das Band löschen muss. Einfacher ist dies mit einem speziellen Symbol, das den beschriebenen Bereich des Bandes markiert. Andernfalls muss man dafür sorgen, dass die Maschine keine beliebig großen Abschnitte mit leeren Zellen zwischen gefüllten Zellen produziert.)

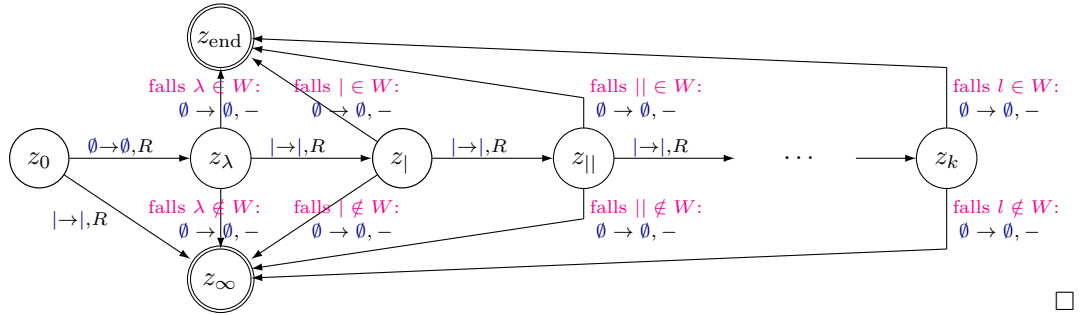
### 3.2 Das Halteproblem und die Unentscheidbarkeit der Prädikatenlogik

Sei  $A$  nun stets ein endliches, nicht-leeres Alphabet.

**Lemma 3.2.1** *Jede endliche Teilmenge von  $A^*$  ist Turing-entscheidbar.*

BEWEIS: Zu  $W = \{w_1, \dots, w_n\} \subseteq A^*$  gibt es eine Turing-Maschine  $\mathcal{C}_W$ , die  $W$  wie folgt entscheidet: Neben dem Startzustand  $z_0$ , dem Endzustand  $z_{\text{end}}$  und einem weiteren Haltezustand  $z_\infty$  gibt es für jedes Anfangsstück  $v$  eines Wortes aus  $W$  einen Zustand  $z_v$  (inklusive des leeren Wortes  $\lambda$ ). Die Maschine  $\mathcal{C}_W$  liest das Eingabewort  $w$  von links nach rechts und merkt sich durch den Zustand die bisher gelesenen Zeichen. Am Ende von  $w$  angelangt geht  $\mathcal{C}_W$  in den Endzustand, wenn  $w \in W$ , und sonst in  $z_\infty$ .

Hier ein Flussdiagramm für das unäre Alphabet  $A = \{\mid\}$ . Für größere Alphabete verzweigt es sich in jedem Zustand  $z_v$  bis zu  $|A|$ -fach. Sei  $l$  (d. h.  $l$  Striche) das längste Wort in  $W$ .



Eine unendliche Menge  $M$  heißt *abzählbar* [countable], falls es eine Bijektion  $\mathbb{N} \rightarrow M$  gibt, und andernfalls *überabzählbar* [uncountable].<sup>47</sup>

**Satz 3.2.2 (Cantor 1877)** Die Potenzmengen  $\text{Pot}(\mathbb{N})$  und  $\text{Pot}(A^*)$  für das endliche Alphabet  $A \neq \emptyset$  sind überabzählbar.

BEWEIS: Eine Bijektion  $\beta : \mathbb{N} \rightarrow A^*$  lässt sich zu einer Bijektion  $\tilde{\beta} : \text{Pot}(\mathbb{N}) \rightarrow \text{Pot}(A^*)$  fortsetzen durch  $\tilde{\beta}(M) = \{\beta(n) \mid n \in M\}$  für  $M \subseteq \mathbb{N}$ . Außerdem steht  $\text{Pot}(\mathbb{N})$  in Bijektion mit der Menge der unendlichen  $\{0, 1\}$ -Folgen, indem man jeder Teilmenge  $M \subseteq \mathbb{N}$  die charakteristische Funktion  $\chi_M$  zuordnet.

Es reicht also, die Überabzählbarkeit der Menge der unendlichen  $\{0, 1\}$ -Folgen zu zeigen, was mit *Cantors Diagonalargument* funktioniert: Angenommen  $\beta : \mathbb{N} \rightarrow \text{Abb}(\mathbb{N}, \{0, 1\})$  ist eine Bijektion. Dann konstruiert man eine Folge  $\alpha$ , die an der  $n$ -ten Stelle verschieden von  $\beta(n)$  ist, indem man  $\alpha(n) := 1 - \beta(n)(n)$  setzt. Nach Annahme gibt es ein  $n_0 \in \mathbb{N}$  mit  $\alpha = \beta(n_0)$ . Dann gilt  $\alpha(n_0) = \beta(n_0)(n_0)$ , aber nach Definition von  $\alpha$  auch  $\alpha(n_0) = 1 - \beta(n_0)(n_0) \neq \beta(n_0)(n_0)$ : Widerspruch! □

Jede Turing-Maschine  $\mathcal{C}$ , die mit dem Alphabet  $A$  arbeitet, lässt sich durch eine endliche Zeichenfolge  $\ulcorner \mathcal{C} \urcorner$  über einem festen endlichen Alphabet  $A_T$  beschreiben: Man wählt zum Beispiel  $A_T = A \cup \{0, \dots, 9, \$\}$ , gibt jedem Zustand  $z$  der Turing-Maschine eine Dezimalzahl  $\#z$  als Nummer ( $z_0$  und  $z_{\text{end}}$  feste Nummern, etwa 0 und 1), und schreibt jeden Eintrag  $(z, a) \mapsto (z', a', n)$  der Übergangsfunktion von  $\mathcal{C}$  als Zeichenfolge  $\#z \wedge a \wedge \#z' \wedge a' \wedge n \wedge \$$ <sup>48</sup>, wobei man für  $n = -1$  eine Ziffer  $\neq 0, 1$  schreibt, z. B. 9. Setzt man alle solchermaßen kodierten Einträge der Übergangsfunktion hintereinander, erhält man eine Möglichkeit für  $\ulcorner \mathcal{C} \urcorner$ . (Man kann auch alles auf berechenbare Weise in  $A^*$  kodieren, bräuchte das Alphabet also nicht unbedingt zu vergrößern.)

Nun gibt es nur abzählbar viele Wörter über  $A_T$ , also nur abzählbar viele Turing-Maschinen, die mit  $A$  arbeiten, und damit auch nur abzählbar viele Turing-(semi-)entscheidbare Teilmengen von  $A^*$ . Die meisten Teilmengen von  $A^*$  – überabzählbar viele – sind also nicht entscheidbar und nicht aufzählbar.

**Lemma 3.2.3**  $\{c \in A^* \mid c = \ulcorner \mathcal{C} \urcorner \text{ für eine Turing-Maschine } \mathcal{C}\}$  ist Turing-entscheidbar.

<sup>47</sup>Man darf *abzählbar* nicht mit *aufzählbar* verwechseln! Jede aufzählbare Menge ist abzählbar, aber nicht umgekehrt.

<sup>48</sup>Das Zeichen  $\wedge$  steht für die *Konkatenation*, d. h. für das Hintereinandersetzen von Wörtern oder Symbolen zu einem längeren Wort.

BEWEIS: Man muss durch eine Turing-Maschine prüfen, ob sich  $c$  in der richtigen Weise zusammensetzt, ob für jede Kombination aus vorkommender Nummer eines Zustands und Symbol aus  $A$  ein Eintrag der Übergangsfunktion vorhanden ist, und ob die Übergangsfunktion für den Endzustand so definiert ist, dass die Maschine darin hält. Das ist intuitiv alles prüfbar, also gibt es (mit Church'scher These) auch eine Turing-Maschine dafür.  $\square$

Turing-Semi-Entscheidbarkeit einer Teilmenge  $X \subseteq A^*$  ist eine besondere Art, eine Menge  $X$  zu beschreiben. Da auch jede andere Art von Beschreibung durch einen Text in einem endlichen Alphabet (Buchstaben, Zahlen, mathematische Symbole) erfolgt, gibt es auch nur abzählbar viele beschreibbare Teilmengen von  $A^*$  (für jede mögliche Präzisierung von „beschreibbar“). Das nächste Ziel ist zu zeigen, dass diese drei Eigenschaften – allgemeine Beschreibbarkeit, Turing-Semi-Entscheidbarkeit und Turing-Entscheidbarkeit – nicht übereinstimmen. Dafür braucht man zunächst eine Verallgemeinerung von Definition 3.1.1:

**Definition 3.2.4** Eine Funktion  $f : A^* \times A^* \dashrightarrow A^*$  ist Turing-berechenbar, wenn es eine Turing-Maschine gibt, die bei Eingabe von  $w_1 \hat{\ } \emptyset \hat{\ } w_2$  nach endlich vielen Schritten mit Ausgabe  $f(w_1, w_2)$  im Endzustand stoppt, falls  $f$  in  $(w_1, w_2)$  definiert ist, und andernfalls nicht stoppt.

Man kann sich überlegen, dass es eine Turing-Maschine gibt, die eine Bijektion  $A^* \rightarrow A^* \times A^*$  berechnet. Insofern kann man auch die Turing-Berechenbarkeit einer Funktion  $A^* \times A^* \dashrightarrow A^*$  auf die Turing-Berechenbarkeit einer Funktion  $A^* \dashrightarrow A^*$  zurückführen.

**Lemma 3.2.5** Wenn  $f : A^* \times A^* \dashrightarrow A^*$  eine Turing-berechenbare Funktion ist, dann ist für jedes  $v \in A^*$  auch die Funktion  $f_v : A^* \dashrightarrow A^*$ ,  $u_v(w) = u(v, w)$  Turing-berechenbar.

BEWEIS: Man konstruiert eine Turing-Maschine, die zunächst  $v = (a_{i_1}, \dots, a_{i_k})$  vor ihre Eingabe  $w$  schreibt und dann die Turing-Maschine  $\mathcal{T}_f$  simuliert, die  $f$  berechnet.

Dazu benötigt man  $k + 1$  neue Zustände  $z'_0, \dots, z'_k$ : Die Maschine läuft vom Startzustand aus nach links und geht in Zustand  $z'_k$ . Für  $j > 0$  schreibt sie im Zustand  $z'_j$  das Symbol  $a_{i_j}$  und geht weiter nach links. Im Zustand  $z'_0$  bleibt sie an ihrer Position und es schließt sich das Programm der Turing-Maschine  $\mathcal{T}_f$  an, wobei  $z'_0$  deren Startzustand ersetzt.  $\square$

**Satz 3.2.6 (a)** Es gibt keine universelle totale Turing-berechenbare Funktion [universal ...], d. h. keine Turing-berechenbare Funktion  $u : A^* \times A^* \rightarrow A^*$ , so dass  $\{u_v : A^* \rightarrow A^* \mid v \in A^*\}$  gerade die Menge aller Turing-berechenbaren Funktionen  $A^* \rightarrow A^*$  ist.

**(b)** Es gibt eine universelle partielle Turing-berechenbare Funktion, d. h. eine Turing-berechenbare Funktion  $u' : A^* \times A^* \dashrightarrow A^*$ , so dass  $\{u'_v : A^* \dashrightarrow A^* \mid v \in A^*\}$  gerade die Menge aller Turing-berechenbaren partiellen Funktionen  $A^* \dashrightarrow A^*$  ist.

BEWEIS: (a) Angenommen es gibt solch ein  $u$ , dann ist auch die Diagonalfunktion  $\delta : A^* \rightarrow A^*$ ,  $w \mapsto u(w, w)$  Turing-berechenbar (man verdoppelt zunächst die Eingabe  $w$  zu  $w \hat{\ } \emptyset \hat{\ } w$  und simuliert dann die Turing-Maschine, die  $u$  berechnet). Für ein beliebig gewähltes Symbol  $a \in A$  ist dann sicher auch  $\delta_a : A^* \rightarrow A^*$ ,  $w \mapsto u(w, w) \hat{\ } a$  Turing-berechenbar, da man die Turing-Maschine nur so abändern muss, dass am Ende das Symbol  $a$  angefügt wird, bevor sie in den Endzustand geht. Nach Annahme gibt es nun ein  $w_0 \in A^*$  mit  $\delta_a = u_{w_0}$ . Dann ist aber  $\delta_a(w_0) = u(w_0, w_0) \hat{\ } a \neq u(w_0, w_0) = u_{w_0}(w_0)$ : Widerspruch

(b) Beweisskizze: Man konstruiert zunächst eine universelle Turing-Maschine  $\mathcal{U}$  (die der besseren Verständlichkeit halber hier zunächst mit dem Alphabet  $A_T$  von Seite 95 arbeitet):



$\mathcal{U}$  prüft zunächst, ob  $c$  von der Form  $\ulcorner \mathcal{C} \urcorner$  ist. Falls ja, simuliert  $\mathcal{U}$  den Lauf von  $\mathcal{C}$  mit Eingabe  $w$  (indem  $\mathcal{U}$  stets in  $c$  „nachschauf“, wie sie sich verhalten soll), und gibt ggf. deren Ausgabe aus. In allen anderen Fällen läuft  $\mathcal{U}$  unendlich lange weiter.

Wenn  $\mathcal{U}$  hält, hat man nach Konstruktion also eine Ausgabe von der Form  $w' \in A^*$ . Somit berechnet  $\mathcal{U}$  eine partielle Funktion  $u'' : A_T^* \times A^* \dashrightarrow A^*$  und  $u''$  ist nach Konstruktion auch universell für partielle Turing-berechenbare Funktionen  $A^* \dashrightarrow A^*$ , d. h. jede partielle Turing-berechenbare Funktionen  $A^* \dashrightarrow A^*$  ist von der Form  $u''_c$  für geeignetes  $c \in A_T^*$ . Kodiert man Turing-Maschinen in  $A^*$  statt in  $A_T^*$ , erhält man die gesuchte Funktion!  $\square$

Der Beweis von Teil (a) ist eine Kopie des Cantor'schen Diagonalarguments. Für partielle Funktionen funktioniert dies nicht, da  $u(w_0, w_0)$  unbestimmt sein kann (und es dann auch sein muss).

**Satz 3.2.7 (Halteproblem [halting problem], Turing 1936)**

$$X := \{(c, w) \in A^* \times A^* \mid c = \ulcorner \mathcal{C} \urcorner \text{ und die Turing-Maschine } \mathcal{C} \text{ stoppt bei Eingabe } w\}$$

ist Turing-semi-entscheidbar, aber nicht Turing-entscheidbar.

BEWEIS: Sei ohne Einschränkung  $1 \in A$ . Man ändert die Turing-Maschine, die die universelle partielle Funktion aus dem vorhergehenden Satz berechnet, so ab, dass sie  $1$  ausgibt, wenn die Maschine hält. Diese abgeänderte Turing-Maschine berechnet dann genau die „partielle charakteristische Funktion“  $\overline{\chi_X}$ . Also ist  $X$  semi-Entscheidbar.

Angenommen  $X$  ist Turing-entscheidbar. Dann konstruiert man folgendermaßen eine universelle Turing-berechenbare Funktion: Bei Eingabe  $(c, w)$ , prüft man zunächst, ob  $c = \ulcorner \mathcal{C} \urcorner$ . Falls ja, prüft man durch die angenommene Turing-Maschine, ob  $\mathcal{C}$  mit Eingabe  $w$  stoppt. Falls auch dies gilt, berechnet man deren Ausgabe und hält im Endzustand. In allen andern Fällen gibt man einen beliebigen festen Wert aus und hält dann ebenfalls im Endzustand. In jedem Fall stoppt die Maschine im Endzustand, berechnet also eine totale Funktion  $u$ . Da jede Turing-berechenbare Funktion  $f$  durch eine Turing-Maschine  $\mathcal{C}$  berechnet wird, ist  $f = u_c$  für  $c = \ulcorner \mathcal{C} \urcorner$ . Für alle anderen  $c$  ist  $u_c$  nach Lemma 3.2.5 ebenfalls Turing-berechenbar. Insgesamt ist  $u$  also eine universelle Turing-berechenbare Funktion: Widerspruch!  $\square$

**Folgerung 3.2.8 Die Menge**

$$Y := \{(c, w) \in A^* \times A^* \mid c = \ulcorner \mathcal{C} \urcorner \text{ und die Turing-Maschine } \mathcal{C} \text{ stoppt nicht bei Eingabe } w\}$$

ist beschreibbar, aber nicht Turing-semi-entscheidbar.

BEWEIS: Zunächst ist das Komplement  $X'$  der Menge  $X$  aus dem Halteproblem nicht Turing-semi-entscheidbar, denn andernfalls wäre  $X$  nach Lemma 3.0.2 Turing-entscheidbar, im Widerspruch zum „Halteproblem“. Gleiches gilt für  $Y$  als Schnitt von  $X'$  mit der entscheidbaren Menge  $\{(c, w) \in A^* \times A^* \mid c = \ulcorner \mathcal{C} \urcorner \text{ für eine Turing-Maschine } \mathcal{C}\}$  (Übung!). Andererseits sind  $X'$  und  $Y$  offenbar beschreibbare Mengen (da sie beschrieben sind).  $\square$

Falls  $\mathcal{L}$  eine höchstens abzählbare Sprache ist, kann man die  $\mathcal{L}$ -Formeln als Wörter über einem endlichen Alphabet  $A$  auffassen. Zum Beispiel kann  $A$  alle Junktoren und Quantoren, das Gleichheitszeichen und Klammern enthalten sowie Symbole  $v, R, f$  und die Ziffern  $0, \dots, 9$ . Eine  $\mathcal{L}$ -Formel wird dafür zunächst als Zeichenkette geschrieben; dann werden die Individuenvariablen, Relations- und Funktionszeichen „aufgelöst“, indem man den Index als Ziffernfolge dem

entsprechenden Symbol  $v$ ,  $R$  oder  $f$  nachstellt. Die Individuenvariable  $v_{17}$  wird beispielsweise zur Folge  $v_{17}$ .

Nun kann man sich zum Beispiel nun die Frage stellen, ob die Menge aller  $\mathcal{L}$ -Formeln als Teilmenge von  $A^*$  Turing-entscheidbar ist. Mit der Church'schen These ist dies klar, da es offenbar ein intuitives Entscheidungsverfahren gibt: Man kann eine Zeichenfolge dahingehend analysieren, ob sie für eine  $\mathcal{L}$ -Formel steht oder nicht. Mit etwas Mühe kann man auch eine konkrete Turing-Maschine dafür angeben. Weniger offensichtlich ist:

**Satz 3.2.9** *Für abzählbares  $\mathcal{L}$  ist die Menge aller allgemeingültigen  $\mathcal{L}$ -Formeln semi-entscheidbar.*

BEWEISSKIZZE: Dies folgt aus der Vollständigkeit des Kalküls  $\mathbb{K}$ : Alle Axiome und Regeln von  $\mathbb{K}$  sind algorithmisch überprüf- bzw. anwendbar. Man kann daher systematisch alle  $\mathbb{K}$ -Beweise auflisten und damit auch alle  $\mathcal{L}$ -Formeln, die am Ende eines  $\mathbb{K}$ -Beweises stehen, d.h. alle allgemeingültigen  $\mathcal{L}$ -Formeln.  $\square$

**Satz 3.2.10 (Unentscheidbarkeit der Prädikatenlogik, Gödel 1931)**

*Falls  $\mathcal{L}$  mindestens ein zweistelliges Relationszeichen oder ein einstelliges Funktionszeichen enthält, ist die Menge aller allgemeingültigen  $\mathcal{L}$ -Formeln nicht entscheidbar.*

BEWEIS: Da  $\varphi$  genau dann allgemeingültig ist, wenn  $\neg\varphi$  nicht erfüllbar ist (und der Übergang von  $\varphi$  zu  $\neg\varphi$  klarerweise berechenbar ist), kann man ebensogut zeigen, dass die Menge der erfüllbaren  $\mathcal{L}$ -Aussagen nicht entscheidbar ist.

Der Beweis führt die Frage auf das Halteproblem zurück: Für jede Turing-Maschine  $\mathcal{C}$  und jede Eingabe  $w$  wird auf berechenbare Weise eine  $\mathcal{L}$ -Formel  $\varphi_{\mathcal{C},w}$  konstruiert, die genau dann erfüllbar ist, wenn  $\mathcal{C}$  bei Eingabe  $w$  nicht stoppt. Wenn die Menge der allgemeingültigen  $\mathcal{L}$ -Aussagen entscheidbar wäre, könnte man also entscheiden, ob  $\varphi_{\mathcal{C},w}$  erfüllbar ist, also ob  $\mathcal{C}$  bei Eingabe  $w$  stoppt, im Widerspruch zur Unentscheidbarkeit des Halteproblems!

Die Zellen des Bandes einer Turing-Maschine kann man sich durch  $\mathbb{Z}$  indiziert vorstellen (mit Startposition 0), die Arbeitsschritte oder Zeitpunkte sind durch  $\mathbb{N}$  indiziert (ebenfalls mit Startwert 0). Die Individuenvariablen werden sowohl für die Zellen als auch für die Zeitpunkte stehen, abhängig davon, an welcher Stelle sie in welchem Relationszeichen vorkommen. Der besseren Verständlichkeit halber schreibe ich  $z$  für die intendierte Bedeutung als *Zellen* und  $t$  für die intendierte Bedeutung als *Zeitpunkte*.

Sei  $A = \{a_1, \dots, a_m\}$  das  $m$ -elementige Alphabet, mit dem die Turing-Maschine arbeitet. Der Einfachheit halber zeige ich den Satz nur für eine von  $A$  und der Maschine  $\mathcal{C}$  abhängigen Sprache  $\mathcal{L} := \mathcal{L}_{\mathcal{C}}$ , die die folgenden Zeichen enthält:

- Ein zweistelliges Relationszeichen  $<$  für die Anordnung der Zellen und der Zeitpunkte.
- Zwei einstellige Funktionszeichen  $N$  und  $V$  für die nachfolgende Zelle/den nachfolgenden Zeitpunkt bzw. die vorangehende Zelle/den vorangehenden Zeitpunkt.
- Eine Konstante  $0$  für die Startposition und den Startzeitpunkt.
- Ein zweistelliges Relationszeichen  $K$ , wobei  $Ktz$  zum Zeitpunkt  $t$  die Position  $z$  des Lese-Schreib-Kopfes angibt.
- Zweistellige Relationszeichen  $A_i$  für  $i = 1, \dots, m$ , wobei  $A_itz$  angibt, dass zum Zeitpunkt  $t$  das Zeichen  $a_i$  in Zelle  $z$  steht.

- Für jeden der Zustände  $z_0, \dots, z_k$  der Turing-Maschine ein einstelliges Relationszeichen  $Z_j$ , wobei  $Z_j t$  angibt, dass sich die Maschine zum Zeitpunkt  $t$  im Zustand  $z_j$  befindet.<sup>49</sup>

Nun drückt man in einer  $\mathcal{L}$ -Aussage  $\tau$  die Grundgegebenheiten einer Turing-Maschine aus. Dabei müsste man eigentlich nur über positive Zeitpunkte  $t$  reden. Um sich im späteren Verlauf des Beweises keine Gedanken über negative Zeitpunkte machen zu müssen, „friert“ man die Maschine am Besten bis zum Zeitpunkt 0 in Startposition ein und lässt erst dann das Programm laufen. Das sieht dann folgendermaßen aus:

- $<$  beschreibt eine lineare Ordnung, die *diskret* ist und keine Endpunkte hat, d. h. jedes Element  $z$  hat einen unmittelbaren Nachfolger  $Nz$  und einen unmittelbaren Vorgänger  $Vz$ . Zum Beispiel wird die Eigenschaft, dass  $Nz$  unmittelbarer Nachfolger von  $z$  ist, beschrieben durch

$$(\forall z z < Nz \wedge \neg \exists y (z < y \wedge y < Nz))$$

Diese Eigenschaften implizieren dann auch  $\forall z NVz \doteq z$  und  $\forall z VNz \doteq z$ .

- In jeder Zelle steht immer maximal ein Symbol:

$$\forall z \forall t \bigwedge_{0 \leq i < j \leq m} \neg (A_i t z \wedge A_j t z)$$

- Die Maschine ist bis zum Startzeitpunkt im Startzustand und zu jedem Zeitpunkt  $t$  in genau einem Zustand:

$$(\forall t (t < N0 \rightarrow Z_0 t) \wedge \forall t ((Z_0 t \vee \dots \vee Z_k t) \wedge \bigwedge_{0 \leq i < j \leq k} \neg (Z_i t \wedge Z_j t)))$$

- Der Lese-Schreib-Kopf steht bis zum Startzeitpunkt in Startposition und zu jedem Zeitpunkt in genau einer Zelle:

$$(\forall t (t < N0 \rightarrow Kt0) \wedge \forall t \exists z (Kt z \wedge \forall z' (Kt z' \rightarrow z \doteq z')))$$

In einer weiteren  $\mathcal{L}$ -Aussage  $\pi_C$  wird die Übergangsfunktion von  $C$  beschrieben. Sie besteht aus einer Konjunktion von Aussagen der Form

$$\forall z \forall t ((V0 < t \wedge Z_j t \wedge A_i t z \wedge Kt z) \rightarrow (Z_{j'} Nt \wedge A_{i'} Nt z \wedge KNt z'))$$

Dabei sind  $z'$ ,  $j'$  und  $i'$  durch die Übergangsfunktion festgelegt und  $z'$  ist entweder  $z$  oder  $Nz$  oder  $Vz$ , je nachdem, ob der Lese-Schreib-Kopf in derselben Zelle bleibt oder sich nach rechts oder links bewegt

Schließlich drückt man in einer  $\mathcal{L}$ -Aussage  $\varepsilon_w$  aus, dass bis zum Startzeitpunkt die Eingabe  $w = (a_{i_1}, \dots, a_{i_l})$  auf dem Band steht:

$$\begin{aligned} \forall t \left( t < V0 \rightarrow (A_{i_1} t N0 \wedge A_{i_2} t NN0 \wedge \dots \wedge A_{i_l} t \overbrace{N \dots N}^{l \text{ mal}} 0 \wedge \right. \\ \left. \wedge \forall z ((z < 0 \vee \underbrace{N \dots N}_{l \text{ mal}} 0 < z) \rightarrow \bigwedge_{i=1}^l \neg A_i t z)) \right) \end{aligned}$$

<sup>49</sup>Um eine von  $C$  unabhängige Sprache zu erhalten, kann man ein zweistelliges Relationszeichen  $Z$  nehmen und durch  $Ztj$  ausdrücken, dass sich die Maschine zum Zeitpunkt  $t$  im Zustand  $Z_j$  befindet, wobei  $j$  das  $j$ -te, in der Anordnung  $<$  auf 0 folgende Element ist. Analog ein dreistelliges Relationszeichen  $A$  anstelle der  $A_i$ .

Angenommen die Turing-Maschine  $\mathcal{C}$  stoppt bei Eingabe  $w$  nicht. Dann ergibt sich aus dem Lauf der Maschine ein Modell  $\mathcal{Z}$  der  $\mathcal{L}$ -Aussage

$$\varphi_{\mathcal{C},w} := \left( \tau \wedge \pi_{\mathcal{C}} \wedge \varepsilon_w \wedge \neg \exists t \left( 0 < t \wedge \bigwedge_{j=0}^k (Z_j t \leftrightarrow Z_j N t) \wedge \forall z (K t z \leftrightarrow K N t z) \wedge \forall z \bigwedge_{i=1}^m (A_i t z \leftrightarrow A_i N t z) \right) \right) :$$

Das Universum des Modells  $\mathcal{Z}$  ist  $\mathbb{Z}$ , und man interpretiert die Zeichen aus  $\mathcal{L}$  so, wie intendiert, d. h. es ist zum Beispiel genau dann  $(n_1, n_2) \in A_i^{\mathcal{Z}}$ , wenn das Symbol  $a_i$  zum Zeitpunkt  $n_1$  in Zelle  $n_2$  steht. (Achtung: Die Elemente des Modells sind sowohl Zellen als auch Zeitpunkte!)

In die andere Richtung: Angenommen die  $\mathcal{L}$ -Aussage  $\varphi_{\mathcal{C},w}$  ist erfüllbar und hat ein Modell  $\mathcal{M}$ . Dann betrachtet man die Menge  $M_0$ , die aus dem Element  $0^{\mathcal{M}}$  und dessen unmittelbaren Nachfolgern  $N \dots N 0^{\mathcal{M}}$  und Vorgängern  $V \dots V 0^{\mathcal{M}}$  besteht. Diese Menge ist dann Universum einer Unterstruktur  $\mathcal{M}_0$  von  $\mathcal{M}$  und als Ordnung bzgl  $<^{\mathcal{M}_0}$  zu  $(\mathbb{Z}, <)$  isomorph.

Auf  $M_0$  beschreibt  $\varphi_{\mathcal{C},w}$  nun den Lauf der Turing-Maschine  $\mathcal{C}$  bei Eingabe  $w$ : Zum Zeitpunkt  $t$  steht in Zelle  $z$  genau dann das Symbol  $a_i$ , wenn  $\varphi_{\mathcal{C},w} \vdash A_i t z$ , etc. In diesem Lauf wird kein Haltezustand erreicht, also stoppt  $\mathcal{C}$  bei Eingabe  $w$  nicht.  $\square$

Man kann sich anschließend noch überlegen, dass man die im Beweis benutzte Sprache  $\mathcal{L}$  durch eine Sprache kodieren kann, die nur ein einziges zweistelliges Relationszeichen oder nur ein einziges einstelliges Funktionszeichen benutzt.

Man könnte analog zur Formel  $\varphi_{\mathcal{C},w}$  versuchen eine  $\mathcal{L}$ -Formel  $\sigma_{\mathcal{C},w}$  aufzustellen, die ausdrückt, dass die Turing-Maschine  $\mathcal{C}$  bei Eingabe  $w$  stoppt, indem man die Negation vor dem Quantor  $\exists t$  weglässt. Tatsächlich hat diese  $\mathcal{L}$ -Formel ein Modell, wenn die Turing-Maschine stoppt. Umgekehrt aber kann man aus der Erfüllbarkeit der Formel nicht schließen, dass die Turing-Maschine stoppt, weil ein Modell von  $\sigma_{\mathcal{C},w}$  nicht wie  $\mathbb{Z}$  aussehen muss und der Stoppzeitpunkt  $t$  „im Unendlichen“ liegen könnte, also nicht von der Form  $N \dots N 0$  sein muss. Im Gegenteil: Wäre es so, dass das Halten einer Turing-Maschine bei fester Eingabe äquivalent zur Erfüllbarkeit einer  $\mathcal{L}$ -Formel wäre, könnte man umgekehrt damit wiederum das Halteproblem entscheiden.

**Satz 3.2.11** *Wenn  $\mathcal{L}$  nur aus Konstanten und null- und einstelligen Relationszeichen besteht, dann ist es entscheidbar, ob eine  $\mathcal{L}$ -Aussage allgemeingültig ist oder nicht.*

BEWEISSKIZZE: Einerseits ist die Menge der allgemeingültigen  $\mathcal{L}$ -Formeln semi-entscheidbar, und damit auch die Menge der nicht erfüllbaren  $\mathcal{L}$ -Formeln.

Andererseits kann man zeigen, dass aufgrund der Beschränkungen an  $\mathcal{L}$  jede erfüllbare  $\mathcal{L}$ -Formel schon ein endliches Modell hat. Für jedes der abzählbar vielen Kombinationen aus  $\mathcal{L}$ -Formel  $\varphi$  und endlicher  $\mathcal{L}$ -Struktur  $\mathcal{M}$  kann man in endlich vielen Schritten herausfinden, ob  $\mathcal{M}$  Modell von  $\varphi$  ist oder nicht. Damit kann man durch systematisches Ausprobieren auch die erfüllbaren  $\mathcal{L}$ -Formeln aufzählen.  $\square$

### 3.3 NP-Vollständigkeit und der Satz von Cook

Für die folgende Betrachtung braucht man eine Variante der Turing-Maschinen, nämlich die *nicht-deterministischen Turing-Maschinen* [nondeterministic Turing machines]. Sie unterschei-

den sich von den gewöhnlichen, *deterministischen Turing-Maschinen* darin, dass die Übergangsfunktion  $Z \times A^+ \rightarrow Z \times A^+ \times \{-1, 0, 1\}$  ersetzt wird durch eine *Übergangsrelation*

$$U \subseteq (Z \times A^+) \times (Z \times A^+ \times \{-1, 0, 1\}).$$

Die Übergangsrelation muss dabei die folgenden beiden Eigenschaften haben:

- Für alle  $(z, a) \in Z \times A^+$  gibt es mindestens ein Tripel  $(z', a', p')$  mit  $(z, a, z', a', p') \in U$ .
- Falls  $(z_{\text{end}}, a, z', a', p') \in U$ , so ist  $z' = z_{\text{end}}$ ,  $a' = a$  und  $p' = 0$ .

Nicht-deterministische Turing-Maschinen werden hier nur für Entscheidungsfragen benutzt, also nicht für die Berechnung von Funktionen. Dies funktioniert folgendermaßen: Eine nicht-deterministische Turing-Maschine *akzeptiert* ein Wort  $w \in A^*$ , wenn es einen möglichen Programmlauf gibt, der den Endzustand erreicht, d. h. wenn es in jedem Programmschritt im Zustand  $z$  und bei aktuellem Bandsymbol  $a$  ein Tripel  $(z', a', p')$  mit  $(z, a, z', a', p') \in U$  gibt, so dass, wenn die Maschine dann in den Zustand  $z'$  geht, das Symbol  $a'$  schreibt und sich gemäß  $p'$  bewegt, irgendwann der Endzustand erreicht wird.

Sei  $A \neq \emptyset$  ein endliches Alphabet. Jede Teilmenge  $M \subseteq A^*$  definiert ein *Entscheidungsproblem*, nämlich die Frage:

*Ist ein gegebenes  $w \in A^*$  in  $M$  oder nicht?*

Verkürzt wird daher die Teilmenge  $M$  ein **Problem** [problem] genannt.

### Definition 3.3.1

(a) Ein Problem ist (nicht-deterministisch) **polynomiell** [(nondeterministic) polynomial], falls es ein Polynom  $T \in \mathbb{R}[X]$  und eine (nicht-deterministische) Turing-Maschine gibt, die für alle  $w \in A^*$  in höchstens  $T(\lg(w))$  Berechnungsschritten entscheidet, ob  $w \in M$ .

Die Menge der polynomiellen Probleme wird mit **P** (oder **PTIME**) bezeichnet, die Menge der nicht-deterministisch polynomiellen Probleme mit **NP**.

(b) Ein Problem  $M$  ist **NP-schwer** [NP-hard], falls es für jedes NP-Problem  $N \subseteq B^*$  eine polynomielle Reduktion auf  $M$  gibt, d. h. eine berechenbare Funktion  $r : B^* \rightarrow A^*$  mit  $w \in N \iff r(w) \in M$  für alle  $w \in B^*$ , so dass  $\lg(r(w)) \leq R(\lg(w))$  für ein Polynom  $R \in \mathbb{R}[X]$ .

(c) Ein Problem  $M$  ist **NP-vollständig** [NP-complete], falls es in **NP** liegt und NP-schwer ist.

**Beispiel:** Sei  $A$  ein endliches Alphabet, in dem aussagenlogische Formeln beschrieben werden können – z. B.  $A = \{\top, \perp, \dots, \leftrightarrow, A, 0, 1, \dots, 9\}$ , wenn aussagenlogische Formeln in Polnischer Notation geschrieben werden und Aussagenvariablen wie  $A_{652}$  ersetzt werden durch das Symbol  $A$  gefolgt von dem Index, hier also die Symbolfolge  $A_{652}$ . Dann ist die Menge der aussagenlogischen Formeln als Teilmenge von  $A^*$  ein polynomielles Problem:

Eine Turing-Maschine kann sukzessive die Formel z. B. von links nach rechts analysieren und abbauen: Sie identifiziert den am weitestens links stehenden Junktor, löscht ihn, findet bei einem zweistelligen Junktor die Trennstelle der beiden Teilformeln und schreibt eine Leerzeichen dazwischen. Dann macht sie mit diesen Teilformeln weiter, bis nur noch Aussagenvariablen übrig bleiben. Sobald andere Zeichen übrig bleiben oder ein Schritt nicht klappt, weil die Syntax nicht stimmt, gibt die Maschine eine negative Antwort. In jedem Abbauschritt muss die Formel im Wesentlichen einmal durchlaufen werden, man hat also einen quadratischen Algorithmus (polynomiell vom Grad 2) in der Länge des Wortes, das die Formel beschreibt.

Die Einordnung von Entscheidungsproblemen in die Klassen P und NP geschieht also durch Einschränkungen an die Berechnungszeit. Die Größe des benötigten Speicherplatzes spielt dabei keine Rolle; dafür gibt es aber die analog definierten Problemklassen PSPACE und NPSPACE. Die *Komplexitätstheorie* untersucht diese und viele andere Problemklassen. Dabei gelten die polynomiellen Probleme als eine Annäherung an die „praktisch berechenbaren“ Probleme, da die Berechnungszeit in einem vernünftigen Verhältnis zur Eingabegröße steht, wobei es in der Praxis natürlich auf den Grad und die Größe der Koeffizienten des Polynoms  $T$  ankommt. Für manche Probleme in NP kennt man dagegen nur Algorithmen mit exponentieller Berechnungszeit in Abhängigkeit von der Eingabegröße, so dass sich schon bei recht kleinen Eingaben eine exorbitante Berechnungszeit ergibt.

Man sieht zunächst, dass  $P \subseteq NP$ , da man jede deterministische Turing-Maschine auch als nicht-deterministische Turing-Maschine auffassen kann, indem man die Übergangsfunktion durch ihren Graphen als Übergangsrelation ersetzt.<sup>50</sup>

Ein großes offenes Problem der Logik und theoretischen Informatik ist die Frage, ob  $P = NP$ , also ob nicht jedes nicht-deterministische polynomielle Problem schon polynomiell ist. Es ist eines der sieben *Millennium Problems* des Clay Instituts<sup>51</sup>, für deren Lösung es jeweils eine Million Dollar gibt. Die meisten Fachleute gehen davon aus, dass  $P \neq NP$ ; andererseits wurde für viele überraschend 2002 ein polynomieller Algorithmus für das Primzahlproblem entdeckt (also für die Frage, ob eine gegebene natürliche Zahl eine Primzahl ist). Das elementare Verfahren, alle kleineren Zahlen als Teiler zu testen, ist dagegen exponentiell in der Eingabegröße.

Könnte man von *einem* NP-schweren Problem  $M$  zeigen, dass es in P liegt, dann wäre zum einen dieses Problem automatisch NP-vollständig (weil  $P \subseteq NP$ ) und zum andern hätte man  $P = NP$  gezeigt, da man dann alle NP-Probleme über die polynomielle Reduktion auf  $M$  ebenfalls in polynomieller Zeit lösen könnte. Daher gelten NP-schwere bzw. NP-vollständige Probleme als praktisch nicht berechenbar. Allerdings gibt es für manche NP-vollständigen Fragestellungen sehr gute und praktisch nutzbare Näherungsalgorithmen.

**Lemma 3.3.2** *Ein Problem  $M \subseteq A^*$  liegt genau dann in NP, wenn es ein Alphabet  $B$ , ein Polynom  $Q \in \mathbb{R}[X]$  und ein polynomielles Problem  $M^+ \subseteq A^* \times B^* \subseteq (A \cup B)^*$  gibt, so dass*

$$w \in M \iff \text{es gibt ein } z \in B^* \text{ mit } w \hat{=} z \in M^+ \text{ und } \lg(z) \leq Q(\lg(w))$$

Die Elemente  $z$  heißen dann *Zertifikate* für  $w$ .

BEWEISSKIZZE: Wenn es polynomielle Zertifikate gibt, konstruiert man eine nicht-deterministische Turing-Maschine, die in ihren möglichen Programmläufen alle  $z \in B^*$  der Länge höchstens  $Q(\lg(w))$  daraufhin testet, ob sie ein Zertifikat für  $w$  sind, was für jedes einzelne  $z$  in Polynomialzeit geht.

Hat man umgekehrt eine nicht-deterministische Turing-Maschine, die in höchstens  $T(\lg(w))$  vielen Schritten entscheidet, ob  $w \in M$ , dann kann nach jedem Berechnungsschritt auch nur ein Wort der Länge höchstens  $\lg(w) + T(\lg(w))$  auf dem Band stehen (da jeder Schritt höchstens ein Symbol hinzufügt). Alle diese Bandzustände hintereinander und zum Beispiel jeweils durch ein neues Symbol getrennt und mit zusätzlichen Symbolen für den jeweils aktuellen Zustand

<sup>50</sup>Die Terminologie ist also nicht besonders exakt; anstelle von „nicht-deterministisch“ wäre *nicht notwendigerweise deterministische Turing-Maschine* präziser.

<sup>51</sup><http://www.claymath.org/millennium-problems>

und die jeweils aktuelle Bandposition versehen, beschreiben also den gesamten Programmablauf in einem Wort der Länge höchstens  $T(\lg(w)) \cdot (\lg(w) + T(\lg(w)) + 3)$ . Dieses Wort dient als Zertifikat, da man sicher in polynomieller Zeit überprüfen kann, ob ein gegebenes Wort auf diese Weise einem Programmablauf der Turing-Maschine entspricht. Man findet also Zertifikate mit einer Länge, die durch das Polynom  $Q(X) = T(X) \cdot (T(X) + X + 3)$  beschränkt ist.  $\square$

Ich werde in der Folge einfach von „Formeln“ sprechen, wenn ich eigentlich Darstellungen von Formeln über einem endlichen Alphabet meine. Das Problem hierbei ist, dass die Länge des darstellenden Wortes beliebig viel größer als die Länge der Formel werden kann, wenn Aussagenvariablen mit großen Indizes vorkommen. Für das prinzipielle Verständnis des Satzes von Cook kann man aber die Länge der Formel als Eingabegröße nehmen, indem man entweder annimmt, dass man nur endlich viele Aussagenvariablen hat, oder so tut, als würden die Komplexitätsbetrachtungen auch für ein unendliches Alphabet funktionieren.

**Beispiel:** Die Menge SAT der erfüllbaren aussagenlogischen Formeln ist ein typisches NP-Problem; die erfüllenden Belegungen dienen hier als Zertifikate. Denn mit  $B = \{0, 1\}$  erhält man ein polynomielles Problem  $\text{SAT}^+ = \{F \wedge \beta \mid \beta(F) = 1\}$ , da es – ebenfalls in quadratischer Zeit – möglich ist, den Wahrheitswert einer Formel für eine gegebene Belegung auszurechnen.

SAT

Alternativ kann man zum Beispiel die Methode von Quine nutzen, um direkt eine nicht-deterministische Turing-Maschine für SAT zu konstruieren, indem die Übergangsrelation sukzessive für jede Aussagenvariable die Möglichkeit bietet, sie einerseits durch  $\top$  und andererseits durch  $\perp$  zu ersetzen.

3-SAT ist das Problem der erfüllbaren aussagenlogischen Formeln in KNF, in denen maximal drei Literale pro Klausel vorkommen.

3-SAT

### Satz 3.3.3 (Satz von Cook und Levin, 1971)

Das Erfüllbarkeitsproblem SAT ist NP-vollständig. Sogar: 3-SAT ist NP-vollständig.

Wenn man  $\text{SAT} \in \text{P}$  bzw.  $\text{SAT} \notin \text{P}$  zeigen könnte, hätte man also die  $\text{P}=\text{NP}$ -Frage beantwortet. Bei der elementaren Methode, Erfüllbarkeit über Wahrheitstabellen zu testen, muss man im Allgemeinen exponentiell viele Belegungen in der Anzahl der vorkommenden Aussagenvariablen betrachten. Andere Verfahren wie z. B. die Resolutionsmethode oder Baumkalküle liefern zwar häufig schnellere Ergebnisse, sind aber auch nicht polynomiell.

BEWEIS: Dass SAT in NP liegt, wurde im Beispiel oben erläutert. Sei  $N \subseteq B^*$  ein NP-Problem, das durch eine nicht-deterministische Turing-Maschine  $C_N$  entschieden wird, wobei die Anzahl der Berechnungsschritten im Sinne der Definition durch das Polynom  $T \in \mathbb{R}[X]$  beschränkt sein soll.  $N$  soll nun auf SAT reduziert werden, indem das Verhalten von  $C_N$  für eine feste Eingabe  $w$  durch eine aussagenlogische Formel  $C_{N,w}$  beschrieben wird.

Dazu benutzt man für  $t \in \mathbb{N}$ ,  $n \in \mathbb{Z}$ ,  $a \in A^+$  und  $z \in Z$  die folgenden Aussagenvariablen:

$S_{t,a,n}$  für: Zum Zeitpunkt  $t$  steht das Symbol  $a$  in der  $n$ -ten Speicherzelle.

$Z_{t,z}$  für: Zum Zeitpunkt  $t$  befindet sich die Maschine im Zustand  $z$ .

$K_{t,n}$  für: Zum Zeitpunkt  $t$  befindet sich der Lese-Schreib-Kopf in der  $n$ -ten Speicherzelle.

Dies sind insgesamt abzählbar viele Aussagenvariablen, die man in geeigneter Weise mit den  $(A_i)_{i \in \mathbb{N}}$  identifiziert. Für jedes einzelne  $w \in A^*$  kann in den höchstens  $T_w := \lceil T(\lg(w)) \rceil$  vielen Berechnungsschritten allerdings höchstens eine Speicherzelle im Abstand  $T_w$  von der Startzelle

0 erreicht werden. Man braucht also für jedes  $w$  nur die endlich vielen Aussagenvariablen mit  $t \leq T_w$  und  $|n| \leq T_w$ .

Für fest gewähltes  $w$  und damit auch festes  $T_w$  ist die aussagenlogische Formel  $C_{N,w}$  nun die Konjunktion folgender Formeln:

- Zu jedem Zeitpunkt  $t \leq T_w$  befindet sich die Maschine in genau einem Zustand:

$$\bigwedge_{t \leq T_w} \left( \bigvee_{z \in Z} Z_{t,z} \wedge \bigwedge_{z \neq z'} \neg(Z_{t,z} \wedge Z_{t,z'}) \right)$$

- Zu jedem Zeitpunkt  $t \leq T_w$  befindet sich der Kopf in genau einer der betrachteten Zellen:

$$\bigwedge_{t \leq T_w} \left( \bigvee_{|n| \leq T_w} K_{t,n} \wedge \bigwedge_{n \neq n'} \neg(K_{t,n} \wedge K_{t,n'}) \right)$$

- Zu jedem Zeitpunkt  $t \leq T_w$  steht in jeder Speicherzelle  $n$  mit  $|n| \leq T_w$  genau ein Symbol:

$$\bigwedge_{t \leq T_w} \bigwedge_{|n| \leq T_w} \left( \bigvee_{a \in A^+} S_{t,a,n} \wedge \bigwedge_{a \neq a'} \neg(S_{t,a,n} \wedge S_{t,a',n}) \right)$$

- Zum Zeitpunkt  $t = 0$  befindet sich die Maschine im Startzustand und der Kopf in der 0-ten Speicherzelle:

$$(Z_{0,z_0} \wedge K_{0,0})$$

- Die Eingabe ist  $w = (a_1, \dots, a_l)$  mit  $a_i \in A$ :

$$(S_{0,a_1,1} \wedge S_{0,a_2,2} \wedge \dots \wedge S_{0,a_l,l} \wedge \bigwedge_{|n| \leq T_w, n \notin \{1, \dots, l\}} S_{0,\emptyset,n})$$

- Die Programmschritte laufen in Übereinstimmung mit der Übergangsrelation:

$$\bigwedge_{t \leq T_w - 1} \left( (S_{t,a,n} \wedge Z_{t,z} \wedge K_{t,n}) \rightarrow \bigvee_{(z,a,z',a',p') \in U} (S_{t+1,a',n} \wedge Z_{t+1,z'} \wedge K_{t+1,n+p'}) \right)$$

Da  $A$ ,  $Z$  und  $U$  fest sind, sieht man, dass  $C_{N,w}$  eine polynomielle Länge in  $T_w$  hat, also auch eine polynomielle Länge in  $l = \lg(w)$ .

Die Turing-Maschine  $C_N$  akzeptiert nun  $w$  in höchstens  $T_w$  Berechnungsschritten genau dann, wenn die Formel  $(C_{N,w} \wedge Z_{T_w, z_{\text{end}}})$  erfüllbar ist. Mit Lemma 1.7.1 kann diese Formel in eine erfüllbarkeitsäquivalente 3-SAT-Formel überführt werden, wobei sich die Länge nur um eine Konstante ändert, also polynomiell in  $\lg(w)$  bleibt.  $\square$

### 3.4 Rekursive Funktionen

Im nächsten Abschnitt soll nun ein alternatives Konzept für die Berechenbarkeit von Funktionen vorgestellt werden, das mit Funktionen  $\mathbb{N}^k \rightarrow \mathbb{N}$  auf den natürlichen Zahlen arbeitet (die man aber, wie schon erwähnt, in einer intuitiv berechenbaren Weise mit den Wörtern über einem endlichen, nicht-leeren Alphabet identifizieren kann). Zunächst wird das schwächere (aber für einige Beweise dennoch wichtige) Konzept der *primitiv rekursiven* Funktionen eingeführt, welches dann auf das Konzept der  $\mu$ -*rekursiven* oder auch kurz *rekursiven Funktionen* ausgedehnt wird. Alle rekursiven Funktionen sind sowohl in einem intuitiven Sinn berechenbar als auch Turing-berechenbar (was aber in dieser Vorlesung nicht gezeigt wird).



**Totale Funktionen**

**Definition 3.4.1**

(a) Eine Funktion  $f : \mathbb{N}^k \rightarrow \mathbb{N}$  für  $k \in \mathbb{N}$  heißt **primitiv rekursiv** [primitive recursive], wenn sie eine der Grund- oder Ausgangsfunktionenfunktionen ist oder sich daraus durch endlichfache Anwendung der Regeln [Komposition] und [Primitive Rekursion] ergibt.

Grundfunktionen sind dabei:

- die konstante Nullfunktion  $0 : \mathbb{N}^0 \rightarrow \mathbb{N}$
- die Nachfolgerfunktion  $N : \mathbb{N} \rightarrow \mathbb{N}, n \mapsto n + 1$
- die Projektionsfunktionen  $\pi_i^k : \mathbb{N}^k \rightarrow \mathbb{N}, (n_1, \dots, n_k) \mapsto n_i$

$N : \mathbb{N} \rightarrow \mathbb{N}$

$\pi_i^k : \mathbb{N}^k \rightarrow \mathbb{N}$

und die Regeln lauten:

- Aus Funktionen  $f_1, \dots, f_l : \mathbb{N}^k \rightarrow \mathbb{N}$  und  $g : \mathbb{N}^l \rightarrow \mathbb{N}$  entsteht durch [Komposition] [composition] die Funktion  $h : \mathbb{N}^k \rightarrow \mathbb{N}$  mit

$$h(n_1, \dots, n_k) := g(f_1(n_1, \dots, n_k), \dots, f_l(n_1, \dots, n_k))$$

- Aus Funktionen  $f : \mathbb{N}^k \rightarrow \mathbb{N}$  und  $g : \mathbb{N}^{k+2} \rightarrow \mathbb{N}$  entsteht durch [Primitive Rekursion] [primitive recursion] die Funktion  $h : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$  mit

$$h(0, n_1, \dots, n_k) := f(n_1, \dots, n_k)$$

$$h(n + 1, n_1, \dots, n_k) := g(h(n, n_1, \dots, n_k), n, n_1, \dots, n_k)$$

(b) Eine (totale) Funktion  $f : \mathbb{N}^k \rightarrow \mathbb{N}$  heißt  **$\mu$ -rekursiv** oder kurz **rekursiv** [ $(\mu)$ -recursive], wenn sie eine der Grundfunktionen ist oder sich daraus durch endlichfache Anwendung der Regeln [ $\mu$ -Rekursion] im Normalfall, [Komposition] und [Primitive Rekursion] ergibt.

Die zusätzlich Regel lautet:

- Aus einer Funktion  $f : \mathbb{N}^{k+1} \rightarrow \mathbb{N}$  entsteht, sofern der Normalfall vorliegt, dass für alle  $(n_1, \dots, n_k) \in \mathbb{N}$  ein  $m \in \mathbb{N}$  mit  $f(m, n_1, \dots, n_k) = 0$  existiert, durch [ $\mu$ -Rekursion] [ $\mu$ -recursion] die Funktion  $h : \mathbb{N}^k \rightarrow \mathbb{N}$  mit

$$h(n_1, \dots, n_k) := \mu m [f(m, n_1, \dots, n_k) = 0]$$

wobei  $\mu m [E]$  für das kleinste  $m \in \mathbb{N}$  mit der Eigenschaft  $E$  steht.

$\mu m [\dots]$

**Erste Folgerungen und Varianten:**

- Wenn  $f : \mathbb{N}^k \rightarrow \mathbb{N}$  (primitiv) rekursiv ist und  $\sigma : \{1, \dots, k\} \rightarrow \{1, \dots, k\}$ , dann ist auch die Funktion

$$(n_1, \dots, n_k) \mapsto f(n_{\sigma(1)}, \dots, n_{\sigma(k)}) = f(\pi_{\sigma(1)}^n(n_1, \dots, n_k), \dots, \pi_{\sigma(k)}^n(n_1, \dots, n_k))$$

(primitiv) rekursiv. Insbesondere muss bei den Rekursionsschemata [Primitive Rekursion] und [ $\mu$ -Rekursion] die Rekursion nicht notwendigerweise über die erste Variable laufen.

- Oft wird das Schema der Primitiven Rekursion mit einer  $(k + 1)$ -stelligen Funktion  $g'$  durch  $h(n + 1, n_1, \dots, n_k) = g'(h(n, n_1, \dots, n_k), n_1, \dots, n_k)$  angegeben, also ohne das zusätzliche Argument  $n$ . Diese Variante lässt sich leicht aus den angegebenen herleiten, weil mit  $g'$  auch

$$\begin{aligned} g(m, n, n_1, \dots, n_k) &= g'((m, n_1, \dots, n_k) = \\ &= g'(\pi_1^{k+2}(m, n, n_1, \dots, n_k), \pi_3^{k+2}(m, n, n_1, \dots, n_k), \dots, \pi_{k+2}^{k+2}(m, n, n_1, \dots, n_k)) \end{aligned}$$

(primitiv) rekursiv ist. <sup>52</sup>

---

<sup>52</sup>Die Umkehrung, also dass die „ $g'$ -Variante“ ausreicht, wird hier nicht bewiesen.

- Die rekursiven Funktionen kann man auch ohne das Schema der Primitiven Rekursion charakterisieren, wenn man zusätzliche Grundfunktionen zulässt, nämlich Addition, Multiplikation und die charakteristische Funktion der Ordnung  $<$ .

Alle konstanten Funktionen  $c_m^k : \mathbb{N}^k \rightarrow \mathbb{N}, (n_1, \dots, n_k) \mapsto m$  sind primitiv rekursiv, denn  $c_0^0$  ist eine Grundfunktion und die weiteren konstanten Nullfunktionen bekommt man durch iterierte primitive Rekursion:

$$c_0^{k+1}(n_1, \dots, n_k, 0) = c_0^k(n_1, \dots, n_k)$$

$$c_0^{k+1}(n_1, \dots, n_k, n+1) = c_0^{k+1}(n_1, \dots, n_k, n) = \pi_1^{k+2}(c_0^{k+1}(n_1, \dots, n_k, n), n, n_1, \dots, n_k)$$

Für  $m \neq 0$  erhält man schließlich  $c_m^k = N(N(\dots N(c_0^k)\dots))$  durch  $m$ -fach iterierte Komposition der Nachfolgerfunktion  $N$  mit der konstanten Nullfunktion.

Die Addition  $(m, n) \mapsto m + n$  ist primitiv rekursiv, mit primitiver Rekursion über  $m$ :

$$0 + n = n = \pi_1^1(n)$$

$$(m + 1) + n = (m + n) + 1 = N(m + n)$$

Die Multiplikation  $(m, n) \mapsto m \cdot n$  ist primitiv rekursiv, mit primitiver Rekursion über  $m$ :

$$0 \cdot n = 0 = c_0^1(n)$$

$$(m + 1) \cdot n = (m \cdot n) + n = g(m \cdot n, m, n)$$

Dabei ist  $g$  primitiv rekursiv mit der Bemerkung von oben, da die Summe primitiv rekursiv ist.

Die Exponentiation  $(m, n) \mapsto m^n$  ist primitiv rekursiv, mit primitiver Rekursion über  $n$ :

$$m \cdot 0 = 1 = c_1^1(m)$$

$$m^{n+1} = m^n \cdot m = \pi_1^3(m^n, m, n) \cdot \pi_2^3(m^n, m, n)$$

Die Vorgängerfunktion  $V(0) := 0$  und  $V(n + 1) := n$  ist primitiv rekursiv (die Definition ist bereits die primitive Rekursion), und die modifizierte Differenz  $m \dot{-} n := \max\{m - n, 0\}$  ist primitiv rekursiv durch die primitive Rekursion

$$m \dot{-} 0 = m = \pi_1^1(m)$$

$$m \dot{-} (n + 1) = V(m \dot{-} n)$$

### Definition 3.4.2

(a) Eine Teilmenge  $R \subseteq \mathbb{N}^k$  heißt (primitiv) rekursiv, wenn ihre charakteristische Funktion  $\chi_R : \mathbb{N}^k \rightarrow \mathbb{N}$  (primitiv) rekursiv ist.

(b) Eine Teilmenge  $A \subseteq \mathbb{N}^k$  heißt rekursiv aufzählbar [recursively enumerable], wenn es eine rekursive Teilmenge  $R \subseteq \mathbb{N}^{k+1}$  gibt mit

$$A = \pi[R] = \{(n_1, \dots, n_k) \mid \text{es gibt } m \in \mathbb{N} \text{ mit } (n_1, \dots, n_k, m) \in R\}$$

wobei  $\pi = (\pi_1^{k+1}, \dots, \pi_k^{k+1})$  die Projektion auf die ersten  $k$  Koordinaten ist.

**Bemerkung:** Wenn  $R \subseteq \mathbb{N}^{k+1}$  rekursiv ist mit  $\pi[R] = \mathbb{N}^k$ , dann ist auch die Funktion

$$f(n_1, \dots, n_k) := \mu m [(n_1, \dots, n_k, m) \in R] = \mu m [1 \dot{-} \chi_R(n_1, \dots, n_k, m) = 0]$$

$c_n^k$

$V: \mathbb{N} \rightarrow \mathbb{N}$

$\dot{-}$

rekursiv.

Die  $<$ -Relation und die  $\leq$ -Relation sind primitiv rekursiv, denn

$$\chi_{\leq}(x, y) = 1 \dot{-} (x \dot{-} y) = \begin{cases} 1 & x \leq y \\ 0 & x > y \end{cases}$$

$$\chi_{<}(x, y) = 1 \dot{-} \chi_{\leq}(y, x) = \begin{cases} 1 & x < y \\ 0 & x \geq y \end{cases}$$

Die Menge  $M = \{2^n \mid n \in \mathbb{N}\}$  ist rekursiv aufzählbar, denn mit der konstanten Funktion 2 und der Exponentiation ist auch die Funktion  $F(n) = 2^n$  rekursiv, also nach Lemma 3.4.4 auch ihr Graph  $\Gamma_f$  und damit auch der gespiegelte Graph  $\{(x, y) \mid (y, x) \in \Gamma_f\}$ , dessen Projektion im Sinne von Definition 3.4.2 die Menge  $M$  ist.  $M$  ist sogar rekursiv, aber das ist weniger offensichtlich zu sehen.

**Lemma 3.4.3** Die Menge der rekursiven Teilmengen von  $\mathbb{N}$  ist abgeschlossen bezüglich Schnitt, Vereinigung, Differenz und Komplement

BEWEIS:  $\chi_{X \cap Y} = \chi_X \cdot \chi_Y$  und  $\chi_{X \setminus Y} = \chi_X \dot{-} \chi_Y$ . Da die konstante  $k$ -stellige Funktion  $c_1^k$  rekursiv ist, ist  $\mathbb{N}^k$  rekursiv, und damit mit  $X$  auch sein Komplement  $\mathbb{N}^k \setminus X$ . Schließlich ist  $X \cup Y = \mathbb{N}^k \setminus ((\mathbb{N}^k \setminus X) \cap (\mathbb{N}^k \setminus Y))$ .  $\square$

**Lemma 3.4.4**

(a)  $f : \mathbb{N}^k \rightarrow \mathbb{N}$  ist genau dann rekursiv, wenn der Graph  $\Gamma_f \subseteq \mathbb{N}^{k+1}$  rekursiv ist.

(b)  $X \subseteq \mathbb{N}^k$  ist genau dann rekursiv, wenn  $X$  und  $\mathbb{N}^k \setminus X$  rekursiv aufzählbar sind, und genau dann, wenn es ein rekursives  $Y \subseteq \mathbb{N}$  mit  $X = f[Y]$  gibt, wobei  $f : \mathbb{N} \rightarrow \mathbb{N}^k$  eine rekursive Bijektion ist.

(c)  $X \subseteq \mathbb{N}^l$  ist genau dann rekursiv aufzählbar, wenn es ein rekursives  $Y \subseteq \mathbb{N}^k$  mit  $X = f[Y]$  gibt, wobei  $f : \mathbb{N}^k \rightarrow \mathbb{N}^l$  eine rekursive Funktion ist, und genau dann, wenn  $X$  entweder die leere Menge ist oder der Wertebereich einer rekursiven Funktion  $g : \mathbb{N}^k \rightarrow \mathbb{N}^l$ . In beiden Fällen kann  $k = 1$  gewählt werden.

Eine Funktion  $f = (f_1, \dots, f_l) : \mathbb{N}^k \rightarrow \mathbb{N}^l$  soll dabei rekursiv heißen, wenn jede der Komponentenfunktionen  $f_i$  rekursiv ist.

Teil (c) erklärt den Begriff „rekursive Aufzählbarkeit“, da jede rekursiv aufzählbare Teilmenge  $A \neq \emptyset$  Bild einer rekursiven Funktion  $f : \mathbb{N} \rightarrow \mathbb{N}^l \supseteq A$  ist, also  $A = \{f(0), f(1), f(2), \dots\}$ , wobei die Funktionswerte  $f(0), f(1), f(2), \dots$  sukzessive berechnet werden können. Die rekursiv aufzählbaren Mengen sind also die semi-entscheidbaren Mengen im Sinne des Berechenbarkeitsbegriffs der rekursiven Funktionen. Die in Lemma 3.0.2 dargestellten intuitiven Zusammenhänge gelten also auch hier (teils per Definition).

BEWEIS: (a)

$$f(\bar{x}) = \mu y [( \bar{x}, y) \in \Gamma_f] = \mu y [1 \dot{-} \chi_{\Gamma_f}(\bar{x}, y) = 0]$$

$$\chi_{\Gamma_f}(\bar{x}, y) = 1 \dot{-} |f(\bar{x}) - y| = 1 \dot{-} ((f(\bar{x}) \dot{-} y) + (y \dot{-} f(\bar{x})))$$

(b) „ $\Rightarrow$ “ der ersten Äquivalenz:

Da  $\chi_{X \times \mathbb{N}}(\bar{x}, y) = \chi_X(\pi_1^{k+1}(\bar{x}, y), \dots, \pi_k^{k+1}(\bar{x}, y))$ , ist  $X \times \mathbb{N}$  rekursiv und somit  $X = \pi[X \times \mathbb{N}]$  rekursiv aufzählbar, und entsprechend das nach Lemma 3.4.3 rekursive Komplement  $\mathbb{N}^k \setminus X$ .

„ $\Leftarrow$ “: Ist umgekehrt  $X = \pi[Y]$  und  $\mathbb{N}^k \setminus X = \pi[Z]$  für rekursive  $Y, Z \subseteq \mathbb{N}^{k+1}$ , dann ist  $\chi_X(\bar{x}) = \chi_Y(\bar{x}, \mu y[(\bar{x}, y) \in Y \cup Z])$ , rekursiv, weil  $Y \cup Z$  nach Lemma 3.4.3 rekursiv ist.  $\square$

Der restliche Beweis von (b) und der (c) braucht eine Vorbereitung:

**Lemma 3.4.5** *Für alle  $k, l > 0$  gibt es rekursive Bijektionen  $\beta_l^k : \mathbb{N}^k \rightarrow \mathbb{N}^l$  mit  $\beta_l^k = (\beta_k^l)^{-1}$ , und für jede rekursive Bijektion  $\beta : \mathbb{N}^k \rightarrow \mathbb{N}^l$  mit rekursiver Umkehrfunktion ist  $X \subseteq \mathbb{N}^k$  genau dann rekursiv (aufzählbar), wenn  $\beta[X] \subseteq \mathbb{N}^l$  rekursiv (aufzählbar) ist.*

BEWEIS (OHNE NACHRECHNEN): Es reicht aus, den Fall  $\{k, l\} = \{1, 2\}$  zu betrachten, den Rest bekommt man durch iterierte Komposition. Für  $\beta_1^2$  kann man  $(m, n) \mapsto \frac{1}{2}(m+n)(m+n+1) + m$  wählen, was als Komposition rekursiver Funktionen rekursiv ist – die Division durch 2 bekommt man etwa durch  $\mu m[n \leq 2m + 1]$ . Mit  $h(n) = \mu m[\frac{1}{2}m(m+1) > n]$  ist die Umkehrfunktion rekursiv als  $\beta_2^1(n) = (n \dot{-} \frac{1}{2}h(n)(h(n) \dot{-} 1), \frac{1}{2}h(n)(h(n) + 1) \dot{-} (n + 1))$ .

Sei nun  $\beta : \mathbb{N}^k \rightarrow \mathbb{N}^l$  bijektiv mit rekursiver Umkehrfunktion  $\beta^{-1}$ . Wenn  $X \subseteq \mathbb{N}^k$  rekursiv ist, dann ist  $\chi_{\beta[X]}(\bar{y}) = \chi_X(\beta^{-1}(\bar{y}))$  rekursiv. Wenn  $X = \pi[Y] \subseteq \mathbb{N}^k$  rekursiv aufzählbar ist mit rekursivem  $Y \subseteq \mathbb{N}^{k+1}$ , dann ist demnach auch  $\beta[X] = \pi[(\beta \times \text{id}_{\mathbb{N}})[Y]]$  rekursiv aufzählbar, da  $\beta \times \text{id}_{\mathbb{N}} : \mathbb{N}^{k+1} \rightarrow \mathbb{N}^{l+1}$  ebenfalls bijektiv mit rekursiver Umkehrfunktion  $\beta^{-1} \times \text{id}_{\mathbb{N}}$  ist.  $\square$

BEWEIS VON LEMMA 3.4.4, FORTSETZUNG:

(b) „ $\Leftarrow$ “ der zweiten Äquivalenz ist Teil des vorherigen Lemmas 3.4.5.

Für „ $\Rightarrow$ “ wählt man  $f = \beta_k^1$  und  $Y = \beta_1^k[X]$ .

(c) Eine rekursiv aufzählbare Menge  $X \subseteq \mathbb{N}^k$  ist per Definition Bild einer rekursiven Menge  $Y \subseteq \mathbb{N}^{k+1}$  unter der rekursiven Abbildung  $f = \pi = (\pi_1^{k+1}, \dots, \pi_k^{k+1})$  und mit Lemma 3.4.5 dann auch Bild der rekursiven Menge  $\beta_1^{k+1}[Y] \subseteq \mathbb{N}$  unter der rekursiven Abbildung  $f \circ \beta_{k+1}^1$ . Ist  $x_0 \in X = f[Y] \subseteq \mathbb{N}^k$  rekursiv aufzählbar wie im Lemma, dann ist  $X = g[\mathbb{N}^k]$  für die rekursive Funktion  $g(\bar{y}) := f(\bar{y}) \cdot \chi_Y(\bar{y}) + n_0 \cdot \chi_{\mathbb{N}^{k+1} \setminus Y}(\bar{y})$ .

Wenn  $X = g[\mathbb{N}^k]$  für rekursives  $g$ , ist  $X = g'[\mathbb{N}]$  für die rekursive Abbildung  $g' = g \circ \beta_k^1$ . Nach geeigneter Permutation der Koordinaten ist dann  $X$  die Projektion im Sinne von Definition 3.4.2 des Graphen von  $g'$ , der nach Lemma 3.4.4 (a) rekursiv ist.  $\square$

**Folgerung 3.4.6** *Falls  $X$  rekursiv aufzählbar und  $f$  rekursiv, ist  $f[X]$  rekursiv aufzählbar.*

BEWEIS:  $X$  ist nach Lemma 3.4.4 von der Form  $g[Y]$  für rekursives  $Y$  und rekursives  $g$ , also ist  $f[X] = (f \circ g)[Y]$  rekursiv aufzählbar, weil  $f \circ g$  als Komposition rekursiver Abbildung selbst rekursiv ist.  $\square$

## Partielle Funktionen

### Definition 3.4.7

Eine partielle Funktion  $f : \mathbb{N}^k \dashrightarrow \mathbb{N}$  für  $k \in \mathbb{N}$  heißt **rekursiv [recursive]**, wenn sie eine der Grundfunktionen aus Definition 3.4.1 ist oder sich daraus durch endlichfache Anwendung der Regeln [Komposition], [Primitive Rekursion] und [ $\mu$ -Rekursion] ergibt, die für partielle Funktionen wie folgt angepasst werden:

- Aus partiellen Funktionen  $f_1, \dots, f_l : \mathbb{N}^k \dashrightarrow \mathbb{N}$  und  $g : \mathbb{N}^l \dashrightarrow \mathbb{N}$  entsteht durch [Komposition] die partielle Funktion  $h : \mathbb{N}^k \dashrightarrow \mathbb{N}$  mit

$$h(\bar{a}) := \begin{cases} g(f_1(\bar{a}), \dots, f_l(\bar{a})) & \text{falls } f_1(\bar{a}), \dots, f_l(\bar{a}) \text{ und } g(f_1(\bar{a}), \dots, f_l(\bar{a})) \text{ definiert} \\ \text{unbestimmt} & \text{sonst} \end{cases}$$

- Aus partiellen Funktionen  $f : \mathbb{N}^k \dashrightarrow \mathbb{N}$  und  $g : \mathbb{N}^{k+2} \dashrightarrow \mathbb{N}$  entsteht durch [Primitive Rekursion] die partielle Funktion  $h : \mathbb{N}^{k+1} \dashrightarrow \mathbb{N}$  mit

$$h(0, \bar{a}) := \begin{cases} f(\bar{a}) & \text{falls } f(\bar{a}) \text{ definiert} \\ \text{unbestimmt} & \text{sonst} \end{cases}$$

$$h(n+1, \bar{a}) := \begin{cases} g(h(n, \bar{a}), n, \bar{a}) & \text{falls } h(n, \bar{a}) \text{ und } g(h(n, \bar{a}), n, \bar{a}) \text{ definiert} \\ \text{unbestimmt} & \text{sonst} \end{cases}$$

- Aus einer partiellen Funktion  $f : \mathbb{N}^{k+1} \dashrightarrow \mathbb{N}$  entsteht durch [ $\mu$ -Rekursion] die partielle Funktion  $h : \mathbb{N}^k \dashrightarrow \mathbb{N}$  mit

$$h(\bar{a}) := \begin{cases} \mu n [f(n, \bar{a}) = 0] & \text{falls ein } m \text{ mit } f(m, \bar{a}) = 0 \text{ existiert} \\ & \text{und } f(n, \bar{a}) \text{ für alle } n \leq m \text{ definiert ist} \\ \text{unbestimmt} & \text{sonst} \end{cases}$$

**Bemerkung 3.4.8 [ohne Beweis]** Die Definitionen von partiellen und totalen rekursiven Funktionen sind verträglich: Wenn eine totale Funktion im Sinne von Definition 3.4.7 rekursiv ist, ist sie auch gemäß Definition 3.4.1 rekursiv.

Eine partielle Funktion ist genau dann rekursiv, wenn ihr Graph rekursiv aufzählbar ist.<sup>53</sup>

Bilder rekursiv aufzählbarer Mengen unter partiellen rekursiven Funktionen sind wieder rekursiv aufzählbar und der Definitionsbereich einer partiellen rekursiven Funktion ist stets rekursiv aufzählbar. Eine Menge  $X$  ist genau dann rekursiv aufzählbar, wenn ihre partielle charakteristische Funktion  $\overline{\chi_X}$  rekursiv ist.

**Satz 3.4.9** Die rekursiven partiellen / totalen Funktionen sind genau die partiellen bzw. totalen Turing-berechenbaren Funktionen.

BEWEISIDEE: Die Grundfunktionen lassen sich durch Turing-Maschinen berechnen, und die verschiedenen Kompositions- und Rekursionsschritte lassen sich mit Turing-Maschinen nachbilden (im Detail aufwendig, aber prinzipiell nicht so schwer). Umgekehrt kann man zeigen, dass sich die Arbeitsweise von Turing-Maschinen durch rekursive Funktionen beschreiben lassen (das ist nicht so leicht; der nächste Abschnitt gibt ein paar Anhaltspunkte, wie es gehen könnte).  $\square$

### 3.5 Gödelisierung und die Arithmetik

Der folgende Abschnitt soll nur einen kurzen Ausblick geben, wie man mit rekursiven Funktionen die Unentscheidbarkeit zeigen kann. Auf Beweise wird vollständig verzichtet.

<sup>53</sup>Auch das ist in Einklang mit Lemma 3.4.4 (a): Wenn der Graph einer totalen Funktion rekursiv aufzählbar ist, ist er schon rekursiv!

Es gibt eine (primitiv) rekursive Bijektion

$$\langle \cdot \rangle : \mathbb{N}^* \rightarrow \mathbb{N}, n_1 \dots n_k \mapsto p_1^{n_1+1} \cdot \dots \cdot p_k^{n_k+1} - 1$$

wobei  $p_i$  die  $i$ -te Primzahl ist, also  $p_1 = 2, p_2 = 3, \dots$ . Das „leere Produkt“ ist 1; das leere Wort wird also auf 0 abgebildet. Mit Hilfe der eindeutigen Primfaktorzerlegung kann man sich leicht davon überzeugen, dass es sich um eine Bijektion handelt. Die Rekursivität zu zeigen, ist mit mehr Aufwand verbunden, aber nicht wirklich kompliziert. Außerdem ist auch die Umkehrfunktion in folgendem Sinne (primitiv) rekursiv: Man kann mit (primitiv) rekursiven Funktionen aus  $n = \langle n_1 \dots n_k \rangle$  die Länge  $k$  des kodierten Wortes sowie für jedes  $i$  die  $i$ -te Komponente  $n_i$  berechnen.

Nun kann man alles Mögliche als natürliche Zahlen kodieren, etwa Übergangsfunktionen von Turing-Maschinen oder endliche Programmläufe von Turing-Maschinen mit fester Eingabe oder  $\mathcal{L}$ -Formeln für eine abzählbare Sprache  $\mathcal{L}$ . Letzteres funktioniert zum Beispiel folgendermaßen: Man ordnet jedem Zeichen in  $\mathcal{L}$  und allen weiteren, in  $\mathcal{L}$ -Formeln vorkommenden Zeichen  $Z$  einen Code  $\ulcorner Z \urcorner \in \mathbb{N}$  zu, etwa:

$\ulcorner \top \urcorner = 0$	$\ulcorner \exists \urcorner = 7$	$\ulcorner f_0 \urcorner = 12$
$\ulcorner \perp \urcorner = 1$	$\ulcorner \forall \urcorner = 8$	$\ulcorner R_0 \urcorner = 13$
$\ulcorner \neg \urcorner = 2$	$\ulcorner \dot{=} \urcorner = 9$	$\ulcorner v_0 \urcorner = 14$
$\ulcorner \wedge \urcorner = 3$	$\ulcorner ( \urcorner = 10$	$\ulcorner f_1 \urcorner = 15$
$\ulcorner \vee \urcorner = 4$	$\ulcorner ) \urcorner = 11$	$\ulcorner R_1 \urcorner = 16$
$\ulcorner \rightarrow \urcorner = 5$		$\ulcorner v_1 \urcorner = 17$
$\ulcorner \leftrightarrow \urcorner = 6$		$\vdots$
		$\vdots$

Wichtig ist dabei, dass es eine Systematik gibt, die dazu führt, dass zum Beispiel die Menge der Codes aller Individuenvariablen eine rekursive Menge ist. Auch muss die Stelligkeit eines Funktions- oder Relationszeichens aus seinem Code berechenbar sein. Nötigenfalls muss man dazu die Sprache erweitern, so dass man etwa abzählbar viele Funktions- oder Relationszeichens jeder Stelligkeit hat.

Nun kodiert man  $\mathcal{L}$ -Formeln  $\varphi$  in  $\mathbb{N}$ , indem man sie zunächst als Zeichenkette  $Z_1 \dots Z_l$  schreibt und ihnen dann den Code

$$\ulcorner \varphi \urcorner := \langle \ulcorner Z_1 \urcorner \dots \ulcorner Z_l \urcorner \rangle$$

zuweist. Dieser Code  $\ulcorner \varphi \urcorner \in \mathbb{N}$  heißt auch **Gödel-Nummer** [Gödel number] von  $\varphi$ .

Die (bereits als Zeichenkette vorliegende) Formel  $\varphi = \exists v_0 f_1 v_0 \dot{=} f_0$  wird kodiert durch

$$\begin{aligned} \ulcorner \exists v_0 f_1 v_0 \dot{=} f_0 \urcorner &= \langle \ulcorner \exists \urcorner \ulcorner v_0 \urcorner \ulcorner f_1 \urcorner \ulcorner v_0 \urcorner \ulcorner \dot{=} \urcorner \ulcorner f_0 \urcorner \rangle \\ &= \langle 7 \ 4 \ 15 \ 14 \ 9 \ 12 \rangle \\ &= 2^8 \cdot 3^5 \cdot 5^{16} \cdot 7^{15} \cdot 11^{10} \cdot 13^{13} - 1 \end{aligned}$$

Wenn man alles richtig macht, kann man die meisten Eigenschaften von  $\mathcal{L}$ -Formeln berechenbar aus dem Code ablesen. Zum Beispiel sind

- $\{\ulcorner \varphi \urcorner \mid \varphi \text{ ist eine } \mathcal{L}\text{-Formel}\}$
- $\{\ulcorner \varphi \urcorner \mid \varphi \text{ ist eine } \mathcal{L}\text{-Formel mit genau einer freien Variablen } v_0\}$
- $\{\ulcorner \varphi \urcorner \mid \varphi \text{ ist eine } \mathcal{L}\text{-Tautologie}\}$

primitiv rekursive Teilmengen von  $\mathbb{N}$ . (Der Beweis ist langwierig, aber unspektakulär.)

Sei nun  $\mathcal{L}$  stets eine abzählbare Sprache, und es soll eine Kodierung der  $\mathcal{L}$ -Formeln als natürliche Zahlen festgelegt sein, die die gewünschten schönen Eigenschaften besitzt.

**Definition 3.5.1** Eine  $\mathcal{L}$ -Theorie  $T$  heißt **rekursiv axiomatisierbar** [recursively axiomatisable], wenn es ein  $T_0 \subseteq T$  gibt mit  $\{\varphi \mid T_0 \vdash \varphi\} = \{\varphi \mid T \vdash \varphi\}$  und so, dass  $\{\ulcorner \varphi \urcorner \mid \varphi \in T_0\}$  rekursiv aufzählbar ist, und **(semi-)entscheidbar**, wenn  $\{\ulcorner \varphi \urcorner \mid T \vdash \varphi\}$  rekursiv (aufzählbar) ist.

**Lemma 3.5.2** *Rekursiv axiomatisierbare  $\mathcal{L}$ -Theorien sind semi-entscheidbar, vollständige rekursiv axiomatisierbare  $\mathcal{L}$ -Theorien sind entscheidbar.*

BEWEISIDEE: Der erste Teil folgt aus der Vollständigkeit eines Kalküls wie  $\mathbb{K}$ : Da  $T$  ein rekursiv aufzählbares Axiomensystem hat, kann man die Axiome von  $\mathbb{K}$  rekursiv aufzählen, und damit dann auch alle  $\mathbb{K}$ -Beweise, und damit auch alle Codes von  $\mathcal{L}$ -Formeln, die in  $\mathbb{K}$ -Beweisen vorkommen.

Wenn  $T$  zusätzlich vollständig ist, kann man auch  $\{\ulcorner \varphi \urcorner \mid T \not\vdash \varphi\} = \{\ulcorner \varphi \urcorner \mid T \vdash \neg\varphi\}$  rekursiv aufzählen. Da außerdem  $\{\ulcorner \varphi \urcorner \mid \varphi \text{  $\mathcal{L}$ -Formel}\}$  rekursiv ist, bekommt man insgesamt die Entscheidbarkeit von  $T$ .  $\square$

Sei  $\mathcal{L}_{\mathcal{N}} = \{+, \cdot, S, 0, <\}$  und  $\mathcal{N}$  die  $\mathcal{L}_{\mathcal{N}}$ -Struktur der natürlichen Zahlen, d. h.

$$\mathcal{N} = (\mathbb{N}; +^{\mathbb{N}}, \cdot^{\mathbb{N}}, S^{\mathbb{N}}, 0^{\mathbb{N}}, <^{\mathbb{N}})$$

wobei  $S^{\mathbb{N}} = N$  die Nachfolgerfunktion sein soll.

**Satz 3.5.3** *Alle rekursiven Funktionen und alle rekursiv aufzählbaren Mengen sind in  $\mathcal{N}$  definierbar.*

Die Definierbarkeit von Mengen ist hier im Sinne von Bemerkung 2.3.9 zu verstehen:  $X \subseteq \mathbb{N}^k$  ist in  $\mathcal{N}$  definierbar, wenn es eine  $\mathcal{L}_{\mathcal{N}}$ -Formel  $\varphi(v_1, \dots, v_k)$  gibt mit  $X = \varphi^{\mathcal{N}}$ . Eine Funktion  $f : \mathbb{N}^k \dashrightarrow \mathbb{N}$  heißt definierbar, wenn ihr Graph definierbar ist. (Dies ist ein umfassenderes Konzept als die Definierbarkeit durch Terme aus Bemerkung 2.3.9!)

Es ist einfach zu sehen, dass die Grundfunktionen definierbar sind und dass sich Komposition und  $\mu$ -Rekursion definierbar ausdrücken lassen. Die Primitive Rekursion entzieht sich dagegen einer Definierbarkeit, da die zu definierende Funktion in der Definition selbst vorkommt. Zum Beweis des Satzes braucht man daher die oben bereits erwähnte Charakterisierung der rekursiven Funktionen ohne Primitive Rekursion, aber mit mehr Grundfunktionen.

Wenn man es sich genauer anschaut, stellt man fest, dass sich die rekursiven Funktionen und die rekursiv aufzählbaren Mengen durch  $\Sigma_1$ -Formeln definieren lassen. Das sind Formeln, die im Wesentlichen aus quantorenfreien Formeln durch Existenzquantifikation und *beschränkte Allquantifikation* hervorgehen. Beschränkte Allquantifikation einer Formel  $\varphi$  ist eine Formel von der Form  $\forall v_i (v_i < \tau(n) \rightarrow \varphi)$ , wobei  $\tau(n)$  ein Term von der Form  $S0 + \dots + S0$  ist, der eine natürliche Zahl beschreibt. Umgekehrt kann man zeigen, dass alle  $\Sigma_1$ -Formeln rekursiv aufzählbare Mengen definieren. Damit sind die  $\Sigma_1$ -definierbaren Funktionen gerade die rekursiven Funktionen, womit man eine weitere Charakterisierung der rekursiven bzw. der berechenbaren Funktionen erhält.

**Satz 3.5.4** *Die Arithmetik, also die vollständige Theorie der natürlichen Zahlen  $\text{Th}(\mathcal{N}) := \{\varphi \text{  $\mathcal{L}_{\mathcal{N}}$ -Aussage} \mid \mathcal{N} \models \varphi\}$ , ist unentscheidbar.*

**Folgerung 3.5.5** Jedes rekursiv aufzählbares Axiomensystem für  $\mathbb{N}$  ist unvollständig.

Beispiele für solche rekursiv aufzählbaren Axiomensysteme sind die *Peano-Arithmetik* oder die unten definierte *Robinson-Arithmetik*  $Q$ .

Da für jede  $\mathcal{L}_{\mathcal{N}}$ -Aussage  $\psi$  entweder  $\psi$  oder  $\neg\psi$  in  $\mathbb{N}$  gelten muss, wird das Corollar oft durch den Slogan „Es gibt in  $\mathbb{N}$  wahre Sätze, die nicht beweisbar sind“ paraphrasiert.

BEWEISIDEE FÜR DEN SATZ: Sei  $\alpha : \mathbb{N} \rightarrow \mathbb{N}$  eine rekursive Aufzählung aller Gödel-Nummern der  $\mathcal{L}_{\mathcal{N}}$ -Formeln mit einer freien Variablen  $v_0$ , und  $\alpha(n) = \ulcorner \varphi_n(v_0) \urcorner$ . Wenn  $\text{Th}(\mathcal{N})$  entscheidbar wäre, also rekursiv, wäre auch  $A := \{n \in \mathbb{N} \mid \mathcal{N} \models \neg\varphi_n[\frac{\tau(n)}{v_0}]\}$  rekursiv. Dann wäre aber  $A$  durch eine  $\mathcal{L}_{\mathcal{N}}$ -Formeln  $\varphi_{n_0}(v_0)$  definierbar. Nun führt ein Diagonalargument zum Widerspruch:

$$\begin{aligned} n_0 \in A &\iff \mathcal{N} \models \varphi_{n_0}[n_0] && \text{da } \varphi_{n_0} \text{ } A \text{ definiert} \\ &\iff \mathcal{N} \models \varphi_{n_0}[\frac{\tau(n_0)}{v_0}] && \text{da sich } n_0 \in \mathbb{N} \text{ durch den Term } \tau(n_0) \text{ beschreiben lässt} \\ &\iff \mathcal{N} \not\models \neg\varphi_{n_0}[\frac{\tau(n_0)}{v_0}] \\ &\iff n_0 \notin A && \text{nach Definition von } A \qquad \square \end{aligned}$$

Zum Beweis der Unvollständigkeit reicht ein endliches Axiomensystem für  $\mathbb{N}$ , nämlich die (nach Raphael Robinson benannte) *Robinson-Arithmetik*  $Q$ :

$$\begin{aligned} &\forall v_0 (v_0 + 0 \doteq v_0) \\ &\forall v_0 \forall v_1 (v_0 + S(v_1) \doteq S(v_0 + v_1)) \\ &\forall v_0 (v_0 \cdot 0 \doteq 0) \\ &\forall v_0 \forall v_1 (v_0 \cdot S(v_1) \doteq (v_0 \cdot v_1) + v_0) \\ &\forall v_0 \neg v_0 < 0 \\ &\forall v_0 \forall v_1 (v_0 < S(v_1) \leftrightarrow (v_0 < v_1 \vee v_0 \doteq v_1)) \end{aligned}$$

Damit bekommt man dann auch die Unvollständigkeit der Prädikatenlogik in der Sprache  $\mathcal{L}_{\mathcal{N}}$ , denn wenn  $\chi$  die Konjunktion aller Axiome der Robinson-Arithmetik ist, wären mit der Prädikatenlogik auch alle  $\mathcal{L}_{\mathcal{N}}$ -Formeln der Form  $(\chi \rightarrow \varphi)$  entscheidbar, und damit wäre  $Q$  entscheidbar.

Zum Abschluss sei ein schwerer Satz erwähnt, der zeigt, dass die rekursiv aufzählbaren Mengen durch Formeln einfachster Art definierbar sind:

**Satz 3.5.6 (Satz von (Davis, Putnam, Robinson und) Matiyasevich<sup>54</sup>, 1970)**

$R \subseteq \mathbb{N}^k$  ist genau dann rekursiv aufzählbar, wenn es Polynome  $P, Q \in \mathbb{N}[X_1, \dots, X_{k+l}]$  für ein  $l \in \mathbb{N}$  gibt mit

$$R = \{(n_1, \dots, n_k) \in \mathbb{N}^k \mid \text{es gibt } m_1, \dots, m_l \in \mathbb{N} \text{ mit } P(n_1, \dots, n_k, m_1, \dots, m_l) = Q(n_1, \dots, n_k, m_1, \dots, m_l)\}$$

d. h. wenn  $R$  durch eine Formel

$$\exists v_{k+1} \dots \exists v_{k+l} \tau_1(v_1, \dots, v_k, v_{k+1}, \dots, v_{l+k}) \doteq \tau_2(v_1, \dots, v_k, v_{k+1}, \dots, v_{l+k})$$

mit  $\{+, \cdot, 0, 1\}$ -Termen  $\tau_1, \tau_2$  definiert ist.

<sup>54</sup>Davis, Putnam und Julia Robinson haben entscheidende Vorarbeiten geleistet, Matiyasevich dann den Schlusspunkt gesetzt.



## 4 Anhang

### 4.1 Ein Sequenzenkalkül für die Aussagenlogik

In diesem Abschnitt soll für die Aussagenlogik eine Variante des von Gerhard Gentzen eingeführten *Sequenzenkalküls* [sequent calculus] vorgestellt werden. In ihm kann man logische Folgerungen in einer Art kombinatorischem Spiel erzeugen. Diese Version ist sehr symmetrisch aufgebaut und macht dadurch die Dualität deutlich sichtbar.

**Definition 4.1.1** Eine Sequenz [sequent] besteht aus zwei Mengen von je endlich vielen Formeln, einer „linken“ und einer „rechten Menge“. Man schreibt die Sequenz, indem man die Elemente der beiden Mengen auf den entsprechenden Seiten eines Trennzeichens auflistet:

$$F_1 F_2 \dots F_m \Vdash G_1 G_2 \dots G_n$$

$\Vdash$

Die intendierte Bedeutung der Sequenz ist

$$(F_1 \wedge F_2 \wedge \dots \wedge F_m) \vdash (G_1 \vee G_2 \vee \dots \vee G_n)$$

Eine Sequenz ist ableitbar, wenn sie nach den unten stehenden Regeln konstruiert werden kann, und korrekt, wenn ihre intendierte Bedeutung stimmt.

Die Reihenfolge der Formeln auf jeder Seite ist gleichgültig, d. h. man darf links und rechts des Trennzeichens  $\Vdash$  beliebig umsortieren. Eine Seite kann auch die leere Mengen an Formeln enthalten (d. h.  $m = 0$  bzw.  $n = 0$  ist zulässig).

Es gilt nun (Beweis auf Seite 115):

**Satz 4.1.2** Der Sequenzenkalkül ist korrekt (sound) und vollständig, d. h. jede ableitbare Sequenz ist korrekt und jede korrekte Sequenz ist ableitbar.

Insbesondere ist also für jede Tautologie  $F$  die Sequenz  $\Vdash F$  ableitbar und für jede logische Äquivalenz  $F \sim G$  sind die beiden Sequenzen  $F \Vdash G$  und  $G \Vdash F$  ableitbar. Die leere Sequenz „  $\Vdash$  “ ist dagegen nicht ableitbar.

Es gibt zwei Arten von Regeln: ein Axiomenschema, das gewisse Sequenzen als ableitbar setzt, und Ableitungsregeln, mit denen sich aus ableitbaren Sequenzen weitere ableitbare Sequenzen ergeben. Die Ableitungsregeln des Kalküls lassen sich dabei als Einführungs- und Eliminationsregeln für die Junktoren verstehen. Der Übersichtlichkeit halber schreibe ich  $\Phi$  und  $\Psi$  als Abkürzung für endliche Mengen von Formeln.

*Identitätsregel:* Jede Sequenz, in der eine Formel sowohl links wie rechts auftaucht, ist ableitbar.

(Id) $\Phi F \Vdash \Psi F$
-----------------------------

Die Regeln für die Junktoren sind folgendermaßen zu lesen: Wenn die Sequenz oder die beiden Sequenzen oberhalb der waagrechten Linie ableitbar sind, dann ist auch die Sequenz unterhalb der Linie ableitbar. In dieser Weise erhält man aus der Liste unten die *Einführungsregeln* der Junktoren, jeweils für die linke und die rechte Seite (nur für Verum und Falsum gibt es nur eine Seite). Diese Regeln sind aber auch in der umgekehrten Richtung zu lesen: Wenn die Sequenz unterhalb der Linie ableitbar ist, dann ist auch die Sequenz oder sind auch die beiden

Sequenzen oberhalb der Linie ableitbar. Das sind dann die entsprechenden *Eliminationsregeln* für links bzw. rechts.

*Junktorenregeln:*

( $\top$ -Einf/Elim <sub>links</sub> )	$\frac{\Phi \Vdash \Psi}{\Phi \top \Vdash \Psi}$	$\frac{\Phi \Vdash \Psi}{\Phi \Vdash \Psi \perp}$	( $\perp$ -Einf/Elim <sub>rechts</sub> )
( $\neg$ -Einf/Elim <sub>links</sub> )	$\frac{\Phi \Vdash \Psi \ F}{\Phi \neg F \Vdash \Psi}$	$\frac{\Phi \ G \Vdash \Psi}{\Phi \Vdash \Psi \neg G}$	( $\neg$ -Einf/Elim <sub>rechts</sub> )
( $\wedge$ -Einf/Elim <sub>links</sub> )	$\frac{\Phi \ F \ G \Vdash \Psi}{\Phi (F \wedge G) \Vdash \Psi}$	$\frac{\Phi \Vdash \Psi \ F \ G}{\Phi \Vdash \Psi (F \vee G)}$	( $\vee$ -Einf/Elim <sub>rechts</sub> )
( $\vee$ -Einf/Elim <sub>links</sub> )	$\frac{\Phi \ F \Vdash \Psi \quad \Phi \ G \Vdash \Psi}{\Phi (F \vee G) \Vdash \Psi}$	$\frac{\Phi \Vdash \Psi \ F \quad \Phi \Vdash \Psi \ G}{\Phi \Vdash \Psi (F \wedge G)}$	( $\wedge$ -Einf/Elim <sub>rechts</sub> )
( $\rightarrow$ -Einf/Elim <sub>links</sub> )	$\frac{\Phi \Vdash \Psi \ F \quad \Phi \ G \Vdash \Psi}{\Phi (F \rightarrow G) \Vdash \Psi}$	$\frac{\Phi \ F \Vdash \Psi \ G}{\Phi \Vdash \Psi (F \rightarrow G)}$	( $\rightarrow$ -Einf/Elim <sub>rechts</sub> )
( $\leftrightarrow$ -Einf/Elim <sub>links</sub> )	$\frac{\Phi \Vdash \Psi \ F \ G \quad \Phi \ F \ G \Vdash \Psi}{\Phi (F \leftrightarrow G) \Vdash \Psi}$	$\frac{\Phi \ F \Vdash \Psi \ G \quad \Phi \ G \Vdash \Psi \ F}{\Phi \Vdash \Psi (F \leftrightarrow G)}$	( $\leftrightarrow$ -Einf/Elim <sub>rechts</sub> )

### Beispiele:

Das *ex-falso-quodlibet*-Gesetz ist ableitbar:

- (1)  $\perp \Vdash \perp \ F$  (Id)
- (2)  $\perp \Vdash \ F$  mit ( $\perp$ -Elim<sub>rechts</sub>) aus (1)

Das Prinzip des ausgeschlossenen Dritten ist ableitbar:

- (1)  $F \Vdash F$  (Id)
- (2)  $\Vdash F \neg F$  mit ( $\neg$ -Einf<sub>links</sub>) aus (1)
- (3)  $\Vdash (F \vee \neg F)$  mit ( $\vee$ -Einf<sub>rechts</sub>) aus (2)

Die Gesetze von *de Morgan* sind ableitbar; hier beispielhaft eines von beiden:

- (1)  $F \Vdash F \ G$  (Id)
- (2)  $G \Vdash F \ G$  (Id)
- (3)  $(F \vee G) \Vdash F \ G$  mit ( $\vee$ -Einf<sub>links</sub>) aus (1) und (2)
- (4)  $\neg F \neg G \Vdash \neg(F \vee G)$  mit ( $\neg$ -Einf<sub>rechts</sub>) und zweimal ( $\neg$ -Einf<sub>links</sub>) aus (3)
- (5)  $(\neg F \wedge \neg G) \Vdash \neg(F \vee G)$  mit ( $\wedge$ -Einf<sub>links</sub>) aus (4)
- (6)  $F \Vdash (F \vee G)$  mit ( $\vee$ -Einf<sub>rechts</sub>) aus (1)
- (7)  $G \Vdash (F \vee G)$  mit ( $\vee$ -Einf<sub>rechts</sub>) aus (2)
- (8)  $\Vdash \neg F (F \vee G)$  mit ( $\neg$ -Einf<sub>rechts</sub>) aus (6)
- (9)  $\Vdash \neg G (F \vee G)$  mit ( $\neg$ -Einf<sub>links</sub>) aus (7)
- (10)  $\Vdash (\neg F \wedge \neg G) (F \vee G)$  mit ( $\wedge$ -Einf<sub>rechts</sub>) aus (8) und (9)
- (11)  $\neg(F \vee G) \Vdash (\neg F \wedge \neg G)$  mit ( $\neg$ -Einf<sub>links</sub>) aus (10)

BEWEIS VON SATZ 4.1.2 : Ich schreibe der Einfachheit halber  $\bigwedge \Phi$  für die Konjunktion der Formeln in  $\Phi$  und  $\bigvee \Psi$  für die Disjunktion der Formeln in  $\Psi$ .

Die Korrektheit von ( $\top$ -Einf<sub>links</sub>), ( $\top$ -Elim<sub>links</sub>), ( $\perp$ -Einf<sub>rechts</sub>) und ( $\perp$ -Elim<sub>rechts</sub>) folgt daraus, dass

$$\bigwedge \Phi \vdash \bigvee \Psi \iff (\bigwedge \Phi \wedge \top) \vdash (\bigvee \Psi \vee \perp)$$

Die Korrektheit von ( $\wedge$ -Einf<sub>links</sub>) und ( $\vee$ -Einf<sub>rechts</sub>) ist trivial. Die Korrektheit der Negationsregeln, beispielsweise ( $\neg$ -Einf<sub>links</sub>)/( $\neg$ -Elim<sub>links</sub>) sieht man so:

$$\begin{aligned} (F_1 \wedge \dots \wedge F_m) \vdash (\bigvee \Psi \vee F) &\iff \vdash ((F_1 \wedge \dots \wedge F_m) \rightarrow (\bigvee \Psi \vee F)) \\ &\iff \vdash (\neg(F_1 \wedge \dots \wedge F_m) \vee F \vee \bigvee \Psi) \\ &\iff \vdash (\neg F_1 \vee \dots \vee \neg F_m \vee \neg F \vee \bigvee \Psi) \\ &\iff \dots \iff (F_1 \wedge \dots \wedge F_m \wedge \neg F) \vdash \bigvee \Psi \end{aligned}$$

Mit den Negationsregeln kann man nun sukzessive alles auf eine Seite bringen; die Korrektheit der ( $\wedge$ -Einf<sub>rechts</sub>) reduziert sich dadurch auf

$$\vdash (\bigwedge \Phi \rightarrow (F \wedge G)) \iff \vdash (\bigwedge \Phi \rightarrow F) \text{ und } \vdash (\bigwedge \Phi \rightarrow G)$$

was man leicht dadurch nachprüft, dass  $(\bigwedge \Phi \rightarrow (F \wedge G)) \sim ((\bigwedge \Phi \rightarrow F) \wedge (\bigwedge \Phi \rightarrow G))$ . Die ( $\vee$ -Einf<sub>links</sub>) lässt sich durch Kontraposition darauf zurückführen. Für die Implikations- und Äquivalenzregeln setzt man die Definition der entsprechenden Junktoren ein. Insgesamt ist damit die Korrektheit des Sequenzenkalküls gezeigt.

Für die Vollständigkeit des Kalküls überlegt man sich, dass man die Ableitbarkeit einer korrekten Sequenz durch sukzessive Elimination der führenden Junktoren – also durch Anwenden der (korrekten!) Eliminationsregeln – auf die Ableitbarkeit von korrekten Sequenzen, in denen keine Junktoren außer möglicherweise  $\top$  rechts oder  $\perp$  links vorkommen, zurückführen kann.

Eine Sequenz  $\Phi \Vdash \Psi \top$ , in der  $\top$  rechts vorkommt, ist aus ( $\text{Id}$ )  $\Phi \top \Vdash \Psi \top$  mit ( $\top$ -Elim<sub>links</sub>) ableitbar. Entsprechend ist eine Sequenz, in der  $\perp$  links vorkommt, aus ( $\text{Id}$ )  $\Phi \perp \Vdash \Psi \perp$  mit ( $\perp$ -Elim<sub>rechts</sub>) ableitbar. (Alle solchen Sequenzen sind stets korrekt.)

Es bleibt also die Ableitbarkeit korrekter Sequenzen zu zeigen, in denen keine Junktoren mehr vorkommen, also von Sequenzen der Form  $A_{i_1} \dots A_{i_m} \Vdash A_{j_1} \dots A_{j_m}$ . Eine solche Sequenz ist genau dann korrekt, wenn  $\vdash (\neg A_{i_1} \vee \dots \vee \neg A_{i_m} \vee A_{j_1} \vee \dots \vee A_{j_m})$ , was genau dann der Fall ist, wenn eine Aussagenvariable  $A_{i_k}$  der linken Seite auch als  $A_{j_l}$  auf der rechten Seite vorkommt. Damit ist die Sequenz aber als Instanz der Identitätsregel ableitbar!  $\square$

Nach Beweis des Satzes kann man in den Sequenzen dann auch mit gutem Gewissen  $\vdash$  statt  $\Vdash$  schreiben!

Dieser Beweis ist übrigens konstruktiv, d. h. er zeigt nicht nur, dass jede korrekte Sequenz ableitbar ist, sondern er liefert auch eine Anleitung, wie man eine korrekte Sequenz mit Hilfe der Regeln ableitet. Ein wesentliches Merkmal dieser Ableitungen ist, dass die Formeln sukzessive komplexer werden. Daher kann man den Sequenzenkalkül der Aussagenlogik auch als Entscheidungsverfahren benutzen: Wenn eine Sequenz nicht mittels des Verfahrens aus dem Beweis ableitbar ist, ist sie generell nicht ableitbar. Für Erweiterungen des Sequenzenkalküls auf die Prädikatenlogik gilt dies nicht mehr, da die Prädikatenlogik nicht entscheidbar ist.

## Literatur

Es gibt eine Vielzahl von Büchern zur (mathematischen) Logik, von denen ich hier nur diejenigen aufführe, mit denen ich gearbeitet habe. Es gibt in der Logik leider weder eine Standardterminologie noch Standardnotationen. Besonders ältere Bücher sind daher manchmal sehr gewöhnungsbedürftig, aber auch neuere sind manchmal schwer zu verstehen, wenn man sie mit-tendrin aufschlägt. Warnen möchte ich vor der Vielzahl von Einführungen in die formalisierte Logik für Nicht-Mathematiker und Nicht-Informatiker (zum Beispiel für Philosophen). Diese Bücher haben oft einen anderen Fokus, sind manchmal ideologisch und teilweise schlichtweg schlecht.

- René Cori & Daniel Lascar: *Logique mathématique*, 2. Auflage, Masson, Paris 1994. (Gibt es auch in englischer Übersetzung.)
- Heinz-Dieter Ebbinghaus, Jörg Flum & Wolfgang Thomas: *Einführung in die mathematische Logik*, 5. Auflage, Spektrum Verlag, Berlin 2007.
- Uwe Schöning: *Logik für Informatiker*, 5. Auflage, Spektrum Verlag, Berlin 2005.
- Martin Ziegler: *Mathematische Logik*, Birkhäuser, Basel 2010.

Zu der Vorlesung „Logik für Studierende der Informatik“ gibt es von Martin Ziegler ein Kurzs-kript vom WS 2006/07 (letzte Version WS 2012/13), von Heike Mildenerger ein ausführliches Skript vom WS 2011/12 und von Amador Martín Pizarro ein Skript vom WS 2020/21. Diese Vorlesungen hatten zum Teil andere Inhalte oder Schwerpunkte.

Nicht abgedeckt in den obigen Büchern und Skripten ist der Abschnitt über intuitionistische Logik. Die hier vorgestellte Semantik versteht man am besten über die Interpretation der in-tuitionistische Aussagenlogik in der Modallogik S4. Vorschläge für weitere Lektüre in diese Richtung sind:

- Melvin Fitting: *Intuitionistic Logic, Model Theory, and Forcing*, North-Holland, Amster-dam 1969.
- Patrick Blackburn, Maarten de Rijke & Yde Venema: *Modal Logic*, Cambridge University Press 2001.

## Namenspaten und wichtige Personen

*George Boole* (1815–1864): formale Aussagenlogik; Boole'sche Algebra

*Luitzen Egbertus Jan Brouwer* (1881–1966): Begründer des Intuitionismus

*Georg Cantor* (1845–1918): Begründer der Mengenlehre; Diagonalargument

*Alonzo Church* (1903–1995): Church'sche These

*Stephen Cook* (\* 1939): Satz von Cook und Levin

*Haskell Brooks Curry* (1900–1982): Currying, Curry-Howard-Korrespondenz (und Programmiersprache Haskell)

*Martin Davis* (\* 1928): Satz von Davis, Putnam, Robinson und Matiyasevich

*Gottlob Frege* (1848–1925): formale Prädiaktenlogik; aus seinem „Behauptungsstrich“ ist das Zeichen  $\vdash$  entstanden

*Gerhard Gentzen* (1909–1945): Gentzen-Kalküle

*Kurt Gödel* (1906–1978): Vollständigkeitssatz, Unvollständigkeitssatz

*Helmut Hasse* (1898–1979): Hasse-Diagramme

*Leon Henkin* (1921–2006): Henkin-Modelle, -Konstanten und -Axiome

*Jacques Herbrand* (1908–1931): Herbrand-Normalform, Satz von Herbrand

*Arend Heyting* (1898–1980): formaler Intuitionismus; Heyting-Algebren

*David Hilbert* (1862–1943): Hilbert-Kalküle

*Alfred Horn* (1918–2001): Horn-Formeln, Horn-Klauseln

*William Howard* (\* 1926): Curry-Howard-Korrespondenz

*Jean-Louis Krivine* (\* 1939): Krivines Hut

*Leonid Levin* (\* 1948): Satz von Cook und Levin

*Adolf Lindenbaum* (1904–1941): Tarski-Lindenbaum-Algebra

*Yuri Matiyasevich* (\* 1947): Satz von Davis, Putnam, Robinson und Matiyasevich

*Augustus de Morgan* (1806–1871): Regeln von de Morgan

*Charles Sanders Peirce* (1839–1914): formale Prädiaktenlogik; Peirce-Funktion: anderer Name für den NOR-Junktor

*Hilary Putnam* (1926–2016): Satz von Davis, Putnam, Robinson und Matiyasevich

*Willard Van Orman Quine* (1908–2000): Methode von Quine

*John Alan Robinson* (1930–2016): Resolution und Unifikation

*Julia Robinson* (1919–1985): Satz von Davis, Putnam, Robinson und Matiyasevich

*Raphael Robinson* (1911–1995): Robinson-Arithmetik  $Q$

*Henry Maurice Sheffer* (1882–1964): Sheffer-Strich: anderer Name für den NAND-Junktor

*Albert Thoralf Skolem* (1887–1963): Skolem-Funktionen und -Normalform, Skolemisierung

*Marshall Stone* (1903–1989): Darstellungssatz von Stone

*Alfred Tarski* (1901–1983): Tarski-Lindenbaum-Algebra

*Alan Turing* (1912–1954): Turing-Maschine

*John Venn* (1834–1923): Venn-Diagramme

*Martin Ziegler* (\* 1948): Symbol  $\doteq$  für das Gleichheitszeichen in der Prädikatenlogik

## Schlussbemerkung und „Plagiats-Disclaimer“

Die erste Version dieses Skript ist zu der im Wintersemester 2016/17 an der Albert-Ludwigs-Universität in Freiburg gehaltenen Vorlesung „Logik für Studierende der Informatik“ entstanden, wurde im Wintersemester 2017/18 und im Wintersemester 2020/21 substantiell überarbeitet. Nicht alle Abschnitte wurden in jedem dieser Semester behandelt.

Das Skript ist nach in der Mathematik gängiger Vorgehensweise angefertigt. Dies bedeutet, dass es keinen Anspruch auf eine eigene wissenschaftliche Leistung erhebt und keine eigenen Ergebnisse wiedergibt, sondern die Ergebnisse anderer darstellt (die im Wesentlichen zwischen 1850 und 1975 entstanden sind). Die konkrete Darstellung und Ausformulierung stammt allerdings von mir (inklusive aller Fehler).

Da Mathematik weitgehend ahistorisch betrieben wird, ließe sich nur mühsam zurückverfolgen, von wem welche Fragestellungen, Begriffe, Sätze, Beweise oder Beweistechniken stammen. Zu einem großen Teil muss man die Entwicklung eines mathematischen Teilgebiets auch als ein Gemeinschaftswerk einer mathematischen *Community* verstehen. Ich habe mich nicht um eine historische Darstellung bemüht; Zuweisungen von Sätzen oder Beweisen zu Mathematikern folgen der Überlieferung und müssen nicht immer historisch exakt sein.

Der Aufbau der Vorlesung folgt wesentlich der von Martin Ziegler im Wintersemester 2012/13 gehaltenen Vorlesung, hat sich aber in Teilen weiterentwickelt. Meine Darstellung ist insgesamt natürlich stark beeinflusst von den Logik-Vorlesungen, die ich bei Jacques Stern, Jean-Louis Krivine und Ramez Labib-Sami in Paris gehört und bei Martin Ziegler in Freiburg als Assistent betreut habe, sowie von den Skripten und Büchern von Kollegen, die im Literaturverzeichnis angegeben sind. Diese verschiedenen Einflüsse sind nicht zu trennen und können daher nicht einzeln dargelegt werden.

Korrekturen und Verbesserungsvorschläge verdanke ich Andreas Claessens und Roland Munt-schick sowie unzähligen teils anonymen Studierenden der Informatik. Über die Mitteilung verbliebener Fehler bin ich dankbar.