Einführung in die Logik

Jiří Adámek
weiterentwickelt von Jürgen Koslowski

Sommersemester 2013 + 2014 + 2016 + 2017

Version vom 12. Juli 2017

Institut für Theoretische Informatik

Inhaltsverzeichnis

1 Einleitung: Logische Systeme **4**

I Aussagenlogik **6**

2 Bausteine der Aussagenlogik **7**
2.1 Geschichte 7
2.2 Alphabet und Syntax der Aussagenlogik 8
2.3 Semantik der Aussagenlogik 11
2.4 Abgeleitete Symbole 13
2.5 Bindungskonventionen 15
2.6 Wahrheitstafeln 16
2.7 Erfüllbarkeit und Tautologien 17
2.8 Äquivalenz von Formeln 19

3 Eigenschaften von Formeln **21**
3.1 Eigenschaften der Negation 21
3.2 Eigenschaften von Kon- und Disjunktion 22
3.3 Adäquate Junktoren 23

4 Normalformen **25**
4.1 Negation-Normalform (NNF) 25
4.2 Konjunktive Normalform 28
4.3 Disjunktive Normalform 31

5 Logische Konsequenz **33**

6 Resolutionsmethode der Aussagenlogik **37**

7 Beweistheorie **46**
7.1 Natürliche Deduktion (ND) 47
7.2 ND-Regeln für die Konjunktion 49

7.3 ND-Regeln der Implikation 50
7.4 Präzisierung des Tableaux-Formalismusses 53
7.5 Regeln der Disjunktion 53
7.6 ND-Regeln der Negation und Absurdität 55
7.7 Zusammenfassung der ND-Regeln 62
7.8 Korrektheit und Vollständigkeit der Natürlichen Deduktion 64
7.9 Hilberts Axiomatisierung 66

8 Hornlogik **68**
8.1 Hornformeln 68
8.2 Die SLD-Resolutionsmethode 70
8.3 Der Markierungsalgorithmus 71
8.4 Zusammenfassung 74

II Prädikatenlogik **75**

9 Syntax der Prädikatenlogik **76**
9.1 Einleitung 76
9.2 Signaturen 80
9.3 Das Alphabet der Prädikatenlogik 82
9.4 Die Syntax der Prädikatenlogik 83
9.5 Die Mindestaxiome der Prädikatenlogik 90

10 Semantik der Prädikatenlogik **92**

11 Logische Äquivalenz **98**

12 Normalformen **103**

13 Herbrandsche Modelle und abstrakte Datentypen **108**

14 Resolutionsmethode der Prädikatenlogik **113**

A Mathematische Grundlagen **119**
A.1 Mengen 119
A.2 Relationen und Funktionen 121
A.3 Geordnete Mengen 123
A.4 Graphen und Bäume 124
A.5 Äquivalenzklassen und Partitionen 126
A.6 Tupel als Funktionen und die Natur des leeren Worts ε 127

A.7 Abzählbarkeit 127

Index **130**

1. Einleitung: Logische Systeme

Formale Logik kann man als den Versuch auffassen, das Denken zu mechanisieren. Als „wahr“ gilt nicht, was aufgrund einer argumentativen Begründung plausibel gemacht, sondern das, was mittels einer systematischen Abfolge mechanischer Arbeitsschritte nachgewiesen werden kann.

Ein logisches System überprüft dabei nicht den Wahrheitsgehalt einzelner Aussagen, sondern es ermöglicht festzustellen, was zusätzlich wahr ist, wenn eine Menge von Aussagen als wahr vorausgesetzt wird. Diese Erweiterung bereits vorhandenen Wissens ist das Forschungsgebiet der Logik.

Es gibt diverse Arten formaler logischer Systeme, die sich zum Teil unterschiedliche Fragestellungen widmen.

In diesem Skript werden nur einige dieser Systeme eingehender betrachtet: klassische Aussagenlogik, Gleichungslogik und Prädikatenlogik. Die große Auswahl existierender logischer Systeme entspricht ähnlich wie die Vielfalt an Programmiersprachen den unterschiedlichen Typen von Anwendungen, die Logik in der Informatik und auch in anderen Gebieten hat.

Die hier vorgestellten Logiken arbeiten mit streng formulierten Sätzen – Formeln –, die aus einem klar definierten Vorrat an Zeichen – Symbolen – zusammengesetzt werden. Allerdings gilt nicht alles was formulierbar ist, als wahr. Deshalb gehört zu jeder Logik immer auch ein Teil, der prüft, welche der korrekt formulierten Sätze auch tatsächlich im System gültig, das heißt „wahr“, sind. Dem semantischen Ansatz zur Wahrheitssuche steht dabei ein syntaktischwer Ansatz ur Seite, die sog. Beweistheorie. Dabei geht es darum, gemäß welcher syntaktischer Umfomrungen man aus als wahr vorausgesetzten Prämissen nur wahre Schlußfolgerungen ziehen kann. Deren Wahrheitsgehalt braucht dann nicht mehr semantisch überprüft zu werden.

Jedes der behandelten logischen Systeme besitzt die folgenden Grundbausteine:

- **Alphabet:** Diese besteht aus allen Symbolen, die die spezifische Logik nutzt, um formal korrekte Formeln aufzubauen.

  Dazu zählt etwa die Mengen

  - ▷ aller *atomaren Aussagen* (oder *Variablen*) „A_0“, „A_1“, „A_2“, …;
  - ▷ aller logischen *Junktoren* zum Verknüpfen einer bestimmten Anzahl von Aussagen, so etwa „$\top$“ (**wahr**, 0-stellig), „$\bot$“ (**falsch**, 0-stellig), „$\wedge$“ (**und**, 2-stellig), „$\vee$“ (**oder**, 2-stellig) und „$\neg$“ (**nicht**, 1-stellig) und ggf. weitere, sowie
  - ▷ der *Hilfssymbole* wie etwa der Klammern „ (“ und „) “.

- **Syntax:** Dies sind die Regeln, die bestimmen, wie man formal korrekte Formeln aus den Symbolen des Alphabets zusammensetzt.

  Meist werden diese Regeln wie folgt in zwei Schritten angegeben.

  ▷ Zuerst legt man fest, was die *atomaren*, nicht weiter zerlegbaren Formeln sind. Normalerweise kommen hier die atomaren Aussagen und evt. die 0-stelligen Junktoren $\top$ und $\bot$ vor.

  ▷ Anschließend wird bestimmt, wie aus diesen mit Hilfe der höherstelligen Junktoren *zusammengesetzte* Formeln korrekt aufgebaut werden können.

  Dieses zweistufige Verfahren wird als *strukturelle Rekursion* bezeichnet. Mit der zugehörigen *strukturellen Induktion* lassen sich dann Eigenschaften korrekt gebildeter Formeln beweisen.

  In der unten eingeführten Aussagenlogik ist etwa die Formel „ $A_0 \vee A_1$ “ formal korrekt, während „ $A_0 \vee A_1 \wedge$ “ nicht formal korrekt ist.

- **Semantik:** Diese weist den Formeln ihre „Bedeutung“ zu, was verschiedene Formen annehmen kann. Auf jeden Fall handelt es sich um eine *Abbildung* von Formeln auf geeignete Wahrheitswerte, meist 0 oder 1, s.u.. Ihre Definition bedient sich, wie auch andere Abbildungen auf der Menge aller Formeln, meist der oben erwähnten strukturellen Rekursion, indem zuerst atomaren Formeln eine Semantik zugewiesen wird, bevor man die Semantik zusammengesetzter Formeln aus der Semantik ihrer Bestandteile aufbaut.

  Achtung: damit das funktionieren kann, muß die Menge der semantischen Werte eine innere Struktur aufweisen, die es erlaubt, Werte entsprechend den verwendeten logischen Junktoren zu kombinieren! Stichwort: Boole'sche Algebra.

  In der *Aussagenlogik* geht es darum den *Wahrheitsgehalt* von korrekt gebildeten Formeln zu bestimmen, sie können entweder wahr oder falsch sein. Diese Wahrheitswerte werden oft binär mit 1 für „wahr“ und 0 für „falsch“ dargestellt; die Semantik-Abbildung hat als Zielbereich also die Menge $2 = \{0, 1\}$.

  Alternativ befasst sich die *intuitionistische Logik* mit der *Beweisbarkeit* von Aussagen, was einen fundamentalen Unterschied darstellt.

- **Logische Kalküle:** Diese beinhalten Regeln zur syntaktischen Transformation korrekter Formeln jenseits ihres Aufbaus mittels struktureller Induktion aber *unter Beibehaltung ihrer Semantik*. Konkret fällt darunter die Beweistheorie, in der man versucht, aus gegebenen Prämissen, die als wahr vorausgesetzt sind, bestimmte wahre Schlußfolgerungen zu ziehen, ohne dabei die Semantik dieser Schlußfolgerungen nochmal separat untersuchen zu müssen.

  Um die Semantik einer Formel F zu bestimmen kann es neben der strukturellen Induktion also andere, im Idealfall effizientere Methoden geben, etwa indem man F als Ergebnis einer zulässigen Transformation einer Formel E von bekannter Semantik darstellt.

Das Hauptinteresse der Logik gilt natürlich der Semantik und den logischen Kalkülen, Syntax als Selbstzweck ist eher uninteressant, aber als Grundlage für Semantik und Kalküle unverzichtbar.

Teil I

Aussagenlogik

2. Bausteine der Aussagenlogik

2.1 Geschichte

Schon die Griechen (Aristoteles, 384vC–322vC) haben den *formalen* Charakter logischen Schließens erkannt:

Alle Menschen sind sterblich.
Alle Griechen sind Menschen.
Daher sind alle Griechen sterblich.

Man braucht nicht zu wissen, was „Menschen“, „sterblich“ und „Griechen“ bedeutet, um die Korrektheit deser Schlußfolgerung einzusehen. Man kann stattdessen Symbole verwenden:

Alle B sind C.
Alle A sind B.
Daher sind alle A auch C.

Das erinnert an Algebra (wo die Symbole viel später eingerführt wurden). Man kann die Schlußfolgerung mechanisch vornehmen, ohne die Bedeutung der Symbole zu verstehen.

Schon Gottfried Wilhelm Leibniz (1646–1716) hatte die Idee, logisches Schließen (in verschiedenen Bereichen, z.B. in der Justiz) auf *Berechnungen* zu reduzieren:

> Das einzige Mittel, unsere Schlußfolgerungen zu verbessern, ist, sie ebenso anschaulich zu machen, wie es die der Mathematiker sind, derart, daß man seinen Irrtum mit den Augen findet und, wenn es Streitigkeiten unter den Leuten gibt, man nur zu sagen braucht: „Rechnen wir!“ ohne eine weitere Förmlichkeit, um zu sehen, wer recht hat.

Laut Wikipedia befasste sich Leibniz intensiv mit Logik und propagierte erstmals eine symbolische Logik in Kalkülform. Seine Logikkalkül-Skizzen veröffentlichte er allerdings nicht; erst sehr verspätet (1840, 1890, 1903) wurden sie publiziert. Seine charakteristischen Zahlen aus dem Jahr 1679 sind ein arithmetisches Modell der Logik des Aristoteles. Seinen Hauptkalkül entwickelte er in den Generales Inquisitiones von 1686. Er entwarf dort die erste Gleichungslogik und leitete in ihr fast zwei Jahrhunderte vor der Boole-Schule die Gesetze der booleschen Verbandsordnung ab.

In ihrer heutigen Form geht die Aussagenlogik auf George Boole (1815–1864) zurück.

2.2 Alphabet und Syntax der Aussagenlogik

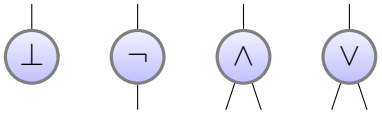
2.2.1 Definition. Das **Alphabet der Aussagenlogik** besteht aus

(a) einer abzählbaren Menge[1] $\mathcal{P}$ **atomarer Aussagen** $A_0, A_1, A_2, \ldots$ auch **Variablen** oder **Propositionen** genannt; für die Baumdarstellung handelt es sich um *Blätter*:

(b) Operatoren vorgegebener Stelligkeit[2] auf der Menge der atomaren Aussagen, den sogenanten **Junktoren**

- $\bot$ **Absurdität**, „bottom“, 0-stellig
- $\neg$ **Negation**, „nicht“, 1-stellig
- $\wedge$ **Konjunktion**, „und“, 2-stellig
- $\vee$ **Disjunktion**, „oder“, 2-stellig

oder für die Baumdarstellung



Vereinfachend schreiben wir oft $A,\ B,\ C, \ldots$ statt $A_0,\ A_1,\ A_2, \ldots$ um atomare Aussagen zu bezeichnen.

2.2.2 Bemerkung. Die Namen und Stelligkeiten der Junktoren sind natürlich im Hinblick auf die intendierte Semantik gewählt, die im folgenden Abschnitt beschrieben wird. Werden die Junktoren nach Stelligkeit sortiert, so bezeichnet man die resultierende Folge Σ von Mengen, oben etwa $\Sigma_0 = \{\bot\}$, $\Sigma_1 = \{\neg\}$, $\Sigma_2 = \{\wedge, \vee\}$ und $\Sigma_n = \varnothing$ für $n > 2$, auch als *Signatur*.

Falls jemand andere ihr schon bekannte Junktoren vermißt, diese werden in Kürze eingeführt.

2.2.3 Definition. [Sprache der Aussagenlogik (Syntax), Baum-Version]
Die Menge $\mathcal{F}$ der **Formeln** der Aussagenlogik (auch **Aussagen** genannt) ist die kleinste Menge von geordneten Bäumen, so dass

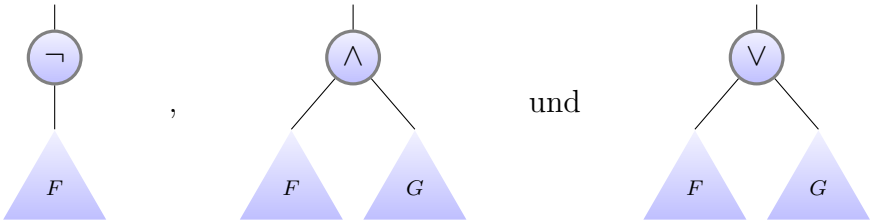
(a) alle Blätter

Formeln sind, d.h., $\mathcal{P} \cup \{\bot\} \subseteq \mathcal{F}$;

[1] siehe: Mathematische Grundagen im Anhang.
[2] Unter der *Stelligkeit* einer Funktion bzw. eines Operators versteht man die Anzahl der Argumente.

(b) falls

Formeln sind, dann gilt dies auch für



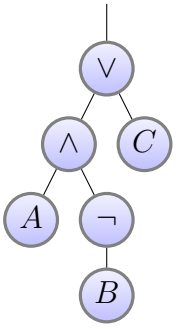
Entsprechend bezeichnet $\mathcal{F}(\mathcal{M})$ die kleinste Menge der aus $\mathcal{M} \subseteq \mathcal{P}$ mittels $\neg$, $\wedge$ sowie $\vee$ aufbaubaren Formeln.

Die sog. **Meta-Variablen** F, G, $H\ldots$ bezeichnen generische Formeln.

Es handelt sich bei dieser Definition um ein Beispiel für die oben erwähnte strukturelle Rekursion. Hier spezifizieren die Bedingungen (a) und (b) sogenannte *Abschlußeigenschaften* der Menge $\mathcal{F}$. Das Besondere an der Menge der Formeln ist ihre diesbezügliche *Minimalität*: es gibt keine anderen Formeln als die durch (a) und (b) spezifizierten.

2.2.4 Bemerkung. Aus mathematischer Sicht ist die Menge der Formeln die *freie* Σ-*Algebra* über der Menge der atomaren Aussagen. Dabei verweist der Begriff „frei“ darauf, dass zwei Formeln nur dann gleich sind, wenn sie syntaktisch, d.h., als Zeichenreihen, übereinstimmen. Dieser Gleichheitsbegriff wird sich für die Praxis als zu streng erweisen. Wir werden später mehr daran interessiert sein, wann Formeln “semantisch gleich” sind, d,.h., dasselbe bedeuten, vergl. Abschnitt 2.8.

2.2.5 Bemerkung. So schön und konzeptionell nützlich die Definition von Formeln als Bäume auch ist, zur praktischen Arbeit (und aus typographischen Gründen) brauchen wir eine *lineare* oder *1-dimensionale* Darstellung von Formeln. Dafür gibt es verschiedene Möglichkeiten, je nachdem, ob man für die binären Junktoren die klassische *Infix*-Notation verwendet, die *Präfix*- oder *polnische* Notation, oder besser noch noch die *Postfix*- oder *umgekehrt polnische* Notation (kurz *RPN*), wie folgendes Beispiel zeigt:



- Infix-Schreibweise (erfordert Klammern):

  $((A \wedge (\neg B)) \vee C)$ oder kürzer $(A \wedge \neg B) \vee C$

- Präfix-Schreibweise (polnische Notation):

  $$\vee \wedge A \neg B C$$

- Postfix-Schreibweise, oder RPN:

  $$A B \neg \wedge C \vee$$

Die Infix-Notation für $\wedge$ und $\vee$ erzwingt die Verwendung von Klammern, ähnlich wie in der Arithmetik. Wenn man Klammern einsparen möchte, kann man sich mit Klammerersparnis-Reglen und sogenannten *Bindungskonventionen* behelfen, siehe Abschnitt 2.5. Auch bestimmte Eigenschaften der Konjunktion und Disjunktion erlauben es, das Klammerdickicht etwas zu lichten. Die äußeren Klammern dienen nur dazu, jede Formel sofort als Baustein für weitere Formeln verwenden zu können.

Präfix- und Postfix-Notation kommen wie die Bäume selbst ohne Klammern aus. Erstere wurde 1924 vom polnischen Logiker Jan Łukasiewicz eingeführt, wird aber heute nicht mehr häufig verwendet, letztere wurde Mitte der 1950'er Jahre von Burks, Warren und Wright [BWW54] sowie vom australischen Philosophen und Informatiker Charles Hamblin vorgeschlagen und weiterentwickelt [Ham57]. In diesem Zusammenhang dürfen auch die Miterfinder des Kellerprinzips (oder Stacks), Friedrich Ludwig Bauer und Klaus Samelson nicht unerwähnt bleiben [BS57]. RPN findet sich beispielsweise in wissenschaftlichen Taschenrechnern von Hewlett Packard, in der Programmiersprache Forth und in der Seitenbeschreibungssprache PostScript.

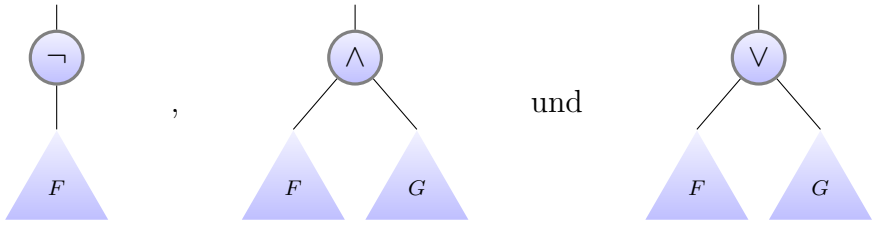
Trotz ihrer Nachteile werden wir in dieser VL weitgehend die Infix-Notation verwenden. Diese formalisieren wir nun:

2.2.6 Definition. Die Infix-Darstellung $\iota(F)$ einer Formel F ist rekursiv definiert:

(a) Alle Blätter

haben als Infix-Darstellung A_i bzw. $\bot$.

(b) Zusammengesetzte Formeln



haben die Infix-Darstellung

$$(\neg\iota(F)) \quad , \quad (\iota(F) \wedge \iota(G)) \quad \text{sowie} \quad (\iota(F) \vee \iota(G))$$

In der Praxis werden wir den Operator ι und die zugehörigen Klammern meistens weglassen, da aus dem Kontext klar sein dürfte, was gemeint ist.

Manche Autoren beginnen direkt mit den Infix-Ausdrücken als Formeln und führen später sog. Syntaxbäume ein. Das scheint uns die Unterscheidung zwischen primärem mathemaitschen Objekt und einer Darstellung desselben auf den Kopf zu stellen.

2.2.7 Beispiel. Für atomare Aussagen A und B sind etwa $((A \vee B) \wedge (\neg B)))$ und $(\neg(\neg(A \vee B)))$ Infix-Darstellungen von Formeln, deren RPN-Darstellung etwas gewöhnungsbedürftig aussieht: $AB \vee B\neg\wedge$ bzw. $AB \vee \neg\neg$.

Wir kommen nun zum Betriff der *Größe* einer Formel, was natürlich der Größe von Bäumen entspricht, und sich von der Länge ihrer (vollständig geklammerten) Infix-Darstellung unterscheiden kann.

2.2.8 Definition. Die **Größe** $|F|$ einer Formel F wie auch die **Länge** $\|F\|$ ihrer Infx-Darstellung definieren wir mit Hilfe struktureller Induktion, also über den Aufbau von F wie folgt:

▷ Falls F mit $\bot$ übereinstimmt oder atomar ist, gelte

$$|F| := 1 =: \|F\|$$

▷ Falls F die Form $(\neg G)$ hat und $|G|$, bzw. $\|G\|$, bekannt ist, setze

$$|(\neg G)| := |G| + 1 \qquad \text{bzw.} \qquad \|(\neg G)\| := \|G\| + 3$$

▷ Falls F die Form $(G \wedge H)$ oder $(G \vee H)$ hat und $|G|$ sowie $|H|$, bzw. $\|G\|$ sowie $\|H\|$, bekannt sind, setze

$$|(G \wedge H)| := |G| + |H| + 1 \qquad \text{bzw.} \qquad \|(G \wedge H)\| := \|G\| + \|H\| + 3$$

und analog für die Disjunktion.

Damit ist die Größe einer Formel nichts weiter als die Anzahl ihrer Baumknoten.

2.3 Semantik der Aussagenlogik

Die Semantik der Aussagenlogik untersucht, wie aus Belegungen der atomaren Aussagen mit Wahrheitswerten diejenigen aller korrekten Formeln mechanisch abgeleitet werden können. Interpretiert man 1 als „wahr“ und 0 als „falsch“, so geht es darum, beliebige Funktionen von der Menge aller atomaren Aussagen in die Menge $2 = \{0, 1\}$ auf die Menge aller korrekten Formeln sinnvoll fortzusetzen.

Gegeben sei eine Formel F. Ihre atomare Teilaussagen liegen in einer endlichen Untermenge $\mathcal{M} \subseteq \mathcal{P}$ aller atomaren Aussagen. Falls wir die Wahrheitswerte der Elemente von $\mathcal{M}$ kennen, soll der Wahrheitswert der ganzen Formel F eindeutig bestimmt werden können. Während für einzelne Formeln die Beschränkung auf endliche Teilmengen der Variablenmenge sinnvoll erscheint, sind später, beim Kompaktheitssatz, nicht notwendig endliche Mengen von Formeln zu betrachten, in denen auch unendlich viele Variablen auftreten können. Daher:

2.3.1 Definition. Eine **Belegung** ist eine Abbildung α von einer (nicht notwendig endlichen) Menge $\mathcal{M}$ atomarer Aussagen in die Menge $\{0, 1\}$ der Wahrheitswerte. Kurz:

$$\alpha : \mathcal{M} \longrightarrow \{0, 1\} \quad \text{für} \quad \mathcal{M} \subseteq \mathcal{P}\ ^{3}$$

[3] Man kann dies auch als *partielle Abbildung* $\mathcal{P} \xrightarrow{\alpha} \{0, 1\}$ interpretieren.

2.3.2 Beispiel. Falls $\mathcal{M} = \{A, B\}$ mit der Belegung $\alpha(A) = 1$ und $\alpha(B) = 0$ ist, legt die intuitive Bedeutung der Wörter „oder“ und „nicht“ nahe, dass $(A \vee B)$ mit 1 und $(\neg(A \vee B))$ mit 0 bewertet werden sollten. Dies gilt es nun zu formalisieren.

2.3.3 Definition. Für jede Menge $\mathcal{M} \subseteq \mathcal{P}$, jede Formel $F \in \mathcal{F}(\mathcal{M})$ und jede Belegung $\alpha : \mathcal{M} \longrightarrow \{0, 1\}$ definieren wir den **Wert** $\widehat{\alpha}(F)$ der Formel F **unter der Belegung** α mittels struktureller Rekursion, d.h., Rekursion über den Aufbau der Formel F:

$|F| = 1$: Entweder stimmt F mit der Absurdität überein, dann setzen wir

$$\widehat{\alpha}(\bot) := 0$$

Oder F ist eine atomare Formel aus $\mathcal{M}$ und wir setzten

$$\widehat{\alpha}(F) := \alpha(F)$$

$|F| > 1$: Hat F eine Größe > 1, so hat F die Form $(\neg G)$ oder $(G \wedge H)$ oder $(G \vee H)$, wobei die Teilformeln G und H echt kleiner sind. Für sie ist $\widehat{\alpha}$ durch die Rekursion also bereits definiert. Insofern definieren wir $\widehat{\alpha}(F)$ durch

$$\widehat{\alpha}(\neg G) := \begin{cases} 1 & \text{falls } \widehat{\alpha}(G) = 0 \\ 0 & \text{sonst} \end{cases}$$

$$\widehat{\alpha}(G \wedge H) := \begin{cases} 1 & \text{falls } \widehat{\alpha}(G) = 1 \text{ und } \widehat{\alpha}(H) = 1 \\ 0 & \text{sonst} \end{cases}$$

$$\widehat{\alpha}(G \vee H) := \begin{cases} 1 & \text{falls } \widehat{\alpha}(G) = 1 \text{ oder } \widehat{\alpha}(H) = 1 \\ 0 & \text{sonst} \end{cases}$$

2.3.4 Bemerkung. Warum wird $\widehat{\alpha}$ gerade so definiert? Und wie verhält es sich mit der Anmerkung auf Seite 5, dass die Menge der semantischen Werte, hier $2 = \{0, 1\}$, eine innere Struktur benötigt, die das Nachspielen der logischen Junktoren erlaubt?

▷ Die Negation soll offenbar die Wahrheitswerte in ihr Komplement umwandeln, was ausgedrückt werden kann durch

$$\widehat{\alpha}(\neg G) = 1 - \widehat{\alpha}(G)$$

▷ Damit eine Konjunktion wahr ist, darf keine der Teilformeln einen geringeren Wahrheitswert als 1 annehmen, anders ausgedrückt

$$\widehat{\alpha}(G \wedge H) = \inf\{\widehat{\alpha}(G), \widehat{\alpha}(H)\}$$

Die *Infimum-Operation* operiert eigentlich auf Teilmengen geordneter Mengen und weist diesen ihre *größte untere Schranke* zu, falls eine solche existiert. Solange man nur in höchstens 2-elementigen Mengen interessiert ist, etwa auf $2 = \{0, 1\}$, wird speziell in Infix-Schreibweise auch gern die Bezeichnung $\wedge$ verwendet. Dann können wir schreiben

$$\widehat{\alpha}(G \wedge H) = \widehat{\alpha}(G) \wedge \widehat{\alpha}(H)$$

was eine Interpretation der Konjunktion als verallgemeinerter Infimum-Operation nahelegt und $\widehat{\alpha}$ als Homomorphismus erscheinen läßt. [4]

[4] Die Menge der Wahrheitswerte muß nicht á priori linear geordnet sein. In solchen Fällen, kann für zwei Elemente, wie auch für unendliche Teilmengen, ein Infimum existieren, ohne daß ein kleinstes Element, d.h., ein Minimum, existiert. Daher bevorzugen wir es, mit dem Infimum zu arbeiten.

▷ Bei der Interpretation von „oder“ gibt es unterschiedliche Ansätze. In der Mathematik und in der theoretischen Informatik meint „oder“ immer die nicht-exklusive Form, die mit $\vee$ bezeichnet wird. Nach dieser Präzisierung ist klar, dass eine Disjunktion schon wahr ist, wenn mindestens eine der Teilformeln wahr ist, anders ausgedrückt

$$\widehat{\alpha}(G \vee H) = \sup\{\widehat{\alpha}(G), \widehat{\alpha}(H)\} = \widehat{\alpha}(G) \vee \widehat{\alpha}(H)$$

Diese Formeln legen es nahe, die Junktoren $\wedge$ und $\vee$ in einem noch zu präzisierenden Sinn als „dual“ zu verstehen, vergleiche Abschnitt 3.2.

Das „exklusive Oder“, häufig mit $\oplus$ bezeichnet, läßt sich auch modellieren, selbst wenn wir es nicht als Teil der Aussagenlogik eingeführt haben, vergleiche Abschnitt 2.4.

Anstelle der obigen Fallunterscheidungen oder Formeln kann man die Wahrheitswerte zusammengesetzter Formel auch in sog. Wahrheitstabellen auflisten:

$\neg$:

A	$\neg A$
0	1
1	0

$\wedge$:

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

$\vee$:

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

(2.1)

2.3.5 Beispiel. Die Semantik der Formel

$$(A \vee (B \wedge (\neg C)))$$

ist bei jeder Belegung α der Atome A, B, C definiert. Die Formel ist wahr, falls A wahr ist oder $(B \wedge (\neg C))$ wahr ist. Letzteres erfordert sowohl die Wahrheit von B als auch die von $\neg C$. Man überzeuge sich anhand einer Wahrheitstabelle davon, dass dies die einzigen Möglichkeiten sind, der zusammengesetzten Formel den Wert wahr zuzuweisen.

2.3.6 Definition. Eine Belegung $\alpha : \mathcal{M} \longrightarrow \{0, 1\}$ heißt **passend** für eine Formel F falls all deren atomaren Aussagen in $\mathcal{M}$ liegen, d.h., falls $F \in \mathcal{F}(\mathcal{M})$.

Jeder Formel werden genau durch die passenden Belegungen Werte 0 oder 1 zugeordnet, andere Belegungen sind dafür uninteressant.

2.4 Abgeleitete Symbole

Andere Autoren zählen die zwei-stelligen Junktoren $\Rightarrow$ (Implikation) und $\Leftrightarrow$ (Äquivalenz) sowie den null-stelligen Junktor $\top$ (Tautologie) zu den Grundbausteinen der Aussagenlogik. Wir hingegen werden sie, ebenso wie das exklusive Oder $\oplus$, als abgeleitete Symbole verwenden, definiert als Abkürzungen längerer Formeln, die nur $\neg$, $\wedge$ oder $\vee$ verwenden. Dass dies möglich ist, ergibt sich aus der Analyse der *intendierten klassischen Semantik* für die um diese neuen Junktoren erweiterte Syntax:

▷ Die Tautologie $\top$ als Komplement der Absurdität $\bot$ ist unproblematisch.

▷ Die Implikation $(G \Rightarrow H)$ soll als Abstraktion von „wenn G, dann H“ verstanden werden. Das bedeutet rein formal, dass der Wahrheitswert von G kleiner oder gleich dem Wahrheitswert von H sein soll. Achtung: inhaltlich braucht kein Zusammenhang zwischen den Aussagen G und H zu bestehen, was die Akzeptanz der obigen Interpretation erschweren mag. Wem dies nicht gefällt, muß eine bessere Semantik suchen, Kandidaten dafür gibt es.

▷ Die Äquivalenz $(G \Leftrightarrow H)$ soll „G genau dann wenn H“ ausdrücken und kann als Konjunktion zweier entgegengesetzter Implikationen $(G \Rightarrow H)$ und $(H \Rightarrow G)$ aufgefasst werden. Mit anderen Worten, die Wahrheitswerte von G und H stimmen überein.

▷ Schließlich soll $(G \oplus H)$ genau dann wahr sein, wenn genau eine der Teilformeln wahr ist, mit anderen Worten, wenn beide Teilformeln verschiedene Wahrheitswerte haben.

Diese Vorgaben lassen sich wie folgt formalisieren:

2.4.1 Definition. Für jede, bzw. jede für G und H passende Belegung setzen wir

$$\begin{aligned}
\widehat{\alpha}(\top) &:= 1 \\
\widehat{\alpha}(G \Rightarrow H) &:= \begin{cases} 1 & \text{falls } \widehat{\alpha}(G) \leq \widehat{\alpha}(H) \\ 0 & \text{sonst} \end{cases} \\
\widehat{\alpha}(G \Leftrightarrow H) &:= \begin{cases} 1 & \text{falls } \widehat{\alpha}(G) = \widehat{\alpha}(H) \\ 0 & \text{sonst} \end{cases} \\
\widehat{\alpha}(G \oplus H) &:= \begin{cases} 1 & \text{falls } \widehat{\alpha}(G) \neq \widehat{\alpha}(H) \\ 0 & \text{sonst} \end{cases}
\end{aligned}$$

Wenn es gelingt, die spezifizierten Relationen zwischen den Wahrheitswerten in eine oder mehrere Gleichungen umzusetzen, die evtl. die Konstante 1 verwenden, können wir die neu eingeführten Junktoren durch die ursprünglichen ausdrücken.

2.4.2 Proposition. *In der Aussagenlogik mit um $\top$, $\Rightarrow$, $\Leftrightarrow$ sowie $\oplus$ erweiterter Syntax sind folgenden Formeln immer semantisch gleichwertig, d.h., haben bzgl. jeder passenden Belegung denselben Wahrheitswert:*

(a) $\top$ *bsw.* $\neg\bot$*;*

(b) $(G \Rightarrow H)$ *bzw.* $((\neg G) \vee H)$*;*

(c) $(G \Leftrightarrow H)$ *bzw.* $(G \Rightarrow H) \wedge (H \Rightarrow G))$ *bzw.* $((G \wedge H) \vee ((\neg G) \wedge (\neg H)))$*;*

(d) $(G \oplus H)$ *bzw.* $(\neg(G \Leftrightarrow H))$ *bzw.* $((G \wedge (\neg H)) \vee ((\neg G) \wedge H))$.

Folglich können wir die Junktoren $\top$, $\Rightarrow$, $\Leftrightarrow$ und $\oplus$ ohne Einbuße an Ausdrucksfähigkeit wieder aus der Syntax der Aussagenlogik entfernen.

Beweis. Im Folgenden ist α eine beliebige passende Belegung.

(a) Klar wegen $1 = 1 - 0$.

(b)

$$\begin{aligned}\widehat{\alpha}(G) \leq \widehat{\alpha}(H) \quad &\text{gdw} \quad \widehat{\alpha}(G) = 0 \quad \text{oder} \quad \widehat{\alpha}(H) = 1\\ &\text{gdw} \quad \widehat{\alpha}(\neg G) = 1 \quad \text{oder} \quad \widehat{\alpha}(H) = 1\\ &\text{gdw} \quad \sup\{\widehat{\alpha}(\neg G), \widehat{\alpha}(H)\} = 1\\ &\text{gdw} \quad \widehat{\alpha}(\neg G \vee H) = 1\end{aligned}$$

(c) Die erste Variante ergibt sich aus

$$\widehat{\alpha}(G) = \widehat{\alpha}(H) \quad \text{gdw} \quad \widehat{\alpha}(G) \leq \widehat{\alpha}(H) \quad \text{und} \quad \widehat{\alpha}(H) \leq \widehat{\alpha}(G)$$

was sich gemäß (b) weiter umformen läßt zu

$$\begin{aligned}&\text{gdw} \quad \widehat{\alpha}((\neg G) \vee H) = 1 \quad \text{und} \quad \widehat{\alpha}((\neg H) \vee G) = 1\\ &\text{gdw} \quad \widehat{\alpha}(((\neg G) \vee H) \wedge ((\neg H) \vee G)) = 1\end{aligned}$$

Alternativ erhalten wir

$$\begin{aligned}\widehat{\alpha}(G) = \widehat{\alpha}(H) \quad &\text{gdw} \quad \widehat{\alpha}(G) = 1 = \widehat{\alpha}(H) \quad \text{oder} \quad \widehat{\alpha}(G) = 0 = \widehat{\alpha}(H)\\ &\text{gdw} \quad \widehat{\alpha}((G \wedge H) \vee ((\neg G) \wedge (\neg H)))\end{aligned}$$

(d) Die erste Variante ist unmittelbar klar. Alternativ erhalten wir

$$\begin{aligned}\widehat{\alpha}(G) \neq \widehat{\alpha}(H) \quad &\text{gdw} \quad \widehat{\alpha}(G) = 1 = \widehat{\alpha}(\neg H) \quad \text{oder} \quad \widehat{\alpha}(G) = 0 = \widehat{\alpha}(\neg H)\\ &\text{gdw} \quad \widehat{\alpha}(G) = 1 = \widehat{\alpha}(\neg H) \quad \text{oder} \quad \widehat{\alpha}(\neg G) = 1 = \widehat{\alpha}(H)\\ &\text{gdw} \quad \widehat{\alpha}((G \wedge (\neg H)) \vee ((\neg G) \wedge H))\end{aligned}$$

□

In Abschnitt 2.8 wird sich die semantische Gleichwertigkeit als nützlichere Relation auf $\mathcal{F}$ erweisen als die syntaktischen Gleichheit zwischen Formeln.

2.5 Bindungskonventionen

Wie gerade gesehen, können selbst Formeln recht simpler Semantik schnell eine komplizierte und unübersichtliche Infix-Notation liefern. Um unnötige Klammern zu vermeiden, d.h., zur Vereinfachung der Infix-Notation, vereinbaren wir die folgende

2.5.1 Notationelle Konvention.

- Die äußeren Klammern um zusammengesetzte Aussagen können weggelassen werden
- $\neg$ bindet stärker als alle zwei-stelligen Junktoren ($\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\oplus$).
- $\wedge$ und $\vee$, die „alten“ zwei-stelligen Junktoren, binden gleich stark, aber stärker als die abgeleiteten Junktoren $\Rightarrow$, $\Leftrightarrow$ und $\oplus$.

2.5.2 Beispiel. $\neg A \Rightarrow B \vee \neg C$ wird dadurch eine korrekte Formel. Es ist die Abkürzung für die „offizielle“ Langfassung $\big((\neg A) \Rightarrow (B \vee (\neg C))\big)$. Auch die durch $G \oplus H$ abgekürzte Formel vereinfacht sich zu $\neg((H \vee \neg G) \wedge (G \vee \neg H))$.

Dennoch können nicht alle Klammern entfernt werden. Ausdrücke wie $A \vee B \wedge C$ oder $A \wedge B \wedge C$ oder auch $A \Rightarrow B \Rightarrow C$ lassen sich nicht aufgrund der obigen Bindungskonventionen aus einer Langfassung herleiten. Mit Hilfe späterer Ergebnisse kann man $A \wedge B \wedge C$ interpretieren, während $A \Rightarrow B \Rightarrow C$ eine zusätzliche Konvention erfordert. Aber $A \vee B \wedge C$ wird sich als uninterpretierbar herausstellen.

2.6 Wahrheitstafeln

Die Semantik jeder Formel $F \in \mathcal{F}(\{A_0, A_1, \ldots, A_{n-1}\})$ kann man als **Wahrheitstafel** darstellen: jede Zeile entspricht einer der 2^n Belegungen $\alpha : \mathcal{M} \longrightarrow \{0, 1\}$. Diese sind gemäß der Binärdarstellung der Zahlen von 0 bis $2^n - 1$ angeordnet:

A_0	A_1	$\ldots$	A_{n-1}	F
0	0	$\ldots$	0	$\widehat{\alpha}(F)$ für $\alpha(A_i) = 0$ für alle $i < n$
0	0	$\ldots$	1	$\widehat{\alpha}(F)$ für $\alpha(A_n) = 1$, sonstige $\alpha(A_i) = 0$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
1	1	$\ldots$	1	$\widehat{\alpha}(F)$ für $\alpha(A_i) = 1$ für alle $i < n$

Letztendlich lassen sich alle Wahrheitstafeln auf diejenigen für Negation, Konjunktion und Disjunktion zurückführen, die wir schon in Abschnitt 2.3 eingeführt hatten, vergl. Tabellen 2.1. Das läßt sich z.B. mit Hilfe der sog. „kanonischen konjunktiven Normalform“ zeigen, vergl. Kapitel 4 und HA.

Der Übersichtlichkeit halber und zu Vergleichszwecken ist es auch zulässig, Hilfsspalten mit Teilergebnissen einzufügen. Diese können auch semantischen Spezifikationen umsetzen. Für die Implikation und die Äquivalenz erhalten wir nun

	A	B	$\neg A$	$\neg A \vee B$	$\alpha(A) \leq \alpha(B)$
	0	0	1	1	1
$\Rightarrow$:	0	1	1	1	1
	1	0	0	0	0
	1	1	0	1	1

	A	B	$A \Rightarrow B$	$B \Rightarrow A$	$(A \Rightarrow B) \wedge (B \Rightarrow A)$	$\alpha(A) = \alpha(B)$
	0	0	1	1	1	1
$\Leftrightarrow$:	0	1	1	0	0	0
	1	0	0	1	0	0
	1	1	1	1	1	1

während das exklusive Oder folgende Tabelle hat:

	A	B	$\neg(A \Leftrightarrow B)$	$\alpha(A) \neq \alpha(B)$
	0	0	0	0
$\oplus$:	0	1	1	1
	1	0	1	1
	1	1	0	0

2.6.1 Beispiel. Wahrheitstafeln ermöglichen eine Analyse der jeweils dargestellten Formel, etwa von $F := C \Rightarrow A \vee (B \wedge A)$:

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Gemäß der Tafel ist die Formel fast immer wahr, außer in zwei Fällen:

- $\alpha(A) = 0 = \alpha(B)$ und $\alpha(C) = 1$
- $\alpha(A) = 0$ und $\alpha(B) = 1 = \alpha(C)$

Daraus ist zu erkennen, dass der Wert von F *unabhängig* von dem Wert von B, also von $\alpha(B)$, ist. Wir können also die B-Spalte aus der Tabelle entfernen:

A	C	F
0	0	1
0	1	0
1	0	1
1	1	1

Damit ist $\widehat{\alpha}(F) = 1$, falls $\alpha(A) = 1$ oder $\alpha(C) = 0$ gilt. Damit können wir kürzere Formeln mit dieselbe Semantik angeben

$$G := A \vee \neg C \qquad \text{oder noch kürzer} \qquad H := C \Rightarrow A$$

2.7 Erfüllbarkeit und Tautologien

Grundsätzlich kann man Formeln danach klassifizieren, ob überhaupt passende Belegungen existieren, die sie wahr bzw. falsch machen.

2.7.1 Definition. Eine Formel F heißt

(a) **erfüllbar**, falls sie unter wenigstens einer Belegung wahr ist;

(b) **Tautologie**, falls sie für jede passende Belegung wahr ist (in Anlehnung an den 0-stelligen Junktor $\top$).

Hinsichtlich der Wahrheitstabelle bedeutet dies, dass die rechte Spalte mindestens eine Eins enthält, bzw. nur aus Einsen besteht. Offenbar enthält die Wahrheitstabelle der Negation jeder Tautologie nur Nullen in der rechten Spalte, folglich sind Negationen von Tautologien nicht erfüllbar. Das läßt sich auch positiv formulieren:

2.7.2 Satz. *Eine Formel F ist genau dann erfüllbar, wenn $\neg F$ keine Tautologie ist.*

Beweis. F ist genau dann erfüllbar, wenn es eine passende Belegung α mit $\widehat{\alpha}(F) = 1$, oder äquivalent $\widehat{\alpha}(\neg F) = 1 - \widehat{\alpha}(F) = 0$ gibt. Aber genau dies charakterisiert $\neg F$ als keine Tautologie. □

2.7.3 Beispiel. Die Formel $A \Rightarrow B$ ist erfüllbar, da sie etwa dann wahr wird, wenn man A mit 0 oder B mit 1 belegt. Das gilt speziell im Fall $B = \neg A$, was zunächst irritierend sein mag, aber der sehr simplen Semantik der Aussagenlogik geschuldet ist, die nur die Wahrheitswerte der Teilformeln betrachtet.

2.7.4 Beispiel. Die Formel $F \vee \neg F$ ist eine Tautologie, denn F und $\neg F$ nehmen aufgrund der Semantik von $\neg$ immer verschiedene Werte an. Damit tritt immer genau einmal der Wert 1 auf, so dass aufgrund der Semantik von $\vee$ als Supremum auch hier der Wert 1 angenommen wird.

Auch $(\neg F \Rightarrow \neg G) \Rightarrow (G \Rightarrow F)$ ist eine Tautologie, wie man anhand der Wahrheitstabelle unschwer feststellt. Sie bildet die Grundlage für das Beweisprinzip der *Contraposition*. Wenn einem der Berweis von $F \Rightarrow G$ nicht gleich gelingt, kann man sich stattdessen an $\neg G \Rightarrow \neg F$ versuchen. Im Erfolgsfall folgt daraus die ursprüngliche Behauptung (zumindest in der klassischen Aussagenlogik).

2.7.5 Beispiel. Eine evtl. überraschende Tautologie ist

$$(F \Rightarrow G) \vee (G \Rightarrow F)$$

In der Tat, bei jeder passenden Belegung α gilt entweder $\widehat{\alpha}(F) = 0$, woraus die Wahrheit von $F \Rightarrow G$ folgt, oder $\widehat{\alpha}(F) = 1$, woraus die Wahrheit von $G \Rightarrow F$ folgt. Und für die Wahrheit einer Disjunktion genügt die Wahrheit einer Teilformel.

Die obige Formel besagt nur, dass der Wertebereich $2 = \{0, 1\}$ der Semantik der Aussagenlogik total geordnet ist, also

$$0 \leq 0 < 1 \leq 1$$

Damit ergibt sich für die beiden Wahrheitswerte $\widehat{\alpha}(F)$ und $\widehat{\alpha}(G)$ trivialerweise (hier ist das Wort tatsächlich mal erlaubt)

$$\widehat{\alpha}(F) \leq \widehat{\alpha}(G) \qquad \text{oder} \qquad \widehat{\alpha}(G) \leq \widehat{\alpha}(F)$$

Es wäre irreführend, die obige Aussage dahingehend zu interpretieren, dass für zwei beliebige Aussagen F und G immer mindestens eine aus der anderen *hergeleitet werden kann*. Das setzt eine andere Semantik voraus, die auf Beweisbarkeit statt auf Wahrheit basiert. Dort kann die obige Formel auch formuliert werden, ist aber keine Tautologie mehr. Das unterstreicht, dass der Begriff „Tautologie“ nur relativ zu einer gegebenen Semantik Sinn ergeben kann.

Im „realen Leben“, etwa in der Bahnindustrie, werden zur Steuerung von Weichenanlagen Formeln mit bis zu 300.000 atomaren Aussagen formuliert und deren Erfüllbarkeit getestet. Dieses Problem wird in der Theoretischen Informatik intensiv erforscht und heißt dort **SAT** (von englisch „satisfiable“). Im übernächsten Kapitel entwickeln wir einen Algorithmus dafür.

2.8 Äquivalenz von Formeln

Nachdem wir schon in Proposition 2.4.2 das Konzept der semantischen Gleichwertigkeit verwendet hatten, um die Einführung von abgeleiteten Symbolen zur Abkürzung komplexer Formeln als unbedenklich nachzuweisen, wollen wir es nun auch auf Formeln in der Originalsyntax anwenden. Zunächst brauchen wir aber einen einprägsameren bzw. kürzeren Namen:

2.8.1 Definition. Zwei Formeln F und G der Aussagenlogik heißen **äquivalent**, wenn sie semantisch gleichwertig sind, d.h. wenn für jede zu F und G passende Belegung $\alpha : \mathcal{M} \longrightarrow \{0,1\}$ gilt

$$\widehat{\alpha}(F) = \widehat{\alpha}(G)$$

Wir führen dafür die Notation $F \equiv G$ ein.

Beachte: $\equiv$ ist *kein* Junktor und gehört somit nicht zum Alphabet der Aussagenlogik. Folglich ist $F \equiv G$ auch *keine* Formel der Aussagenlogik, sondern beide Ausdrücke gehören zur sog. *Metasprache*, mit der wir *über* die Aussagenlogik sprechen.

2.8.2 Beispiel. $C \Rightarrow A \vee (B \wedge A)$ ist äquivalent zu $A \vee \neg C$. Dies haben wir in Beispiel 2.6.1 durch Analyse der Wahrheitstafel festgestellt.

Der Begriff „äquivalent“ läßt sich in zweifacher Hinsicht rechtfertigen:

2.8.3 Satz. *Zwei aussagenlogische Formeln F und G sind genau dann äquivalent, wenn $F \Leftrightarrow G$ eine Tautologie ist.*

Beweis. Dies ist nur eine Umformulierung der semantischen Interpretation von $F \Leftrightarrow G$: die Wahrheitswerte von F und G stimmen für jede passende Belegung überein. □

Dieses Ergebnis rechtfertigt es, die Relation $\equiv$ auf Formeln als „Externalisierung“ des Junktors $\Leftrightarrow$ aufzufassen; diesen kann man bei Bedarf präziser mit „interner Äquivalenz“ bezeichnen.

2.8.4 Satz. *Die Relation $\equiv$ ist eine Äquivalenzrelation auf der Menge $\mathcal{F}$ aller aussagenlogischen Formeln, d.h., sie ist reflexiv, transitiv und symmetrisch.*

Beweis. Alle drei Eigenschaften lassen sich auf die entsprechenden Eigenschafen der Gleichheit „=“ im semantischen Wertebereich $2 = \{0,1\}$ zurückführen, die von allen passenden Belegungen „reflektiert“ werden. □

Es gilt sogar noch mehr: $\equiv$ ist zudem mit den Junktoren „verträglich“, was $\equiv$ zu einer sogenannten *Kongruenzrelation* macht. D.h., (Teil-)Formeln dürfen bedenkenlos durch äquivalente (Teil-)Formeln ersetzt werden, ohne dass sich die Semantik ändert.

2.8.5 Satz. *Falls $F \equiv F'$, dann gilt*

$$\neg F \equiv \neg F' \quad \textit{und} \quad F \vee G \equiv F' \vee G \quad \textit{sowie} \quad F \wedge G \equiv F' \wedge G \quad \textit{für alle Formeln } G$$

Beweis. Ist α eine für F und F' passende Belegung, dann gilt nach Voraussetzung $\widehat{\alpha}(F) = \widehat{\alpha}(F')$, und folglich

$$\widehat{\alpha}(\neg F) = 1 - \widehat{\alpha}(F) = 1 - \widehat{\alpha}(F') = \widehat{\alpha}(\neg F')$$

Falls α zudem für G passend ist, erhalten wir

$$\widehat{\alpha}(F) \wedge \widehat{\alpha}(G) = \widehat{\alpha}(F') \wedge \widehat{\alpha}(G)$$

und analog für die Disjunktion. □

2.8.6 Folgerung. *Aus $F \equiv F'$ und $G \equiv G'$ folgt*

$$F \wedge G \equiv F' \wedge G' \quad , \quad F \vee G \equiv F' \vee G' \quad ,$$

$$F \Rightarrow G \equiv F' \Rightarrow G' \quad , \quad F \Leftrightarrow G \equiv F' \Leftrightarrow G' \quad , \quad F \oplus G \equiv F' \oplus G' \qquad \square$$

Wir hatten festgestellt, dass die Menge der korrekten aussagenlogischen Formeln „frei“ erzeugt worden war, d.h., Gleichheit wirklich syntaktische Gleichheit von Zeichenketten bedeutet. Das ist nicht besonders interessant. Die eben eingeführte Äquivalenz $\equiv$ ist eine „bessere Gleichheit“ auf der Menge der Formeln, da sie semantische Aspekte mit einbezieht. Wir sind letztlich nur an Formeln bis auf die obige Äquivalenz, oder kurz „modulo $\equiv$“, interessiert.

3. Eigenschaften von Formeln

Neben den Eigenschaften der Gleichheit auf dem semantischen Wertebereich $2 = \{0, 1\}$ für die klassische Aussagenlogik lassen sich auch Eigenschaften dort definierter Operationen in die Menge der Formeln modulo $\equiv$ reflektieren, so dass diese Quotienten- oder Faktormenge schließlich ebenfalls die Struktur einer sog. *Boole'schen Algebra* annimmt.

3.1 Eigenschaften der Negation

3.1.1 Satz. *Modulo $\equiv$ ist die Negation* ***selbstinvers****, d.h., für jede Formel $F \in \mathcal{F}$ gilt $\neg\neg F \equiv F$.*

Beweis. Jede passende Belegung α erfüllt

$$\widehat{\alpha}(\neg\neg F) = 1 - \widehat{\alpha}(\neg F) = 1 - (1 - \widehat{\alpha}(F)) = \widehat{\alpha}(F) \qquad \square$$

Dieses Ergebnis ist die Basis für den (allerdings nur klassisch akzeptierten) *Beweis durch Widerspruch*: statt F zu beweisen kann man $\neg F$ widerlegen, was einem Beweis von $\neg\neg F$ gleichkommt. Nach Satz 2.8.3 ist nun $\neg\neg F \Leftrightarrow F$ eine Tautologie, deren Langfassung die Form $(\neg\neg F \Rightarrow F) \wedge (F \Rightarrow \neg\neg F)$ ist. Insbesondere müssen dann beide Teilformeln $G = (\neg\neg F \Rightarrow F)$ und $H = (F \Rightarrow \neg\neg F)$ auch Tautologien der Aussagenlogik sein. Während G in so gut wie jedem logischen System gilt, ist das für H keineswegs der Fall. In solchen Systemen, wie z.B. der intuitionistischen Logik, reicht die Widerlegung von $\neg F$ nicht als Nachweis von F.

3.1.2 Satz. *Modulo $\equiv$ gelten in der klassischen Aussagenlogik die* ***De Morganschen Regeln****, d.h., alle Formeln F, G der Aussagenlogik erfüllen*

(a) $\neg(F \wedge G) \equiv \neg F \vee \neg G$

(b) $\neg(F \vee G) \equiv \neg F \wedge \neg G$

Beweis. Nach Definition der Semantik gilt für jede passende Belegung α

(a) $$1 - \inf\{\widehat{\alpha}(F), \widehat{\alpha}(G)\} = \sup\{1 - \widehat{\alpha}(F), 1 - \widehat{\alpha}(G)\}$$

(b) $$1 - \sup\{\widehat{\alpha}(F), \widehat{\alpha}(G)\} = \inf\{1 - \widehat{\alpha}(F), 1 - \widehat{\alpha}(G)\}$$ $\square$

3.2 Eigenschaften von Kon- und Disjunktion

3.2.1 Satz. *Die Konjunktion $\wedge$, modulo $\equiv$,*

- ▷ *ist* ***kommutativ:*** $F \wedge G \equiv G \wedge F$;
- ▷ *ist* ***assoziativ:*** $(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$;
- ▷ *ist* ***idempotent:*** $F \wedge F \equiv F$
- ▷ *hat $\top$ als* ***neutrales Element:*** $F \wedge \top \equiv F$.
- ▷ *hat $\bot$ als* ***absorbierendes Element:*** $F \wedge \bot \equiv \bot$.

Beweis. Das folgt aus den entsprechenden Eigenschaften der Infimum-Operation $\wedge$ auf dem semantischen Wertebereich $2 = \{0, 1\}$. Diese werden von jeder passenden Belegung α reflektiert. □

Völlig analog erhält man

3.2.2 Satz. *Die Disjunktion $\vee$ ist modulo $\equiv$ kommutativ, assoziativ, idempotent und hat $\bot$ als neutrales sowie $\top$ als absorbierendes Element.* □

3.2.3 Notationelle Konvention.

1. Die Assoziativität erlaubt es Formeln wie $A \wedge B \wedge C$ oder $A \wedge B \wedge C \wedge D$ usw. ohne gruppierende Klammern zu schreiben, analog für die Disjunktion.
2. Abkürzend schreiben wir auch $\bigvee_{i<n} F_i$ statt $F_0 \vee \ldots \vee F_{n-1}$ und $\bigwedge_{i<n} F_i$ statt $F_0 \wedge \ldots \wedge F_{n-1}$. Genauer: wir definieren das Symbol $\bigvee_{i<n} F_i$ induktiv:

$$\bigvee_{i<0} F_i = \bot \qquad \text{und} \qquad \bigvee_{i<n+1} F_i = \Big(\bigvee_{i<n} F_i\Big) \vee F_n$$

   Analoges gilt für $\bigwedge_{i<n} F_i$, speziell $\bigwedge_{i<0} F_i = \top$. In beiden Fällen liefert die 0-fache Anwendung des Junktors ($i < 0$) das neutrale Element des Junktors.

3.2.4 Bemerkung. Die De Morganschen Regeln kann man wie folgt verallgemeinern:

$$\neg \bigwedge_{i<n} F_i \equiv \bigvee_{i=1}^{n} \neg F_i \qquad \text{sowie} \qquad \neg \bigvee_{i<n} F_i \equiv \bigwedge_{i=1}^{n} \neg F_i$$

Der Beweis erfolgt analog zu Satz 3.1.2.

3.2.5 Satz. *Modulo $\equiv$ erfüllen Konjunktion und Disjunktion die* ***Distributivgesetze****, d.h., für alle Formeln F und G_i, $i < n$ gilt*

$$F \wedge \bigvee_{i<n} G_i \equiv \bigvee_{i<n} (F \wedge G_i) \qquad \textit{sowie} \qquad F \vee \bigwedge_{i<n} G_i \equiv \bigwedge_{i<n} (F \vee G_i)$$

Beweis. Für $n = 0$ stimmt die Aussage mit der Absorbtionseigenschaft von $\bot$ bzw. $\top$ bzgl. der Konjunktion bzw. Disjunktion überein, vergleiche Satz 3.2.1.

Für $n = 1$ besteht syntaktische Gleichheit, woraus wegen der Reflexivität von $\equiv$ die Behauptung folgt.

Aufgrund der Assoziativität von $\equiv$ genügt nun der Beweis für den Fall $n = 2$. Aus Symmetriegründen beschränken wir uns auf das erste Distributivgesetz. Zu zeigen ist

$$F \wedge (G_0 \vee G_1) \equiv (F \wedge G_0) \vee (F \wedge G_1)$$

Ist α eine passende Belegung, so gilt es in $2 = \{0, 1\}$

$$\widehat{\alpha}F \wedge \big(\widehat{\alpha}(G_0) \vee \widehat{\alpha}(G_1)\big) = \big(\widehat{\alpha}(F) \wedge \widehat{\alpha}(G_0)\big) \vee \big(\widehat{\alpha}(F) \wedge \widehat{\alpha}(G_1)\big)$$

nachzuweisen. Die linke Seite hat genau denn den Wert 1, wenn beide Argumente der Konjunktion den Wert 1 haben, d.h., wenn F und mindestens eine der Formel G_i, $i < 2$, wahr ist.

Die rechte Seite hat genau dann den Wert 1, wenn mindestens ein Argument der Disjunktion den Wert 1 hat, d.h., wenn F und mindestens eine der Formeln G_i, $i < 2$, wahr sind. Damit ist die Behauptung beweisen. □

3.2.6 Bemerkung. Äquivalente Formeln haben dieselbe Wahrheitstabelle. Aufgrund der Kommutativität und der Idempotenz der Konjunktion wie der Disjunktion haben wir damit genau 4 potentiell verschiedene Formeln, die nur ein Atom A verwenden:

$$A,\ \neg A,\ A \vee \neg A \quad \text{und} \quad A \wedge \neg A$$

Ob diese Formeln wirklich verschieden sind, entnehmen wir den Wahrheitstabellen

A	A
0	0
1	1

A	$\neg A$
0	1
1	0

A	$A \vee \neg A$
0	1
1	1

A	$A \wedge \neg A$
0	0
1	0

In der Tat werden alle Möglichkeiten ausgeschöpft.

Analog haben die Wahrheitstabellen für genau zwei verschiedene Atome vier Zeilen mit $2^4 = 16$ möglichen rechten Spalten. Also können modulo $\equiv$ höchstens 16 nicht äquivalente Formeln mit zwei verschiedenen Atomen auftreten.

Allgemein können wir mit genau n verschiedenen Atomen höchstens 2^{2^n} paarweise nicht-äquivalente Formeln aufbauen.

3.3 Adäquate Junktoren

Die De Morganschen Regeln und die Tatsache, dass $\neg$ selbstinvers ist, zeigen, dass die von uns verwendeten „Basis-Junktoren“ $\bot$, $\neg$, $\wedge$ und $\vee$ noch Redundanz aufweisen. So ist $A_0 \wedge \neg A_0$ äquivalent zu $\bot$. Weiterhin kann z.B. $\vee$ eliminiert werden: aus $\neg(F \vee G) \equiv \neg F \wedge \neg G$ folgt durch Negation beider Seiten

$$F \vee G \equiv \neg(\neg F \wedge \neg G)$$

3.3.1 Definition. Eine Menge $\mathcal{J}$ von Junktoren definierter Semantik heißt **adäquat**, falls jede Formel aus $\mathcal{F}$ zu einer Formel über dem Alphabet $\mathcal{P} + \mathcal{J} + \{(,)\}$ äquivalent ist, d.h., zu einer Formel, in der nur Junktoren aus $\mathcal{J}$ vorkommen.

3.3.2 Beispiel.

(a) Wie wir gerade gesehen haben, formen $\wedge$ und $\neg$ eine adäquate Menge. Analog sind auch $\vee$ und $\neg$ adäquat.

(b) Die Junktoren $\Rightarrow$ und $\neg$ sind adäquat. In der Tat: die Disjunktion können wir wie folgt ersetzen:

$$F \vee G \equiv \neg\neg F \vee G = \neg F \Rightarrow G$$

und dann benutzen wir die Tatsache, dass $\vee$ und $\neg$ adäquat sind.

(c) Die Junktoren $\Rightarrow$ und $\bot$ sind adäquat. Da $F \Rightarrow \bot$ eine Abkürzung für $\bot \vee \neg F$ ist, impliziert die Neutralität von $\bot$ bzgl. $\vee$ sofort $F \Rightarrow \bot \equiv \neg F \vee \bot \equiv \neg F$. Nun folgt die Behauptung aus (b).

3.3.3 Beispiel. Die Junktoren $\wedge$ und $\bot$ sind nicht adäquat. In der Tat, für eine atomare Aussage A ist die Formel $\neg A$ zu keiner Formel F äquivalent, die nur $\wedge$ und $\bot$ als Junktoren verwendet. Falls nämlich die Formel F das Symbol $\bot$ nicht enthält, hat sie die Form $F = \bigwedge_{i<n} B_i$, mit $B_i \neq A$ atomar. Die Belegung, die A und B_i, $i < n$, den Wert 1 zuordnet, erfüllt dann $\widehat{\alpha}(\neg A) = 0 \neq 1 = \widehat{\alpha}(F)$; und solche Belegungen existieren. Falls andererseits F das Symbol $\bot$ enthält, gilt $\widehat{\alpha}(F) = 0$ für jede passende Belegung α. Wählen wir speziell die Belegung, die A und allen B_i, $i < n$, den Wert 0 zuordnet, dann gilt $\widehat{\alpha}(\neg A) = 1 \neq 0 = \widehat{\alpha}(F)$.

3.3.4 Beispiel. Der NAND-Junktor $\uparrow$ (auch als "Sheffer-Stroke" bekannt) ist der Junktor "nicht beide", der durch

$$A \uparrow B = \neg(A \wedge B)$$

definiert wird. Die Aussagenlogik kann durch den NAND Junktor allein aufgebaut werden; die Menge $\{\uparrow\}$ ist adäquat:

Negation: $\neg A \equiv \neg(A \wedge A) \equiv A \uparrow A$

Konjunktion: $A \wedge B \equiv \neg(A \uparrow B) \equiv (A \uparrow B) \uparrow (A \uparrow B)$

Disjunktion: $A \vee B \equiv \neg(\neg A \wedge \neg B) \equiv (\neg A) \uparrow (\neg B) = (A \uparrow A) \uparrow (B \uparrow B)$

Absurdität: $\bot \equiv A \wedge \neg A \equiv (A \uparrow (A \uparrow A)) \uparrow (A \uparrow (A \uparrow A))$

Der Preis für die geringe Anzahl an Junktoren ist natürlich eine gewisse Unübersichtlichkeit der resultierenden Formeln (speziell bei Verwendung umgekehrt polnischer Notation, vergl. HA).

4. Normalformen

Es ist oft wichtig mit Formeln einer gewissen „schönen“ Gestalt arbeiten zu können. Die wichtigste solcher Formen heißt konjunktive Normalform (KNF) und wir zeigen hier, wie sie berechnet werden kann. Zwei andere Normalformen werden ebenfalls eingeführt.

4.1 Negation-Normalform (NNF)

Die De Morganschen Regeln und die Tatsache, dass $\neg$ selbstinvers ist, ermöglichen es, Formeln so umzugestalten, dass die Negation höchstens vor atomaren Aussagen auftritt.

4.1.1 Beispiel.

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

4.1.2 Definition. Eine Formel $F \in \mathcal{F}$ hat **Negations-Normalform** (NNF), falls alle Negationen in F direkt vor atomaren Aussagen auftreten. $\mathcal{F}^{NNF}$ bezeichnet die entsprechende Teilmenge von $\mathcal{F}$.

Wir können für jede Formel F ihre NNF berechnen indem wir die De Morganschen Regeln rekursiv anwenden: wegen

$$F = \neg(G \vee H) \equiv \neg G \wedge \neg H$$

sind nur die NNF für $\neg G$ sowie $\neg H$ zu berechnen. Analog für $F = \neg(G \vee H)$. Weiterhin gilt im Falle $F = \neg\neg G$ natürlich $F \equiv G$. Dies formalisiert der folgende rekursive Algorithmus:

4.1.3 Algorithmus (NNF)**.**
Eingabe: Eine aussagenlogische Formel F mit Junktoren $\vee$, $\wedge$ und $\neg$.
(Andere Junktoren wie $\top$, $\Rightarrow$, $\Leftrightarrow$, $\oplus$ etc. sind ggf. erst zu ersetzen.)
Ausgabe: Eine zu F äquivalente Formel aus $\mathcal{F}^{NNF}$, bezeichnet als $\mathsf{NNF}(F)$

Wir verwenden strukturelle Rekursion:

0. $F = \bot$ oder F atomar: wir setzen

$$\mathsf{NNF}(F) := F$$

1. $F = G_0 \wedge G_1$: wir setzen

$$\mathsf{NNF}(G_0 \wedge G_1) := \mathsf{NNF}(G_0) \wedge \mathsf{NNF}(G_1)$$

2. $F = G_0 \vee G_1$: wir setzen

$$\mathsf{NNF}(G_0 \vee G_1) := \mathsf{NNF}(G_0) \vee \mathsf{NNF}(G_1)$$

3. $F = \neg G$: Hier müssen wir eine zweite Rekursion starten, diesmal in G.

   3.0 $G = \bot$ oder G atomar: wir setzen

$$\mathsf{NNF}(\neg G) := \neg G$$

   3.1 $G = H_0 \wedge H_1$: wir setzen

$$\mathsf{NNF}(\neg(H_0 \wedge H_1)) := \mathsf{NNF}(\neg H_0) \vee \mathsf{NNF}(\neg H_1)$$

   3.2 $G = C_0 \vee C_1$: wir setzen

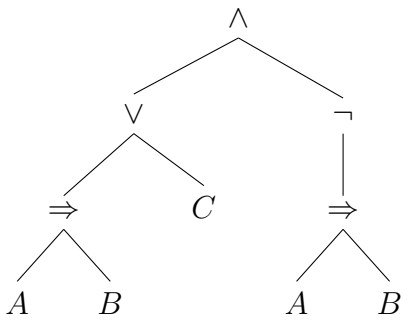
$$\mathsf{NNF}(\neg(H_0 \vee H_1)) := \mathsf{NNF}(\neg H_0) \wedge \mathsf{NNF}(\neg H_1)$$

   3.3 $G = \neg H$: wir setzen

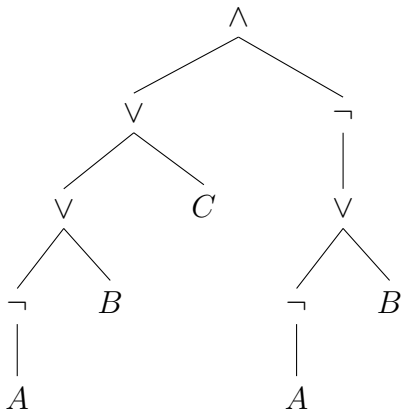
$$\mathsf{NNF}(\neg\neg H) := \mathsf{NNF}(H)$$

Anschaulich verschiebt der Algorithmus NNF die $\neg$-Knoten im Baum nach unten, bis sie sich gegen andere $\neg$-Knoten aufheben, oder vor Blättern landen.

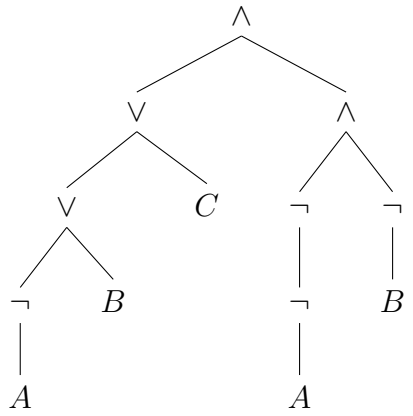
4.1.4 Beispiel. $((A \Rightarrow B) \vee C) \wedge \neg(A \Rightarrow B)$ hat als (vereinfachter) Baum die Form



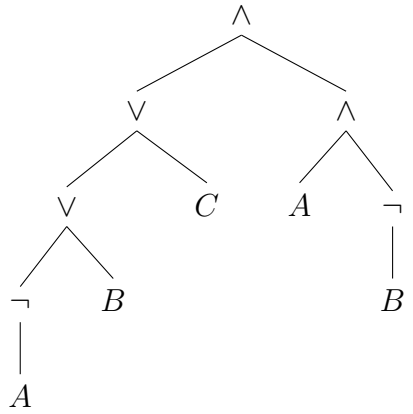
Zunächst ersetzen wir die $\Rightarrow$-Knoten:



Jetzt wird im rechten Unterbaum eine der De Morganschen Regeln angewendet:



und schließlich wird die resultierende doppelte Negation entfernt:



In linearer Form erhalten wir nun $(\neg A \vee B \vee C) \wedge A \wedge \neg B$.

4.1.5 Satz. *Der Algorithmus* NNF *ist korrekt, genauer: für jede Formel $F \in \mathcal{F}$ wird $\mathsf{NNF}(F) \in \mathcal{F}^{NNF}$ in endlich vielen Schritten berechnet und ist äquivalent zu F.*

Beweis. Die endliche Schrittzahl folge aus der Tatsache, dass für eine Formel F der Größe $n > 1$ der Algorithmus NNF ein oder zweimal auf Formeln der Gesamtlänge $n - 1$ anzuwenden ist. Da für Formeln der Größe 1 nichts zu tun ist, ist NNF insgesamt höchstens so oft aufzurufen, wie die Größe der ursprünglichen Formel angibt.

Die Äquivalenz $F \equiv \mathsf{NNF}(F)$ sowie $\mathsf{NNF}(F) \in \mathcal{F}^{NNF}$, müssen wieder mittels struktureller Induktion bewiesen werden. Falls $F = \bot$ oder F atomar, gilt natürlich $\mathsf{NNF}(F) = F \in \mathcal{F}^{NNF}$. Stimmt die Aussage für G_0 und G_1, verwenden wir für $F = G_0 \wedge G_1$ die Definition

$$\mathsf{NNF}(F) := \mathsf{NNF}(G_0) \wedge \mathsf{NNF}(G_1)$$

und benutzen dann die Kongruenzeigenschaft aus Satz 2.8.5: aus $G_i \equiv \mathsf{NNF}(G_i)$ für $i < 2$ folgt

$$\mathsf{NNF}(F) \equiv G_0 \wedge G_1 = F$$

Wegen $\mathsf{NNF}(G_0), \mathsf{NNF}(G_1) \in \mathcal{F}^{NNF}$, folgt dies unmittelbar auch für $\mathsf{NNF}(F)$, da keine neue Negation zum Einsatz kommt.

Der Fall $F = G_0 \vee G_1$ ist analog zu bearbeiten. Dagegen ist für $F = \neg G$ die innere Rekursion über G aus Schritt 3 anzuwenden, was aber ganz ähnlich wie oben geht. □

4.1.6 Beispiel. Wir wollen eine NNF der Formel

$$(C \Rightarrow \neg A \vee B) \wedge (A \wedge C \Rightarrow \neg B \vee C)$$

berechnen. Bevor wir den rekursiven Algorithmus starten, müssen wir $\Rightarrow$ ersetzen. Wir arbeiten mit der Formel

$$G = \big(\neg C \vee (\neg A \vee B)\big) \wedge \big(\neg(A \wedge C) \vee (\neg B \vee C)\big)$$

Die rekursive Berechnung wird separat auf der linken und der rechten Seite entwickelt:

$$\begin{aligned}
\mathsf{NNF}(G) &= \big(\mathsf{NNF}(\neg C) \vee \mathsf{NNF}(\neg A \vee B)\big) \wedge \big(\mathsf{NNF}(\neg(A \wedge C) \vee (\neg B \vee C)\big) \\
&= \big(\neg C \vee \mathsf{NNF}(\neg A) \vee \mathsf{NNF}(B)\big) \wedge \big(\mathsf{NNF}(\neg(A \wedge C)) \vee \mathsf{NNF}(\neg B \vee C)\big) \\
&= \big(\neg C \vee \neg A \vee B\big) \wedge \big(\mathsf{NNF}(\neg A) \vee \mathsf{NNF}(\neg C) \vee \mathsf{NNF}(\neg B) \vee \mathsf{NNF}(C)\big) \\
&= \big(\neg C \vee \neg A \vee B\big) \wedge \big(\neg A \vee \neg C \vee \neg B \vee C\big) \\
&\equiv \big(\neg C \vee \neg A \vee B\big) \wedge \top \\
&\equiv \neg C \vee \neg A \vee B
\end{aligned}$$

Die letzten beiden Vereifachungen sind unabhängig vom Algorithmus, daher steht das Ergebnis $\mathsf{NNF}(G)$ in der drittletzten Zeile.

4.2 Konjunktive Normalform

4.2.1 Definition. Unter einem **Literal** verstehen eine atomare Aussage oder die Negation einer solchen, formal also

$$\mathcal{L} := \mathcal{P} + \neg\mathcal{P}$$

wobei $\neg\mathcal{P}$ abkürzend für $\{\, \neg A_i \,:\, i \in I\!N \,\}$ steht. Wir sprechen auch von positiven Literalen A oder negativen Literalen $\neg A$ (A atomar). Da die Negation $\neg$ selbstinvers ist, ist $\mathcal{L}$ hinsichtlich Äquivalenz unter Negation abgeschlossen.

4.2.2 Definition. Eine Disjunktion von Literalen heißt **Klausel**.

4.2.3 Beispiel.

(a) Die Formel $A \vee A \vee \neg B \vee C$ ist eine Klausel. Sie ist äquivalent zur kürzeren Formel $A \vee \neg B \vee C$, die auch eine Klausel ist.

(b) Für n atomare Aussagen $X_0, \ldots, X_{n-1}$ ist $\bigvee_{i<n} X_i$ eine Klausel. Ihr Wert ist 1 unter genau den passenden Belegungen α, für die ein Index $i < n$ mit $\alpha(X_i) = 1$ existiert. Im Fall $n = 0$ stimmt die Klausel mit $\bot$ überein und ist somit falsch.

4.2.4 Bemerkung. Der Wahrheitswert $\widehat{\alpha}(F)$ läßt sich besonders einfacher bestimmen wenn F eine Konjunktionen $K_0 \wedge \ldots \wedge K_{n-1}$ von Klauseln ist: $\widehat{\alpha}(F)$ ist genau dann wahr, wenn alle Klauseln K_i wahr sind. Dafür muß für jedes $i < n$ ein positives Literal A mit $\alpha(A) = 1$ in K_i vorkommen, oder ein negatives Literal $\neg A$ mit $\alpha(A) = 0$. Daher verdienen solche Formeln einen besonderen Namen.

4.2.5 Definition. Eine Formel in **Konjunktiver Normalform** (KNF) besteht aus einer Konjunktion von Klauseln.

4.2.6 Beispiel. Die Formel $(B \wedge \neg A \wedge C) \vee (\neg A \wedge \neg B)$ wollen wir in KNF darstellen. Dazu benutzten wir die Distributivgesetze:

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

Zum Schluss kamen die Idempotenz von $\vee$ und die Neutraltäit von $\top$ zur Anwendung.

Die vorletzte Zeile des obigen Beispiels liegt also bereits in KNF vor. Wir stellen jetzt einen Algorithmus vor, der eine KNF berechnet. Sein Kern besteht in der Verwendung eines Distributivgesetzes. Wir formulieren erst diesen Kern, und fügen diesen dann in unserem Algorithmus ein:

4.2.7 Satz. *Falls F und G in KNF mit Klauseln F_i mit $i < n$ bzw. G_j mit $j < m$ vorliegen, ist $F \vee G$ äquivalent zur Konjunktion der Klauseln $F_i \vee G_j$ für $i < n$ und $j < m$.*

Beweis. Es gilt, aufgrund des Distributivgesetzes:

$$F \vee G = (\bigwedge_{i<n} F_i) \vee G \equiv \bigwedge_{i<n} (F_i \vee G)$$

Für jedes i berechnen wir analog

$$F_i \vee G \equiv F_i \vee (\bigwedge_{j<m} G_j) \equiv \bigwedge_{j<m} (F_i \vee G_j)$$

Daraus folgt:

$$F \vee G \equiv \bigwedge_{i<n} (F_i \vee G) \equiv \bigwedge_{i<n} \bigwedge_{j<m} (F_i \vee G_j)$$

Weil die Formeln F_i und G_j Klauseln sind ist aufgrund der Assoziativität von $\vee$ auch $F_i \vee G_j$ eine Klausel. Somit liegt die Gesamtformel in KNF vor. □

Der folgende Algorithmus verwendet wieder strukturelle Rekursion.

4.2.8 Algorithmus (KNF)**.**
Eingabe: Eine Formel $F \in \mathcal{F}^{NNF}$
Ausgabe: Eine zu F äquivalente Formel $\mathsf{KNF}(F) \in \mathcal{F}^{KNF}$

0. F atomar: wir setzen

$$\mathsf{KNF}(F) := F$$

1. $F = G \wedge H$, wobei $\mathsf{KNF}(G)$ und $\mathsf{KNF}(H)$ bekannt sind: wir setzen

$$\mathsf{KNF}(F) := \mathsf{KNF}(G) \wedge \mathsf{KNF}(H)$$

2. $F = G \vee H$, wobei $\mathsf{KNF}(G)$ und $\mathsf{KNF}(H)$ folgende Form als Konjunktion von Klauseln haben

$$\mathsf{KNF}(G) = \bigwedge_{i<n} G_i \quad \text{bzw.} \quad \mathsf{KNF}(H) = \bigwedge_{j<m} H_j$$

Dann setzen wir gemäß Satz 4.2.7

$$\mathsf{KNF}(G \vee H) := \bigwedge_{i<n} \bigwedge_{j<m} (G_i \vee H_j)$$

3. $F = \neg G$. Hier muss $G = \bot$ gelten oder G atomar sein, da F nach Voraussetzung in NNFvorliegt. Wir setzen

$$\mathsf{KNF}(\neg\bot) := \top \qquad \text{bzw.} \qquad \mathsf{KNF}(\neg G) := \neg G.$$

4.2.9 Satz. *Der Algorithmus* KNF *ist korrekt: für jede Formel F berechnen wir in endlich vielen Schritte eine zu F äquivalente Formel aus $\mathfrak{F}^{KNF}$.*

Beweis. Der Beweis hat dieselbe Struktur wie der von Satz 4.1.5. Der einzige nichttriviale Schritt ist zu beweisen, dass $\mathsf{KNF}(F)$ für $F = G \vee H$ zu F äquivalent ist. Aber dies folgt aus Satz 4.2.7. □

4.2.10 Beispiel. Die KNF der Formel

$$F = A \Leftrightarrow (B \Rightarrow \neg C \wedge A)$$

berechnen wir in 3 Schritten:

(a) $\Leftrightarrow$ und $\Rightarrow$ sind zu ersetzen, vergleiche Proposition 2.4.2 (b) und (c):

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

(b) Der Algorithmus NNF liefert

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

(c) Algorithmus KNF wird angewendet. Das zweite Argument der äußeren Disjunktion liegt bereits in KNF vor. In ihrem ersten Argument hat das zweite Argument der Konjunktion (rot) noch nicht die Form einer KNF, es handelt sich aber um die Disjunktion zweier Formeln in KNF. Behandlung gemäß Schritt 2, d.h. Anwendung von Satz 4.2.7 liefert hier eine Formel in KNF (grün):

$$F \equiv (A \wedge (\neg B \vee \neg C) \wedge (\neg B \vee A)) \vee (\neg A \wedge B \wedge (C \vee \neg A))$$

Auf die resultierende Disjunktion zweier Formeln in KNF kann nun ebenfalls Satz 4.2.7 angewendet werden, was die Kombination von jeweils 3 Klauseln erfordert:

$$\begin{aligned}\mathsf{KNF}(F) = (A \vee \neg A) \wedge (A \vee B) \wedge (A \vee C \vee \neg A) \wedge (\neg B \vee \neg C \vee \neg A)\\ \wedge (\neg B \vee \neg C \vee B) \wedge (\neg B \vee \neg C \vee C \vee \neg A)\\ \wedge (\neg B \vee A \vee \neg A) \wedge (\neg B \vee A \vee B)\\ \wedge (\neg B \vee A \vee C \vee \neg A)\end{aligned}$$

Schließlich kann man noch, unabhängig vom KNF=Algorithmus, die offensichtlichen Tautologien zusammenfassen, und dann Klauseln mit $\top$ entfernen:

$$\begin{aligned}\mathsf{KNF}(F) &\equiv \top \wedge (A \vee B) \wedge (\top \vee C) \wedge (\neg B \vee \neg C \vee \neg A)\\ &\quad \wedge (\neg C \vee \top) \wedge (\top \vee \neg B \vee \neg A)\\ &\quad \wedge (\top \vee \neg B) \wedge (A \vee \top) \wedge (\neg B \vee C \vee \top)\\ &\equiv (A \vee B) \wedge (\neg B \vee \neg C \vee \neg A)\end{aligned}$$

4.2.11 Beispiel. Für $G_2 = (A_0 \wedge B_0) \vee (A_1 \wedge B_1)$ ergibt sich $\mathsf{KNF}(G_2)$ aufgrund von Satz 4.2.7, d.h. nur unter Verwendung des Distributivgesetzes, zu:

$$\mathsf{KNF}(G_2) = (A_0 \vee A_1) \wedge (A_0 \vee B_1) \wedge (B_0 \vee A_1) \wedge (B_0 \vee B_1)$$

Analog: für die Formel $G_{n-1} = \bigvee_{i<n}(A_i \wedge B_i)$ ergibt sich $\mathsf{KNF}(G_{n-1})$ als Konjunktion aller Klauseln des Typs

$$A_0 \vee A_1 \vee \ldots A_{n-1} \quad , \quad B_0 \vee A_1 \vee \ldots A_{n-1} \quad , \quad \ldots \quad , \quad B_0 \vee B_1 \vee \ldots B_{n-1}$$

wobei immer n Atome mit aufsteigenden Indizes $i < n$ auftreten und als Namen zunächst B, dann später A.

4.2.12 Bemerkung. Die Formel G_{n-1} hat Größe $4n - 1 \in O(n)$. Aber $\mathsf{KNF}(G_{n-1})$ hat 2^n Klauseln, da für jeden Index $1, \ldots, n$ die Wahl zwischen A oder B vorkommt. Deswegen liegt die Größe von $\mathsf{KNF}(G_{n-1})$ in $O(2^n)$. Dies bedeutet, dass der Algorithmus KNF nicht effizient ist: für Formeln der Größe 200 ist es i.A. praktisch unmöglich eine KNF zu berechnen. In der Tat: die Lichtgeschwindigkeit ist $3 \cdot 10^8\, {}^{m}/_{sec}$ und der Durchschnitt eines Protons ist $10^{-15}m$ dann lieferte ein Rechner, der jede Operation in der Zeit erledigt, die das Licht braucht um den Durchmesser eines Protons zurückzulegen, $3 \cdot 10^{23}$ Operationen pro Sekunde. Wegen $10 < 2^4$ gilt $3 \cdot 10^{23} < 2^{100}$. Falls also ein Algorithmus 2^{200} Schritte braucht, müsste auch ein so schneller Rechner wenigstens 2^{100} Sekunden rechnen - dies ist aber mehr, als die Zeit ($4 \cdot 10^{17} < 2^{59}$ Sekunden) seit dem Urknall.

4.3 Disjunktive Normalform

4.3.1 Definition. Eine Formel hat **disjunktive Normalform** (DNF), falls sie als Disjunktion von Co-Klauseln vorliegt, wobei Co-Klauseln Konjunktionen von Literalen sind.

4.3.2 Beispiel. Wir wandeln die Formel $C \wedge A \Rightarrow (B \wedge \neg C)$ in DNF um:

$$C \wedge A \Rightarrow (B \wedge \neg C) \equiv \neg(C \wedge A) \vee (B \wedge \neg C) \equiv \neg C \vee \neg A \vee (B \wedge \neg C)$$

4.3.3 Algorithmus. Um eine Formel G in DNF umzuwandeln, können wir die de Morganschen Regeln auf eine KNF für $\neg G$ anwenden.

1. Wir berechnen eine KNF $H = \mathsf{KNF}(\mathsf{NNF}(\neg G))$ für $\neg G$;

2. Wir berechnen eine NNF von $\neg H$:

$$\begin{aligned} G &\equiv \mathsf{NNF}(\neg \mathsf{KNF}(\mathsf{NNF}(\neg G))) \\ &\equiv \mathsf{NNF}(\neg(H_0 \wedge H_1 \wedge \ldots \wedge H_{n-1})) \\ &\equiv \mathsf{NNF}(\neg H_0) \vee \mathsf{NNF}(\neg H_1) \vee \ldots \vee \mathsf{NNF}(\neg H_{n-1}) \end{aligned}$$

   Die Anwendung von NNF auf die negierten Klauseln besteht in der Anwendung der De Morganschen Regeln und ggf. der Eliminierung doppelter Negationen, liefert also Co-Klauseln, und insgesamt eine DNF.

Wie in einer frühen Hausaufgabe gesehen, läßt sich eine DNF für F auch direkt aus einer Wahrheitstabelle für F herleiten. Allerdings dürfte diese Methode eher noch langsamer sein als obiger Algorithmus, da die Wahrheitstabelle bei einer Formel mit n Variablen 2^n Zeilen besitzt.

5. Logische Konsequenz

Ganz ähnlich, wie wir mit der Äquivalenz $\equiv$ den Junktor $\Leftrightarrow$ externalisiert haben, wollen wir nun mit dem Junktor $\Rightarrow$ für die Implikation verfahren. Die im Folgenden beschriebene Methode wird außerdem die Konjunktion $\wedge$ externalisieren.

5.0.1 Definition. Gegeben seien eine nicht notwendig endliche Menge Γ von Formeln und eine einzelne Formel F.

(a) Eine für Γ passende Belegung α (siehe Definition 2.3.6) **erfüllt** Γ, wenn $\widehat{\alpha}$ jedes Element $G \in \Gamma$ auf 1 abbildet. Existiert keine derartige Belegung, so heißt Γ **nicht erfüllbar**.

(b) Die Formel F **folgt** aus der Menge Γ, oder ist deren **logische Konsequenz**, geschrieben $\Gamma \models F$, wenn jede für $\Gamma \cup \{F\}$ passende und Γ erfüllende Belegung auch F erfüllt.

Die Elemente von Γ werden dabei als **Prämissen** bezeichnet.

Wir stellen fest, dass die Folge-Relation $\models$ letztendlich mittels der Inklusion zwischen bestimmten Mengen von Belegungen definiert ist. Insbesondere läßt sich die Nicht-erfüllbarkeit von Γ durch $\Gamma \models \bot$ charakterisieren. In den HA wird bewiesen:

5.0.2 Satz. *$\Gamma \models F$ gilt genau dann, wenn $\Gamma \cup \{\neg F\}$ nicht erfüllbar ist.* □

5.0.3 Notation. Für endliche Mengen $\Gamma = \{G_0, \ldots, G_{n-1}\}$ lässt man die Mengenklammern weg und schreibt

$$G_0, \ldots, G_{n-1} \models F$$

5.0.4 Satz. *Für jede endliche Menge Γ von Formeln und jede Formel F sind folgende Aussagen der Metasprache äquivalent:*

(a) $\Gamma \models F$*;*

(b) $\bigwedge \Gamma \Rightarrow F$ *ist eine Tautologie;*

(c) $\bigwedge \Gamma \models F$*.*

(d) $\bigwedge \Gamma \wedge \neg F$ *ist nicht erfüllbar.*

Beweis. Die Implikation $\bigwedge \Gamma \Rightarrow F$ ist eine Tautologie wenn jede für $\Gamma \cup \{F\}$ passende Belegung α eine der folgenden Bedingungen erfüllt: $\widehat{\alpha}(\bigwedge \Gamma) = 0$ oder $\widehat{\alpha}(F) = 1$. Aber dies sagt genau, dass jede passende Belegung α mit $\widehat{\alpha}(\bigwedge \Gamma) = \bigwedge \widehat{\alpha}[\Gamma] = 1$ auch $\widehat{\alpha}(F) = 1$

erfüllt, was gleichbedeutend mit $\widehat{\alpha}\,(\bigwedge \Gamma \wedge \neg F) = \bigwedge \widehat{\alpha}[\Gamma \cup \{\neg F\}] = 0$ ist. Man beachte, dass $\bigwedge\{\,\widehat{\alpha}(G) \,:\, G \in \Gamma\,\} = 1$ genau dann gilt, wenn jedes $G \in \Gamma$ die Bedingung $\widehat{\alpha}(G) = 1$ erfüllt. □

5.0.5 Bemerkung. Die Relation $\models$ setzt nicht notwendig endliche Mengen von Prämissen mit einzelnen Formeln in Beziehung und ist insofern asymmetrisch. Es ist aber möglich, auch allgemeinere logische Konsequenzen zu betrachten, bei denen auf der rechten Seite mehr als eine Formel vorkommt, etwa $A, B \models C, D$. Dies soll als $A \wedge B \models C \vee D$ verstanden werden, d.h., die Interpretation des Kommas ist „kontextsensitiv“: links von $\models$ dient es als Konjunktion, rechts von $\models$ als Disjunktion. Auf der syntaktischen Ebene, d.h., in der **Beweistheorie** (Kapitel 7) ist das beim Sequenzenkalkül für die klassische Logik von Vorteil, siehe etwa [Cal11].

Satz 5.0.4 gilt speziell für $\Gamma = \varnothing$. Das führt zu einer alternativen Charakterisierung von Tautologien.

5.0.6 Folgerung. *Eine Formel F ist genau dann eine Tautologie, wenn $\models F$ gilt, was zu $\top \models F$, und somit zur Tautologie-Eigenschaft von $\top \Rightarrow F$ gleichwertig ist.* □

5.0.7 Beispiel.

(a) $A \wedge B \models A$.

(b) $A, B \models A \wedge B$.

(c) A, $A \Rightarrow B \models B$

5.0.8 Satz. *Die folgenden fünf Aussagen sind gleichwertig:*

(a) $A \wedge B \models F$

(b) $A, B \models F$

(c) $A \models B \Rightarrow F$

(d) $\models A \Rightarrow (B \Rightarrow F)$

(e) $\models A \wedge B \Rightarrow F$

Beweis. Die Gleichwertigkeit von (a) und (b), ebenso wie die von (c) und (d), bzw. von (e) und (a) folgt unmittelbar aus Satz 5.0.4.

Nachzuweisen bleibt die Gleichwertigkeit von (b) und (c), d.h., dass die Prämissen einzeln, und nicht nur zur Gänze nach rechts transportiert werden können. Aber aus Satz 5.0.4 folgt die Gleichwertigkeit von $B \models F$ mit $\models B \Rightarrow F$, eine Aussage die alle für B und F passenden Belegungen betrifft. Die Einschränkung auf diejenigen Belegungen, die zudem A wahr machen, liefert die Behauptung. □

5.0.9 Bemerkung. Wir haben absichtlich in der Formulierung des Satzes den Begriff „gleichwertig“ anstelle des sonst üblichen „äquivalent“ verwendet, um potentielle Verwirrung zu vermeiden. Es wäre nämlich schon die dritte Variante von „Äquivalenz“ gewesen:

- ▷ Der Junktor $\Leftrightarrow$ im Alphabet der Aussagenlogik heißt „Äquivalenz“.
- ▷ Diesen Junktor hatten wir in Abschnitt 2.8 externalisiert zu einer Relation $\equiv$, die auch als „Äquivalenz“ bezeichnet wurde. Dies ist Teil der sogenannten *Meta-Sprache*, mit der wir über die Aussagenlogik sprechen.
- ▷ Satz 5.0.8 vergleicht schließlich mehrere meta-sprachliche Aussagen hinsichtlich ihres Wahrheitsgehalts und stellt fest, dass sie alle denselben haben. Üblicherweise wird das als „Äquivalenz“ dieser meta-sprachlichen Aussagen formuliert, womit wir uns aber im Bereich der Meta-Meta-Sprache bewegen!

Diese Probleme könnten umgangen werden, wenn man die Sprach-Ebene mit einfließen läßt. Allerdings ist es auf Dauer doch recht umständlich, von „0-Äquivalenz“, „1-Äquivalenz“, „2-Äquivalenz“ usw. zu sprechen.

Im Hinblick auf die Sätze 5.0.4 und 5.0.8 stellt sich die Frage, wozu die Externalisierung $\models$ der Implikation $\Rightarrow$ überhaupt gut ist. Wir erinnern uns, dass die Definition von $\Gamma \models F$ keine Größenbeschränkung für Γ enthielt, insbesondere kann Γ unendlich sein. Und dafür gibt es offenbar keine interne Entsprechung.

5.0.10 Beispiel.

(a) Die Menge Γ möge aus allen Aussagen bestehen, die höchstens $\vee$ oder $\wedge$ als Junktoren enthalten. Gilt für Atome A und B die logische Konsequenz

$$\Gamma \models A \Rightarrow B \quad ?$$

Jede für A und B passende Belegung, die B wahr macht, macht auch $A \Rightarrow B$ wahr. Und falls eine Belegung alle Formeln aus Γ wahr macht, macht sie insbesondere auch $B \in \Gamma$ wahr. Daher ist die obige Aussage korrekt.

(b) Wie steht es mit der logischen Konsequenz

$$\Gamma \models \bot$$

für die Menge Γ aller Literale? Auch diese gilt, weil keine Belegung existiert, die alle Formeln aus Γ wahr macht: für jedes Atom A enthält Γ die Formeln A sowie $\neg A$. Folglich ist für keine Belegung zu überprüfen, ob sie $\bot$ auf 1 abbildet.

5.0.11 Bemerkung. Für jede Formelmenge $\Gamma' \subseteq \Gamma$ ist es klar, dass aus der Gültigkeit von $\Gamma' \models F$ auch die von $\Gamma \models F$ folgt: jede Belegung, die alle Formeln aus Γ wahr macht, macht alle Formeln aus Γ' wahr und somit auch F.

Insofern werden logische Konsequenzen der Form $\Gamma \models F$ umso schwächer, je größer die Mengen Γ werden: je mehr Prämissen erfüllt werden müssen, desto weniger Belegungen können dies leisten.

Nachdem wir in Beispiel 5.0.10 explizit unendliche Mengen Γ betrachtet haben, wollen jetzt beweisen, dass die Gültigkeit einer logischen Konsequenz $\Gamma \models F$ immer schon mit einer endlichen Untermenge von Γ überprüft werden kann.

5.0.12 Kompaktheitssatz. *Gilt die logische Konsequenz $\Gamma \models F$, so existieren endlich viele Formeln $G_i \in \Gamma$, $i < n$, mit $G_0, \ldots, G_{n-1} \models F$.*

Beweis. Wir unterscheiden zwei Fälle:

0. Falls alle Formeln aus $\Gamma \cup \{F\}$ nur endlich viele, etwa k, Atome enthalten, gibt es nach Bemerkung 3.2.6 $n \leq 2^{2^k}$ Formeln G_i in Γ, so dass alle übrigen zu einer von diesen äquivalent sind. Dann gilt aber die logische Konsequenz $G_0, \ldots, G_{n-1} \models F$: jede Belegung, die $G_0, \ldots, G_{n-1}$ wahr macht, macht auch alle anderen Formeln in Γ wahr, nach Voraussetzung also auch F.

1. Ist hingegen die Menge aller Atome in den Formeln aus $\Gamma \cup \{F\}$ unendlich, etwa $\{ C_i \,:\, i \in \mathbb{N} \}$, beweisen wir die Contraposition: ist für jede endliche Teilmenge $\Gamma_0 \subseteq \Gamma$ die logische Konsequenz $\Gamma_0 \models F$ falsch, so ist auch $\Gamma \models F$ falsch.

   ObdA möge F genau die t Atome C_i, $i < t$, enthalten. Für jedes $k > t$ bezeichne Γ_k die Menge der Formeln in Γ, in denen höchstens die Atome C_i, $i < k$, vorkommen. Die Contraposition von Teil (0) zeigt, dass die logische Konsequenz $\Gamma_k \models F$ nicht gültig ist. Damit existiert eine Belegung α_k der Atome C_i, $i < k$, mit

   $$\widehat{\alpha}_k(F) = 0 \qquad \text{und} \qquad \widehat{\alpha}_k(G) = 1 \quad \text{für alle} \quad G \in \Gamma_k$$

   Wir konstruieren nun eine Belegung α aller Variablen C_i, $i \in \mathbb{N}$, die alle Formeln in Γ wahr macht, nicht aber F; folglich ist $\Gamma \models F$ falsch. Wir setzen erst

   $$\alpha(C_0) := \begin{cases} 1 & \text{falls } \ \alpha_k(C_0) = 1 \text{ für unendlich viele Indizes } k \\ 0 & \text{sonst} \end{cases}$$

   Jetzt entfernen wir alle Belegungen α_k mit $\alpha(C_0) \neq \alpha_k(C_0)$ aus unserer Liste von Belegungen. Nach Konstruktion verbleiben unendlich viele Belegungen α_k und wir setzen

   $$\alpha(C_1) := \begin{cases} 1 & \text{falls } \ \alpha_k(C_1) = 1 \text{ für unendlich viele der restlichen Indizes } k \\ 0 & \text{sonst} \end{cases}$$

   usw. Die resultierende Belegung α hat folgende Eigenschaft: für jede Zahl n stimmen die Werte $\alpha(C_0), \ldots, \alpha(C_{n-1})$ mit $\alpha_k(C_0), \ldots, \alpha_k(C_{n-1})$ für unendlich viele Indizes $k \in \mathbb{N}$ überein.

   Wähle ein verbliebenes $k \geq t$ mit $\alpha(C_i) = \alpha_k(C_i)$ für alle $i < t$. Dann gilt $\widehat{\alpha}(F) = \widehat{\alpha}_k(F)$. Aus der Konstruktion der Belegung α_k folgt aber $\widehat{\alpha}(F) = 0$.

   Analog wird jede Formel $G \in \Gamma$ von $\widehat{\alpha}$ auf 1 abgebildet, denn auch G enthält nur endlich viele Atome und es gibt deswegen einen Index l mit $\widehat{\alpha}(G) = \widehat{\alpha}_l(G) = 1$.

   Die Belegung α realisiert somit den erwünschte Widerspruch zur Gültigkeit der logischen Konsequenz $\Gamma \models F$. □

6. Resolutionsmethode der Aussagenlogik

Wie kann man in der Praxis die Gültigkeit von $\Gamma \models F$ feststellen? Alle für $\Gamma \cup \{F\}$ passenden Belegungen α zu bestimmen, die Γ erfüllen, erscheint sehr umständlich.

In diesem und im folgenden Kapitel werden syntaktische Methoden vorgestellt, die dieses Problem lösen können. Wenn die Prämissenmenge Γ aus endlich vielen Formeln in KNF, oder, was äquivalent dazu ist, aus endlich vielen Klauseln besteht, läßt sich die sogenannte Resolutionsmethode algorithmisch anwenden, wie dieses Kapitel zeigt. Für allgemeines Γ ist leider kein Algorithmus bekannt. Man wird dann nach einme syntaktischen *Beweis* dafür suchen, dass F aus Γ folgt. Das nächste Kapitel widmet sich daher der Beweistheorie.

Die Resolutionsmethode prüft algorithmisch die Erfüllbarkeit einer Formel in KNF. Ist Γ eine endliche Menge von Klauseln und G deren Konjunktion, dann ist $\Gamma \models F$ nach Satz 5.0.4 äquivalent zur Nichterfüllbarkeit von $G \wedge \neg F$, bzw. von $G \wedge \mathsf{KNF}(\neg F)$, was auch in KNF vorliegt.

Die Methode basiert auf den folgenden Beobachtungen:

6.0.1 Bemerkung. Gegeben ist eine Formel H in KNF.

(a) Falls H komplementäre Literale A und $\neg A$ als Klauseln enthält, ist H unerfüllbar, da für jede passende Belegung nicht beide Literale gleichzeitig wahr sein können.

(b) Falls F die Klauseln $K_0 \vee C$ und $K_1 \vee \neg C$ mit den komplementären Literalen C und $\neg C$ enthält, dürfen wir die Klausel $K_0 \vee K_1$ per Konjunktion zu H hinzufügen – ohne dass sich die Erfüllbarkeit ändert: ist H erfüllbar, etwa durch die Belegung α, so gilt $\widehat{\alpha}(K_0 \vee C) = 1 = \widehat{\alpha}(K_1 \vee \neg C)$, und je nach Wert von $\alpha(C)$ muß $\alpha(K_0)$ oder $\alpha(K_1)$ den Wert 1 annehmen, also auch $\widehat{\alpha}(K_0 \vee K_1)$. Da umgekehrt eine erfüllende Belegung jede Klausel einer KNF wahr machen muß, folgt aus der Erfüllbarkeit von $H \wedge (K_0 \vee K_1)$ auch die von H.

Iterierte Anwendung von (b) unter Berücksichtigung von (a) ermöglicht eine Entscheidung darüber, ob F erfüllbar ist. Einen entsprechenden Algorithmus entwickeln wir in diesem Abschnitt.

6.0.2 Notationelle Konvention (Mengentheoretische Schreibweise)**.**

0. Läßt man, etwa um Platz zu sparen, alle binären Junktoren einer KNF weg, so bleibt eine Liste von Listen von Literalen übrig; aus dieser ist die ursprüngliche KNF eindeutig rekonstruierbar.

Wegen der Idempotenz und Kommutativität von $\vee$ kann man die den Klauseln entsprechenden inneren Listen durch Mengen von Literalen ersetzen. Analog könnte man mit der äußeren Liste verfahren. Die Reihenfolge der Klauseln bzw. Literal-Mengen ist für die Erfüllbarkeit irrelevant, Wiederholungen ebenso.

1. In diesem Abschnitt verwenden wir daher alternativ Literal-Mengen statt Klauseln. Die leere Klausel $\bot$ wird dann mit $\varnothing$ bezeichnet. Das Vorkommen eines Literals X in einer Klausel K kann nunmehr mit Hilfe der Element-Relation $\in$ ausgedrückt werden: entweder gilt $X \in K$ oder $X \notin K$. Klauseln sind unter Vereinigung abgeschlossen.

   Die Konjunktion von Klauseln wird listennäher als Konkatenation (Verkettung) der entsprechenden Mengen von Literalen dargestellt. Z.B. läßt sich $\mathsf{KNF}(G_2)$ in Beispiel 4.2.11 wie folgt schreiben

$$\{A_0, A_1\}\,\{A_0, B_1\}\,\{B_0, A_1\}\,\{B_0, B_1\}$$

2. Falls die Klausel K das Literal X enthält, bezeichnet $K - X$ jene Klausel, die durch Entfernen von X aus K entsteht. Andernfalls ist $K - X := K$ undefiniert.

6.0.3 Beispiel. Für $K = \{A, \neg B\}$ gilt $K - A = \{\neg B\}$ und $K - \neg B = \{A\}$. Für $L = \{A, \neg B, C\}$ haben wir $L - C = \{A, \neg B\}$, aber $L - \neg A$ ist nicht definiert. Schließlich gilt $\{A\} - A = \varnothing$.

6.0.4 Definition. Sind K und K' Klauseln und ist X ein Literal mit $X \in K$ und $\neg X \in K'$, so nennen wir die Klausel

$$(K - X) \cup (K' - \neg X)$$

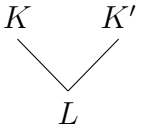
eine **Resolvente** von K und K'.

Ausführlicher: Sind zwei Klauseln $K := \{X_0, \ldots, X_{n-1}\}$ und $K' := \{X'_0, \ldots, X'_{m-1}\}$ gegeben mit $X_i = \neg X'_j$ für ein Paar von Indezes $i < n$ und $j < m$, dann ist die Klausel

$$\{X_0, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{n-1}, X'_0, \ldots, X'_{j-1}, X'_{j+1}, \ldots, X'_{m-1}\}$$

eine Resolvente von K und K'.

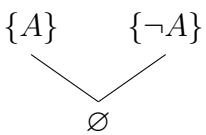
6.0.5 Notationelle Konvention. Ist L eine Resolvente von K und K', so stellen wir das graphisch wie folgt dar:



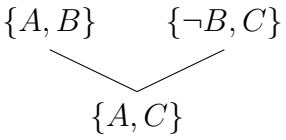
Aus solchen Bausteinen lassen sich **Resolventengraphen** zusammensetzen.

6.0.6 Beispiel.

- $\{A\}$ und $\{\neg A\}$ haben die Resolvente



- $\{A, B\}$ und $\{\neg B, C\}$ haben die Resolvente $\{A, C\}$, in Baumform:



- $\{A, B, \neg C\}$ und $\{A, \neg B, C\}$ haben zwei Resolventen

  die aufgrund der Tautologien und der absorbierenden Eigenschaft von $\top$ beide zu $\top$ äquivalent sind. Es wird sich in Kürze herausstellen, dass derartige Resolventen keinen Beitrag zur Lösung der Entscheidbarkeitsfrage liefern und daher ersatzlos gestrichen werden dürfen.

  Aufpassen! Die Resolventenbildung erfolgt immer nur bezogen auf **eine** atomare Aussage und das zugehörige Paar aus einem positiven und einem negativen Literal. Daher tritt $\{A\}$ hier nicht als Resolvente auf. Wollte man eine Resolvente bzgl. $B \vee \neg C$ bilden, brauchte man als Gegenstück die Negation $\neg(B \vee \neg C) \equiv \neg B \wedge C$, was als Konjunktion nicht Teil einer Klausel sein kann.

6.0.7 Satz (Resolutionslemma)**.** *Für zwei Klauseln K und K' mit komplementären Literalen $X \in K$ sowie $\neg X \in K'$ und Resolvente $L := (K - X) \cup (K' - \neg X)$ gilt für jede passende Belegung $\widehat{\alpha}(K \wedge K') \leq \widehat{\alpha}(L)$ und somit $\widehat{\alpha}(K \wedge K') = \widehat{\alpha}(K \wedge K' \wedge L)$, mit anderen Worten, $K \wedge K'$ ist zu $K \wedge K' \wedge L$ äquivalent.*

Beweis. Nach Konstruktion ist jede für $K \wedge K'$ passende Belegung auch passend für L. Wir unterscheiden zwei Fälle:

▷ Aus $\widehat{\alpha}(K \wedge K') = 1$ folgt nach Definition der Semantik $\widehat{\alpha}(K) = 1 = \widehat{\alpha}(K')$. Falls $\alpha(X) = 0$, dann impliziert $\widehat{\alpha}(K) = 1$ sofort $\widehat{\alpha}(K - X) = 1$, denn K muss ein von X verschiedenes wahres Literal enthalten. Analog folgt aus $\alpha(X) = 1$ sofort $\widehat{\alpha}(K' - \neg X) = 1$, denn $\neg X$ kann nicht das wahre Literal von K' sein. In beiden Fällen gilt:

$$\widehat{\alpha}(L) = \widehat{\alpha}(K - X) \vee \widehat{\alpha}(K' - \neg X) = 1 = \widehat{\alpha}(K \wedge K')$$

und folglich

$$\widehat{\alpha}(K \wedge K' \wedge L) = \widehat{\alpha}(K \wedge K') \wedge \widehat{\alpha}(L) = 1 \wedge 1 = 1$$

▷ Aus $\widehat{\alpha}(K \wedge K') = 0$ folgt $0 = \widehat{\alpha}(K \wedge K') \leq \widehat{\alpha}(L)$ und $\widehat{\alpha}(K \wedge K' \wedge L) = 0$. □

Wir wollen nun die Erfüllbarkeit einer Formel in KNF analysieren, indem wir sie unter Resolution „sättigen“, d.h., solange neue Resolventen bestimmen, bis entweder $\varnothing$ auftritt, oder keine weiteren Resolventen mehr gebildet werden können. Hierbei schreiben wir Formeln in KNF als Liste von Mengen von Literalen.

6.0.8 Definition. Für eine Formel H in KNF entsteht $\mathrm{Res}(H)$ durch Hinzunahme aller Resolventen zweier H-Klauseln, oder in herkömmlicher Notation als Konjunktion

$$\mathrm{Res}(H) := H \wedge \bigwedge \{\, L \,:\, L \not\equiv \top \text{ ist neue Resolvente zweier } H\text{-Klauseln} \,\}$$

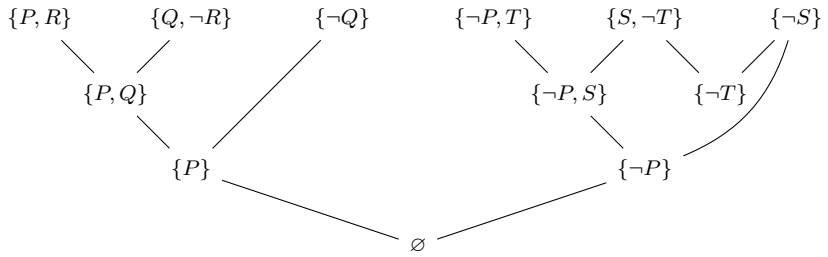
Zwecks Arbeitserleichterung werden dabei entstehende Klauseln, die Tautologien enthalten, wie in Beispiel 6.0.6(c) sofort durch $\top$ ersetzt, was wegen der Neutralität bzgl. $\wedge$ eliminiert werden kann.

6.0.9 Beispiel. Für $H = (P \vee R) \wedge (Q \vee \neg R) \wedge \neg Q \wedge (\neg P \vee T) \wedge \neg S \wedge (S \vee \neg T)$ wollen wir die Res-Funktion iterieren:

$$\begin{aligned}
H &= \{P, R\}\,\{Q, \neg R\}\,\{\neg Q\}\,\{\neg P, T\}\,\{\neg S\}\,\{S, \neg T\} \\
\mathrm{Res}(H) &= F\,\{P, Q\}\,\{\neg R\}\,\{R, T\}\,\{\neg P, S\}\,\{\neg T\} \\
\mathrm{Res}^2(H) &= \mathrm{Res}(F)\,\{R, S\}\,\{P\}\,\{Q, T\}\,\{\neg P\}\,\{Q, S\}\,\{T\}\,\{R\} \\
\mathrm{Res}^3(H) &= \mathrm{Res}^2(H)\,\{Q\}\,\{S\}\,\varnothing \\
\mathrm{Res}^4(H) &= \mathrm{Res}^3(H)
\end{aligned}$$

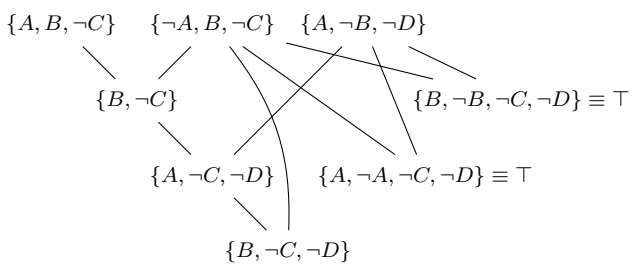
Die Iteration muß nach endlich vielen Schritten ein stabiles Ergebnis liefern, da die Anzahl k der verfügbaren Variablen die Größe der möglichen Klauseln ohne Wiederholungen beschränkt. Sobald $\varnothing$ als Resolvente auftritt, kann man das Verfahren abbrechen, denn aufgrund des Resolutionslemmas ist die ursprüngliche Formel ebenso wie die aktuelle Formel nicht erfüllbar.

Hier läßt sich ein partieller Resolventengraph erstellen, der zwar nicht alle Resolventen enthält, aber dafür $\varnothing$:



Nun stellt sich die Frage, ob eine Formel erfüllbar ist, wenn $\varnothing$ nicht bei der Iteration der Resolventen auftritt.

6.0.10 Beispiel. $H := (A \vee B \vee \neg C) \wedge (\neg A \vee B \vee \neg C) \wedge (A \vee \neg B \vee \neg D)$ liefert folgenden vollständigen Resolventengraphen:



In diesem Fall sind keine weiteren Resolventen konstruierbar. Zwar können wir aus den Resolventen keine Belegung ablesen, mit der H wahr wird. Trotzdem hoffen wir, dass aus der Abwesenheit von $\varnothing$ die Erfüllbarkeit von H folgt.

6.0.11 Satz. *Eine Formel $H \in \mathcal{F}^{KNF}$ ist genau dann erfüllbar wenn $\varnothing$ keine (iterierte) Resolvente von H ist.*

Beweis. Die Behauptung ist äquivalent dazu, dass H genau dann unerfüllbar ist, wenn $\varnothing$ eine (iterierte) Resolvente von H ist.

Tritt $\varnothing$ als Resolvente auf, so gilt nach dem Resolutionslemma $H \equiv H \wedge \bot \equiv \bot$. Somit ist H nicht erfüllbar.

Die umgekehrte Richtung beweisen wir mit vollständiger Induktion über die Anzahl k der verschiedenen Literale in H. Im Folgenden sei H also nicht erfüllbar.

Induktionsanfang $k = 0$: Dann ist $H = \bot = \emptyset$.

Induktionsvoraussetzung $k > 0$: Wir nehmen an, dass aus allen nicht erfüllbaren Formeln mit $< k$ verschiedenen Literalen die Resolvente $\varnothing$ ableitbar ist.

Induktionsschritt $k > 0$: OBdA können wir annehmen, dass H mit k verschiedenen Literalen keine tautologischen Klauseln enthält. Wähle ein $X \in \mathcal{P}$, das positiv und negativ in H auftritt. Wegen der Unerfüllbarkeit von H existiert solch ein X. Wir sortieren die Klauseln wie folgt in disjunkte Teilformeln, die jeweils weniger als k Literale haben:

$$H = H_X \wedge H_{\neg X} \wedge H_O \qquad \text{wobei}$$

H_X die Konjunktion aller Klauseln ist, die X enthalten;

$H_{\neg X}$ die Konjunktion aller Klauseln ist, die $\neg X$ enthalten;

H_O die Konjunktion aller Klauseln ist, die weder X noch $\neg X$ enthalten.

Ist eine Teilformel oder die Konjunktion zweier dieser Teilformeln nicht erfüllbar, so folgt die Behauptung.

Andernfalls haben $H_X \wedge H_O$ und $H_O \wedge H_{\neg X}$ verschiedene erfüllende Belegungen. Wir zeigen, dass diese Formeln dann die Resolventen $\{X\}$ bzw. $\{\neg X\}$ haben:

Bezeichnet $\widetilde{H}_X$ die Formel, die aus H_X entsteht, wenn jede Klausel K durch $K - X$ ersetzt und somit echt kleiner wird, so darf $\tilde{H}_X \wedge H_O$ nicht erfüllbar sein: aus jeder erfüllenden Belegung, die auf X undefiniert ist ließe sich dann eine F erfüllende Belegung γ mit $\gamma(X) = 0$ konstruieren. Damit hat $\widetilde{H}_X \wedge H_O$ nach Induktionsvoraussetzung $\varnothing$ als Resolvente. Deren Konstruktion muß wegen der Erfüllbarkeit von H_O in jedem Schritt Resolventen von $\widetilde{H}_X$ verwenden.

Spielt man diese Konstuktion mit der erfüllbaren Formel $H_X \wedge H_O$ nach, so erhält man die Resolvente $\{X\}$. Aus Symmetriegründen hat dann $H_O \wedge H_{\neg X}$ die Resolvente $\{\neg X\}$, woraus sich $\varnothing$ als Resolvente von H ergibt. □

Damit erhalten wir folgenden Algorithmus zur Bestimmung der Erfüllbarkeit von aussagenlogischen Formeln:

6.0.12 Algorithmus (naive Resolutionsmethode)**.**
Eingabe: Eine Formel H in KNF.
Ausgabe: Entscheidung über die Erfüllbarkeit von H.

Iteriere die Res-Funktion so lange, bis

- ▷ entweder die leere Resolvente $\varnothing$ entsteht, Ausgabe „H ist unerfüllbar“;
- ▷ oder die Res-Funktion sich stabilisiert hat, d.h., keine weiteren Resolvenen mehr gebildet werden können, und alle Resolventen nichtleer sind, Ausgabe „H ist erfüllbar.“

6.0.13 Beispiel. Familie Z will im kommenden Jahr eine Waschmaschine, ein Auto und ein Moped anschaffen. Aber falls Herr Z seinen üblichen Bonus nicht bekommt, können sie sich nicht alles leisten. Die Waschmaschine ist aber unverzichtbar. Zudem braucht die Familie mindestens ein Fortbewegungmittel. Wenn sie in den Urlaub fahren will, kann sie sich kein Auto leisten. Wenn sie nicht in den Urlaub fahren, muß sie aber ein Moped kaufen, um den Sohn zu besänftigen.

Zeigen Sie, dass Familie Z ein Moped und kein Auto anschafft, sofern Herr Z seinen üblichen Bonus nicht bekommt.

Die Übersetzung in atomare Aussagen und aussagenlogische Formeln in Form von Klauseln liefert:

- “Aber falls Herr Z seinen üblichen Bonus nicht bekommt, können sie sich nicht alles leisten.”
$$\neg B \Rightarrow \neg(W \wedge A \wedge M) \equiv \neg A \vee B \vee \neg M \vee W$$
- “Die Waschmaschine ist aber unverzichtbar.”
$$W$$
- “Die Familie braucht mindestens ein Fortbewegungsmittel.”
$$A \vee M$$
- “Wenn sie in den Urlaub fahren will, kann sie sich kein Auto leisten.”
$$U \Rightarrow \neg A \equiv \neg A \vee \neg U$$
- “Wenn sie nicht in den Urlaub fahren, muß sie aber ein Moped kaufen.”
$$\neg U \Rightarrow M \equiv M \vee U$$

Zusammen mit der Voraussetzung $\neg B$ erhalten wir folgende Konjunktion aller Prämissen:

$$G = \neg B \wedge (\neg A \vee B \vee \neg M \vee \neg W) \wedge W \wedge (A \vee M) \wedge (\neg A \vee \neg U) \wedge (M \vee U)$$

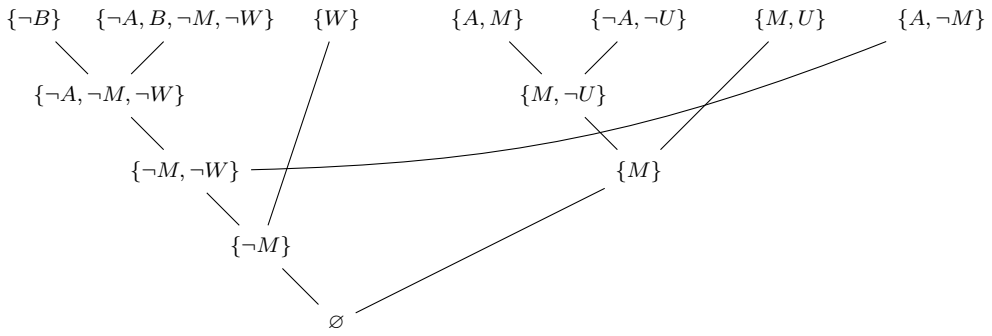
bzw. in Mengenschreibweise

$$G = \{\neg B\}\{\neg A, B, \neg M, \neg W\}\{W\}\{A, M\}\{\neg A, \neg U\}\{M, U\}$$

Wir fügen nun die negierte Schlußfolgerung $\neg F = \neg(\neg A \wedge M) \equiv A \vee \neg M$ als Klausel hinzu und versuchen, $\varnothing$ als Resolvente von

$$H = \{\neg B\}\{\neg A, B, \neg M, \neg W\}\{W\}\{A, M\}\{\neg A, \neg U\}\{M, U\}\{A, \neg M\}$$

zu finden. Das gelingt recht schnell, wenn auch unsystematisch, z.B.:



Sollte sich für eine Menge Γ von Klauseln die Vermutung $\Gamma \models F$ als falsch erweisen, so hätte man alle iterierten Resolventen von $H = \bigwedge \Gamma \wedge \mathsf{NNF}(\neg F)$ zu bestimmen. Das kann sehr aufwändig werden, da viele überflüssige Resolventen auftreten. Das folgende Ergebnis erlaubt es, diese frühzeitig zu eliminieren.

6.0.14 Proposition. *Entfernt man aus H in KNF alle Klauseln K, die eine andere H-Klausel echt umfassen, so ist die resultierene Formel J genau dann erfüllbar, wenn dies für H gilt. Weiterhin existiert zu jeder Klausel $M \in \mathrm{Res}(H)$ eine Klausel $M' \in \mathrm{Res}(J)$ mit $M' \subseteq M$.*

Beweis. Paßt α für H, so folgt aus $\widehat{\alpha}(J) = 1$ auch $\widehat{\alpha}(H) = 1$, denn jede H-Klausel K, die in J fehlt, umfaßt mindestens eine J-Klausel K'. Nach Voraussetzung gilt bereits $\widehat{\alpha}(K') = 1$, also auch $\widehat{\alpha}(K) = 1$..

Wird umgekehrt jede H-Klausel von α erfüllt, so auch jede J-Klausel.

Wegen der Transitivität von $\supseteq$ ist die letzte Behauptung für jede Klausel aus H klar. Falls $M = (K - A) \cup (L - \neg A) \in \mathrm{Res}(H)$ für Klauseln K und L von H und ein Literal A, existieren also Klauseln K' und L' in J mit $K' \subseteq K$ und $J' \subseteq J$.

- Bei Gleichheit in beiden Fällen folgt $M \in \mathrm{Res}(J)$.
- Andernfalls, sofern $A \in K'$ und $\neg A \in J'$, läßt sich die Resolvente $M' = (K' - A) \cup (J' - \neg A)$ dieser G-Klauseln bilden und ist nach Voraussetzung in M enthalten.
- Und wenn $A \notin K'$ oder $\neg A \notin J'$ ist zumindest K' bzw. J' in M enthalten. □

6.0.15 Definition. $H \in \mathcal{F}^{KNF}$ heißt **rein**, wenn

1. In jeder Klausel von H kein Literal mehrfach auftritt;
2. H keine tautologische Klauseln enthält;
3. H keine Klausel Obermenge einer anderen Klausel ist.

Rein(H) entsteht aus H duch Eliminierung der obigen Probleme.

6.0.16 Folgerung. *$H \in \mathcal{F}^{KNF}$ ist genau dann erfüllbar, wenn* Rein(H) *erfüllbar ist.* □

Nun vereinfacht sich obiger Resolutionsalgorithmus 6.0.12, wenn man nach dem Bilden aller neuen Resolventen bzgl. einer Variablen das Ergebnis reinigt.

6.0.17 Algorithmus (praktikable Resolutionsmethode)**.**
Eingabe: Eine reine Formel H in KNF.
Ausgabe: Entscheidung über die Erfüllbarkeit von H.
Iteriere folgende Schritte, so lange wie möglich:

▷ Für eine Variable X, die positiv und negativ in H vorkommt, zerlege $H = H_X \wedge H_{\neg X} \wedge H_O$ wie oben und füge alle Resolventen hinzu, die aus Klauseln in H_X bzw. $H_{\neg X}$ bzgl. X und $\neg X$ gebildet werden können;

▷ reinige das Ergebnis.

Die Auswahl der Variablen X im jeweiligen Schritt kann mehr oder weniger geschickt erfolgen. Das Erzeugen neuer sollte Vorrang vor dem Reproduzieren alter Resolventen haben.

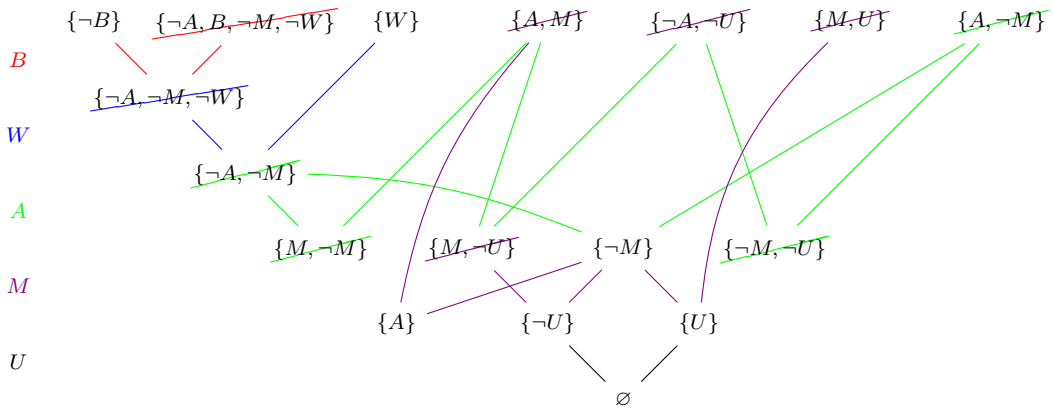
6.0.18 Beispiel. Ausgehend von

$$H = \{\neg B\}\{\neg A, B, \neg M, \neg W\}\{W\}\{A, M\}\{\neg A, \neg U\}\{M, U\}\{A, \neg M\}$$

aus Beispiel 6.0.13 berechnen wir die neuen Resolventen

- bzgl. B, was $\{\neg A, \neg M, \neg W\}$ liefert und erlaubt, $\{\neg A, B, \neg M, \neg W\}$ zu eliminieren;
- bzgl. W, was $\{\neg A, \neg M\}$ liefert und erlaubt $\{\neg A, \neg M, \neg W\}$ zu eliminieren;
- bzgl. A, was $\{M, \neg M\}$, $\{\neg M\}$, $\{M, \neg U\}$ und $\{\neg A, M\}$ liefert und erlaubt $\{M, \neg M\}$, $\{A, \neg M\}$, $\{\neg A, \neg M\}$ und $\{\neg M, \neg U\}$ zu eliminieren;
- bzgl. M, was $\{A\}$, $\{U\}$ und $\{\neg U\}$ liefert und erlaubt $\{A, M\}$, $\{\neg A, \neg U\}$, $\{M, \neg U\}$ und $\{\neg M, \neg U\}$ zu eliminieren;
- bzgl. U, was $\varnothing$ liefert (und erlaubt, alle übrigen Klauseln zu eliminieren).

Die farbliche Codierung im Graphen soll nur den zeitlichen Ablauf der einzelnen Schritte nachvollziehbar machen.



Somit ist H nicht erfüllbar.

Bei Formeln der Größe $O(n)$ kann bereits die Umwandlung in KNF eine Zeitkomplexität von $O(2^n)$ aufweisen. Aber selbst bei $\mathcal{F}^{KNF}$-Formeln kann die Resolutionsmethode exponentiell viele Schritte benötigen um minimale Resolventen zu liefern:

6.0.19 Proposition. *Die Anwendung der vereinfachten Resolutionsmethode kann eine Formel in KNF mit n Klauseln in eine Formel mit $2^n - 1$ minimalen Resolventen verwandeln.*

Beweis. Wir konstruieren induktiv Formeln F_k in KNF, in denen höchstens $2k$ Variablen vorkommen und die k Klauseln und $2^k - 1$ minimale Resolventen haben.

$k = 0$: $F_0 = \bot$ bzw. $\varnothing$. Die Resolutionsmethode liefert $2^0 - 1 = 0$ minimale Resolventen.

Annahme: Die Konjunktion $F_k = \bigwedge_{i<k} K_{k,i}$ aus k Klauseln $K_{k,i}$, $i < k$, hat $2^k - 1$ minimale Resolventen.

$k + 1$: Definiere F_{k+1} als $\mathsf{KNF}(F_k \vee A_{2k}) \wedge (\neg A_{2k} \vee A_{2k-1})$, was im linken Teil nur die Anwendung des Distributivgesetzes erfordert.

Die Resolventenbildung für F_{k+1} ohne Beteiligung der letzten Klausel liefert durch Hinzufügen von A_{2k} zu allen $2^k - 1$ minimalen Resolventen von F_k wieder $2^k - 1$ Resolventen von F_{k+1}. Aus diesen lassen sich unter Verwendung der letzten Klausel weitere $2^k - 1$ Resolventen konstruieren, durch Hinzufügen von A_{2k-1} zu allen minimalen Resolventen von F_k. Die Minimalität des so konstruierten Resolventen folgt aus $A_{2k} \neq A_{2k=1}$. Zusammen mit der ebenfalls minimalen letzten Klausel von F_{k+1} erhalten wir somit $2 \cdot \left(2^k - 1\right) + 1 = 2^{k+1} - 1$ viele minimale Resolventen. □

In Kapitel 8 zeigen wir, dass die Resolutionsmethode zumindest für eine bestimmte Klasse von Formeln, die sogenannten „Hornformeln“, effizient ist.

7. Beweistheorie

Die Resolutionsmethode war ein erster Schritt, die logische Konsequenz $\models$ auf syntaktischem Wege zu überprüfen; dabei waren die Prämissen auf KNFs, also letztlich Klauseln beschränkt. Im allgemeinen Fall orientieren wir uns an, *Beweisen, wie sie in der Mathematik üblich sind.* Auch diese versuchen *Schlussfolgerungen* über die Wahrheit von Formeln aus *Annahmen* über die Wahrheit bestimmter Prämissen zu ziehen.

Aber Vorsicht: typische mathematische Beweise, wie man sie in Fachaufsätzen, Lehrbüchern oder Vorträgen findet, etwa der obige Beweis des Kompaktheitssatzes, wirken häufig informell und sind mehr oder weniger verkürzt, gemäß den Vorkenntnissen der intendieren Leser/Hörer, die die fehlenden oder nur angedeuteten Schritte ergänzen können sollten. Dennoch folgen solche Beweise bestimmten Regeln, auf die sich die (meisten) Mathematiker verständigt haben.

Es gilt also zunächst, die typischen Schlußregeln solcher Beweise zu extrahieren, bevor man sie formalisieren kann, denn erst formale Beweise sind selber mathematischen Methoden zugänglich.

Diese Schlußregeln werden eine ähnliche Form wie die logischen Konsequenz aus Kapitel 5 annehmen: $G_0, \ldots, G_{n-1} \vdash F$ soll bedeuten, dass die Formel F aus endlich vielen Prämissen herleitbar ist. Instanzen dieser syntaktischen (!) Herleitungsrelation $\vdash$ heißen **Sequenzen**; die Manipulation solcher Sequenzen fällt, ähnlich wie die Konstuktion von Resolventen, unter unter den anfangs erwähnten Begriff des *logischen Kalküls*.

In jedem Fall stellen sich unmittelbar zwei entscheidende Fragen:

- ▷ Ist alles wahr, was syntaktisch hergeleitet (und somit formal bewiesen) werden kann? Dann nennt man das Kalkül **korrekt**.
- ▷ Kann alles, was wahr ist, syntaktisch hergeleitet werden? Im positiven Fall nennt man das Kalkül **vollständig**.

Solange wir die Antworten auf diese Fragen nicht kennen, müssen wir zwischen logischer Konsequenzen $\models$ (vergl. Kapitel 5) und syntaktischer Herleitbarkeit $\vdash$ unterscheiden. Damit lassen sich die Fragen nach der Korrektheit bzw. Vollständigkeit als Inklusionen dieser Relationen formulieren:

$$\text{Korrektheit:} \quad \vdash \subseteq \models \qquad \text{bzw.} \qquad \text{Vollständigkeit} \quad \models \subseteq \vdash$$

Wenn beide Fragen positiv beantwortet worden sind, also logische Konsequenz $\models$ mit syntaktischer Herleitbarkeit $\vdash$ übereinstimmt, könnte man im Prinzip diese Unterscheidung wieder vergessen.

Im Rahmen der Prädikatenlogik (vergl. Teil II) suchte man seit der zweiten Hälfte des 19. Jahrhundersts geeignete Formalisierungen für $\vdash$. Ursprünglich verfolgte man dabei die *axiomatische Methode*, bei der man für das Gebiet von Interesse gewisse „offensichtliche“ oder „unwiderlegbare“ Grundannahmen festlegte, *Axiome* genannt, um aus diesen soviele korrekte Aussagen über dieses Gebiet herzuleiten, wie möglich [Hil1899]. Bekannteste Beispiele dürften die Peano-Axiome für die natürlichen Zahlen von 1889 sein, und der axiomatische Zugang zur Mengenlehre zunächst durch Zermelo (1905–1908) und später durch Fraenkel und unabhängig davon durch Skolem (jeweils 1922). Hilberts Programm sah die vollständige Formalisierung der Mathematik vor. Mit der axiomatischen Methode verbinden sich neben dem Namen von Hilbert auch die von Frege und Łukasiewizc (dem Erfinder der „polnischen Notation“).

Leider zeigten es sich zu Beginn des 20. Jahrhunderts Schwierigkeiten, Hilberts Programm umzusetzen, beginnend mit der Entdeckung des Russel'schen Paradoxons (Zermelo 1900, unveröffentlicht; Russel 1901). 1930 konnte Kurt Gödel schließlich zeigen, dass die von Hilbert angestrebte Formalisierung im Rahmen jedes hinreichend ausdrucksstarken Systems prinzipiell nicht erreichbar ist. (Das betrifft zwar nicht die Aussagenlogik, aber die nachfolgend zu behandelnde Prädikatenlogik.) Zudem modellierte die axiomatische Methode nicht wirklich die typische Arbeitsweise von Mathematikern. Sie war so umständlich, dass selbst ihre Protagonisten sie nicht konsequent verwendeten. Im Jahr 1934 erschienen unabhängig voneinander gleich zwei Vorschläge für ein System, das die tatsächliche Arbeitsweise von Mathematikern besser wiederspiegeln sollte: die sogenannte **natürliche Deduktion**: von Stanisław Jaśkowski [Jaś34] (mit Vorarbeiten seit mindestens 1929) und Gerhard Gentzen [Gen34] (basierend auf seiner Dissertation von 1933).

7.1 Natürliche Deduktion (ND)

Die englische Wikipedia-Seite bietet einen einen ordentlichen Überblick über die Facetten dieses interessanten Themas und weiterführende Hinweise.

Anstatt alle mahtematischen Sätze mit wenigen Schlußregeln aus Axiomen herleiten zu wollen, werden bei der natürlichen Deduktion typischerweise mathematische Sätze unter bestimmten Voraussetzungen oder globalen Prämissen formuliert, die ggf. wieder entfernt und in die Schlußfolgerung integriert werden können. Dazu kommen eine Reihe von Schlußregeln, deren lokale Prämissen nicht nur Formeln, sondern in manchen Fällen auch ganze Herleitungen mit bestimmten vorgegebenen globalen Prämissen sein können, es werden aber keine Axiome benötigt.

Die natürliche Deduktion (kurz ND) war zunächst für Beweise in der intuitionistischen Logik konzipiert und ermöglicht dort die recht effiziente Konstruktion von Beweisen. Dabei hilft, dass sich alle Teilformeln, die im Verlauf des Beweises benötigt werden, auch in der Schlußfolgerung auftreten müssen, das schränkt die Suche nach einem Beweis stark ein. Durch Hinzufügen einer einzigen Schlußregel kann die ND auf die klassische Logik erweitert werden. Nicht alle klassisch gültigen Formeln gelten auch intuitionistisch, und deren klassische Beweise enthalten typischerweise Teilformeln, die nicht notwendig in der Schlußfolgerung auftreten, weshalb solche Beweise bei Intuitionisten als „artifiziell“gelten. Dies erschwert zudem die Suche nach klassischen Beweisen, siehe etwa [D'A05].

Die Präsentation von Beweisen in natürlicher Deduktion kann im Wesentlichen auf zwei Weisen erfolgen: mittels einer Baumstruktur, mit zusätzlichen Markierungen von Teil-Herleitungen, die als lokale Prämissen dienen (was auf Gentzen zurückgeht), oder in tabellarischer Form mit Annotationen, aus welchen früheren Zeilen die aktuelle Formel aufgrund einer Schlußregel hergeleitet wurde und welchen Gültigkeitsbereich (Skope) bestimmte Teilherleitungen haben (was auf Jaśkowski zurückgeht, dessen ursprüngliches Verfahren allerdings auch diagrammatischer Natur war). Beide Varianten der Darstellung sind im Laufe der Zeit von einer Reihe von Autoren optimiert und verfeinert worden, wobei die Baumdarstellung eher für die Erforschung des Kalküls nützlich ist, und die tabellarische Darstellung anwendungsorientierer zu sein scheint, weshalb wir uns auf diese konzentrieren.

Ab der Mitte des vergangenen Jahrhunderts war die natürliche Deduktion im anglo-amerikanischen Raum die Methode der Wahl, die in Lehrbüchern für Logik sehr weitgehende Verbreitung erfuhr, und damit Generationen von Logik-Nutzern (Philosophen wie Mathematikern und Informatikern) prägte. In Europa hielt sich Hilberts axiomatische Methode länger[1].

In der Fortsetzung [Gen35] des oben genannten Artikels stellte Gentzen auch noch eine zweite Beweismethode vor, den sogenannten **Sequenzenkalkül**. Ursprünglich als technisches Hilfmittel gedacht, hat sich dieser seit seiner modernen Präsentation durch Kleene [Kle52] zu einem ernstzunehmenden Konkurrenten der natürlichen Deduktion entwickelt, speziell im Bereich der klassischen Logik, siehe etwa [Cal11].

Folgende Aspekte finden sich in praktisch allen Spielarten der natürlichen Deduktion, auch wenn sie dieselbe nicht charakterisieren:

- ▷ Kaum Einschränkung auf bestimmte Junktoren oder Normalformeln.
- ▷ Die Regeln (inference rules) der Natürlichen Deduktion treten häufig paarweise auf, zur Einführung und zur Eliminierung von Junktoren. Erstere dienen dazu, Schlussfolgerungen aus einfacheren Voraussetzungen ziehen zu können, letztere zur Auflösung von Voraussetzungen in ihre Bestandteile. Allerdings erfolgt der "Informationsfluß" bei diesen Regeln in entgegengesetzter Richtung, was einer Automatisierung des Verfahren entgegensteht.
- ▷ Die Regeln sind „natürlich“ oder „intuitiv“ richtig, bilden also Denk- und Argumentationsmuster ab, die beim Verwendung natürlicher Sprache typisch sind. (Das ist natürlich hochgradig subjektiv.)

In der hier präsentierten Variante der Natürlichen Deduktion für die Aussagenlogik finden die Junktoren $\wedge$, $\vee$, $\neg$, $\Rightarrow$ und $\bot$ Verwendung. Nur $\Leftrightarrow$ wollen wir wie gewohnt durch $(A \Rightarrow B) \wedge (B \Rightarrow A)$ ersetzen. Die natürlichen Deduktionsregeln haben die Form

$$P_0 \vdash Q \qquad \text{oder} \qquad P_0, P_1 \vdash Q \qquad \text{oder} \qquad P_0, P_1, P_2 \vdash Q$$

wobei P_i die Prämissen der Regel sind und Q die Folgerung. Alternativ sind vertikale, spezifisch für eine Baum-Darstellung von Beweisen geeignete Varianten dieser Regeln verbreitet:

$$\frac{P_0}{Q} \qquad \text{oder} \qquad \frac{P_0 \quad P_1}{Q} \qquad \text{oder} \qquad \frac{P_0 \quad P_1 \quad P_2}{Q}$$

[1] Z.B. in Herrn Professor Pfeiffers Logik Vorlesung Ende der 1970'er Jahre in Hannover.

Interpretation: Aus jeder Obermenge Γ der Prämissen läßt sich die Folgerung herleiten. In graphentheoretischer Hinsicht sollte man die „Bruchstriche“ als Knoten mit durch Prämissen gelabelten Input-Kanten und durch die Schlußfolgerung gelabelter Output-Kante interpretieren. (Das oben kurz erwähnte Sequenzen-Kalkül verwendet zwar ebenfalls „Bruchstriche“, aber dort wird spezifiziert, wie eine oder mehrere Sequenzen im „Zähler“ in eine neue Sequenz im „Nenner“ transformiert werden dürfen. Die Sequenzen selber können auch hier diagrammatisch interpretiert werden.)

7.2 ND-Regeln für die Konjunktion

Gehören die Prämissen G und H zu Γ, dann läßt sich ihre Konjunktion aus Γ herleiten, geschrieben $\Gamma \vdash G \wedge H$. Die zugehörige Regel formulieren wir für die Minimalversion von Γ

$$G, H \vdash G \wedge H$$

Diese Regel, nennen wir **Introduktion der Konjunktion** oder kürzer $(\wedge i)$. Sie wird typischerweise verwendet, um ein gewünschtes (Zwischen-)Ergebnis aus einfacheren Bausteinen zusammenzusetzen.

Wir benötigen jedoch auch eine Regel zur **Elimination der Konjunktion**, kurz $(\wedge e)$, etwa um eine komplizierte Voraussetzung in ihre Bestandteile zu zerlegen. Da unsere Sequenzen rechts des Relationszeichens nur eine Formel aufweisen, brauchen wir diese Regel in zwei Varianten. Zusammengefasst:

7.2.1 Definition. Die Regeln der Konjunktion lauten

$$G, H \vdash G \wedge H \ (\wedge i) \qquad \text{und} \qquad G \wedge H \vdash G \ (\wedge e) \quad \text{sowie} \quad G \wedge H \vdash H \ (\wedge e)$$

oder in Baum-Form

$$\frac{G \quad H}{G \wedge H}\ (\wedge i) \qquad \text{und} \qquad \frac{G \wedge H}{G}\ (\wedge e) \qquad \text{sowie} \qquad \frac{G \wedge H}{H}\ (\wedge e)$$

Mit diesen ersten Regeln der natürlichen Deduktion lässt sich bereits etwas beweisen.

7.2.2 Beispiel. Wir demonstrieren das Verfahren, indem wir Assoziativität der Konjunktion nachweisen. Konkret wollen wir die Gültigkeit der syntaktischen Sequenz $F \wedge (G \wedge H) \vdash (F \wedge G) \wedge H$ zunächst tabellarisch zeigen:

Beweis.

1	$F \wedge (G \wedge H)$	Prämisse der Sequenz
2	F	$(\wedge e)$, Zeile 1
3	$G \wedge H$	$(\wedge e)$, Zeile 1
4	G	$(\wedge e)$, Zeile 3
5	H	$(\wedge e)$, Zeile 3
6	$F \wedge G$	$(\wedge i)$, Zeilen 2 und 4
7	$(F \wedge G) \wedge H$	$(\wedge i)$, Zeilen 6 und 5

Der eigentliche Beweis steht links in nummerierten Zeilen, während die Kommentare auf der rechen Seite die angewendete Regel angeben und die Zeilen, in denen deren Voraussetzungen zu finden sind. Im Folgenden lassen wir das Wort „Zeile“ jedoch weg.

Und nun alternativ als Baum:

$$\dfrac{\dfrac{\dfrac{F \wedge (G \wedge H)}{F}\,(\wedge e) \qquad \dfrac{\dfrac{F \wedge (G \wedge H)}{G \wedge H}\,(\wedge e)}{G}\,(\wedge e)}{F \wedge G}\,(\wedge i) \qquad \dfrac{\dfrac{F \wedge (G \wedge H)}{G \wedge H}\,(\wedge e)}{H}\,(\wedge e)}{(F \wedge G) \wedge H}\,(\wedge i)$$

Man beachte, dass dieselbe Prämisse mehrfach verwendet werden kann, was in der Baumstruktur deutlicher zum Ausdruck kommt. Bäume werden zudem gern von unten nach oben entwickelt, da man die Anzahl der benötigten Zweige nicht vorher kennt.

7.2.3 Beispiel. Als weiteres Beispiel soll die Kommutativität von $\wedge$ bewiesen werden:

$$H \wedge G \vdash G \wedge H$$

Beweis.

1	$H \wedge G$	Prämisse
2	H	$(\wedge e), 1$
3	G	$(\wedge e), 1$
4	$G \wedge H$	$(\wedge i), 3, 2$

Alternativ als Baum:

$$\dfrac{\dfrac{H \wedge G}{G}\,(\wedge e) \qquad \dfrac{G \wedge H}{H}\,(\wedge e)}{G \wedge H}\,(\wedge i)$$

7.3 ND-Regeln der Implikation

Auch für die Implikation gibt es zwei Regeln. Die Elimination, klassisch bekannt als *modus ponens*, erlaubt uns aus den Voraussetzungen G und $G \Rightarrow H$ die Schlussfolgerung H zu ziehen. Umgekehrt ermöglicht es die Introduktionsregel, aus einem Beweis von H spezifisch unter der Voraussetzung G, und evtl. unter weiteren vorher eingeführten Voraussetzungen, die Herleitbarkeit von $G \Rightarrow H$ ohne die spezifische Voraussetzung G zu folgern. Während bisher die Prämissen direkt zu verwendet waren, handelt es sich hier um eine Regel mit „Fernwirkung“.

Ein Beispiel soll dies illustrieren: in Beispiel 7.2.3 haben wir einen Beweis in mehreren Schritten präsentiert, dass aus $H \wedge G$ die Formel $G \wedge H$ herleitbar ist, also $H \wedge G \vdash G \wedge H$. Dies erlaubt uns, die Voraussetzung in die Schlussfolgerung einzubeziehen, ohne die Zwischenschritte:

$$\vdash H \wedge G \Rightarrow G \wedge H$$

Dies als Regel zu formulieren erfordert eine neuartige Notation.

7.3.1 Definition. Die Regeln der Implikation lauten:

$$G, G \Rightarrow H \vdash H \ (\Rightarrow e) \qquad \text{sowie} \qquad [G] \ldots H \vdash G \Rightarrow H \ (\Rightarrow i)$$

oder in Baumform

$$\frac{G \quad G \Rightarrow H}{H}\,(\Rightarrow e) \qquad \text{sowie} \qquad \frac{\begin{array}{c}[G]\\ \vdots \\ H\end{array}}{G \Rightarrow H}\,(\Rightarrow i)$$

Bei der $\Rightarrow$-Introduktion steht der Klammer-Ausdruck links bzw. oben immer für eine ausgewählte Prämisse, hier G, eines formalen Beweis-Teils, dessen Ergebnis, hier H, nach den Punkten folgt. Dieser Beweis-Teil darf auch vorangegangene Ergebnisse von außerhalb verwenden. Aber nachdem die $\Rightarrow$-Introduktion $G \Rightarrow H$ ausgeführt worden ist, sind die „lokalen Variablen“ dieses derart markierten Beweis-Teils nicht mehr zugänglich! Bei zeilenweiser Schreibweise wird der durch Punkte angedeutete Gültigkeitsbereich (scope) der ausgewählten Prämisse in einen Kasten eingeschlossen. Das rechtfertigt den Namen *Kasten-Prämisse*; sie bedarf keiner externen Begründung. Achtung: es ist nicht nötig, dass die Kasten-Prämisse G tatsächlich bei der Herleitung von H verwendet wird. In der Baumdarstellung kann das zu Irritationen führen!

7.3.2 Beispiel. Wir zeigen die Gültigkeit von $F \Rightarrow G, F \Rightarrow H \vdash F \Rightarrow G \wedge H$:

1	$F \Rightarrow G$	Prämisse
2	$F \Rightarrow H$	Prämisse
3	F	**Kasten-Prämisse**
4	G	$(\Rightarrow e), 1, 3$
5	H	$(\Rightarrow e), 2, 3$
6	$G \wedge H$	$(\wedge i), 4, 5$
7	$F \Rightarrow\ G \wedge H$	$(\Rightarrow i), 3-6$

Man beachte, dass in der letzten Zeile neben $(\Rightarrow i)$ alle Zeilen angegeben werden, über die sich der relevante Kasten erstreckt.

In Baumform läßt sich der Kasten schlecht realisieren:

$$\frac{\dfrac{\dfrac{[F] \quad F \Rightarrow G}{G}\,(\Rightarrow e) \quad \dfrac{[F] \quad F \Rightarrow H}{H}\,(\Rightarrow e)}{G \wedge H}\,(\wedge i)}{F \Rightarrow G \wedge H}\,(\Rightarrow i)$$

Dafür erkennt man deutlicher, dass die Kasten-Prämisse (in eckigen Klammern) mehrfach verwendet wird.

Hinweis:

- ▷ Keine Zeile, die in einem Kasten liegt, darf außerhalb des Kastens (als Begründung) benutzt werden. In der Baum-Darstellung gilt dies für die Zweige von der relevanten Introduktionsregel bis zu den durch eckige Klammern markierten Auftreten der Kastenprämisse.

▷ Kästen dürfen hierarchisch verschachtelt werden. Innerhalb eines Kastens darf ein anderer eröffnet werden, dieser muss allerdings vor dem äußeren wieder geschlossen werden. D.h. die Linien der Kästen dürfen sich nicht kreuzen. Das ist in Baumform nur schwer darstellbar, wir werden es mit Farbe versuchen.

7.3.3 Beispiel.

$$F \wedge G \Rightarrow H \vdash F \Rightarrow (G \Rightarrow H)$$

Beweis. Aufgrund der Form der rechten Seite erwarten wir zwei ineinandergeschachtelte Boxen, eine innere mit Prämisse G und letzter Zeile H, also der Schlußfolgerung $G \Rightarrow H$, und eine mit Prämisse F und letzter Zeile $G \Rightarrow H$, also der Schlußfolgerung $F \Rightarrow (G \Rightarrow H)$:

1	$F \wedge G \Rightarrow H$	Prämisse
2	F	Kasten-Prämisse
3	G	Kasten-Prämisse
4	$F \wedge G$	$(\wedge i), 2, 3$
5	H	$(\Rightarrow e), 1, 4$
6	$G \Rightarrow H$	$(\Rightarrow i), 3 - 5$
7	$F \Rightarrow (G \Rightarrow H)$	$(\Rightarrow i), 2 - 6$

und als Baum:

$$\dfrac{\dfrac{\dfrac{[F] \quad [G]}{F \wedge G}\,(\wedge i) \qquad F \wedge G \Rightarrow H}{H}\,(\Rightarrow e)}{G \Rightarrow H}\,(\Rightarrow i)}{F \Rightarrow (G \Rightarrow H)}\,(\Rightarrow i)$$

7.3.4 Beispiel. Wir wollen die Gültigkeit der Sequenz $F \Rightarrow G \wedge H \vdash (H \Rightarrow G) \Rightarrow F$ beweisen:

1	$F \Rightarrow G \wedge H$	Prämisse
2	$H \Rightarrow G$	Kasten-Prämisse
3	F	Kasten-Prämisse
4	$G \wedge H$	$(\Rightarrow e), 1, 3$
5	G	$(\wedge e), 4$
6	H	$(\wedge e), 4$
7	???	
8		
9	F	
10	$(H \Rightarrow G) \Rightarrow F$	

Wie kann man von $H \Rightarrow G$ zu F gelangen? Das scheint nicht zu funktionieren. In der Tat gilt die logische Konsequenz $F \Rightarrow G \wedge H \models (H \Rightarrow G) \Rightarrow F$ nicht. Dies zeigt

die Belegung $\alpha(F) = 0, \alpha(G) = 1 = \alpha(H)$, für die $\widehat{\alpha}(F \Rightarrow G \wedge H) = 1$ gilt, aber auch $\widehat{\alpha}(H \Rightarrow G) = 0$. Da sich unser Kalkül als korrekt herausstellen wird, kann man keine ungültigen Sequenzen beweisen. Bei Schwierigkeiten beim Beweis sollte man also durchaus überprüfen, ob die zu beweisende Sequenz evtl. ungültig ist.

7.4 Präzisierung des Tableaux-Formalismusses

Auch wenn noch nicht alle Schlußregeln eingeführt wurden, wollen wir die bisher informell verwendete Tableaux-Methode genauer fassen.

Formale Beweise der natürlichen Deduktion, die die Gültigkeit einer Sequenz nachweisen, operieren nach so strengen formalen Kriterien, dass im Prinzip ein Rechner einen gegebenen Beweis leicht überprüfen kann. In dieser Hinsicht ist die Baum-Notation klar im Nachteil gegenüber der tabellarischen Notation. Das umgekehrte Problem, Beweise gemäß natürlicher Deduktion automatisiert erstellen zu lassen, ist zumindest in der klassischen Logik viel schwieriger.

7.4.1 Definition. Ein **formaler Beweis** der Sequenz $\Gamma \vdash F$ in **Tableaux-Form** ist eine durchnummerierte endliche Liste von Formeln, wobei F in der letzten Zeile steht. Die Liste wird durch hierarchische organisierte Kästen strukturiert, so dass mindestens die letzte Zeile zu keinem Kasten gehört. Die einzelnen Formeln

- sind entweder als Prämissen in Γ markiert und liegen dann außerhalb aller Kästen;
- oder erscheinen frei wählbar als erste Zeile eines Kastens und sind als **Kasten-Prämisse** markiert,
- oder tragen eine Begründung, wie diese Formel aus bestimmten Formeln in vorangegangenen Zeilen oder ganzen Kästen (beschrieben durch einen Bereich aufeinanderfolgender Zeilen) durch Anwendung einer spezifischen Regel folgt.

  Dabei muß jeder Kasten, der eine Zeile bzw. einen Kasten aus der Begründung von Zeile i in der Hierarchie umfaßt, auch Zeile i umfassen.

Die präzise Spezifikation eines formalen Beweises in Baumform wäre hinsichtlich der Regeln mit Fernwirkung auch sehr mühsam. Andererseits spiegeln Bäume die innere Struktur eines Beweies recht gut wider, und man sieht besser, ob gewisse Prämissen mehrfach verwendet werden. Insofern kann die Baumform zumindest am Anfang beim Auffinden von Beweisen in Listenform nützlich sein. Für theoretische Überlegungen über die Beweistheorie ist sie ohnehin vorzuziehen.

7.5 Regeln der Disjunktion

Im Gegensatz zur Implikation ist die Introduktion für die Disjunktion klar, dafür ist die Elimination etwas schwieriger. Wenn G oder H zu den Voraussetzung gehört, soll $G \vee H$ herleitbar sein. Das ergibt zwei Varianten der Introduktionsregel.

Bei der Elimination nehmen wir an, dass $G \vee H$ zur Menge der Prämissen gehört. Welche Formeln K sollen dann aus Γ herleitbar sein? Dazu reicht es, einen Beweis von K mit

(Kasten-)Prämisse G zu besitzen, sowie einen zweiten mit (Kasten-)Prämisse H; diese Prämissen bedürfen jeweils keiner Begründung. Analog zur Introduktion der Implikation liefert dies eine Regel mit Fernwirkung. diesmal sogar in zweifacher Hinsicht:

7.5.1 Definition. Die Regeln der Disjunktion sind:

$$G \vdash G \vee H \ (\vee i) \quad \text{oder} \quad H \vdash G \vee H \ (\vee i) \qquad \text{bzw.} \qquad \frac{G}{G \vee H}\,(\vee i) \quad \text{oder} \quad \frac{H}{G \vee H}\,(\vee i)$$

sowie

$$G \vee H, [G] \ldots K, [H] \ldots K \vdash K \ (\vee e) \qquad \text{bzw.} \qquad \frac{G \vee H \quad \begin{array}{c}[G]\\ \vdots \\ K\end{array} \quad \begin{array}{c}[H]\\ \vdots \\ K\end{array}}{K}\,(\vee e)$$

7.5.2 Beispiel. Die Implikation

$$G \wedge H \Rightarrow G \vee H$$

ist eine Tautologie. In der Tat, hier ist ein Beweis (ohne Prämissen):

1	$G \wedge H$	Kasten-Prämisse
2	G	$(\vee e), 1$
3	$G \vee H$	$(\vee i), 2$
4	$G \wedge H \Rightarrow G \vee H$	$(\Rightarrow i) 1 - 3$

In Baumform haben wir

$$\frac{\dfrac{\dfrac{[G \wedge H]}{G}\,(\wedge e)}{G \vee H}\,(\vee i)}{G \wedge H \Rightarrow G \vee H}\,(\Rightarrow i)$$

Dieser Baum hat nur einen Zweig, daher ist der Unterschied zur Tableaux-Version gering.

Einmal hergeleitete Tautologien können in späteren Herleitungen natürlich genutzt werden können, ohne sie erneut herleiten zu müssen.

7.5.3 Beispiel. Wir beweisen, dass $\vee$ kommutativ ist:

$$G \vee H \vdash H \vee G$$

1	$G \vee H$	
2	G	Kasten-Prämisse
3	$H \vee G$	$(\vee i), 2$
4	H	Kasten-Prämisse
5	$H \vee G$	$(\vee i), 4$
6	$H \vee G$	$(\vee e), 1, 4 - 5, 2 - 3$

In Baumform:

$$\frac{G \vee H \quad \dfrac{[G]}{H \vee G}\,(\vee i) \quad \dfrac{[H]}{H \vee G}\,(\vee i)}{H \vee G}\,(\vee e)$$

7.5.4 Beispiel. Wir leiten eines Teil eines der Distributivgesetze her:

$$F \vee (G \wedge H) \vdash (F \vee G) \wedge (F \vee H)$$

1	$F \vee (G \wedge H)$	Prämisse
2	F	Kasten-Prämisse
3	$F \vee G$	$(\vee i), 2$
4	$G \wedge H$	Kasten-Prämisse
5	G	$(\wedge e), 4$
6	$F \vee G$	$(\vee i), 5$
7	$F \vee G$	$(\vee e), 1, 2-3, 4-6$
8	F	Kasten-Prämisse
9	$F \vee H$	$(\vee i), 8$
10	$G \wedge H$	Kasten-Prämisse
11	H	$(\wedge e), 10$
12	$F \vee H$	$(\vee i), 11$
13	$F \vee H$	$(\vee e), 1, 8-9, 10-12$
14	$(F \vee G) \wedge (F \vee H)$	$(\wedge i), 7, 13$

Der entsprechende Baum ufert etwas aus:

$$\dfrac{\dfrac{F \vee (G \wedge H) \quad \dfrac{[F]}{F \vee G}(\vee i) \quad \dfrac{\dfrac{[G \wedge H]}{G}(\wedge e)}{F \vee G}(\vee i)}{F \vee G}(\vee e) \quad \dfrac{F \vee (G \wedge H) \quad \dfrac{[F]}{F \vee H}(\vee i) \quad \dfrac{\dfrac{[G \wedge H]}{H}(\wedge e)}{F \vee H}(\vee i)}{F \vee H}(\vee e)}{(F \vee G) \wedge (F \vee H)}(\wedge i)$$

7.6 ND-Regeln der Negation und Absurdität

Die Introduktion der Negation $\neg G$ ist simpel, beruht aber wieder auf Fernwirkung: wir nehmen an, dass es uns gelungen ist, aus G die Absurdität ($\bot$) herzuleiten, das erlaubt uns eine Herleitung von $\neg G$.

Wie wird die Negation eliminiert? Falls G sowie $\neg G$ Prämissen in Γ sind, ist Γ nicht erfüllbar und erlaubt daher die Herleitung der Absurdität. Insofern fallen die Elimination von $\neg$ und die Introduktion von $\bot$ zusammen. Die Elimination von $\bot$ orientiert sich an der Tautologie $\bot \Rightarrow G$ für beliebiges G, die unter dem arg verkürzten Namen „ex falso quodlibet“ bekannt ist, frei übersetzt „aus Falschem läßt sich Beliebiges herleiten“, wobei „sequitur“ unter den Tisch gefallen ist.

Leider genügt dies noch nicht, um doppelte Negationen zu eliminieren. Das erfordert eine Extra-Regel, die nicht nur das bisherige Schema der paarweisen Introduktions- und Eliminationsregeln für die Junktoren bricht, sondern auch nicht von allen Mathematikern

als evident angesehen wird. In der sog. intuitionistischen Logik ist die Elimination der doppelten Negation nicht erlaubt, aber dies ist tatsächlich der einzige Unterschied zur klassischen Logik. (Es gibt noch weitere Schlußregeln, die dieselbe Rolle spielen können, etwa die Annahme, $F \vee \neg F$ sei eine Tautologie, also ohne Prämissen herleitbar; vgl. HA.)

7.6.1 Definition. Die Regeln für Negation und Absurdität lauten

$$[G] \ldots \bot \vdash \neg G \ (\neg i) \text{ bzw. } \dfrac{\begin{array}{c}[G]\\ \vdots \\ \bot\end{array}}{\neg G}\ (\neg i)$$

sowie

$$G, \neg G \vdash \bot \ (\neg e) \qquad \text{bzw.} \qquad \dfrac{G \quad \neg G}{\bot}\ (\neg e)$$

und schließlich

$$\neg\neg G \vdash G \ (\neg\neg e) \quad \text{bzw.} \quad \dfrac{\neg\neg G}{G}\ (\neg\neg e) \qquad \text{sowie} \qquad \bot \vdash G \ (\bot e) \quad \text{bzw.} \quad \dfrac{\bot}{G}\ (\bot e)$$

7.6.2 Beispiel. Wir sind nun in der Lage das Beweisprinzip der *Contraposition*, auch bekannt als *Beweis durch Widerspruch*, zu etablieren. Zunächst betrachten wir

$$F \Rightarrow G \vdash \neg G \Rightarrow \neg F \tag{7.1}$$

Beweis:

1	$F \Rightarrow G$	Prämisse
2	$\neg G$	Kasten-Prämisse
3	F	Kasten-Prämisse
4	G	$(\Rightarrow e), 1, 3$
5	$\bot$	$(\neg e), 4, 2$
6	$\neg F$	$(\neg i), 3-5$
7	$\neg G \Rightarrow \neg F$	$(\Rightarrow i), 2-6$

Im zugehörigen Baum sind die Scoping-Regeln besonders unübersichtlich:

$$\dfrac{\dfrac{[\neg G] \qquad \dfrac{[F] \quad F \Rightarrow G}{G}\ (\Rightarrow e)}{\bot}\ (\neg e)}{\dfrac{\neg F}{\neg G \Rightarrow \neg F}\ (\Rightarrow i)}\ (\neg i)$$

Nun sind wir an einer Herleitung von $F \Rightarrow G$ aus der Prämisse $\neg G \Rightarrow \neg F$ interessiert.

$$\neg G \Rightarrow \neg F \vdash F \Rightarrow G \tag{7.2}$$

Beweis:

1	$\neg G \Rightarrow \neg F$	Prämisse
2	F	Kasten-Prämisse
3	$\neg G$	Kasten-Prämisse
4	$\neg F$	$(\Rightarrow e), 1, 3$
5	$\bot$	$(\neg e), 4, 2$
6	$\neg\neg G$	$(\neg i), 3-5$
7	G	$(\neg\neg e), 6$
8	$F \Rightarrow G$	$(\Rightarrow i), 2-7$

Die Struktur des Beweises ist bis auf die Verwendung von F anstelle von $\neg\neg F$ in Schritt 2 und auf Schritt 7 ganz ähnlich wie oben. Das gilt natürlich auch für den Beweisbaum:

$$\dfrac{\dfrac{\dfrac{[F] \quad \dfrac{[\neg G] \quad \neg G \Rightarrow \neg F}{\neg F}\,(\Rightarrow e)}{\bot}\,(\neg e)}{\neg\neg G}\,(\neg i)}{G}\,(\neg\neg e)}{F \Rightarrow G}\,(\Rightarrow i)$$

Je eine Anwendung der Regel $(\Rightarrow i)$ liefert die beiden Teile der gewünschten Äquivalenz:

$$\vdash (F \Rightarrow G) \Leftrightarrow \neg G \Rightarrow \neg F$$

7.6.3 Beispiel. Obwohl $(\neg\neg e)$ eine separate Regel ist, brauchen wir für $(\neg\neg i)$ keine eigene Regel, denn die Herleitbarkeit von $\neg\neg G$ aus G können wir beweisen:

1	G	Prämisse
2	$\neg G$	Kasten-Prämisse
3	$\bot$	$(\neg e), 1, 2$
4	$\neg\neg G$	$(\neg i), 2-3$

oder in Baumform

$$\dfrac{\dfrac{G \quad [\neg G]}{\bot}\,(\neg e)}{\neg\neg G}\,(\neg i)$$

7.6.4 Beispiel. Die erste De Morgansche Regel, $\neg(G \vee H) \Leftrightarrow \neg G \wedge \neg H$ läßt sich herleiten. Für die Implikation $\neg(G \vee H) \Rightarrow \neg G \wedge \neg H$ zeigen wir zunächst

$$\neg(G \vee H) \vdash \neg G \wedge \neg H$$

1	$\neg(G \vee H)$	Prämisse
2	G	Kasten-Prämisse
3	$G \vee H$	$(\vee i), 2$
4	$\bot$	$(\neg e), 3, 1$
5	$\neg G$	$(\neg i) 2 - 4$
6	H	Kasten-Prämisse
7	$G \vee H$	$(\vee i), 6$
8	$\bot$	$(\neg e), 7, 1$
9	$\neg H$	$(\neg i), 6 - 8$
10	$\neg G \wedge \neg H$	$(\wedge i), 5, 9$

Hier ist derselbe Beweis in Baumform:

$$\dfrac{\dfrac{\neg(G \vee H) \quad \dfrac{[G]}{G \vee H}\,(\vee i)}{\dfrac{\bot}{\neg G}\,(\neg i)} \quad \dfrac{\neg(G \vee H) \quad \dfrac{[H]}{G \vee H}\,(\vee i)}{\dfrac{\bot}{\neg H}\,(\neg i)}}{\neg G \wedge \neg H}\,(\wedge i)$$

Eine weitere Anwendung von $(\Rightarrow i)$ liefert die Herleitbarkeit von $\neg(G \vee H) \Rightarrow \neg G \wedge \neg H$. Der Beweis der anderen Richtung folgt aus

1	$\neg G \wedge \neg H$	Prämisse
2	$G \vee H$	Kasten-Prämisse
3	G	Kasten-Prämisse
4	$\neg G$	$(\wedge e), 1$
5	$\bot$	$(\neg e), 3, 4$
6	H	Kasten-Prämisse
7	$\neg H$	$(\wedge e), 1$
8	$\bot$	$(\neg e), 6, 7$
9	$\bot$	$(\vee e), 2, 3 - 5, 6 - 8$
10	$\neg(G \vee H)$	$(\neg i), 2 - 9$

oder in Baumform

$$\dfrac{\dfrac{[G \vee H] \quad \dfrac{\dfrac{\neg G \wedge \neg H}{\neg G}\,(\wedge e) \quad [G]}{\bot}\,(\neg e) \quad \dfrac{\dfrac{\neg G \wedge \neg H}{\neg H}\,(\wedge e) \quad [H]}{\bot}\,(\neg e)}{\bot}\,(\vee e)}{\neg(G \vee H)}\,(\neg i)$$

Eine weitere Anwendung von $(\Rightarrow i)$ liefert die Herleitbarkeit von $\neg G \wedge \neg H \Rightarrow \neg(G \vee H)$.

7.6.5 Beispiel. Wie steht es mit der zweiten De Morganschen Regel $\neg(F \wedge G) \vdash \neg F \vee \neg G$?. In Anlehnung an Beispiel 7.6.4 könnte man wie folgt beginnen:

1	$\neg(F \wedge G)$	
2	$\neg(\neg F \vee \neg G)$	Kasten-Prämisse
3	$\neg F$	Kasten-Prämisse
4	$\neg F \vee \neg G$	$(\vee i), 3$
5	$\bot$	$(\neg e), 4, 2$
6	$\neg\neg F$	$(\neg i) 3 - 5$
7	F	$(\neg\neg e), 6$
8	$\neg G$	Kasten-Prämisse
9	$\neg F \vee \neg G$	$(\vee i), 8$
10	$\bot$	$(\neg e), 9, 2$
11	$\neg\neg G$	$(\neg i), 8 - 10$
12	G	$(\neg\neg e), 11$
13	$F \wedge G$	$(\wedge i), 7, 12$
14	$\bot$	$(\neg e), 13, 1$
15	$\neg\neg(\neg F \vee \neg G)$	$(\neg i), 2 - 14$
16	$\neg F \vee \neg G$	$(\neg\neg e), 15$

Die Zeilen 2–13 entsprechen im Wesentlichen dem ersten Beweis von 7.6.4 mit $\neg F$ anstelle von G und $\neg G$ anstelle von H, ausgenommen die Zeilen 7 und 12, wo die doppelte Negation sofort eliminiert wird, statt sich später mit $\neg\neg F \wedge \neg\neg G$ herumzuplagen. Letzteres in $F \wedge G$ umzuwandeln erfordert mehr Aufwand, als die Bausteine gleich zu vereinfachen.

Hier ist der zugehörige Baum:

$$\dfrac{\neg(F \wedge G) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\neg(\neg F \vee \neg G)] \quad \dfrac{[\neg F]}{\neg F \vee \neg G}\,(\vee i)}{\bot}\,(\neg e)}{\neg\neg F}\,(\neg i)}{F}\,(\neg\neg e) \qquad \dfrac{\dfrac{\dfrac{\dfrac{[\neg(\neg F \vee \neg G)] \quad \dfrac{[\neg G]}{\neg F \vee \neg G}\,(\vee i)}{\bot}\,(\neg e)}{\neg\neg G}\,(\neg i)}{G}\,(\neg\neg e)}{F \wedge G}\,(\wedge i)}{\bot}\,(\neg e)}{\neg\neg(\neg F \vee \neg G)}\,(\neg i)}{\neg F \vee \neg G}\,(\neg\neg e)$$

Anstelle einer weiteren expliziten Herleitung von $\neg F \vee \neg G \vdash \neg(F \wedge G)$ wollen wir versuchen, dieses Ergebnis auf geschicktere Weise mit Hilfe bereits hergeleiteter Tautologien herzuleiten.

Der zweite Teil von Beispiel 7.6.4 hat als Instanz $\neg\neg F \wedge \neg\neg G \vdash \neg(\neg F \vee \neg G)$. Mittels

Contraposition, siehe Beispiel 7.6.2, gilt somit auch

$$\neg\neg(\neg F \vee \neg G) \vdash \neg(\neg\neg F \wedge \neg\neg G) \tag{7.3}$$

Substitution der Prämisse $\neg F \vee \neg G$ in Beispiel 7.6.3 liefert

$$\neg F \vee \neg G \vdash \neg\neg(\neg F \vee \neg G) \tag{7.4}$$

Die Kombination von Beispiel 7.6.3 mit $(\wedge i)$ liefert andererseits $F \wedge G \vdash \neg\neg F \wedge \neg\neg G$, aufgrund von Contraposition also

$$\neg(\neg\neg F \wedge \neg\neg G) \vdash \neg(F \wedge G) \tag{7.5}$$

Zusammen mit den obigen Ergebnissen also

$$\neg F \vee \neg G \vdash \neg\neg(\neg F \vee \neg G) \vdash \neg(\neg\neg F \wedge \neg\neg G) \vdash \neg(F \wedge G)$$

Hier wurden beschrieben, wie Instanzen bereits existierender Herleitungen miteinander kombiniert werden können. Das ist letztendlich unabhängig von der Darstellung als Tableaux oder als Baum.

Grundsätzlich ist man also an der Wiederverwendbarkeit natürlicher Deduktionsbeweise interessiert, modulo geeigneter Substitutionen. Das gilt für beide Arten der Darstellung. Beim Zusammensetzen von Beweisen können im Prinzip Redundanzen auftreten. Streamlining, also deren Elimination, sollte aber erst zum Schluß erfolgen.

7.6.6 Beispiel. Fortsetzung von Beispiel 7.5.4. Unter Verwendung früherer Ergebnisse zeigen wir die Herleitbarkeit der anderen Richtung eines Distributivgesetzes:

$$(F \vee G) \wedge (F \vee H) \vdash F \vee (G \wedge H)$$

Bei Verwendung der Tableaux-Form ist insbesondere auf die korrekten Zeilennummern

zu achten.

1	$(F \vee G) \wedge (F \vee H)$	Prämisse
2	$F \vee G$	$(\wedge e), 1$
3	$F \vee H$	$(\wedge e), 2$
4	$\neg(F \vee (G \wedge H))$	Kasten-Prämisse
12	$\vdots$	(8 Zeilen aus Beispiel 7.6.4)
13	$\neg F \wedge \neg(G \wedge H)$	
14	$\neg F$	$(\wedge e), 13$
15	$\neg(G \wedge H)$	$(\wedge e), 13$
29	$\vdots$	(14 Zeilen aus Beispiel 7.6.5)
30	$\neg G \vee \neg H$	
31	$\neg G$	Kasten-Prämisse
32	F	Kasten-Prämisse
33	G	Kasten-Prämisse
34	$\bot$	$(\neg e), 33, 31$
35	F	$(\bot e), 34$
36	F	$(\vee e), 2, 32, 33-35$
37	$\neg H$	Kasten-Prämisse
38	F	Kasten-Prämisse
39	H	Kasten-Prämisse
40	$\bot$	$(\neg e), 39, 37$
41	F	$\bot, 40$
42	F	$(\vee e), 3, 38, 39-41$
43	F	$(\vee e), 30, 31-35, 37-42$
44	$\bot$	$(\neg e), 43, 14$
45	$\neg\neg(F \vee (G \wedge H))$	$(\neg i), 4-46$
46	$F \vee (G \wedge H)$	$(\neg\neg e), 45$

Aus offensichtlichen Gründen ersparen wir uns hier den Baum.

7.7 Zusammenfassung der ND-Regeln

Zusammenfassung der Regeln der Natürlichen Deduktion

	Introduktion	Elimination
Konjunktion	$\dfrac{G \quad H}{G \wedge H}$	$\dfrac{G \wedge H}{G}$ sowie $\dfrac{G \wedge H}{H}$
	$G, H \vdash G \wedge H$	$G \wedge H \vdash G \ , \ G \wedge H \vdash H$
Implikation	$\dfrac{\begin{array}{c}[G] \\ \vdots \\ H\end{array}}{G \Rightarrow H}$	$\dfrac{G \quad G \Rightarrow H}{H}$
	$[G] \ldots H \vdash G \Rightarrow H$	$G, G \Rightarrow H \vdash H$
Disjunktion	$\dfrac{G}{G \vee H}$ sowie $\dfrac{H}{G \vee H}$	$\dfrac{G \vee H \quad \begin{array}{c}[G] \\ \vdots \\ K\end{array} \quad \begin{array}{c}[H] \\ \vdots \\ K\end{array}}{K}$
	$G \vdash G \vee H \ , \ H \vdash G \vee H$	$G \vee H, [G] \ldots K, [H] \ldots K \vdash K$
Negation	$\dfrac{\begin{array}{c}[G] \\ \vdots \\ \bot\end{array}}{\neg G}$	$\dfrac{G \quad \neg G}{\bot}$ und $\dfrac{\neg\neg G}{G}$ und $\dfrac{\bot}{G}$
	$[G] \ldots \bot \vdash \neg G$	$G, \neg G \vdash \bot \ , \ \neg\neg G \vdash G \ , \ \bot \vdash G$

In der Literatur finden sich diverse Varianten obiger Regeln.

7.7.1 Beispiel. Im Rahmen seiner Bachelorarbeit hat Herrn Grunwald zur rechnerunterstützten Natürlichen Deduktion erstellt. Allerdings verwendet er Schlußregeln für Disjunktion und Negation, die gegenüber den obigen leicht abgewandelt sind (umseitig). Das resultiert einerseits daraus, dass er die Absurdität $\bot$ nicht als Junktor verwendet, anderseits gewisse Programmteile für $(\Rightarrow i)$ bei $(\vee e)$ wiederverwenden wollte. Ein formaler Beweis der Gleichwertigkeit beider Kalküle steht noch aus. Um ein Gefühl für die Unterschiede in den Herleitungen zu bekommen, wird empfohlen, $\neg G \wedge \neg H \vdash \neg(G \vee H)$ aus Beispiel 7.6.4 mit Hilfe von Herrn Grunwalds Reglen nachzuweisen.

	Introduktion	Elimination
Disjunktion	$\dfrac{G}{G \vee H}$ sowie $\dfrac{H}{G \vee H}$	$\dfrac{G \vee H \quad \begin{array}{c}[G]\\ \vdots \\ G \Rightarrow K\end{array} \quad \begin{array}{c}[H]\\ \vdots \\ H \Rightarrow K\end{array}}{K}$
	$G \vdash G \vee H \ , \ H \vdash G \vee H$	$G \vee H, [G] \ldots G \Rightarrow K, [H] \ldots H \Rightarrow K \vdash K$
Negation	$\dfrac{\begin{array}{c}[G]\\ \vdots \\ H, \neg H\end{array}}{\neg G}$	$\dfrac{\neg\neg G}{G}$
	$[G] \ldots \neg H \ldots H \cdots \vdash \neg G$	$\neg\neg G \vdash G$

Praktische Aspekte der Natürlichen Deduktion

- ▷ Bei der natürlichen Deduktion werden Formeln rein syntaktisch als Zeichenketten behandelt, die nach den obigen 12 Methoden transformiert werden dürfen. Es dürfen keine semantischen Äquivalenzen verwendet werden, also darf man z.B. $F \Rightarrow G$ nicht direkt durch $G \vee \neg F$ ersetzen; stattdessen muß man $F \Rightarrow G \vdash G \vee \neg F$ ebenfalls syntaktisch herleiten. (Einmal hergeleitet, darf man dieses Ergebnis natürlich später wiederverwenden.)
- ▷ Der häufigste Fehler bei der natürlichen Deduktion ist die Verwendung „lokaler Variablen“ außerhalb ihres Gültigkeitsbereichs (scope). Dieses Problem ist in der Baumdarstellung gelegentlich schwer zu sehen, dafür zeigt die Tableaux-Darstellung die Struktur des Beweises weniger deutlich.
- ▷ In der Klausur dürfen nur die obigen Regeln verwendet werden, also keine Regeln, die in der Vorlesung oder Übung daraus abgeleitet worden sind. Letztere sind bei Bedarf neu abzuleiten. Wenn nicht explizit anders verlangt, sollten Beweise in Tableaux-Form angegeben werden.

7.7.2 Definition. Eine Formel F ist aus einer Menge Γ von Formeln **herleitbar**, geschrieben $\Gamma \vdash F$, wenn eine Herleitung von F existiert, deren sämtliche Voraussetzungen zu Γ gehören. (Solche in eckigen Klammern, d.h., Kasten-Prämissen, gehören nicht dazu.) F heißt **Theorem**, falls $\vdash F$ gilt.

7.8 Korrektheit und Vollständigkeit der Natürlichen Deduktion

Wir wenden uns nun den zu Beginn angesprochenen Fragen zu, die den Zusammenhang zwischen $\models$ und $\vdash$ betreffen. Da die Herleitbarkeit $\vdash$ durch die Existenz einer Herleitung und somit rekursiv definiert ist, bietet sich für den Nachweis der Korrektheit ein Induktionsbeweis an.

7.8.1 Satz (Korrektheit)**.** *Wenn* $\Gamma \vdash F$*, dann auch* $\Gamma \models F$*.*

Beweis. Wir verwenden Induktion über die Tiefe t des Herleitungsbaumes für $\Gamma \vdash F$ und überprüfen, ob die obigen Regeln, angewendet im letzten Schritt, Wahrheit erhalten.

$t = 0$**:** Existiert ein Herleitungsbaum ohne Knoten, so gehört F zwangsläufig zu Γ, woraus automatisch $\Gamma \models F$ folgt.

Wir nehmen nun an, dass für alle Sequenzen $\Gamma \vdash F$, für die ein Herleitungsbaum der Tiefe $< t$ existiert, $\Gamma \models F$ gilt.
Ist t die minimale Tiefe der Herleitungsbäume für $\Gamma \vdash F$, wählen wir einen derartigen Baum T aus und betrachten den letzten Knoten.

($\wedge i$): Für $F = G \wedge H$ betrachte in T die Herleitungsbäume R und S für G bzw. H der Tiefe $< t$ mit Prämissenmengen $\Omega \subseteq \Gamma$ bzw. $\Delta \subseteq \Gamma$. Nach Voraussetzung gilt $\Omega \models G$ und $\Delta \models H$.

Jede $\Gamma \supseteq \Omega \cup \Delta$ erfüllende und für G sowie H passende Belegung α erfüllt sowohl Ω als auch Δ, also gilt nach Voraussetzung $\widehat{\alpha}(G) = 1 = \widehat{\alpha}(H)$, und damit $\widehat{\alpha}(F) = \widehat{\alpha}(G \wedge H) = 1$. Existiert keine derartige Belegung, ist $\Gamma \models G \wedge H$ trivialerweise korrekt.

($\wedge e$): S sei der Herleitungsbaum für $F \wedge G$ der Tiefe $t-1$ mit Prämissenmenge Γ. Nach Voraussetzung gilt $\Gamma \models F \wedge G$.

Betrachte eine für Γ und F passende Belegung α, die Γ erfüllt. Diese erweitern wir zu einer auch für G passende Belegung β, indem wir allen Varialblen von G, auf denen α nicht definiert, beliebige Werte zuweisen. β erfüllt weiterhin Γ, und nach Voraussetzung gilt $\widehat{\beta}(F \wedge G) = 1$, also auch $\widehat{\alpha}(F) = \widehat{\beta}(F) = 1$.

($\Rightarrow i$): Der zugehörige Herleitungsbaum für $\Gamma, G \vdash F$ hat die Tiefe $t - 1$, folglich gilt $\Gamma, G \models F$.

Betrachte eine Γ erfüllende und für $G \Rightarrow F$ passende Belegung α. Falls $G \in \Gamma$, folgt $\Gamma \models F$ und somit $\Gamma - \{G\} \models G \Rightarrow F$, also auch $\Gamma \models G \Rightarrow F$.

Falls $G \notin \Gamma$ folgt für $\widehat{\alpha}(G) = 1$ nach Voraussetzung $\widehat{\alpha}(F) = 1$, und damit $\widehat{\alpha}(G \Rightarrow F) = 1$. Andererseits impliziert $\widehat{\alpha}(G) = 0$ ebenfalls $\widehat{\alpha}(G \Rightarrow F) = 1$. Also gilt auch in diesem Fall $\Gamma \models G \Rightarrow F$.

($\Rightarrow e$): R und S seien die Herleitungsbäume für G bzw. $G \Rightarrow F$ der Tiefe $< t$ in T, mit Prämissenmengen Ω und Δ. Nach Voraussetzung gilt $\Omega \models G$ und $\Delta \models G \Rightarrow F$.

Ist α eine $\Gamma \supseteq \Omega \cup \Delta$ erfüllende und für F passende Belegung, so erweitern wir diese beliebig (s.o.) zu einer für G passenden Belegung β; diese erfüllt Γ immer noch. Nach Voraussetzung gilt $\widehat{\beta}(G) = \widehat{\beta}(G \Rightarrow F) = 1$, und folglich auch $\widehat{\alpha}(F) = \widehat{\beta}(F) = 1$.

($\vee i$): Eine Herleitung von $\Gamma \vdash F = G \vee H$ der Tiefe t hat vor dem letzten Schritt obdA eine Herleitung $\Gamma \vdash G$ der Länge $t-1$, folglich gilt $\Gamma \models G$.

Jede Γ erfüllende und für $G \vee H$ passende Belegung α paßt auch für G und erfüllt daher $\widehat{\alpha}(G) = 1$. Folglich gilt auch $\widehat{\alpha}(G \vee H) = \widehat{\alpha}(G) \vee \widehat{\alpha}(H) = 1$.

($\vee e$): In einem Herleitungsbaum der Länge t von $\Gamma \vdash F$ möge ($\vee e$) als letzter Schritt kürzere Teilherleitungen $\Omega \vdash G \vee H$ sowie $\Delta, G \vdash F$ und $\Phi, H \vdash F$ aufweisen. Nach Voraussetzung gilt $\Omega \models G \vee H$, $\Delta, G \models F$ sowie $\Phi, H \models F$.

Betrachte eine $\Gamma \supseteq \Omega \cup \Delta \cup \Phi$-erfüllende und für F passende Belegung α. Diese paßt automatisch auch für $G \vee H$, insofern hat mindestens einer der Werte $\widehat{\alpha}(G)$ und $\widehat{\alpha}(H)$ den Wert 1, und damit auch $\widehat{\alpha}(F)$.

($\neg i$): Eine Herleitung $\Gamma \vdash \neg G$ der Länge t basiert auf einer Herleitung $\Gamma, G \vdash \bot$ der Länge $t-1$, nach Voraussetzung also $\Gamma, G \models \bot$. Auf der semantischen Ebne dürfen wir nun G durch $\neg\neg G$ ersetzen, weil $\widehat{\alpha}(\neg\neg G) = \widehat{\alpha}(G)$. Das liefert $\Gamma, \neg\neg G \models \bot$. Gemäß Satz 5.0.2 gilt damit auch $\Gamma \models \neg G$, wie gewünscht.

($\neg e$): Aus zwei Herleitungen $\Gamma \vdash G$ und $\Gamma \vdash \neg G$ der Länge $> t$ folgt, dass $\Gamma \models G$ sowie $\Gamma \models \neg G$. Damit ist Γ nicht erfüllbar, d.h., $\Gamma \models \bot$.

($\neg\neg e$): Die Herleitung $\Gamma \vdash F$ der Länge t enthält eine Herleitung $\Gamma \vdash \neg\neg F$ der Länge $t-1$, also gilt nach Voraussetzung $\Gamma \models \neg\neg F$. Aber auf der semantischen Ebene dürfen wir $\neg\neg F$ durch F ersetzen (s.o.).

($\bot e$): Die Herleitung $\Gamma \vdash F$ der Länge t beruht auf einer Herleitung $\Gamma \vdash \bot$ der Länge $t-1$, nach Voraussetzung also $\Gamma \models \bot$, oder äquivalent, Γ ist unerfüllbar. Da jede der Γ erfüllenden Belegungen F wahr macht, haben wir $\Gamma \models F$. □

Aufgrund dieses Ergebnisses sind Theoreme, also ohne Voraussetzungen herleitbare Formeln (siehe Definition 7.7.2), automatisch Tautologien, etwa das Prinzip der Contraposition (Beweis durch Widerspruch) aus Beispiel 7.6.2, und speziell dessen zweite Hälfte

$$T_2 := (\neg G \Rightarrow \neg H) \Rightarrow (H \Rightarrow G) \tag{7.6}$$

die sich aus der Sequenz 7.6.2 ergibt. Ebendieses Prinzip kann auch auf der Meta-Ebene Anwendung finden um nachzuweisen, dass manche Formeln keine Theoreme sind: einfach indem man zeigt, dass sie keine Tautologien sind.

7.8.2 Beispiel. Die Formel

$$T_0 := G \Rightarrow (H \Rightarrow G)$$

ist eine Tautologie. Hier ist der Beweis für $\vdash T_0$:

1	G	Kasten-Prämisse
2	H	Kasten-Prämisse
3	$G \wedge H$	$(\wedge i), 1, 1$
4	G	$(\wedge e), 3$
5	$H \Rightarrow G$	$(\Rightarrow i), 2-4$
6	$G \Rightarrow (H \Rightarrow G)$	$(\Rightarrow i), 1-5$

7.8.3 Beispiel. Die Formel

$$T_1 := (F \Rightarrow (G \Rightarrow H)) \Rightarrow ((F \Rightarrow G) \Rightarrow (F \Rightarrow H))$$

ist ebenfalls eine Tautologie:

1	$F \Rightarrow (G \Rightarrow H)$	Kasten-Prämisse
2	$F \Rightarrow G$	Kasten-Prämisse
3	F	Kasten-Prämisse
4	G	$(\Rightarrow e), 2, 3$
5	$G \Rightarrow H$	$(\Rightarrow e), 1, 3$
6	H	$(\Rightarrow e), 5, 4$
7	$F \Rightarrow H$	$(\Rightarrow i), 3-6$
8	$(F \Rightarrow G) \Rightarrow (F \Rightarrow H)$	$(\Rightarrow i), 2-7$
9	$(F \Rightarrow (G \Rightarrow H)) \Rightarrow ((F \Rightarrow G) \Rightarrow (F \Rightarrow H))$	$(\Rightarrow i), 1-8$

Die Umkehrung des Korrektheitssatzes ist schwieriger zu beweisen, siehe z.B. [HR09, Satz 1.4.4].

7.8.4 Satz (Vollständigkeit)**.** *Wenn* $\Gamma \models F$*, dann auch* $\Gamma \vdash F$*.* □

Wir können die Natürliche Deduktion als einen Algorithmus betrachten, der versucht, zu einer gegebenen Sequenz $\Gamma \models F$ einen formalen Beweis der Länge 1, 2, 3, ... zu finden. Auch diese Methode, genau wie die Resolutionsmethode, ist aber ineffizient. Das Problem der Erfüllbarkeit von Aussagenlogik ist als „NP-vollständig“ bekannt und deswegen ist es sehr unwahrscheinlich, dass je ein effizienter Algorithmus mit polynominaler Laufzeit in der Größe der Sequenz (d.h., der Größe von $\bigwedge \Gamma \wedge \neg F$) gefunden wird.

Es gibt viele effiziente Methoden, die für Formeln eines speziellen Typs ihre Erfüllbarkeit effizient testen, etwa die Hornlogik (Kapitel 8).

7.9 Hilberts Axiomatisierung

Wie in der Einführung erwähnt, hat David Hilbert zu Beginn des letzten Jahrhunderts einen anderen Stil für formale Beweise in der Aussagenlogik entwickelt: Er benutzt

nur eine Regel, den Modus Ponens ($\Rightarrow e$), in Verbindung mit einer Reihe sogenannter *Axiome*. Dabei handelt es sich um intuitiv als gültig angesehen vorgegebene Formeln bzw. Formel-Schemata. Aus geeignet instanziierten Axiomen (für F, G, $H\ldots$) können dann alle anderen Tautologien mittels ($\Rightarrow e$) hergeleitet werden. Sein System benutzte nur die Junktoren $\Rightarrow$ und $\neg$. Diese sind adäquat (siehe Definition 3.3.1). Wie in Beispiel 3.3.2 gesehen, lassen sich Konjunktion und Disjunktion wie folgt darstellen:

$$F \vee G \equiv \neg F \Rightarrow G \qquad \text{sowie} \qquad F \wedge G \equiv \neg(F \Rightarrow \neg G)$$

Hilbert verwendete nur drei Axiome, nämlich die Tautologien T_0, T_1, T_2 aus den vorangegangenen Beispielen 7.8.2, 7.8.3 und 7.6.2.

7.9.1 Satz. *Hilberts Kalkül ist vollständig: Jede Tautologie F hat einen Beweis der Sequenz*

$$T_0, T_1, T_2 \vdash F$$

der nur Modus Ponens benutzt. □

8. Hornlogik

8.1 Hornformeln

Die Hornlogik beschränkt sich auf eine Untermenge der Formeln der Aussagenlogik in KNF, für die die Resolutionsmethode effizient funktioniert. Diese Formeln sind zudem für die Logik-Programmierung wichtig und werden etwa in der Programmiersprache PROLOG verwendet. Erinnern wir uns, dass ein Literal entweder positiv ist, d.h., eine atomare Aussage, oder die Negation einer atomaren Aussage.

8.1.1 Definition (vergl. Alfred Horn, 1951)**.** Eine Formel in KNF heißt **Hornformel**, sofern jede Klausel höchstens ein positives Literal und kein Literal doppelt enthält.

Wir unterscheiden folgende Typen von Klauseln:

- **Regeln** oder **definite clauses** mit genau einem positiven Literal, die somit zu Implikationen von einer endlichen Konjunktion von Variblen in eine einzelne Variable äquivalent sind, etwa

  $$\neg A_0 \vee \neg A_2 \vee \cdots \vee \neg A_{n-1} \vee B \equiv A_0 \wedge A_2 \wedge \ldots \wedge A_{n-1} \Rightarrow B$$

  Fehlen die negativen Literale, hat also die obige Implikation de Prämisse $\top$, so spricht man auch von **Tatsachen**;

- **Frage-Klauseln** ohne positives Literal, die folglich zur Negation einer Co-Klausel äquivalent sind, etwa

  $$\neg A_0 \vee \neg A_2 \vee \cdots \vee \neg A_{n-1} \equiv \neg(A_0 \wedge A_2 \wedge \ldots \wedge A_{n-1}) \equiv A_0 \wedge A_2 \wedge \ldots \wedge A_{n-1} \Rightarrow \bot$$

  Die zugrundeliegende Frage bezieht sich auf die Gültigkeit dieser Co-Klausel $A_0 \wedge A_2 \wedge \ldots \wedge A_{n-1}$. Diese ist natürlich gegeben, wenn die Hornformel zusammen mit der Frage-Klausel *nicht* erfüllbar ist, vergl. Satzt 5.0.2. Sind mehrere Frage-Klauseln vorhanden, so bezieht sich die Frage auf die Gültigkeit der Disjunktion der entprechenden Co-Klauseln, d.h., einer Formel in DNF mit lauter positiven Literalen.

8.1.2 Definition. Die sogenannte **Hornlogik** hat dasselbe Alphabet wie die Aussagenlogik, mit einer auf die Teilmenge $\mathcal{H} \subseteq \mathcal{F}$ der Hornformeln eingeschränkten Syntax. Die Semantik entspricht der Semantik der Aussagenlogik.

8.1.3 Beispiel. Die Formel $F = Q \wedge (Q \wedge S \Rightarrow P) \wedge (Q \wedge R \Rightarrow P) \wedge (\neg R \vee \neg S)$ ist strenggenommen keine Hornformel, aber sie ist zu der Hornformel

$$Q \wedge (\neg Q \vee \neg S \vee P) \wedge (\neg Q \vee \neg R \vee P) \wedge (\neg R \vee \neg S)$$

äquivalent. Diese können wir als eine Frage nach der Gültigkeit von $R \wedge S$ verstehen, unter der Voraussetzung, dass Q eine Tatsache und somit atomar ist, und die Implikationen $Q \wedge S \Rightarrow P$ sowie $Q \wedge R \Rightarrow P$ gelten. Die Formel F ist tatsächlich erfüllbar: für $\alpha(P) = \alpha(Q) = 1,\ \alpha(S) = \alpha(R) = 0$ hat sie Wert 1. Daher ist die Frage nach der Gültigkeit von $R \wedge S$ negativ zu beantworten.

8.1.4 Beispiel. An einen Wintertag (W) wurde festgestellt, dass es stark regnet (R). Jedesmal, wenn der Damm voll ist (D) und es stark regnet, ereignet sich eine Katastrophe (K). Und im Winter ist der Damm immer voll. Wir wissen auch, dass der Dammwächter nur im Winter im Urlaub (U) ist. Folgt daraus, dass es zu einer Katastrophe kommen wird?

Die Tatsachen sind W und R, und die übrigen Regeln sind

$$D \wedge R \Rightarrow K \quad , \quad W \Rightarrow D \quad , \quad U \Rightarrow W$$

Die Frage K, ob eine Katastrophe eintritt, hat genau dann die Antwort „ja“, wenn die offenbar zu einer Horn-Formel äquivalente Formel

$$W \wedge R \wedge (D \wedge R \Rightarrow K) \wedge (W \Rightarrow D) \wedge (U \Rightarrow W) \wedge \neg K$$

unerfüllbar ist. Das kann man direkt ablesen: aus der Tatsache W folgt D, was in Konjunktion mit der Tatsache R impliziert, dass K gilt. Das macht die Hornformel in der Tat unerfüllbar, und die Antwort auf die Frage K lautet: ja, die Katastrophe tritt ein.

8.1.5 Bemerkung. In der Logik-Programmierung befaßt man sich mit der Bearbeitung von Horn-Klauseln, typischerweise in der Programmiersprache PROLOG. Hier wird die Syntax (Implikation von rechts nach links!)

$$\mathrm{A.} \quad , \quad \mathrm{B} :- \mathrm{A}_0, \mathrm{A}_2, \cdots, \mathrm{A}_{n-1}. \quad , \quad ?- \mathrm{A}_0, \mathrm{A}_2, \cdots, \mathrm{A}_{n-1}.$$

für die Tatsache A, die Regel $A_0 \wedge A_2 \wedge \ldots \wedge A_{n-1} \Rightarrow B$, bzw. die Frage $\neg A_0 \vee \neg A_2 \vee \cdots \vee \neg A_{n-1}$ verwendet. Dabei soll $:-$ eine Implikation $\Leftarrow$ andeuten.

8.1.6 Beispiel. Das PROLOG-Programm

```
A:- B, C.
A:- D.
B:- A, D.
C.
D:- E.
? - D.
```

entspricht der folgenden Hornformel:

$$(A \vee \neg B \vee \neg C) \wedge (A \vee \neg D) \wedge (B \vee \neg A \vee \neg D) \wedge C \wedge (D \vee \neg E) \wedge \neg D$$

8.1.7 Bemerkung. Jede Hornformel ohne Frage-Klauseln ist erfüllbar: Wir können einfach alle positiven Literale mit 1 belegen.

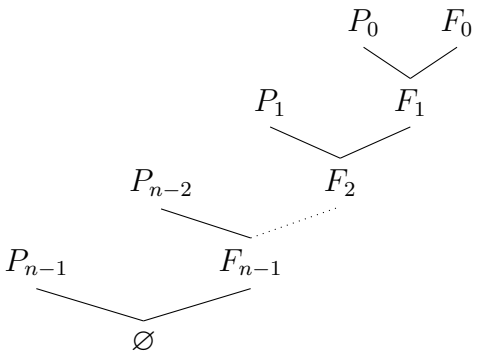
8.2 Die SLD-Resolutionsmethode

Die Resolventenbildung benötigt also zwingend eine Regel als Argument. Die Resolvente zweier Regeln ist wieder eine Regel, da genau eines der beiden positiven Literale verschwindet. Hier verwenden wir, dass kein Literal mehrfach in einer Klausel auftreten darf. Dagegen ist die Resolvente einer Regel mit einer Frage wieder eine Frage: das einzige positive Literal verschwindet. Damit sind Resolventen zweier Regeln bei der Suche nach $\varnothing$ irrelevant.

8.2.1 Algorithmus. Unter dem Begriff **SLD-Resolution** (*Selective Linear Definite clause resolution*)[1]versteht man eine Einschränkung der Resolutionsmethode 6.0.17 für Hornformeln, die die Bildung von Resolventen zweier Regeln vermeidet.

Man beginnt mit einer positiven Klausel P_0 und einer Frage F_0 und versucht eine Resolvente F_1 zu formen – die eine Frage sein muß. Existiert F_1, versucht man mit Hilfe einer weiteren Regel P_1 der ursprünglichen Hornformel eine nächste Frage F_2 als Resolvente zu bilden, usw. Dabei können Obermengen wie zuvor gestrichen werden.

Wenn die ursprüngliche Formel nicht erfüllbar ist, muß eine Frage-Klausel F_o und eine direkte Herleitung der leeren Resolvente aus F_0 existieren:

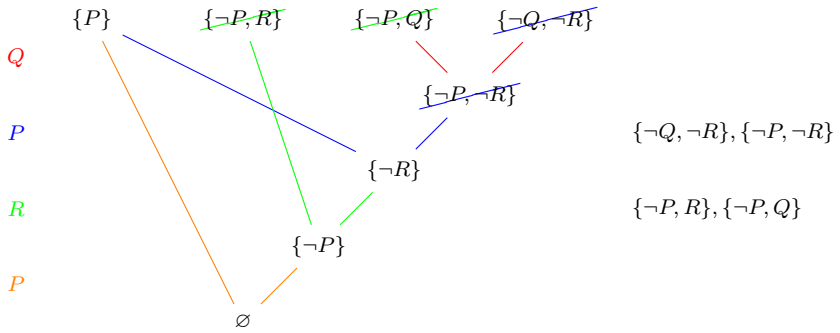


Dabei brauchen die Regeln P_i nicht paarweise verschieden zu sein.

8.2.2 Beispiel. Die Formel

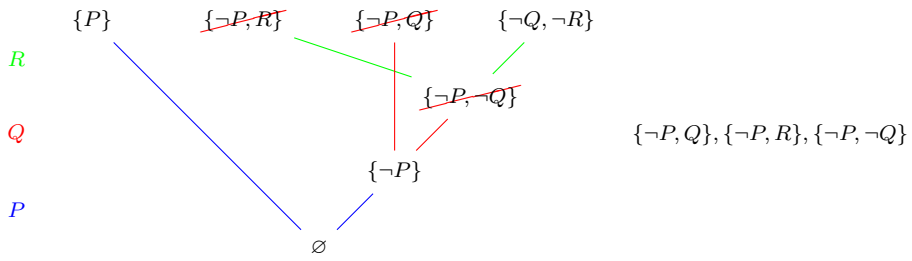
$$F := P \wedge (\neg P \vee R) \wedge (\neg P \vee Q) \wedge (\neg Q \vee \neg R)$$

mit einer einzigen Frage-Klausel am Ende hat z.B. folgende SLD-Resolution:



[1]Eigentlich geht es um „Linear resolution with unrestricted Selection for Definite clauses“, aber die Abkürzung „LSD“ mag manchem zu unheimlich gewesen zu sein. Die im Netz noch häufiger auftretende Abkürzung „SDL“ bezieht sich dagegen auf die *Simple Direct media Layer*.

und ist somit nicht erfüllbar. Es fällt auf, dass die Variable P zweimal zur Resolventenbildung verwendet wurde (das erste solche Beispiel). Das deutet an, dass man schneller zum Ziel kommen könnte. In der Tat gilt auch

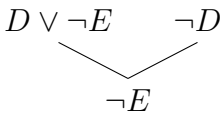


Die in der ursprünglichen Methode möglichen Resolventen $\{R\}$ und $\{Q\}$ werden in der SDL-Variante ignoriert.

8.2.3 Beispiel. Für das vorige PROLOG-Beispielprogramm 8.1.6 mit der Hornformel

$$(A \vee \neg B \vee \neg C) \wedge (A \vee \neg D) \wedge (B \vee \neg A \vee \neg D) \wedge C \wedge (D \vee \neg E) \wedge \neg D$$

gibt es nur eine SLD-Resolvente:



Deswegen ist die Formel erfüllbar und die Frage nach D negativ zu beantworten.

8.2.4 Satz. *Für Hornformeln ist die SLD-Resolution ist korrekt und vollständig: Eine Hornformel ist genau dann erfüllbar, wenn keine SLD-Resolvente leer ist.*

Einen Beweis findet man z.B. in Kreuzer, Kühling „Logik für Informatiker“ (Pearson Studium 2006), Satz 3.17.

8.3 Der Markierungsalgorithmus

Der folgende Algorithmus funktioniert für jede Formel F in KNF, wird aber erst für Hornformeln effizient.

Vorüberlegung: F wird wahr, wenn

- jede Klausel mindestens ein positives Literal enthält; man wählt ein solches Literal pro Klausel, belegt die entsprechende Variablen mit 1, und alle übrigen Variablen beliebig;
- jede Klausel mindestens ein negatives Literal enthält, man wählt ein solches Literal pro Klausel, belegt die entsprechende Variablen mit 0, und alle übrigen Variablen beliebig.

Wir denken uns zunächst alle vorkommenden Variablen mit 0 belegt, etwa durch α_0. Wird F auf diese Weise nicht 1 zugewiesen, also $\widehat{\alpha}_0(F) = 0$, so versuchen wir, diese Belegung rekursiv geschickt abzuwandeln, zu α_1, $\alpha_2 \ldots$, um dadurch mehr Klauseln den Wert 1 zuzuweisen, ohne den Wert von Klauseln zu ändern, denen schon 1 zugewiesen ist. Für die nunmehr auf 1 abgebildeten Variablen markieren wir alle Vorkommen in F, daher der Name. Das hilft auch klauseln zu identifizieren, deren Wahrheitswert noch von 0 auf 1 geändert werden muß.

8.3.1 Definition. Für $i \in \mathbb{N}$ sei $\mathcal{N}_i$ die Menge aller Klauseln, in denen noch kein positives Literal, aber schon alle negativen Literale markiert sind. Speziell für Hornformeln besteht $\mathcal{N}_0$ genau aus den Tatsachen.

8.3.2 Algorithmus.
Eingabe: $F \in \mathcal{F}^{KNF}$.

Solange $\mathcal{N}_i$ nicht leer ist,

- wähle ein positives Literale in den Formeln in $\mathcal{N}_i$ aus, etwa A_k (zwecks technischer Vereinfachung etwa dasjenige mit dem kleinsten Index), und markiere alle Vorkommen von A_k in F (auch die negativen); das spezifiziert $\alpha_{n+1}(A_k) = 1$;
- entferne alle Klauseln mit dem gerade markierten positiven Literal A_k aus $\mathcal{N}_i$; füge stattdessen alle Klauseln hinzu, in denen nunmehr alle negativen Literale markiert sind, aber kein positives Literal.
- Sobald $\mathcal{N}_i$ leer ist,
  - ▷ sind alle Klauseln mit mindestens einem negativen Literal ohne Markierung wahr, da die zugehörige Variable immer noch mit 0 belegt ist;
  - ▷ haben alle Klauseln mit mindestens einem positiven Literal, in denen die Variablen aller negativen Literale markiert sind, auch ein markiertes positives Literal und sind folglich wahr;
  - ▷ sind alle Klauseln ohne positive Literale (vulgo „Fragen“), deren sämtliche Variablen markiert sind, falsch.
- Solange mit 0 bewertete Frage-Klauseln auftreten, versuchen wir durch Backtracking die bisher konstruierte Belegung zu modifizieren: wir suchen den letzten Schritt, in dem neben der tatsächlich zur Markierung gewählten Variablen A_k eine andere (mit größerem Index) zur Auswahl stand, und fahren mit dieser anstelle von A_k fort; A_k wird wieder auf 0 gesetzt.
- Genau dann, wenn für jede der so konstruierten Belegungen eine mit 0 belegte Frage-Klausel existiert, ist F nicht erfüllbar.

Das Backtracking kann zu exponentieller Laufzeit führen. Beschränkt man sich hingegen auf Horn-Formeln, so entfällt die Notwendigkeit des Backtrackings, denn jede Klausel in $\mathcal{N}_i$ hat genau ein positives Literal und bietet damit keine Wahlmöglichkeit. Der Algorithmus vereinfacht sich dann entsprechend:

8.3.3 Algorithmus.
Eingabe: Hornformel F.

(M0) Markiere jedes Vorkommen jeder Tatsache A; diese formen $\mathcal{N}_0$, und solange noch Tatsachen in $\mathcal{N}_i$ vorkommen, wählen wir deren Variablen statt solche mit kleinstem Index unter den positiv in $\mathcal{N}_i$ auftretenden;

(M1) (Rekursiv) Sobald in einer Regel $\neg A_0 \vee \neg A_2 \vee \cdots \vee \neg A_{n-1} \vee B$, die Variablen aller negativen Literale markiert sind, markiere jedes Vorkommen der Variable B. Falls keine neuen Variablen markiert werden können, gehe zu Schritt (M2).

(M2) Besteht mindestens eine Frage-Klausel nur aus Literalen mit markierten Variablen, erfolgt die Ausgabe „F unerfüllbar“, andernfalls „F erfüllbar“.

8.3.4 Beispiel. In Beispiel 8.2.2 kann man in

$$F := P \wedge (\neg P \vee Q) \wedge (\neg Q \wedge \neg R) \wedge (\neg P \vee R)$$

zunächst das Atom P markieren (einzige Tatsache)

$$F := P \wedge (\neg P \vee Q) \wedge (\neg Q \wedge \neg R) \wedge (\neg P \vee R)$$

während Schritt (M1) die Literale Q und R markiert werden.

$$F := P \wedge (\neg P \vee Q) \wedge (\neg Q \wedge \neg R) \wedge (\neg P \vee R)$$

Das liefert eine vollständig markierte Frage, also ist F nicht erfüllbar.

8.3.5 Satz. *Der Markierungsalgorithmus ist korrekt und braucht Zeit $O(n^2)$ für Hornformeln mit n Literalen (einschließlich Wiederholungen).*

Beweis.
Korrektheit: Wir zeigen erst, dass jede die Hornformel F erfüllende Belegung α alle markierten Variablen mit 1 belegt. Für Schritt (M0) ist das trivial, denn falls A eine Tatsache und somit eine Klausel ist, muss $\alpha(A) = 1$ gelten. Beim rekursiven Schritt (M1) erinnern wir uns an die Implikationsform $A_0 \wedge A_1 \wedge \ldots A_{n-1} \Rightarrow B$ der Regeln: sind bereits alle A_i markiert und gilt damit nach Voraussetzung $\alpha(A_i) = 1$, $i < n$, so wird auch B markiert. Damit die Implikation, und somit die Klausel, wahr wird, muß wegen $\widehat{\alpha}(\bigvee_{i<n} A_i) = 1$ auch $\alpha(B) = 1$ gelten.
In Schritt (M2) kann wegen $\alpha(F) = 1$ keine Frageklausel gefunden werden, in der alle Variablen markiert sind. Die Ausgabe „erfüllbar“ ist damit richtig.
Umgekehrt verhindert eine Frage mit lauter markierten Variablen die Existenz einer Belegung α mit $\widehat{\alpha}(F) = 1$, insofern ist die Ausgabe “unerfüllbar” korrekt.
Lassen sich für keine Frage alle Variablen markieren, so setzen wir

$$\alpha(A) := \begin{cases} 1 & \text{falls } A \text{ markiert} \\ 0 & \text{sonst} \end{cases}$$

Damit haben alle positiven Klauseln den Wert 1: Für Tatsachen ist das klar, während bei Regeln entweder alle Variablen markiert sind, insbesondere die im positiven Literal,

oder mindestens eine Variable in einem negativen Literal nicht markiert ist, was dem Literal den Wert 1 zuweist. Dieses letzte Argument greift auch bei Fragen. Damit werden alle Klauseln von $\widehat{\alpha}$ auf 1 abgebildet, also auch F.

Zeitkomplexität: Bei n Literalen treten höchstens n Variablen auf, also auch höchstens n Tatsachen. Also braucht Schritt (M0) Zeit $O(n)$ und Schritt (M1) hat höchstens n Durchläufe, die jeweils eine atomare Aussage abarbeiten. Ein Durchgang in Schritt (M1) läuft in $O(n)$ Schritten über alle Literale der Formel, weshalb Schritt (M1) als ganzes in der Zeitklasse $O(n^2)$ liegt. Schritt (M2) benötigt $O(n)$ Zeit. Insgesamt ergibt sich eine Zeitkomplexität von $O(n + n^2 + n) = O(n^2)$. □

8.4 Zusammenfassung

In der Aussagenlogik stellen wir Fragen des Typs:

- Ist eine Formel erfüllbar?
- Ist eine Formel eine Tautologie?
- Folgt eine Formel aus gegebenen Formeln?

Diese drei Probleme sind äquivalent: siehe dazu die Sätze 2.7.2 und 5.0.4.

Wir haben verschiedenen Algorithmen vorgestellt, die diese Fragen beantworten: die Berechnung der Wahrheitstabelle, die Resolutionsmethode und die Natürliche Deduktion. Keiner dieser Algorithmen ist effizient: alle können Zeit exponentiell in der Größe der Formel benötigen. Und es ist eher unwahrscheinlich, dass je ein Polynomialzeit-Algorithmus dafür gefunden wird, da die Erfüllbarkeit ein NP-vollständiges Problem ist (vergl. Theoretische Informatik 2).

Die Situation für Hornformeln ist wesentlich besser: wir verfügen über den Markierungsalgorithmus mit Zeitkomplexität $O(n^2)$ sowie eine effiziente Variante der Resolutionsmethode.

Teil II

Prädikatenlogik

9. Syntax der Prädikatenlogik

9.1 Einleitung

Die Aussagenlogik ist universell in allen Bereichen anwendbar, wo potentiell wahre oder falsche Aussagen formuliert und miteinander kombiniert werden können, sei es die Herstellung von Würstchen oder die Theorie der Quantengraviation. Ihre recht einfache mathematische Struktur ist Konsequenz dieser beschränkten Ausdrucksfähigkeit. So ist die Aussagenlogik nicht in der Lage, Aussagen über die Elemente einer spezifischen (nichtleeren) mathematischen Struktur zu formulieren, z.B. die natürlichen Zahlen. Dazu bedarf es der Prädikatenlogik (auch als „Logik der ersten Stufe“ bekannt), die sich an die Gegebenheiten der mathematischen Strukturen von Interesse anpassen läßt und dann viel ausdrucksstärker ist als die Aussagenlogik.

Praktisch jede mathematische Struktur von Interesse läßt sich spezifizieren als (nichtleere?) Menge A, auf der

- bestimmte Operationen $A^{n_i} \xrightarrow{f_i} A$, $i \in I$, und
- bestimmte Relationen $A^{m_j} \xrightarrow{R_j} \mathbf{2} = \{0, 1\}$, $j \in J$, (aufgefaßt als charakteristische Funktionen der entsprechenden Teilmengen $R_j \subseteq A^{m_j}$)

definiert sind, die gewissen Axiomen genügen. Sollten auch partielle Operationen gewünscht sein, so ist dies keine Einschränkung, denn man kann dies mittels der charakteristeischen Funktion des Definitionsbereichs ausdrücken.

Z.B. ist auf den natürlichen Zahlen $\mathbb{N}$ eine Addition $+$ und eine Multiplikation $\cdot$ mit neutralen Elementen 0 bzw. 1, sowie eine lineare Ordnung $\leq$ mit kleinstem Element 0 definiert. Die Operationen $+$ und $\cdot$ sind assoziativ und kommutativ und genügen den Distributivgesetzen.

Ähnlich kann man Gruppen, Ringe, Verbände, topologische Räume, ja selbst die Aussagenlogik beschreiben.

Für die Elemente einer mathematischen Struktur will man in der Lage sein, über

– Eigenschaften solcher Elemente,

– Relationen zwischen derartigen Elementen, und

– Funktionen, die auf Teilmengen solcher Elemente definiert sind.

zu sprechen. Die Prädikatenlogik stellt die entsprechende Sprache zur Verfügung, angepaßt an die jeweilige mathematische Struktur.

Wir werden synonym auch die Begriffe „Operation“ anstelle von Funktion und „Prädikat“ anstelle von Relation verwenden.

Die Grundidee besteht darin, die mathematische Struktur von Interesse zu **abstrahieren**. Das geschieht in drei Schritten:

- Zunächst werden **Platzhalter** für die Operationen (oder Funktionen) und Prädikate (oder Relationen) von Interesse in der gewünschten Stelligkeit zu einer sog. **Signatur** Σ zusammengefaßt. So kann etwa ein 2-stelliges Funktionssymbol die Addition, oder ein 2-stelliges Relationssymbol die Teilbarkeit von Zahlen symbolisieren.

  Achtung: Es wird immer ein spezielles 2-stelliges Prädikat „$\doteq$“ geben, das die Gleichheit symbolisiert und eine besondere Rolle spielt, vergl. Abschnitt 9.5.

  Da bisher keine Grundmenge spezifiziert ist, sind obige Platzhalter zunächst **uninterpretiert**, erst im Rahmen der Semantik werden sie **interpretiert**, d.h. ihnen werden konkrete Operationen bzw. Relationen auf einer konkreten Menge zugewiesen.

  Neben der intendierten Interpretation sind i.A. auch viele andere Interpretationen möglich!

- Mit Hilfe einer abzählbaren Menge von **Variablen** x_i, $i \in I\!N$, und der Funktionssymbole in der Signatur läßt sich rein syntaktisch eine kanonische Grundmenge konstruieren um darauf Logik betreiben zu können: die Menge $\mathcal{T}_\Sigma$ der **Σ-Terme**. Dabei handelt es sich um Bäume mit Variablen als Blättern und und Funktionssymbolen der entsprechenden Stelligkeit als weiteren Knoten (ggf. auch weiteren Blättern bei Stelligkeit 0), ganz analog zur Menge $\mathcal{F}$ der Aussagenlogik. Auf $\mathcal{T}_\Sigma$ lassen sich jedem n-stelligen Funktionssymbol f aus Σ kanonisch eine n-stellige Funktion $(\mathcal{T}_\Sigma)^n \xrightarrow{\bar{f}} \mathcal{T}_\Sigma$ zuordnen, die n Bäume unter einer neuen Wurzel mit Label f zu einem Baum zusammenfaßt. Mit diesen Operationen spricht man dann auch von der **freien Term-Algebra** über Σ.

- Die Relationssymbole aus Σ wie auch $\doteq$ haben im Gegensatz zu den Funktionssymbolen keine kanonsche Interpretation in $\mathcal{T}_\Sigma$. Man wird allerdings verlangen, dass die Interpretation von $\doteq$ hinsichtlich der oben beschriebenen Interpretation der Funktionssymbole eine Kongruenzrelation ist, vergl. Abschnitt 9.5, es braucht aber nicht die syntaktische Gleichheit von Termen sein.

  Mit Hilfe der Relationssymbole aus Σ und von $\doteq$ konstruiert man nun mit Argumenten aus $\mathcal{T}_\Sigma$ die **atomaren Formeln** der Prädikatenlogik: die Aussage, dass gewisse Terme zueinander in Relation stehen, oder „gleich“ sind, kann wahr oder falsch sein. Atomare Formeln können dann mit den Junktoren der Aussagenlogik zu allgemeineren Formeln kombiniert werden.

  Aber weil die atomaren Formeln aus Termen aufgebaut und somit durch Variablen parametrisiert sind, kann man auch fragen, ob Formeln für all ihre Parameterwerte wahr sind, oder ob überhaupt Parameterwerte existieren, die die Formel wahr machen. Mit anderern Worten, man kann Formeln **quantifizieren**. Dazu wird das Alphabet der Logik um zwei Symbole $\forall$ und $\exists$, die sog. **Quantoren** erweitert; Ausdrücke der Form $\forall : x$ und $\exists : x$ formalisieren dann „für alle x“ bzw. „es existiert ein x“.

In den resultierenden Formeln kann man dann das Auftreten von Variablen danach klassifizieren, ob sie **gebunden** auftreten, also im Wirkungsbereich eines Quantors, oder nicht, im letzteren Fall spricht man von **freien Variablen**.

- Verglichen mit der intendierten Interpretation der Funktionssymbole der Signatur können deren kanonischen Interpretationen in der Term-Algebra noch bestimmte Eigenschaften fehlen. Abstrahiert z.B. $f \in \Sigma$ die Addition von Zahlen, so wird die Interpretation $\bar{f}$ in der Term-Algebra nicht notwendig kommutativ sein, zumindest sind die Terme $f(x_0, x_1)$ und $f(x_1, x_0)$ sind syntaktisch verschieden. Um die „Gleichheit" bestimmter Terme zu „erzwingen", kann man eine Menge E von Term-Paaren vorgeben, sog. **Gleichungen**, und verlangen, dass die entsprechenden atomaren Formeln mit $\doteq$ gelten sollen. Außerdem kann man die Gültigkeit bestimmter Formeln ohne $\doteq$ als Wurzel vorschreiben: ist R die Abstaktion der $\leq$ -Relation auf den Zahlen, und ist die Konstante $\bar{0}$ die Abstraktion der kleinsten Zahl 0, so hat die Formel $R(\bar{0}, x_0)$ in der Term-Algebra unter einer Belegung α nicht notwendig den Wert 1. Alle Formeln, die immer wahr sein sollen, faßt man unter dem Begriff **Axiome** zusammen.

Die Signatur und die Axiome bilden zusammen eine sog. **Theorie**. Die Semantik beschäftigt sich dann mit **Modellen** dieser Theorie, konkreten Mengen mit konkreten Interpretationen der Opertionen und Variablen (und folglich der Terme) sowie der Relationen (und folglich der Formeln), so dass alle Axiome erfüllt sind. Was die Operationen angeht, so liefert die Interpretation der Variablen einen Homomorphismus von der freien Term-Algebra in eine Algebra auf der Grundmenge des Modells. Für alle Formeln kann man in einem solchen Modell prinzipiell, wenn auch nicht notwendig effizient, feststellen, ob sie erfüllt sind.

Das legt eine weitere Klassifikation der Formeln der Prädikatenlogik nahe: Formeln, die in jedem Modell einer gegebenen Theorie gelten, heißen **allgemeingültig**, solche, die nur in manchen Modellen gelten heißen **erfüllbar**.

Bevor wir die hier eingeführten Begriffe formalisieren, wollen wir anhand eines praxisorientierten (unmathematischen) Beispiels den auch dort sinnvollen Einsatz von Prädikaten, Funktionen und Quantoren demonstrieren.

9.1.1 Beispiel (politisch korrekte Version)**.** Sagt eine Frau, die ein Porträt beobachtet: „Geschwister habe ich keine. Und die Mutter dieser Frau ist meiner Mutter Tochter." Wer ist da abgebildet?

Wir führen zwei Konstanten für die beiden Frauen im Beispiel ein:

$a =$ die abgebildete Frau

$b =$ die Beobachterin.

Die erste Aussage arbeitet mit dem zweistelligen Prädikat „Geschwister"

$$G(x, y) \quad \text{steht für} \quad „x \text{ und } y \text{ sind Geschwister}".$$

Die Beobachterin b behauptet hier: $G(b, y)$ ist immer, d.h. für jedes y, falsch. Das schreiben wir mit dem **All-Quantor** $\forall$ in der Form

$$\forall y : \neg G(b, y) \tag{9.1}$$

Für die zweite Aussage brauchen wir das zweistellige Prädikat „Tochter von“

$$T(x, y) \quad \text{steht für} \quad „x \text{ ist Tochter von } y“.$$

Wir könnten auch „Mutter von“ als Prädikat nehmen, aber wir haben etwas besseres: eine Funktion $m(x)$, die jeder Person x ihre (eindeutig bestimmte) Mutter $y = m(x)$ zuweist. Die zweite Aussage der Beobachterin formulieren wir nun als

$$T(m(a), m(b)) \quad \text{oder effizienter als} \quad m(m(a)) = m(b) \tag{9.2}$$

Die alternative Aussage benutzt noch ein wichtiges zweistelliges Prädikat: die Gleichheit „=“. Sie nimmt eine Sonderstellung ein und wird als immer verfügbares Prädikat mit fester Interpretation verwendet.

Im Beispiel nicht explizit erwähnt, aber implizit zur Aufgabe gehörig, ist die Tatsache, dass jemand (Variable z) ohne Geschwister das einzige Kind ihrer Mutter ist:

$$\big(\forall y : \neg G(z, y)\big) \Rightarrow \forall u : (m(u) = m(z) \Rightarrow u = z) \tag{9.3}$$

Was folgt aus den Prämissen (9.1), (9.2) und (9.3)? Wir können für die Variable z in (9.3) die Konstante b substituieren und erhalten

$$\forall y : \neg G(b, y) \Rightarrow \forall u : (m(u) = m(b) \Rightarrow u = b)$$

Dies, zusammen mit (9.1), ergibt (wegen modus ponens)

$$\forall u : (m(u) = m(b) \Rightarrow (u = b))$$

Ließt man dies als verallgemeinerte Konjunktion, so müssen im Falle ihrer Wahrheit alle individuellen Instanzen wahr sein. Substituieren wir speziell $m(a)$ für u, so gilt

$$m(m(a)) = m(b) \Rightarrow m(a) = b.$$

Modus ponens liefert aufgrund von (9.2) $m(a) = b$. D.h., a ist die Tochter der Beobachterin b.

Wir haben hier mit konkreten zweistelligen Relationen G (Geschwister) und $=$ (Gleichheit) gearbeitet (die Tochter-Relation T wurde nur übergangsweise verwendet). Dazu kamen 0-stellige Funktionen oder Konstanten a und b und die einstellige Mutter-Funktion $m(x)$. Wichtig ist, dass jeder Name ein eindeutig bestimmtes Objekt (Relation bzw. Funktion) auf der konkret spezifizierten Grundmenge (hier: alle Personen) beschreibt.

Man könnte aber auch auf einer ganz anderen Menge, z.B. $\mathbb{N}$, eine zweistellige Relation G (und ggf. auch T) definieren, dazu eine einstellige Funktion m und Konstanten a und b, dann Aussagen formen und deren Wahrheitsgehalt überprüfen. Dabei ist allerdings nicht gewährleistet, dass die Formeln (9.1), (9.2) und (9.3) in dieser neuen Situation gelten. Das hängt von den konkret gewählten Interpretation der Daten in $\mathbb{N}$ ab. Es mag Formeln geben, die in beiden, oder in nur einer, oder in keiner der beiden Interpretionen gelten. Dagegen gilt $x = x$ in jeder möglichen Interpretation, und $\neg(x = x)$ in keiner.

9.2 Signaturen

Mathematische Strukturen auf konkreten Mengen sind meist durch Funktionen (oder Operationen, evtl. auch partiell) und Relationen (oder Prädikate) auf diesen Mengen gegeben. Zunächst sind diese Begriffe zu präzisieren:

9.2.1 Definition. Gegeben sei eine Menge A.

(a) Für jede natürliche Zahl $n \in \mathbb{N}$ bezeichent A^n die Menge der geordneten n-Tupel mit Komponenten in A, genauer, die Menge aller Funktionen $n = \{0, 1, \ldots, n-1\} \longrightarrow A$. Dabei haben wir die Zahl n mit der Menge ihrer Vorgänger identifiziert.

Zwei geordnete n-Tupel $\vec{a} = \langle a_i : i < n \rangle$ und $\vec{b} = \langle b_i : i < n \rangle$ stimmen genau dann überein, wenn sie komponentenweise gleich sind, d.h., $a_i = b_i$, $i < n$.[1]

Speziell im Fall $n = 2$ schreibt man häufig $A \times A$ anstelle von A^2 und spricht vom **cartesischen Produkt**. Weiter besitzt für $n = 0 = \varnothing$ die Menge A^0 genau ein Element ε, die Inklusion der leeren Menge in A; diese wird häufig als **leeres Wort** bezeichnet.

(b) Ein **n-stelliges Prädikat** auf A ist eine Teilmenge R von A^n, dem n-fachen kartesischen Produkt der Menge A mit sich, oder äquivalent eine Funktion $A^n \xrightarrow{R} \{0, 1\}$. Der Wahrheitswert der Behauptung, dass ein n-Tupel $\vec{x} \in A^n$ zur Relation $R \subseteq A^n$ gehört, wird üblicherweise in der Logik durch $R(\vec{x})$ ausgedrückt anstelle von $\vec{x} \in R$; im binären Fall verwendet man gelegentlich auch die Infix-Schreibweise $x\,R\,y$ statt $R(x, y)$, etwa $x < y$.

Speziell im Fall $n = 0$ gibt es genau zwei Teilmengen der ein-elementigen Menge $A^0 = \{\varepsilon\}$: die leere Menge $\varnothing$ und die nichtleere Menge $\{\varepsilon\}$. Entsprechend gibt es genau zwei Abbildungen von $\{\varepsilon\}$ nach $\{0, 1\}$. Diese lassen sich mit den **Wahrheitswerten** 0 und 1 identifizieren

Einstellige Prädikate entsprechen Teilmengen von A, oder **Eigenschaften**.

(c) Eine **n-stellige Operation** ist eine Abbildung $A^n \xrightarrow{f} A$, also eine Teilmenge $f \subseteq A^{n+1}$, die, aufgefaßt als binäre Relation $f \subseteq A^n \times A$, zwei Eigenschaften hat:

- ▷ f ist **total**, d.h., zu jedem $\vec{x} \in A^n$ existiert ein $y \in A$, das zu $\vec{x}$ in Relation steht;
- ▷ f ist **einwertig**, d.h., stehen y und z aus A zu $\vec{x} \in A^n$ in Relation, so gilt $y = z$.

Folglich steht zu jedem $\vec{x} \in A^n$ genau ein $y \in A$ in Relation, das traditionell mit $f(\vec{x})$ bezeichent wird. Achtung: das paßt leider nicht mit der Schreibweise $R(\vec{x})$ aus Teil (b) zusammen! Die überwiegende Verwendung kleiner Buchstaben für Funktionen und großer Buchstaben für Relationen sollte helfen, Verwirrung zu vermeiden.[2]

Weil A^0 ein-elementig ist, können wir 0-stellige Funktionen auf A mit Elementen von A identifizieren. Man spricht dann auch von **Konstanten**.

[1] Dies ist die *Spezifikation* für geordnete n-Tupel, sie kann auf verschiedene Weise *realisiert* werden, aber das interessiert uns hier nicht.

[2] Vermutlich wurden früher Funktionen nicht als spezielle Relationen aufgefaßt.

In einer Signatur sammelt man nur Platzhalter, oder Namen **symbolischer Natur**, die erst nach Spezifikation einer Menge $\mathcal{A}$, an der man konkret interessiert ist, als echte Funktionen bzw. Relationen auf $\mathcal{A}$ interpretiert werden können (siehe Kapitel 10). Genauer:

9.2.2 Definition. Eine **Signatur** Σ besteht aus

- einer Liste $\Sigma_F$von Funktionssymbolen (meist mit kleinen Buchstaben bezeichnet) mit gegebenen Stelligkeiten $0, 1, 2, 3, \ldots$; die 0-stelligen Funktionssymbole heißen auch **Konstanten**.
- einer Liste Σ_P von Prädikatensymbolen (von $\doteq$ verschieden, meist mit großen Buchstaben bezeichnet) mit gegebenen *positiven* Stelligkeiten $1, 2, 3, \ldots$.

Jede dieser Listen darf auch leer sein; das immer verfügbare Prädikat $\doteq$ braucht hier nicht explizit erwähnt zu werden.

9.2.3 Bemerkung. Formal spricht nichts dagegen, auch 0-stellige Prädikatssymbole zuzulassen. Da ihnen in der Semantik aber beliebige Wahrheitswerte zugewiesen werden könnten, entsprechen sie den atomaren Aussagen der Aussagenlogik, tragen also nichts zur Beschreibung der mathematischen Stuktur von Interesse bei. Insofern werden wir in dieser VL auf sie verzichten.

9.2.4 Beispiel. Im Beispiel 9.1.1 haben wir die Signatur $\Sigma = \langle m, a, b; G, T\rangle$ benutzt. Die Prädikatssymbole G und T sind zweistellig, m ist ein einstelliges Funktionssymbol und a, b sind als Konstanten 0-stellig. Allerdings war die Signatur in Beispiel 9.1.1 bereits *interpretiert* worden: die auftretenden Variablen waren Platzhalter für Menschen, G und T waren konkrete Relationen zwischen Menschen, und m war eine konkrete einstellige Funktion. Eine derartige Σ *Struktur* ist der erste Schritt Richtung Semantik, vergl. Definition 10.0.1.

9.2.5 Bemerkung. Eine notationelle Differenzierung zwischen den Symbolen in einer Signatur und ihren Interpretationen in einem Modell wird in Definition 10.0.1 zwar vorgeschlagen, ist in der Praxis aber nicht immer durchzuhalten. Es liegt also in der Verantwortung der Leserin, kontextsensitiv G mal als Platzhalter für eine 2-stellige Relation, oder als konkrete 2-stellige Relation zu verstehen.

9.2.6 Beispiel. In der Arithmetik benutzen wir z.B. die Funktionen

$$+ \quad \text{und} \quad \cdot \quad \text{(beide zweistellig)}$$

die Konstanten 0 und 1 und die Relation

$$< \quad \text{(zweistellig)}$$

Die zugehörige Signatur hat die Form

$$\Sigma = \{+, \cdot, 0, 1 \ <\}$$

mit den entsprechenden Stelligkeiten. Ist folgende Aussage wahr?

$$\forall x : (x \cdot x > 0) \vee (x \cdot x = 0)$$

Das hängt von dem konkreten Model ab, in dem wir die Signatur interprtieren. Falls x aus dem Bereich der natürlichen Zahl stammwn soll, ist die Aussage wahr. Aber wenn x auch eine komplexe Zahl sein darf, lautet die Antwort „nein“. Der Wahrheitsgehalt einer Formel hängt also vom Kontext ab, dies werden wir im nächsten Abschnitt präzise formalisieren.

9.2.7 Beispiel. In der Theorie der gerichteten Graphen arbeitet man mit einem zweistelligen Prädikatensymbol R; wir interpretieren $R(x, y)$ als Existenz einer „gerichteten Kante“ von x nach y, häufig dargestellt als $x \longrightarrow y$. Schleifen (engl. loops), also Kanten von einem Knoten zu sich selbst, sind zulässig, aber es kann höchstens eine Kante zwischen den Knoten x und y geben. Die Liste der Funktionssymbole ist leer. Folgende Formel charakterisiert alle gerichteten Graphen ohne 2-Zyklen

$$\forall x : \forall y : (R(x, y) \wedge R(y, x) \Rightarrow (x = y))$$

Falls man dagegen verlangt, die Relation R möge symmetrisch sein:

$$\forall x : \forall y : (R(x, y) \Rightarrow R(y, x))$$

gehört zu jeder Kante eine Kante in der Gegenrichtung, $x \rightleftharpoons y$, was man als $x \longrightarrow\!\!\!\!\!- \; y$ abkürzen kann. Diese Bedingung charakterisiert also **ungerichtete** Graphen, deren Kanten auch als nichtleere Teilmengen der Knotenmenge mit höchstens zwei Elementen beschrieben werden können. Offenbar ist der Begriff des gerichteten Graphen der fundamentalere, der des ungerichteten Graphen ist ein Spezialfall. Es ist nicht offensichtlich, ob ungerichtete Graphen allein mit einer Signatur ohne weitere Axiome beschreiben werden können; ein Prädikatssymbol genügt jedenfalls nicht dafür.

9.3 Das Alphabet der Prädikatenlogik

Neben den Qantoren $\forall$ und $\exists$, die als neue logische Symbole gelten, brauchen wir in der Prädikatenlogik mehr Hilfssymbole als in der Aussagenlogik:

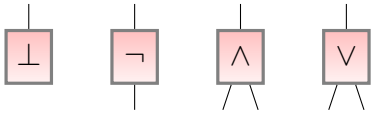
9.3.1 Definition. Das **Alphabet** der Prädikatenlogik besteht aus

(a) einer abzählbar unendlichen Menge von **Variablen** $x_0, x_1, x_2, \ldots$; für die Baumdarstellung der Terme handelt es sich um *Blätter*:

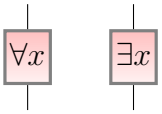
(b) einer zu $\mathcal{V}$ disjunkten Signatur Σ; deren Komponenten einschließlich $\doteq$ werden auch als **nichtlogische Symbole** bezeichnet. In der Baumdarstellung der Terme bzw. atomaren Formeln sind die Funktions- bzw. Relationssymbole der Stelligkeit k als Knoten mit k Ausgaben zu interpretieren

Der Input des R-Knotens wird den Übergang zwischen Termen und Formeln ermöglichen. Mit seiner Hilfe lassen sich die atomaren Formeln der Prädikatenlogik analog zu den atomaren Aussagen der Aussagenlogik behandeln, vergl. Definition 9.4.4.

(c) den Junktoren $\bot$, $\neg$, $\wedge$, $\vee$, und je nach Vereinbarung auch $\Rightarrow$, $\top$, $\Leftrightarrow \ldots$; diese dürfen natürlich nicht in $\mathcal{V}$ oder in Σ vorkommen; oder für die Baumdarstellung der Formeln



(d) den Quantoren $\forall$ und $\exists$, die zusammen mit jeder Variablen x weitere 1-stellige Junktoren liefern:



(e) und für die Lineardarstellung den Hilfssymbolen Klammern „)“ , „(“ , Komma „,“ und Doppelpunkt „:”

9.3.2 Bemerkung. Für die Variablen schreiben wir oft x, y, z, in bestimmten Kontexten nutzen wir auch andere Symbole wie ϵ oder δ. Dabei ist aber zu beachten, dass keine Namen aus der Signatur Σ verwendet werden.

9.4 Die Syntax der Prädikatenlogik

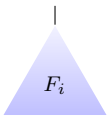
Die Funktions- bzw. Relationssymbole einer Signatur Σ sind für zwei völlig unterschiedliche rekursiv definierte syntaktische Kategorien verantwortlich, die *Terme* und die *Formeln*. Erstere haben einen algebraischen Charakter und dienen als abstrakte *Baupläne* für Elemente einer nicht weiter spezifischen Grundmenge. Mit den Formeln hingegen läßt sich Logik betreiben.

9.4.1 Definition. Die Menge $\mathcal{T}_\Sigma$ aller Σ-**Terme** ist die kleinste Menge von geordneten Bäumen, so dass

(a) alle Blätter

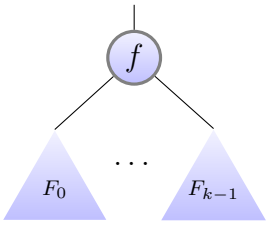
Formeln sind, wobei $c \in \Sigma$ ein 0-stelliges Funktionssymbol ist, , d.h., $\mathcal{V} \cup \Sigma_0 \subseteq \mathcal{T}_\Sigma$;

(b) falls $f \in \Sigma$ k-stelliges Funktionssymbol ist, und



, $i < k$

Terme sind, dann gilt dies auch für



Die Menge aller Σ-Terme, deren Variablen in $\mathcal{M} \subseteq \mathcal{V}$ liegen, bezeichnet man mit $\mathcal{T}_\Sigma(\mathcal{M})$.

Jede der Mengen $\mathcal{T}_\Sigma(\mathcal{M})$ mit $\mathcal{M} \subseteq \mathcal{V}$ ist ein Model für die Funktionssymbole in Σ; die einem k-stelligen Funktionssymbol f entsprechende Funktion faßt ein k-Tupel von Termen unter einer neuen Wurzel f zu einem Term zusammen.

Es gelten dieselben Regeln für die Lineardarstellung von Termen wie in Bemerkung 2.2.5, wobei man üblicherweise die Präfixdarstellung mit Klammern verwendet, hier also $f(F_0, \ldots, f_{k=1})$.

9.4.2 Beispiel.

- Falls $\Sigma = \{s\}$ nur ein 1-stelliges Funktionssymbol s enthält, sind die Terme neben den Variablen x_i genau die Syntaxbäume der Form $s(x_i), s(s(x_i)), \ldots$
- Falls $\Sigma = \{+, 0\}$ aus einer 2-stelligen Funktion $+$ und einer Konstante 0 besteht, haben wir Terme

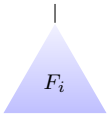
$$x,\ 0,\ x + y,\ x + 0,\ (x + 0) + (0 + 0),\ \ldots$$

9.4.3 Bemerkung. Die Menge $\mathcal{T}_\Sigma(\varnothing)$ aller Terme ohne Variablen, der sog. **Grundterme** wird im Zusammenhang mit der Resolutionsmethode für die Prädikatenlogik wichtig werden, siehe Kapitel 14. Z.B. gehören in der Signatur in Beispiel 9.2.6 die Terme $1 + 0$, $1 + 1$, $1 \cdot (0 + 1)$ zu $\mathcal{T}_\Sigma(\varnothing)$, während in Beispiel 9.4.7 unten $m(0, 2),\ f(a(0)), f(f(f(2)))$ Variablen-freie Terme sind. Für Signaturen ohne Konstanten gilt natürlich $\mathcal{T}_\Sigma(\varnothing) = \varnothing$.

Die Relationssymbole der Signatur (und $\doteq$) erlauben den Übergang von Termen zu prädikatenlogischen Formeln:

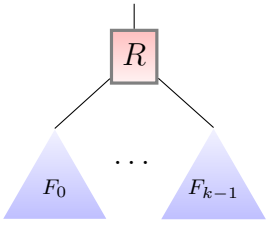
9.4.4 Definition. Die **Formeln** der Prädikatenlogik bilden die kleinste Menge geordneter Bäume, so dass

(a) falls R ein k-stelliges Relationssymbol in Σ oder $\doteq$ ist, und



, $i < k$

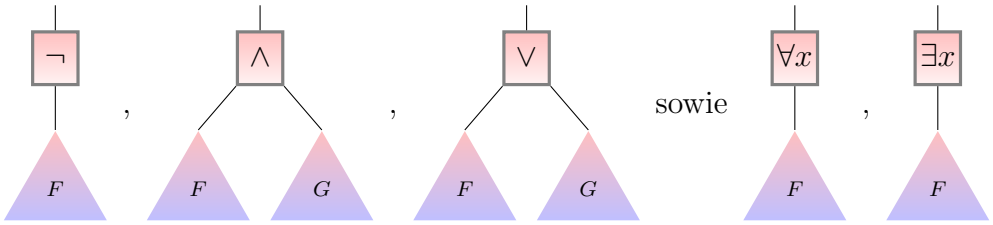
Terme sind, dann ist



eine Formel; man spricht bei diesem Übergang von Termen zu Formeln auch von **atomaren Formeln**

(b) falls

prädikatenlogische Formeln sind, dann gilt dies auch für



d.h., bezüglich der Junktoren $\neg$, $\wedge$ und $\vee$ (je nach Vereinbarung auch anderen) sowie $\forall x$ und $\exists x$, $x \in \mathcal{V}$, ist die Menge der prädikatenlogischen Formeln abgeschlossen.

In Anlehnung an Definition 9.4.1 bezeichnen wir für eine Menge $\mathcal{M}$ von Variablen mit $\mathcal{A}_\Sigma(\mathcal{M})$ bzw. $\mathcal{F}_\Sigma(\mathcal{M})$ die Mengen der (atomaren) Formeln über $\mathcal{M}$

9.4.5 Bemerkung.

(a) Die früheren Bemerkungen zur Lineardarstellung von Bäumen gelten auch für Formeln.

(b) Terme an sich haben keine logische, sondern nur algebraische Bedeutung. Die einzigen atomaren Formeln ohne Terme sind die 0-stelligen Relationssymbole (sofern in der Signatur vorhanden, vergl. Bemerkung 9.2.3); diese entsprechen atomaren Aussagen der Aussagenlogik, da ihr Wahrheitswert frei wählbar ist. Selbst wenn die Signatur keine Funktionssymbole hat, sind alle Variablen Terme, die in passender Zahl in die Relationssymbole bzw. die Gleichheit eingesetzt werden können.

(c) Wie in der Aussagenlogik benutzen wir $F \Rightarrow G$ für $G \vee \neg F$ und $F \Leftrightarrow G$ für $(F \wedge G) \vee (\neg F \wedge \neg G)$.

9.4.6 Notationelle Konvention. Wir erweitern die **Bindungskonventionen** der Aussagenlogik: formal kann man die Ausdrücke $\forall x :$ und $\exists x :$ ebenso wie $\neg$ als einstellige Junktoren auffassen (wenn auch abzählbar unendlich viele), und diese binden stärker als alle 2-stelligen Junktoren. Es bleibt dabei, dass $\wedge$ und $\vee$ stärker binden als $\Rightarrow$ und $\Leftrightarrow$.

Dann dürfen wir schreiben:

$$\forall x : F \vee \exists y : G \Rightarrow \neg H$$

was dann folgendes bedeutet:

$$((\forall x : F) \vee (\exists y : G)) \Rightarrow \neg H$$

Diese Formel darf nicht mit den folgenden verwechselt werden

$$\forall x : (F \vee (\exists y : G \Rightarrow \neg H)) \qquad \text{oder} \qquad (\forall x : F) \vee (\exists y : (G \Rightarrow \neg H))$$

die etwas ganz anderes besagen. Ein praktisches Beispiel für das unterschiedliche Zusammenspiel verschiedener Quantoren bei einer 2-stelligen Relation findet sich unter Wikipedia unter dem Begriff "Loving relation".

9.4.7 Beispiel.

1. Wie formalisiert man Aussagen wie „Für alle ϵ größer 0 gilt $F(\varepsilon)$"? Der Ausdruck „$\forall \epsilon > 0 : F(\varepsilon)$" ist keine syntaktisch korrekte Formel. Aber man kann die atomare Formel $\epsilon > 0$ als Prämisse separieren:

$$\forall \epsilon : \big(\varepsilon > 0 \Rightarrow F(\varepsilon)\big)$$

2. Analog formalisiert man „Es gibt ein δ größer 0 mit $G(\delta)$" als

$$\exists \delta : \big(\delta > 0 \wedge G(\delta)\big)$$

3. In der Analysis findet man die Aussage

   „die Funktion $f(x)$ ist stetig in Punkt 2"

   häufig wie folgt mit Hilfe von Quantoren leicht schlampig formuliert:

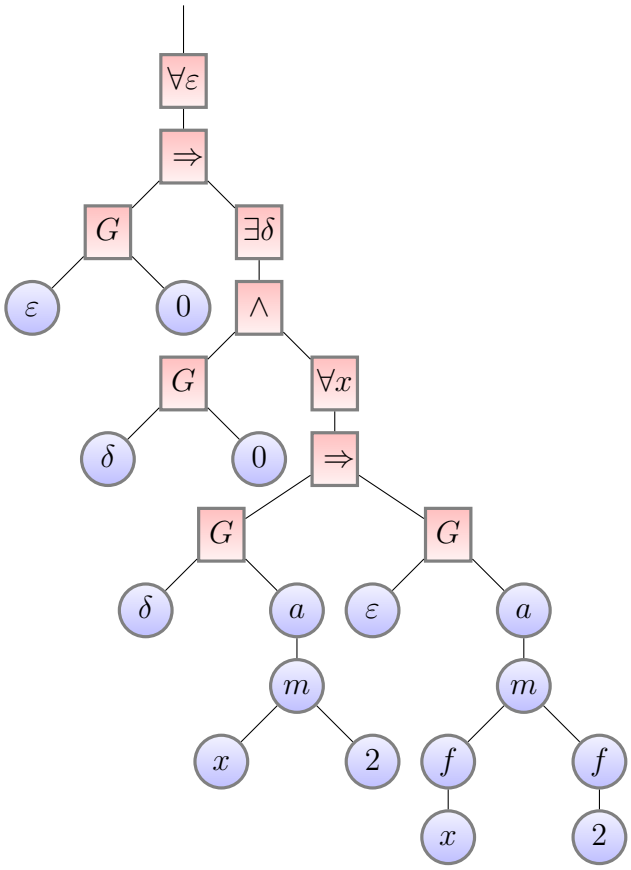
$$\forall \epsilon > 0 : \exists \delta > 0 : \forall x : \big(|x - 2| < \delta \Rightarrow (|f(x) - f(2)| < \epsilon)\big)$$

   Besser unterscheidet man zwischen der konkreten Betragsfunktion bzw. Subtraktion und den ihren Platzhaltern, etwa a, m in Σ, und ebenso zwischen der größer-Relation $>$ und ihrem Platzhalter, etwa G. In der Signatur $\Sigma = \{m, a, f, 0, 2; G\}$ hat die obige Formel die korrekte wenn auch unübersichtlichere Form

$$\forall \varepsilon : \Big(G(\varepsilon, 0) \Rightarrow \exists \delta : \big(G(\delta, 0) \wedge \forall x : \big(G(\delta, a(m(x, 2))) \Rightarrow G(\epsilon, a(m(f(x), f(2))))\big)\big)\Big)$$

   die wir mit Hilfe der Klammergröße versucht haben, besser zu strukturieren. Der

zugrundeliegende Baum ist



9.4.8 Beispiel. Im Land Warling gibt es drei Stämme: Xur, Yzy und Poli. Jemand vom Stamm Xur vermählt sich nur mit jemand vom Stamm Xur oder vom Stamm Yzy. Polis vermählen sich nie, weil sie auf die entscheidende Frage immer mit „Nein“ antworten.

Um dies formal aufzuschreiben benötigen wir keine Operationen, sondern nur die folgenden Relationen:

1-stellig: X, Y und P für die drei Stämme

2-stellig: V für „vermählt sein mit“

J für „beide Partner beantworten die entscheidende Frage mit “Ja“ “.

Die erste Aussage ist

$$F_0 := (\forall x : (X(x) \Rightarrow (\forall y : (V(x,y) \Rightarrow (X(y) \vee Y(y))))))$$

Die zweite ist aus zwei Teilen zusammengesetzt:

$$F_2 := (\forall x : (P(x) \Rightarrow (\forall y : (\neg J(x,y)))))$$

und

$$F_3 := (\forall x : \forall y : (\neg J(x,y) \Rightarrow \neg V(x,y)))$$

Die komplette Aussage über Warling ist $F_0 \wedge F_2 \wedge F_3$.

9.4.9 Beispiel. „Es gab zwei Sorten von Studenten, die die volle Punktezahl erreichten: einige wussten alle Lösungen der Aufgaben und die anderen hatten das Glück, dass ihr bester Freund zur ersten Sorte gehört."

Dies wollen wir durch eine Formel ausdrücken. Sei $W(x)$ das einstellige Prädikat „x weiß alle Lösungen" und $B(x, y)$ das zweistellige Prädikat „y ist der beste Freund von x" – es kann natürlich mehrere y je x geben oder auch keinen. Die Formel für die volle Punktezahl ist dann:

$$P := W(x) \vee \exists y : (B(x, y) \wedge W(y))$$

9.4.10 Bemerkung. Im letzten Beispiel ist die Variable x in der Formel P **frei**, d.h., steht nicht im Einflußbereich (Scope) eines Quantors der sich auf x bezieht: sie bezeichnet damit einfach abstrakt „einen Konsumenten akademischer Bildung". Dagegen ist y durch den Ausdruck $\exists y$: **gebunden**.

Diese Unterscheidung zwischen freien und gebundenen Variablen ist wichtig.

9.4.11 Definition. Die Menge Frei(F) aller **freien Variablen** einer Formel F ist rekursiv definiert

Formel F	Menge Frei(F)	
$t_0 = t_2$	alle Variablen, die in t_0 oder t_2 vorkommen	
$R(t_0, \ldots, t_{n-1})$	alle Variablen, die in $t_0, \ldots, t_{n-1}$ vorkommen	
$F_0 \vee F_2$	$\text{Frei}(F_0) \cup \text{Frei}(F_2)$	
$F_0 \wedge F_2$	$\text{Frei}(F_0) \cup \text{Frei}(F_2)$	
$\neg G$	Frei(G)	d.h. dieselben Variablen wie in G sind frei
$\forall x : G$	$\text{Frei}(G) - \{x\}$	alle freien Variablen von G außer x
$\exists x : G$	$\text{Frei}(G) - \{x\}$	alle freien Variablen von G außer x

Eine Formel ohne freie Variablen heißt **geschlossen** oder auch **Satz**.

9.4.12 Bemerkung. Die obige Definition erfaßt die Variablenproblematik nur zum Teil. Variablen können in einem Formelbaum in zwei Typen von Knoten auftreten: in den neuen Junktoren, die durch Verbindung mit den Quantoren $forall$ und $exists$ entstehen, und in den Blättern der Terme, die mittels Prädikatssymbolen an den Formelbaum angehefet sind. Die neuen Junktoren $\forall x$ und $\exists x$ binden die Variable x in dem durch diesen Knoten bestimmten Teilbaum, es sei denn, dort taucht ein weiterer Junktor $\forall x$ oder $existsx$ auf. Das ist zwar ungeschickt, aber nicht verboten.

Es ist zielführender zu fragen, ob eine Variable x in einem bestimmten Blatt eines Terms der Formel gebunden ist, und wenn ja, durch welchen Quantor. Es bindet der erste Junktor $\forall x$ oder $\exists x$, der auf dem eindeutig bestimmten Pfad von diesem Blatt zur Wurzel der Formel auftritt; gibt es keinen solchen, ist das fragliche Auftreten von x frei.

Man beachte, dass in derselben Formel x in verschiedenen Blättern auftreten und je nach Position von unterschiedlichen oder keinen Quantoren gebunden sein kann, z.B. in

$$F = R(x, y) \wedge \exists x : x > 2$$

Das erste **Vorkommen** von x ist frei, aber das zweite ist gebunden. Solch ein Problem läßt sich immer umgehen, wenn man die Namen der gebundenen Variablen etwas geschickter wählt. Die obige Problem-Formel wird sich als äquivalent herausstellen zu

$$G = R(x, y) \wedge \exists z : z > 2$$

Mit anderen Worten: gebundene Variable können *innerhalb ihres Gültigkeitsbereichs* beliebig umbenannt werden, vergl. Fakt 11.0.10 im nächsten Abschnitt.

9.4.13 Beispiel. In der obigen Formel P aus Beispiel 9.4.9 ist x eine freie und y eine gebundene Variable. Es gilt: $\mathrm{Frei}(F) = \{x\}$.

Zum Beispiel ist die Stetigkeitsformel in Beispiel 9.4.7 ein Satz: die Variablen ϵ, δ und x sind alle gebunden. Für Sätze ist die Frage sinnvoll, ob die Formel wahr oder falsch ist: im konkreten Beispiel hängt dies davon ab, ob die gegebene Funktion $f(x)$ an der Stelle 2 stetig ist oder nicht. Im Gegensatz dazu hängt der Wahrheitswert der obigen Formel P von der Interpretation der freien Variablen x ab. Dies wird im nächsten Kapitel erläutert.

9.4.14 Notation. Gelegentlich werden in der Prädikatenlogik Ausdrücke der Form $F(x)$ oder $F(x, y)$ oder $F(x, y, z)$ etc. verwendet, wobei F eine Meta-Variable für eine Formel ist. Diese sollen ausdrücken, dass $\mathrm{Frei}(F)$ Teilmenge der angegebenen Variablenmenge ist; $\mathrm{Frei}(F)$ braucht ausdrücklich *nicht* mit der angegebenen Variablenmenge übereinzustimmen.

Als Notation ist das etwas unglücklich, denn es liefert die dritte inkompatible Version eines in runde Klammern eingeschlossenen Arguments aus Variablen. Z.B.

$f(x, y)$	der formale Funktionswert der 2-stelligen Funktionssymbols f an der Stelle (x, y), ein **Term**
$R(x, y)$	der formale Wahrheitswert der Behauptung, die Variablen x und y stünden in der formalen 2-stelligen Relation R, eine **Formel**
$F(x, y)$	die freien Variablen der Formel F sind in $\{x, y\}$ enthalten, eine **Aussage der Meta-Sprache**, in der F eine Variable für Formeln ist

Das folgende Beispiel zeigt eine weitere notationelle Falle:

9.4.15 Beispiel. Wenn die Signatur $\Sigma = \langle \bot, \neg, \wedge, \vee \rangle$ aus den aussagenlogischen Junktoren der Stelligkeiten 0, 1, 2 und 2 besteht, gilt

$$\mathcal{T}_\Sigma(\mathcal{M}) = \mathcal{F}(\mathcal{M})$$

für jede Teilmenge $\mathcal{M} \subseteq \mathcal{V}$, vergl. Definition 2.2.3 und Bemerkung 2.2.4. Die Formeln der Aussagenlogik sind also, bis auf die Namen der Variablen, genau die Terme für diese spezielle Signatur. Wie üblich induzieren die Funktionssymbole der Signatur entsprechende Funktionen auf der Term-Menge, die man durchaus mit dem Funktionssymbol benennen kann.

Um der Aussagenlogik noch näher zu kommen, kann man als Gleichungen die Ergebnisse aus Kapitel 3 verwenden. Da es außer $\doteq$ keine Relationensymbole gibt, gibt es keine Axiome.

Ein technisches Problem besteht nun darin, das die Junktoren zur Kombination von prädikatenlogischen Formeln auch in der Signatur vorkommen. Das schließt nicht aus, dass man die Prädikatenlogik auf die Aussagenlogik anwenden kann, aber dann muß man die Junktoren aus der Signatur von denen zum prädikatenlogischen Formelaufbau unterscheiden, z.B. farblich, oder indem man letztere mit dem Index 1 versieht („Logik der ersten Stufe“). Verglichen mit der Aussagenlogik befinden wir uns bei den Formeln der Prädikatenlogik auf dem Level der Meta-Sprache, vergl. Bemerkung 5.0.9.

Im Vorgriff auf das nächste Kapitel stellt sich natürlich auch die Frage nach der Interpretation von „Gleichheit“, z.B. der prädikatenlogischen Formel $A \doteq A \wedge (B \vee A)$. Wie in Kapitel 2 gesehen, ist syntaktische Gleichheit weniger interessant als semantische Gleichheit, für die wir in Definition 2.8.1 die Bezeichnung $\equiv$ eingeführt hatten. Der Verzicht darauf, $\doteq$ in konkreten Fällen zwingend als syntaktische Gleichheit interpretieren zu müssen, stellt sich nun als Vorteil heraus.

Allerdings sollte die Interpretation von $\doteq$ keine beliebige binäre Relation sein dürfen.

9.5 Die Mindestaxiome der Prädikatenlogik

Wie schon erwähnt, soll die Interpretation von $\doteq$ eine Kongruenzrelation bzgl. der interpretierten Funktionssymbole sein, d.h., eine Äquivalenzrelation auf der Trägermenge des Modells, die mit den interpretierten Funktionen verträglich ist.

Der tiefere Sinn dieser Forderung besteht darin, dass die Äquivalenzklassen bzgl. einer Kongruenzrelation wieder eine Algebra desselben Typs bilden wie die ursprüngliche Algebra, die sog. **Faktor-Algebra**.

Aber wie steht es mit den Prädikaten der ursprünglichen Algebra? Sicherlich sollen sich diese auch auf die Faktor-Algebra vererben! Mit anderen Worten, falls zunächst Elemente a_i, $i < n$, der Trägermenge in Relation stehen, sollen hinterher auch die entprechenden Äquivalenzklassen in Relation stehen.

Um diese Forderungen in Axiome zu fassen, müssen wir über alle vorkommenden Variablen universell quantifizieren, denn die Gültigkeit dieser Formeln soll unabhängig von der Interpretation einzelner Variablen sein (vergl. nächstes Kapitel):

9.5.1 Definition. Die **Axiome** der Prädikatenlogik umfassen mindestens die Formeln:

$$\forall x : x \doteq x$$

d.h., $\doteq$ ist **formal reflexiv**;

$$\forall x : \forall y : \forall z \, (x \doteq y \wedge y \doteq z \Rightarrow x \doteq z)$$

d.h., $\doteq$ ist **formal transitiv**;

$$\forall x : \forall y : (x \doteq y \Rightarrow y \doteq x)$$

d.h., $\doteq$ ist **formal symmetrisch**;

$$\forall x_0 : \forall x_1 : \cdots \forall x_{2n-1} : \Big(\Big(\bigwedge_{i<n} x_{2i} \doteq x_{2i+1} \Big) \Rightarrow f(x_0, \ldots, x_{2(n-1)}) \doteq f(x_1, \ldots, x_{2n-1}) \Big)$$

wobei f ein n-stelliges Funktionssymbol aus Σ ist;

$$\forall x_0 : \forall x_1 : \cdots \forall x_{2n-1} : \Big(\Big(\bigwedge_{i<n} x_{2i} \doteq x_{2i+1} \Big) \wedge R(x_0, \ldots, x_{2(n-1)}) \Rightarrow R(x_1, \ldots, x_{2n-1}) \Big)$$

wobei R ein n-stelliges Relationssymbol aus Σ ist.

Praktisch bedeutet dies, dass $\doteq$ immer als Äquivalenzrelation interpretiert werden muß, und dass man die Argumente von f bzw. R durch äquivlente („gleiche“) Argumente ersetzen darf, ohne dass sich der Funktions- bzw. der Wahrheitswert ändert.

10. Semantik der Prädikatenlogik

Ist die Formel $\forall x : (x + 1 > 0)$ wahr? Die Antwort hängt von der mathemaischen Struktur ab, in der x **interpretiert** wird: die Formel ist z.B. wahr, falls x auf die natürlichen Zahlen $\mathbb{N}$ beschränkt wird und $+$, 1 sowie $>$ ihre "normale" Interpretation haben, aber nicht, falls x eine ganze Zahl aus $\mathbb{Z}$ darstellen kann. Wir müssen also zunächst spezifizieren, woher die Elemente stammen, die x darstellt, man spricht auch von der *Trägermenge*, und anschließend, was die Symbole $+$, $>$ und 0 der Signatur in dieser Trägermenge konkret bedeuten sollen (hier waren schon suggestive Namen im Hinblick auf eine Interpretation in den Zahlen gewählt worden). Spezifikationen dieser Art, in denen die gegebeneTheorie interpretiert werden kann, heißen *Strukturen* oder *Modelle*:

10.0.1 Definition. Für eine Signatur Σ besteht eine **Σ-Struktur $\mathcal{A}$**, auch **Σ-Model** genannt, aus

- ▷ einer Menge $\mathcal{A}$ (dem **Träger** von $\mathcal{A}$), die nicht leer sein soll (warum? vergl. Beispiel 11.0.20(b)),
- ▷ einer konkreten n-stelligen Operation

  $$f^{\mathcal{A}} : \mathcal{A}^n \longrightarrow \mathcal{A}$$

  für jedes n-stellige Funktionssymbol f in Σ, und
- ▷ einer konkreten n-stelligen Relation

  $$R^{\mathcal{A}} \subseteq \mathcal{A}^n \quad \text{bzw.} \quad R^{\mathcal{A}} : \mathcal{A}^n \longrightarrow \mathbf{2}$$

  für jedes n-stellige Prädikatssymbol R in Σ.

Die konkreten Operationen $f^{\mathcal{A}}$ bzw. Relationen $R^{\mathcal{A}}$ sind die Instanziierungen der abstrakten Funktions- und Prädikatssymbole aus der Signatur Σ für das Model $\mathcal{A}$. Gelegentlich, z.B. bei suggestiven Namen der Funktions- und Prädikatssymbole in Σ, lassen wir den oberen Index $\mathcal{A}$ auch weg.

Insbesondere wird jeder Konstante c in Σ eine Funktion $c^{\mathcal{A}} : \mathcal{A}^0 = 1 \longrightarrow \mathcal{A}$, d.h., ein Element $c^{\mathcal{A}} \in \mathcal{A}$ zugeordnet. Würde man 0-stellige Prädikate R zulassen, wäre ihnen ein frei wählbarer Wahreitswert $R^{\mathcal{A}} \in \{0, 1\}$ zuzuordnen. Die damit einhergehenden Komplikationen wollen wir uns hier aber ersparen.

10.0.2 Beispiel. Falls $\Sigma = \{R\}$ aus einem 2-stelligen Prädikat R besteht, ist eine Σ-Model $\mathcal{A}$ genau ein (nichtleerer!) gerichteter Graph. Dabei ist $\mathcal{A}$ die Menge der Knoten von $\mathcal{A}$ ist, und $R^{\mathcal{A}} \subseteq A \times A$ die Menge der Kanten von $\mathcal{A}$.

10.0.3 Beispiel. Besteht Σ aus einer Konstante 0, einem einstelligen Funktionssymbol s und einem zweistelligen Prädikatssymbol R können wir z.B. das Σ-Model $\mathcal{N}$ der natürlichen Zahlen betrachten:

$$\begin{aligned} \text{Träger } & \mathbb{N} = \{0, 1, 2, 3, \dots\} \\ & 0^{\mathcal{N}} \text{ ist die Konstante 0;} \\ & s^{\mathcal{N}} \text{ ist die Nachfolgerfunktion, } s^{\mathcal{N}}(n) = n + 1; \\ R^{\mathcal{N}}(x, y) & \text{ bedeutet } x < y. \end{aligned}$$

Ein anderes Σ-Model $\mathcal{Z}$ verwendet als Träger die Menge $\mathbb{Z}$ der ganzen Zahlen, wobei $0^{\mathcal{Z}}$, $s^{\mathcal{Z}}$ und $R^{\mathcal{Z}}$ analog wie oben definiert sind. Ein drittes Σ-Model $\mathcal{A}$ habe den Träger $\{0, 1\}$, wobei $s^{\mathcal{A}}$ die beiden Werte vertauscht, $0^{\mathcal{A}} = 0$ gilt und $R^{\mathcal{A}} = \varnothing$ das leere Prädikat ist.

10.0.4 Beispiel. Wie in Beispiel 9.4.15 gesehen, läßt sich sogar die Aussagenlogik als prädikatenlogische Theorie auffassen. Dabei interessiert uns vorrangig ein Modell **2** mit Trägermenge $2 = \{0, 1\}$ und

$$\neg^{\mathbf{2}}(x) = 1 - x \quad , \quad \wedge^{\mathbf{2}}(x, y) = \inf\{x, y\} \quad , \quad \vee^{\mathbf{2}}(x, y) = \sup\{x, y\}$$

Außerdem möge $\doteq^{\mathbf{2}}$ die echte Gleichheit, also die Diagonalrelation $\{\langle 0, 0\rangle, \langle 1, 1\rangle\}$ auf 2 sein. Belegungen sind dann Funktionen $\alpha : \mathcal{M} \longrightarrow \{0, 1\}$, vergl. Definition 2.3.1.

Auch wenn man häufig für eine Signatur Σ ein bestimmtes Σ-Model im Auge hat, mit dem man sich beschäftigen möchte (die *intendierte* Σ-Stuktur), so gibt es oft viele andere Σ-Modelle, die man zunächst nicht ausschließen kann. In Beispiel 9.8 hatten wir allerdings gesehen, wie man durch die Forderung nach Gültigkeit *zusätzlicher Formeln* (sogenannter **Axiome**, speziell von **Gleichungen**) die passenden Σ-Modele einschränken kann. Von besonderem Interesse ist es daher, eine vorzugsweise endliche Axiomenmenge zu finden, so dass nur noch das intendierte Σ-Model übrigbleibt (bis auf Umbenennung der Elemente), die diese Axiome erfüllt. Das Problem der (endlichen) Axiomatisierbarkeit werden wir hier allerdings nicht weiter behandeln.

Wir können jetzt das Problem angehen, den prädikatenlogischen Formeln über Σ einen Wahrheitswert zuzuweisen. Ähnlich wie in der Aussagenlogik, wo atomaren Formeln mittels *Belegungen* Wahrheitswerte zugewiesen wurden, woraufhin man die Belegungen auf zusammengesetzte Aussagn erweitern konnte, wollen wir nun die Variablen auf Elemente im Träger des aktuellen Σ-Models abbilden, und derartige Abbildungen dann auf die Menge aller Terme erweitern.

Damit erhalten die Interpretationen aller atomaren Formeln Wahrheitswerte, denn auch den evtl. auftretenden Konstanten sind im Rahmen der Model-Spezifikation schon Werte in der Trägermenge zugewiesen worden. Dann geht es darum, die zu untersuchen, ob die Bilder bestimmter Terme unter der Belegung übereinstimmen, oder in spezifischen Relationen stehen.

Wähend die Erweiterung auf aussagenlische Kominationen von Formeln klar ist, erfordert die Interpretation quantifizierter Formeln eine neue Intuition: mit $\forall x :$ bzw. $\exists x :$ beginnende Formeln sollen als potentielle unendliche Konjunktion bzw. Disjunktion aufgefaßt werden. In diesen Fällen möge der Parameter x über alle Elemente der

Trägermenge laufen. Mit anderen Worten, alle freien Vorkommen von x im Argument der quantifizierten Formel sind der Reihe nach mit den Elemente der Trägermenge zu belegen, und dann ist das Infimum bzw. Supremum der resultierenden Menge von Wahrheitswerten zu bilden.

10.0.5 Definition. (Belegungen von Variablen im Träger $\mathcal{A}$ eines Σ-Models)

0. Gegeben ist ein Model mit Träger A. Eine **Belegung** α ist eine partielle Funktion $\mathcal{V} \xrightarrow{\alpha} A$, die jeder Variablen x aus einer Teilmenge $D(\alpha) = \mathcal{M} \subseteq \mathcal{V}$ ein Element aus dem Träger $\mathcal{A}$ zuordnet.

1. Für jede auf $\mathcal{M} \subseteq \mathcal{V}$ definierte Belegung α, jede Variable x und jeden Wert $a \in \mathcal{A}$ im Träger bezeichnet der **Update** von a für x in α die (potentiell modifizierte) Belegung

$$\alpha[{}^{a}/_{x}] : \mathcal{M} \cup \{x\} \longrightarrow A$$

die alle Variablen $y \in \mathcal{M} - \{x\}$ genau wie α belegt, aber der Variable x den Wert a zuweist, formal:

$$\alpha[{}^{a}/_{x}](y) := \begin{cases} \alpha(y) & \text{falls } y \neq x \\ a & \text{sonst} \end{cases}$$

(Wir lesen den Ausdruck $[{}^{a}/_{x}]$ als „a für x“.)

10.0.6 Definition. Jede Belegung $\alpha : \mathcal{M} \longrightarrow \mathcal{A}$ wird rekursiv auf alle Terme in Variablen aus $\mathcal{M}$ erweitert. Wir definieren die Funktion $\overline{\alpha} : \mathcal{T}_\Sigma(\mathcal{M}) \longrightarrow \mathcal{A}$ wie erwartet:

$$\overline{\alpha}(x) = \alpha(x)$$

für alle Variablen $x \in \mathcal{M}$ und

$$\overline{\alpha}(f(t_0, \ldots, t_{n-1})) = f^{\mathcal{A}}(\overline{\alpha}(t_0), \ldots, \overline{\alpha}(t_{n-1}))$$

für alle Funktionssymbolde der Stelligkeit n und alle Terme $t_0, \ldots, t_{n-1}$ in $\mathcal{T}_\Sigma(\mathcal{M})$, für die $\overline{\alpha}(t_i)$ schon definiert wurde.

10.0.7 Beispiel. In der Signatur $\Sigma = \langle 0, s; R\rangle$ des obigen Beispiels 10.0.3 besteht $\mathcal{T}_\Sigma(\emptyset)$ aus allen Termen $0, s(0), s(s(0)), \ldots$, so dass man $\mathcal{T}_\Sigma(\emptyset)$ mit der Menge $\mathbb{N}$ der natürlichen Zahlen identifizieren kann. Für $\mathcal{T}_\Sigma(\{x\})$ haben wir außerdem die Terme $x, s(x), s(s(x)), \ldots$, d.h, $\mathcal{T}_\Sigma(\{x\})$ besteht im Wesentlichen aus zwei disjunkten Kopien von $\mathbb{N}$. Allgemein besteht $\mathcal{T}_\Sigma(\mathcal{M})$ für endliches $\mathcal{M}$ aus $|\mathcal{M}| + 1$ disjunkten Kopien von $\mathbb{N}$. Gegeben sei etwa eine Belegung

$$\alpha : \{x, y\} \longrightarrow \mathbb{Z} \quad \text{mit} \quad \alpha(x) = -7 \quad \text{und} \quad \alpha(y) = 3$$

Dann haben wir

$$\begin{array}{lll} \overline{\alpha}(0) = 0 & \overline{\alpha}(x) = -7 & \overline{\alpha}(y) = 3 \\ \overline{\alpha}(s(0)) = 1 & \overline{\alpha}(s(x)) = -6 & \overline{\alpha}(s(y)) = 4 \\ \overline{\alpha}(s(s(0))) = 2 & \overline{\alpha}(s(s(x))) = -5 & \overline{\alpha}(s(s(y))) = 5 \\ \vdots & \vdots & \vdots \end{array}$$

Wir wollen jetzt die Semantik einer Formel F in einem Σ Model $\mathcal{A}$ definieren. Sofern F freie Variable enthält, hängt der Wahrheitswert von F in $\mathcal{A}$ natürlich von den zu F **passenden** Belegungen $\alpha : \mathcal{M} \longrightarrow \mathcal{A}$ ab, das sind diejenigen Belegungen, die $\mathrm{Frei}(F) \subseteq \mathcal{M}$ erfüllen.

10.0.8 Definition (Semantik der Prädikatenlogik). Gegeben: Eine Signatur Σ, ein Σ-Model $\mathcal{A}$ mit Träger $\mathcal{A}$, eine Formel F mit $\mathrm{Frei}(F) \subseteq \mathcal{M}$ und eine Belegung $\alpha : \mathcal{M} \longrightarrow \mathcal{A}$. Wir definieren $\widehat{\alpha}(F)$ in $\mathcal{A}$ wie folgt:

- atomare Formeln:

$$\begin{aligned} \widehat{\alpha}(t_0 \doteq t_1) = 1 \quad &\text{g.d.w.} \quad \overline{\alpha}(t_0) \doteq^{\mathcal{A}} \overline{\alpha}(t_1) \quad \text{gilt} \\ \widehat{\alpha}(R(t_0, \ldots, t_{n-1})) = 1 \quad &\text{g.d.w.} \quad R^{\mathcal{A}}(\overline{\alpha}(t_0), \ldots, \overline{\alpha}(t_{n-1})) \quad \text{gilt} \end{aligned}$$

- zusammengesetzte Formeln

$$\begin{aligned} \widehat{\alpha}(\neg G) = 1 \quad &\text{g.d.w.} \quad \widehat{\alpha}(G) = 0 \\ \widehat{\alpha}(G \wedge H) = 1 \quad &\text{g.d.w.} \quad \widehat{\alpha}(G) = 1 \text{ und } \widehat{\alpha}(H) = 1 \\ \widehat{\alpha}(G \vee H) = 1 \quad &\text{g.d.w.} \quad \widehat{\alpha}(G) = 1 \text{ oder } \widehat{\alpha}(H) = 1 \\ \widehat{\alpha}(\forall x : G) = 1 \quad &\text{g.d.w.} \quad \text{für alle } a \in \mathcal{A} \text{ gilt } \widehat{\alpha[{}^{a}/_{x}]}(G) = 1 \\ \widehat{\alpha}(\exists x : G) = 1 \quad &\text{g.d.w.} \quad \text{es gibt ein } a \in \mathcal{A} \text{ mit } \widehat{\alpha[{}^{a}/_{x}]}(G) = 1 \end{aligned}$$

10.0.9 Bemerkung.

- Für 1-stellige Prädikatssymbole R gilt die Formel $R(t)$ genau dann in $\mathcal{A}$ unter α, wenn die Interpretation $\overline{\alpha}(t)$ des Terms die Eigenschaft $R^{\mathcal{A}}$ hat – d.h. wenn $R^{\mathcal{A}}(\overline{\alpha}(t))$ gilt. Die Semantik einer 2-stelligen Relation $R^{\mathcal{A}}$ unter α macht $R(t_0, t_1)$ genau dann wahr, wenn die Elemente $\overline{\alpha}(t_0)$ und $\overline{\alpha}(t_1)$ in der Relation $R^{\mathcal{A}}$ stehen.

- Bei der Interpretation von $\forall x : G$ bzw. $\exists x : G$ ist zu beachten, dass bei passenden Belegungen α der Wert $\alpha(x)$, sofern er existiert, irrelevant für den Wahrheitswert der quantifizierten Formel ist, da x dort nicht frei vorkommt. Insofern kann die Funktion α an der Stelle x beliebig abgewandelt werden, was durch die „Updates“ $\alpha[{}^{a}/_{x}]$ realisiert wird. Je nach Quantor ist entscheidend, ob der Wert von G unter allen oder mindestens einem derartigen Update den Wert 1 annimmt.

- In Analogie zu Bemerkung 2.3.4 lassen sich die Wahrheitswerte der Formeln auch anders (und nützlicher) ausdrücken:

$$\begin{aligned} \widehat{\alpha}(\neg G) &= 1 - \widehat{\alpha}(G) \\ \widehat{\alpha}(G \wedge H) &= \inf\{\widehat{\alpha}(g), \widehat{\alpha}(H)\} \\ \widehat{\alpha}(G \vee H) &= \sup\{\widehat{\alpha}(g), \widehat{\alpha}(H)\} \\ \widehat{\alpha}(\forall x : G) &= \inf\{\, \widehat{\alpha[{}^{a}/_{x}]}(G) \,:\, a \in \mathcal{A} \,\} \\ \widehat{\alpha}(\exists x : G) &= \sup\{\, \widehat{\alpha[{}^{a}/_{x}]}(G) \,:\, a \in \mathcal{A} \,\} \end{aligned}$$

10.0.10 Beispiel. In (nichtleeren) gerichteten Graphen $\mathcal{A}$ als Σ-Modellen (vgl. Beispiel 10.0.2) besagt die Gültigkeit der Formel

$$\forall x : \forall y : (R(x, y) \Rightarrow R(y, x))$$

dass der Graph $\mathcal{A}$ symmetrisch oder ungerichtet ist: zu jeder gerichteten Kante gibt es eine Kante in der entgegengesetzten Richtung. Die entsprechende Formel ohne Quantoren, also

$$R(x, y) \Rightarrow R(y, x)$$

hat keinen festen Wahrheitswert: Wir müssen die freien Variablen x und y zuerst als feste Knoten a, b interpretieren, d.h., Belegungen α mit $\alpha(x) := a$ und $\alpha(y) := b$ verwenden. Dann verlangt die Formel: Wenn eine Kante von a nach b führt, dann gibt es auch eine Kante in der Gegenrichtung. Der Wahrheitswert hängt hier von der gewählten Interpretation, genauer, von den ausgewählten Knoten a und b ab. Sie kann für bestimmte Knoten richtig, für andere dagegen falsch sein.

10.0.11 Beispiel.

- Im Σ-Model $\mathcal{N}$ der natürlichen Zahlen aus Beispiel 10.0.3 ist die Formel

  $$\forall x : \exists y : x < y$$

  wahr, man kann z.B. $y = s(x) = x + 1$ setzen. In Ermangelung einer größten natürlichen Zahl liefert die Vertauschung der Quantoren eine falsche Formel:

  $$\exists y : \forall x : x < y$$

- Die Formel $\forall x : (0 < x \vee 0 = x)$, die 0 als kleinstes Element charakterisiert, ist wahr in $\mathcal{N}$ aber nicht in $\mathcal{Z}$.

10.0.12 Satz. *Für eine Signatur Σ und ein Σ-Model $\mathcal{A}$ mit Träger A hängt der Wahrheitswert einer Formel F in $\mathcal{A}$ nur von der Interpretation ihrer freien Variablen ab: Für alle zu F passenden Belegungen α und β gilt*

$$\alpha(x) = \beta(x) \text{ für alle } x \in \mathrm{Frei}(F) \quad \text{impliziert} \quad \widehat{\beta}(F) = \widehat{\alpha}(F)$$

Beweis. Wir stellen zunächst fest, dass die Interpretation der Σ-Propositionen bereits durch das Σ-Model vorgegeben ist.

Aufgrund von Definition 10.0.6 folgt sofort, dass für zu F passende Belegungen α und β, die Erweiterungen $\bar{\alpha}$ und $\bar{\beta}$ auf der Term-Menge $\mathrm{Frei}(F)$ übereinstimmen.

Zwecks struktureller Induktion über den Aufbau von F unterscheiden wir die Fälle:

- F sei atomar. Dann sind alle auftretenden Variablen von F frei und $\widehat{\alpha}(F)$ ist genau dann wahr wenn $\widehat{\beta}(F)$ wahr ist, denn $\overline{\alpha}$ und $\overline{\beta}$ stimmen gemäß der Vorüberlegung bei allen auftretenden Termen überein. Treten keine Terme auf, so handelt es sich bei F um eine Σ-Proposition, und deren Wert ist durch das Σ-Model bereits bestimmt.

- F resultiert aus aussagenlogischer Verknüpfung, hat etwa die Form $G \wedge H$, und die Behauptung gelte für G und H. Dann folgt wegen $\mathrm{Frei}(G), \mathrm{Frei}(H) \subseteq \mathrm{Frei}(F)$

  $$\begin{aligned} \widehat{\alpha}(F) = \widehat{\alpha}(G \wedge H) &= \inf\{\widehat{\alpha}(G), \widehat{\alpha}(H)\} \\ &= \inf\{\widehat{\beta}(G), \widehat{\beta}(H)\} = \widehat{\beta}(G \wedge H) = \widehat{\beta}(F) \end{aligned}$$

  Analog argumentiert man für $F = G \vee H$ und $F = \neg H$.

- F habe die Form $\forall x : G$ und die Behauptung gelte für G. In diesem Fall gilt wegen $\mathrm{Frei}(G) \subseteq \mathrm{Frei}(F) + \{x\}$

$$\begin{aligned}\widehat{\alpha}(F) &= \inf\{\, \widehat{\alpha[{}^{a}/_{x}]}(G) \;:\; a \in \mathcal{A} \,\} \\ &= \inf\{\, \widehat{\beta[{}^{a}/_{x}]}(G) \;:\; a \in \mathcal{A} \,\} = \widehat{\beta}(F)\end{aligned}$$

  Analog argumentiert man für $\exists x : F$. □

10.0.13 Notationelle Konvention. Der Wahrheitswert eines Satzes F in einem Σ-Model $\mathcal{A}$ ist unabhängig von Variablenbelegungen. Wir sagen, F **gilt in** $\mathcal{A}$, wenn der Wahrheitswert unter jeder Variablenbelegung den Wert 1 annimmt.

10.0.14 Beispiel. Die Formel $\forall x : \forall y : (R(x, y) \Rightarrow R(y, x))$ aus 10.0.10 gilt genau in allen symmetrischen, d.h., ungerichteten Graphen.

11. Logische Äquivalenz

Wir fahren fort, Begriffsbildungen aus der Aussagenlogik in die Prädikatenlogik zu übertragen.

11.0.1 Definition. Zwei Formeln F und G der Prädikatenlogik heißen **äquivalent**, in Symbolen

$$F \equiv G$$

falls für jedes Σ-Model $\mathcal{A}$ und jede zu F und G passende Belegung $\alpha : \mathcal{M} \longrightarrow \mathcal{A}$ gilt: $\widehat{\alpha}(F)$ ist genau dann wahr, wenn $\widehat{\alpha}(G)$ wahr ist. Kurz:

$$\widehat{\alpha}(F) = \widehat{\alpha}(G).$$

11.0.2 Bemerkung. Jede logische Äquivalenz der Aussagenlogik ist auch in der Prädikatenlogik gültig, denn die Auswahl eines Models $\mathcal{A}$ und einer Belegung $\mathcal{V} \xrightarrow{\alpha} \mathcal{A}$ ordnen jeder atomaren Formel einen Wahrheitswert zu, was im Wesentlichen einer Belegung im Sinne der Aussagenlogik entspricht. Z.B., gilt die De Morgansche Regel:

$$\neg(F \wedge G) \equiv \neg F \vee \neg G$$

Der Beweis aus Satz 3.1.2 kann wörtlich übernommen werden.

Um neue Äquivalenzen zu finden, sind die neuen Junktoren $\forall x$ und $\exists x$ in Betracht zu ziehen. Aber zuvor wollen wir noch die Verträglichkeit von $\equiv$ mit diesen neuen Junktoren prüfen.

11.0.3 Satz. *Die Äquivalenz $\equiv$ ist auch hinsichtlich der neuen Junktoren $\forall x :$ und $\exists x :$ eine Kongruenzrelation.*

Beweis. Aus $G \equiv H$ folgt sofort für jedes Σ-Model $\mathcal{A}$, jede für G und H passende Belegung α, jede Variable x und jedes Element a des Trägers $\mathcal{A}$

$$\widehat{\alpha[{}^{a}/_{x}]}(G) = \widehat{\alpha[{}^{a}/_{x}]}(H)$$

und somit

$$\widehat{\alpha}(\forall x : G) = \widehat{\alpha}(\forall x : H)$$

Den Existenz-Quantor behandelt man analog. □

11.0.4 Beispiel. Die Aussage

Nicht alle Vögel können fliegen.

ist eine weniger direkte Variante der Aussage

Es gibt einen Vogel, der nicht fliegen kann.

Da es nur endlich viele Vögel gibt, liegt ein Vergleich mit den endlich iterierten Versionen der Junktoren $\wedge$ und $\vee$ nahe, vergl. Bemerkung 3.2.3. Für diese war in Bemerkung 3.2.4 bereits eine Verallgemeinerung der De Morgan'schen Regeln erfolgt, die nun auch folgenden Satz motiviert:

11.0.5 Satz. *Für jede Formel F und jede Variable x gilt*

$$\neg\forall x : F \equiv \exists x : \neg F \qquad \textit{und} \qquad \neg\exists x : F \equiv \forall x : \neg F$$

Beweis. Für jede passende Belegung α gilt

$$\begin{aligned}
\widehat{\alpha}(\neg(\forall x : F)) &= 1 - \widehat{\alpha}(\forall x : F) \\
&= 1 - \inf\{\, \widehat{\alpha[^a/_x]}(F) \,:\, a \in \mathcal{A} \,\} \\
&= \sup\{\, 1 - \widehat{\alpha[^a/_x]}(F) \,:\, a \in \mathcal{A} \,\} \\
&= \widehat{\alpha}(\exists x : \neg F)
\end{aligned}$$

Das zweite Ergebnis folgt analog, oder auch durch Substitution von $\neg F$ für F und anschließende Negation:

$$\neg\forall x : \neg F \equiv \exists x : F \quad \text{und somit} \quad \forall x : \neg F \equiv \neg\exists x : F \qquad \square$$

11.0.6 Folgerung. *Für die Prädikatenlogik sind die Junktoren $\wedge$ und $\neg$ zusammen mit dem All-Quantor $\forall$ adäquat (vergleiche Definition 3.3.1): Alle Formeln der Prädikatenlogik lassen sich mit $\wedge$, $\neg$ und $\forall$ ausdrücken.*

11.0.7 Proposition. *Es gelten die folgenden verallgemeinerten Assoziativ-Gesetze*

$$\forall x : (F \wedge G) \equiv (\forall x : F) \wedge (\forall x : G) \quad \textit{und} \quad \exists x : (F \vee G) \equiv (\exists x : F) \vee (\exists x : G)$$

Beweis. Für jede passende Belegung α gilt

$$\begin{aligned}
\widehat{\alpha}(\forall x : (F \wedge G)) &= \inf\{\, \widehat{\alpha[^a/_x]}(F \wedge G) \,:\, a \in \mathcal{A} \,\} \\
&= \inf\{\, \inf\{\widehat{\alpha[^a/_x]}(F), \widehat{\alpha[^a/_x]}(G)\} \,:\, a \in \mathcal{A} \,\} \\
&= \inf\{\inf\{\, \widehat{\alpha[^a/_x]}(F) \,:\, a \in \mathcal{A} \,\}, \inf\{\, \widehat{\alpha[^a/_x]}(G) \,:\, a \in \mathcal{A} \,\}\,\} \\
&= \inf\{\widehat{\alpha}(\forall x : F), \widehat{\alpha}(\forall x : G)\} \\
&= \widehat{\alpha}(\forall x : F \wedge \forall x : G)
\end{aligned}$$

Das zweite Ergebnis kann man analog herleiten, oder auch aus dem ersten Teil mit Hilfe der De Morgan'schen Regeln (siehe Satz 11.0.5)

$$\begin{aligned}\exists x : (F \vee G) &\equiv \neg\forall x : \neg(F \vee G)\\ &\equiv \neg\forall x : (\neg F \wedge \neg G)\\ &\equiv \neg((\forall x : \neg F) \wedge (\forall x : \neg G))\\ &\equiv (\neg(\forall x : \neg F)) \vee (\neg(\forall x : \neg G))\\ &\equiv (\neg(\neg\exists x : F)) \vee (\neg(\neg\exists x : G))\\ &\equiv (\exists x : F) \vee (\exists x : G)\end{aligned}$$ □

11.0.8 Bemerkung. Genausowenig, wie zwischen Konjunktion $\wedge$ und Disjunktion $\vee$ eine gemische Assoziativität gilt

$$(F \wedge G) \vee H \not\equiv F \wedge (G \vee H)$$

ist auch der All-Quantor mit $\vee$ bzw. der Existenz-Quantor mit $\wedge$ verträglich:

$$\forall x : (F \vee G) \not\equiv (\forall x : F) \vee (\forall x : G)$$

Aufpassen! Oberflächlich betrachtet sieht dieses Beispiel den beiden vorigen ähnlich (nicht wirklich, wenn man sich klarmacht, dass der All-Quantor eine verallgemeinerte Konkunktion und der Existenz-Quantor eine verallgemeinerte Disjunktion ist), doch hier sind die beiden Seiten **nicht äquivalent**. Links steht, dass für jedes x eine der beiden Aussagen zutrifft. Dagegen steht rechts, dass eine der Aussagen für jedes x zutrifft. Wenn F nur auf eine nichtleere echte Teilmenge des Trägers zutrifft und G auf den nichtleeren Rest, dann ist die linke Seite wahr; – die rechte aber nicht. Dies tritt etwa bei der Signatur $\mathcal{N}$ aus Beispiel 10.0.3 bei den folgenden Formeln auf:

$$\forall x : (3 < x \vee x < 4)$$

gilt, während

$$(\forall x : 3 < x) \vee (\forall x : x < 4)$$

nicht gilt.

Wie bereits am Ende von Abschnitt 9.4 erwähnt, dürfen gebundene Variable umbenannt werden. Zunächst erweitern wir die von den Updates her bekannte Schreibweise zum Zweck der **Substitution** freier Variablen:

11.0.9 Definition. Für jede Formel F, jede Variable $x \in \mathrm{Frei}(F)$ und jeden Term $t \in \mathcal{T}_\Sigma(\mathcal{V})$ bezeichnet $F[{}^t/_x]$ diejenige Formel, die entsteht, wenn jedes freie Vorkommen von x durch t substituiert wird.

11.0.10 Fakt. Umbennung gebundener Variablen
Für jede Variable y, die in F nicht vorkommt, gilt

$$\forall x : F \equiv \forall y : F[{}^y/_x] \quad \text{sowie} \quad \exists x : F \equiv \exists y : F[{}^y/_x]$$

11.0.11 Bemerkung. In der Formel $\forall x : x < 3 \Rightarrow R(x)$ ist die Variable x links gebunden und rechts frei (vergl. Bindungskonvention 9.4.6). Das kann verwirren. Aufgrund von Fakt 11.0.10 können wir aber die gebundene Variable umbenennen und bekommen die nicht mehr verwirrende Formel $\forall y : y < 3 \Rightarrow R(x)$.

11.0.12 Notationelle Konvention. Wenn eine Variable in einer Formel sowohl gebunden wie frei vorkommt, benennen wir die gebundene Variable um.

11.0.13 Beispiel. In der Formel

$$W(y) \vee \exists y : (B(x, y) \wedge W(y))$$

aus Beispiel 9.4.9 ist die Variable y sowohl gebunden als frei. Wir können die gebundene Variable umbenennen:

$$W(y) \vee \exists z : (B(x, z) \wedge W(z))$$

Für jede neue Variable z (verschieden von x, y) ist diese neue Formel logisch äquivalent zu der vorigen.

11.0.14 Definition. Eine Formel liegt in **bereinigter Form** vor, wenn

- keine ihrer Variablen frei und gebunden zugleich vorkommt, und
- alle Quantoren unterschiedliche Variablen binden.

11.0.15 Beispiel. Die Formel

$$\forall x : \forall y : (x < y \vee \forall x : x = 0) \vee \exists y : y < u$$

lässt sich wie folgt bereinigen

$$\forall x : \forall y : (x < y \vee \forall z : z = 0) \vee \exists t : t < u$$

11.0.16 Folgerung. *Jede prädikatenlogische Formel ist zu einer Formel in bereinigter Form äquivalent. Diese ergibt sich durch ggf. mehrmalige Anwendung von Fakt 11.0.10.*

11.0.17 Bemerkung.

- Die Reihenfolge von Quantoren desselben Typs spielt keine Rolle:

  $$\forall x : \forall y : F \equiv \forall y : \forall x : F$$

  sowie

  $$\exists x : \exists y : F \equiv \exists y : \exists y : F$$

  Dies ergibt sich eindeutig aus der Definition der Semantik: mehrfaches Bilden von Infima bzw. Suprema ist unabhängig von der Reihenfolge.

- Aber aufpassen! Wie schon mehrfach gesehen dürfen wir im Allgemeinen $\forall$ und $\exists$ nicht vertauschen.

11.0.18 Beispiel. Zum Nachweis von

$$\forall x : \exists y : F \not\equiv \exists y : \forall x : F$$

betrachten wir die Signatur aus 10.0.3; aufgrund der Interpretation von $s^{\mathbb{N}}$ als Nachfolger-Funktion gilt in $\mathcal{N}$ die Formel

$$\forall x : \exists y : x < y$$

aber mangels eines größten Elements nicht die Formel

$$\exists y : \forall x : x < y$$

Andererseits sind die Formeln

$$\forall x : \exists y : (0 < x + 1 \land y = y + y) \quad \text{und} \quad \exists y : \forall x : (0 < x + 1 \land y = y + y)$$

in der Tat äquivalent.

Wir brauchen noch Gegenstücke für die Begriffe der Tautologie und der Erfüllbarkeit in der Aussagenlogik.

11.0.19 Definition. Eine Formel F der Prädikatenlogik heißt

- **allgemeingültig** falls sie in jedem Σ Model unter jeder passenden Belegung wahr ist;
- **erfüllbar** falls es ein Σ-Model und eine passende Belegung gibt, sodass F wahr ist.

11.0.20 Beispiel.

- Jede Tautologie der Aussagenlogik ist allgemeingültig, z.B. $F \Leftrightarrow \neg\neg F$
- Folgende Formel ist allgemeingültig:

  $$\forall x : F \Rightarrow \exists x : F$$

  Ist $\mathcal{A}$ ein Model und α eine Belegung mit $\widetilde{\alpha}(\forall x : F) = 1$, wählen wir ein Element $a \in \mathcal{A} \neq \varnothing$ aus. Wegen der Semantik von $\forall x$ gilt dann

  $$\widehat{\alpha[{}^{a}/_{x}]}(F) = 1$$

  und dies bedeutet $\widehat{\alpha}(\exists x : F) = 1$. Diese Argument setzt voraus, dass der Träger eines Σ-Models nichtleer ist und wird gern zur Rechtfertigung dieser Einschränkung herangezogen.
- Die Sätze 2.8.3 und 11.0.5 implizieren die Allbemeingültigkeit von

  $$\forall x : \neg F \Leftrightarrow \neg\exists x : F$$

- Die Äquivalenz $\exists x : R(x) \Leftrightarrow \forall x : R(x)$ ist nicht allgemeingültig: falls $\mathcal{A}$ das Model mit Träger $\{0, 1\}$ ist und $R^{\mathcal{A}} = \{0\}$, dann gilt $\exists x : R(x)$ in $\mathcal{A}$ aber $\forall x : R(x)$ gilt nicht. Die Äquivalenz ist aber erfüllbar: Sei $\mathcal{A}$ ein Model mit nur einem Element im Träger, dann erfüllt $\mathcal{A}$ die Formel.
- Die Formel $\exists x : W(x) \land \forall x : \neg W(x)$ ist nicht erfüllbar.

12. Normalformen

Aufgrund der verallgemeinerten De Morgan'schen Regeln (Satz 11.0.5) läßt sich der Algorithmus NNF unmittelbar auf prädikatenlogische Formeln erweitern. Für quantorenfreie Formeln bleibt der Begriff der KNF weiterhin sinnvoll.

Nun führen wir zwei neue prädikatenlogische Normalformen ein:

12.0.1 Definition. Eine Formel, deren Quantoren vollständig am Anfang stehen, liegt in **Pränex-Normalform** oder **PNF** vor, etwa

$$\mathsf{Q}_0 y_0 : \mathsf{Q}_2 y_1 : \ldots \mathsf{Q}_{n-1} y_{n-1} : F$$

wobei $\mathsf{Q}_i \in \{\forall, \exists\}$ gilt und F keine Quantoren enthält.

Alternieren zudem All- und Existenz-Quantoren, so spricht man von **alternierender Pränex-Normalform**, oder **aPNF**. Solche Formeln haben die Form

$$\forall y_0 : \exists y_1 : \forall y_2 : \ldots \mathsf{Q}_{n-1} y_{n-1} : F$$

12.0.2 Beispiel. $F \vee \forall x : G$ liegt nicht in PNF vor. Falls $x \notin \mathrm{Frei}(F)$, speziell falls die Ausgangsformel bereinigt ist, finden wir eine äquivalente Formel in PNF, indem wir F „unter den Quantor ziehen"

$$F \vee \forall x : G \equiv \forall x : (F \vee G)$$

Das ist zulässig, da nach Voraussetzung $x \notin \mathrm{Frei}(F)$ gilt und durch das Verschieben in den Gültigkeitsbereich des Quantors keine freie Variable von F gebunden wird. Speziell gilt $\widehat{\alpha[{}^{a}/_{x}]}(F) = \widehat{\alpha}(F)$ für jede zu F passende Belegung und jedes $a \in \mathcal{A}$. Folglich

$$\begin{aligned}
\widehat{\alpha}(F \vee \forall x : G) &= \sup\{\widehat{\alpha}(F), \inf\{\, \widehat{\alpha[{}^{a}/_{x}]}(G) \,:\, a \in \mathcal{A}\,\}\} \\
&= \inf\{\, \sup\{\widehat{\alpha}(F), \widehat{\alpha[{}^{a}/_{x}]}(G)\} \,:\, a \in \mathcal{A}\,\} \\
&= \inf\{\, \sup\{\widehat{\alpha[{}^{a}/_{x}]}(F), \widehat{\alpha[{}^{a}/_{x}]}(G)\} \,:\, a \in \mathcal{A}\,\} \\
&= \inf\{\, \{\widehat{\alpha[{}^{a}/_{x}]}(F \vee G)\} \,:\, a \in \mathcal{A}\,\} \\
&= \widehat{\alpha}(\forall x : (F \vee G))
\end{aligned}$$

12.0.3 Beispiel. Analog dazu gelten für $x \notin \mathrm{Frei}(F)$ die folgenden Äquivalenzen:

$$\begin{aligned}
F \vee \exists x : G &\equiv \exists x : (F \vee G) \\
F \wedge \forall x : G &\equiv \forall x : (F \wedge G) \\
F \wedge \exists x : G &\equiv \exists x : (F \wedge G)
\end{aligned}$$

12.0.4 Algorithmus (PNF)**.**
Zu einer Formel F der Prädikatenlogik wollen wir eine äquivalente Formel

$$\mathsf{PNF}(F) = \mathsf{Q}_0 y_0 : \ldots \mathsf{Q}_{n-1} y_{n-1} : F^*$$

berechnen, wobei F^* keine Quantoren enthält.
ObdA möge F bereits in bereinigter Form vorliegen.

0. F atomar: wir setzen
$$\mathsf{PNF}(F) := F$$

1. $F = \neg G$ mit $\mathsf{PNF}(G) = \mathsf{Q}_0 y_0 : \ldots \mathsf{Q}_{n-1} y_{n-1} : G^*$. Wir setzen
$$\mathsf{PNF}(F) := \bar{\mathsf{Q}}_0 y_0 : \ldots \bar{\mathsf{Q}}_{n-1} y_{n-1} : \neg G^*$$
wobei $\bar{\forall} = \exists$ und $\bar{\exists} = \forall$ gilt.

2. $F = G \wedge H$ mit $\mathsf{PNF}(G)$ wie oben und $\mathsf{PNF}(H) = \mathsf{Q}'_0 z_0 : \ldots \mathsf{Q}'_{m-1} z_{m-1} : H^*$. Wir setzen
$$\mathsf{PNF}(F) := \mathsf{Q}_0 y_0 : \ldots \mathsf{Q}_{n-1} y_{n-1} : \mathsf{Q}'_0 z_0 : \ldots \mathsf{Q}'_{m-1} z_{m-1} : (G^* \wedge H^*)$$
(Achtung: da F bereinigt ist, treten in G und H unterschiedliche gebundene Variablen auf. Weiter sind keine freien Variablen von H in G gebunden und umgekehrt.)

3. $F = G \vee H$: Analog zu 2. mit $\vee$ anstelle von $\wedge$.

4. $F = \forall x : G$ mit $\mathsf{PNF}(G)$ wie oben. Wir setzen
$$\mathsf{PNF}(F) = \forall x : \mathsf{Q}_0 y_0 : \ldots \mathsf{Q}_{n-1} y_{n-1} : G^*$$

5. $F = \exists x : G$ mit $\mathsf{PNF}(G)$ wie oben. Wir setzen
$$\mathsf{PNF}(F) = \exists x : \mathsf{Q}_0 y_0 : \ldots \mathsf{Q}_{n-1} y_{n-1} : G^*$$

12.0.5 Satz. *Der Algorithmus ist korrekt: die oben konstruierte Pränex-Normalform* $\mathsf{PNF}(F)$ *ist äquivalent zur Ausgangsformel* F*.*

Beweis. Fall 1 folgt aus Satz 11.0.5. Die Fälle 2 und 3 folgen aus Beispiel 12.0.3, während Satz 11.0.3 die Fäll 4 und 5 impliziert. □

12.0.6 Bemerkung. In den Fällen 2 und 3 fällt auf, dass die Quantoren der PNF von G vor diejenigen der PNF von H gesetzt werden. Da aber Konjunktion $\wedge$ und Disjunktion $\vee$ kommutativ sind, muß die andere Reihenfolge der Quantoren dasselbe Ergebnis liefern, also erst die Quantoren der PNF von H und dann die Quantoren der PNF von G. Auf diese Weise können legitimerweise All-Quantoren mit Existenz-Quantoren vertauscht werden, vergl. Beispiel 11.0.18. Es muß sogar möglich sein, die beiden Quantorengruppen zu durchmischen, solange nur die Reihenfolge der Quantoren der einzelnen PNFs nicht verändert werden. Achtung: damit ist die Reihenfolge der Quantoren in einer PNF nicht eindeutig bestimmt.

12.0.7 Beispiel. Wir betrachten

$$F = \forall x : \exists y : x < y \wedge \neg \forall z : x + z \doteq z$$

Die Variable x ist im hinteren Teil der Formel frei, deshalb wollen wir am Anfang x durch t ersetzen. Das liefert die bereingte Form

$$F \equiv \forall t : \exists y : t < y \wedge \neg \forall z : x + z \doteq z$$

Die Anwendung von Fall 1 auf den hinteren Teil ergibt nun:

$$F \equiv \forall t : \exists y : t < y \wedge \exists z : \neg(x + z \doteq z)$$

Jetzt verwenden wir Fall 2 und erhalten

$$F \equiv \forall t : \exists y : \exists z : (t < y \wedge \neg(x + z \doteq z))$$

Es wird sich herausstellen, dass man sich bei der Frage nach der Erfüllbarkeit von Formeln in PNF immer auf solche **ohne Existenzquantor** beschränken kann. Wie ein Existenzquantor eliminiert werden kann, wollen wir zunächst mit einem Beispiel illustrieren:

12.0.8 Beispiel. Die Stetigkeitsformel aus Beispiel 9.4.7 wollen wir ohne Existenz-Quantor umschreiben: da die Existenz-Quantifizierung von δ *nach* der universellen Quantifizierung von ϵ erfolgt („zu jedem ε existiert ein δ“), also δ von ϵ *abhängt*, denken wir uns für jedes ϵ ein spezifisches δ mit der gewünschten Eigenschaft *ausgewählt*.

Bezeichnet man die Auswahl mit $\delta = h(\epsilon)$, kann man die Signatur $\Sigma = \langle m, a, f, 0, 2; G\rangle\}$ um ein neues 1-stelliges Funktionssymbol h zu Σ' erweitern. In der neuen Signatur läßt sich die Stetigkeit dann wie folgt formalisieren:

$$\forall \epsilon : \Big(\epsilon > 0 \Rightarrow h(\epsilon) > 0 \wedge \forall x : \big(h(\epsilon) > am(x, 2) \Rightarrow \epsilon > am(f(x), f(2))\big)\Big)$$

Im Weiteren werden wir sehen, wie man diese Idee anwenden kann, um den Quantor $\exists$ vollständig zu eliminieren.

12.0.9 Definition. Eine Pränex-Normalform, in der kein Existenz-Quantor auftritt, heißt auch **Skolemsche Normalform**.

12.0.10 Beispiel. Um in der folgenden Formel in Pränex-Normalform

$$\exists x : \forall y : \exists z : (K(x, z) \wedge K(z, y))$$

den führenden Existenz-Quantor zu eliminieren, können wir die Signatur $\Sigma = \{<\}$ um eine neue Konstante c erweitern, die die Rolle des speziellen Elements x übernehmen soll, und obige Formel umschreiben zu

$$\forall y : \exists z : (K(c, z) \wedge K(z, y))$$

Drückt man das potentiell von y abhängige z wie oben mit Hilfe eines Funktonssymbols f aus, so läßt sich auch der zweite Existenzquantor eliminieren:

$$\forall y : (K(c, f(y)) \wedge K(f(y), y))$$

Insgesamt verwenden wir die erweiterte Signatur $\langle c, f; K\rangle$.

12.0.11 Bemerkung. Die Formeln

$$F := \exists x : \forall y : \exists z : (K(x,z) \wedge K(z,y)) \quad \text{bzw.} \quad S_F := \forall y : (K(c, f(y)) \wedge K(f(y), y))$$

sind **nicht äquivalent** (warum?), aber sie haben die Eigenschaft, dass F genau dann erfüllbar ist, wenn S_F erfüllbar ist.

In der Tat, falls F in einem Model $\mathcal{A}$ erfüllbar ist, haben wir noch die Freiheit c und f in $\mathcal{A}$ zu bestimmen. Wir wählen für c eine Interpretation der Variable x, für die restliche Formel $\forall y : \exists z : x < z < y$ wahr ist. Und $f^{\mathcal{A}}$ definieren wir so, dass für alle y die Formel $c < f^{\mathcal{A}}(y) < y$ gilt. Diese Wahl ist natürlich in keiner Weise kanonisch oder konstruktiv. Umgekehrt: Falls S_F erfüllbar ist, ist es auch F. Denn x in F kann durch die Konstante c in S_F interpretiert werden, und für z kann man immer das Element wählen, das durch $f(y)$ bestimmt wird.

12.0.12 Algorithmus (Skolemisierung)**.** Für eine Signatur Σ wollen wir einer jeden bereinigten Formel in Pränex-Normalform

$$F := \mathsf{Q}_0 y_0 : \ldots \mathsf{Q}_{n-1} y_{n-1} : F'$$

eine Formel $S(F)$ für eine potentiell erweiterte Signatur Σ' zuordnen, die in Skolemscher Normalform vorliegt und genau dann erfüllbar ist, wenn F erfüllbar ist.

Wir entfernen die Existenz-Quantoren von links nach rechts.

Sind alle Quantoren in F All-Quantoren, so setzen wir

$$S(F) := F$$

Andernfalls treten n All-Quantoren vor dem ersten Existenz-Quantor auf, also

$$F := \forall y_0 : \forall y_1 : \ldots \forall y_{n-1} \exists y_n : G$$

wobei G in PNF durchaus weitere Quantoren enthalten kann. Da F in bereinigter PNF vorliegt, gilt dies automatisch auch für G. Wir erweitern nun die Signatur um ein neues n-stelliges Funktionssymbol h und setzen

$$S(F) := \forall y_0 : \forall y_1 \ldots \forall_{y_{n-1}} : S(G)[^{h(y_0,y_1,\ldots,y_{n-1})}/_{y_n}]$$

12.0.13 Satz. *Der Algorithmus ist korrekt: für jede Pränex-Normalform F ist $S(F)$ eine Skolemsche Normalform, die genau dann erfüllbar ist, wenn F es ist.*

Beweis. Den Beweis findet der Leser z.B. in Kreuzer, Kühling „Logik für Informatiker“ (Pearson Studium 2006), Satz 4.22. □

12.0.14 Satz. *Für jede Formel F der Prädikatenlogik gibt es eine Skolemsche Normalform S_F, die genau dann erfüllbar ist, wenn F es ist, und diese hat die Form*

$$S_F := \forall y_0 : \ldots \forall y_{n-1} : F^*$$

wobei F^* in KNF vorliegt was für quantorenfreie Formeln wie in der Aussagenlogik definiert ist, siehe Definition 4.2.5.

In der Tat haben wir einen Algorithmus, der jeder Formel F eine derartige Skolemsche Normalform S_F zuordnet.

12.0.15 Algorithmus. Wir betrachten eine beliebige Formel F der Prädikatenlogik, etwa über der Signatur Σ.

1. F wird bereinigt zu einer äquivalenten Formel F_0;
2. mit Algorithmus 12.0.4 wird eine äquivalente PNF F_1 erzeugt[1];
3. Algorithmus 12.0.12 erweitert ggf. Σ um neue Funktionssymbole und liefert eine Formel $S(F_1)$ in Skolemscher Normalform, die genau dann erfüllbar ist, wenn dies für F gilt;
4. mittels des KNF-Algorithmus 4.2.8 wird der quantorenfreie Teil F_1' von $S(F_1)$ in KNF umgewandelt; das liefert die gewünschte Formel S_F.

[1]Diese braucht nicht eindeutig bestimmt zu sein.

13. Herbrandsche Modelle und abstrakte Datentypen

Ganz analog zur Äquivalenz $\equiv$ läßt sich auch die logische Konsequenz $\models$ auf die Prädikatenlogik erweitern:

13.0.1 Definition. Gegeben seien eine nicht notwendig endliche Menge Γ von Σ-Formeln und eine einzelne Σ-Formel F.

(a) Ein Σ-Model $\mathcal{A}$ und eine für Γ passende Belegung $\mathcal{V} \xrightarrow{\alpha} \mathcal{A}$ **erfüllen** Γ, wenn $\widehat{\alpha}$ jedes Element $G \in \Gamma$ auf 1 abbildet. Existiert keine derartige Belegung, so heißt Γ **nicht erfüllbar**.

(b) Die Σ-Formel F **folgt** aus der Menge Γ, oder ist deren **logische Konsequenz**, geschrieben $\Gamma \models F$, wenn in jedem Σ-Model $\mathcal{A}$ jede für $\Gamma \cup \{F\}$ passende und Γ erfüllende Belegung auch F erfüllt.

Wie zuvor in Kapitel 6 kann der Nachweis von $\Gamma \models F$ erfolgen, indem man $H = \bigwedge \Gamma \wedge \neg F$ als nicht erfüllbar identifiziert.

Zur Überprüfung der Erfüllbarkeit einer prädikatenlogischen Formel H sind im Prinzip überabzählbar viele Modelle und Belegungen zu betrachten. Diese Vielfalt soll dahingehend eingeschränkt werden, dass zumindest der Nachweis der Nichterfüllbarkeit in endlich vielen Schritte möglich ist.

Dazu wandelt man H in eine Skolem'sche Normalform S_H um, mit quantorenfreiem Teil H^*. (Dass letzterer sogar in KNF vorliegt wird erst in nächsten Abschnitt zum Tragen kommen.). In allen der rein syntaktisch aufgebauten sog. Herbrand-Modelle[1] für S_H, die in diesem Abschnitt eingeführt werden, lassen sich aus H^* dann in systematischer Weise dieselben potentiell unendlich vielen geschlossenen Formeln gewinnen. Die Konjunktion endlicher Anfangsstücke dieser Formelfolge kann man auf Erfüllbarkeit untersuchen. Ist H nicht erfüllbar, so tritt das nach endlich vielen Schritten zutage. Dann ist keine Interpretation der in H auftretenden Prädikatssymbole möglich, die S_H und damit H wahr macht. Aufgrund der Geschlossenheit der Klauseln überträgt sich dies auf jedes andere mögliche Model.

Wir betrachten nun die minimale Signatur Σ', in der die gewählte Skolemisierung S_H von H formuliert werden kann. Diese unterscheidet sich i.A. von der Ausgangssignatur, da die Skolemisierung endlich viele Funktionssymbole hinzufügen kann, und H

[1] Jacques Herbrand, französischer Mathematiker (1908-02-12 bis 1931-07-27), beim Bergsteigen tödlich verunglückt

nicht alle Symbole der Ausgangssignatur enhalten mußte. Insofern ist letztere nur von untergeordneter Bedeutung.

Die Werte der variablenfreien Grundterme in $\mathcal{T}_{\Sigma'}(\varnothing)$, vergl. Bemerkung 9.4.3, sind in jedem Σ'-Model unabhängig von den Werten der Variablen; sie werden einzig von der Interpretation der Funktionssymbole bestimmt.

Wie nach Definition 9.4.1 festgestellt, bildet $\mathcal{T}_{\Sigma'}(\varnothing)$ auf kanonische Weise ein Model für die Teilsignatur Σ'_F, die nur die Funktionssymbole aus Σ' enthält.

13.0.2 Beispiel. Falls Σ' aus einer Konstante, 0, und einem einstelligen Operationssymbol, s, besteht, ist die Menge $\mathcal{T}_{\Sigma'}(\varnothing) = \{0, s(0), s(s(0)), \ldots\}$ isomorph zu $\mathbb{N}$. In jedem Σ'-Model $(\mathcal{A}, 0^{\mathcal{A}}, s^{\mathcal{A}})$ ist die Bewertung des Termes $s^n(0)$ (n-fache Anwendung des Funktionssymbols s auf die Konstante 0) gleich dem Element $(s^{\mathcal{A}})^n(0^{\mathcal{A}}) \in \mathcal{A}$, das durch n-fache Anwendung der Funktion $s^{\mathcal{A}}$ auf die Konstante $0^{\mathcal{A}}$ entsteht.

13.0.3 Beispiel. Für $\Sigma' = \langle +, 0, 1\rangle$ mit einer zweistelligen Operation $+$ und Konstanten 0 und 1 besteht $\mathcal{T}_{\Sigma'}(\varnothing)$ aus Termen wie $0, 1, 0+1, 0+0, 0+(0+1), (1+1)+(0+1), \ldots$. In jedem Σ'-Model haben wir eine offensichtliche Bewertung all dieser Grundterme.

13.0.4 Definition.

(a) Ein **Herbrand Universum** von H ist entweder die Menge $\mathcal{T}_{\Sigma'}(\varnothing)$ der Grundterme über Σ', sofern Σ' Konstanten enthält, oder $\mathcal{T}_{\Sigma'}(\{c\})$ für eine neue Konstante c.

(b) Die **Herbrand Basis** eines Herbrand Universums von H ist die kleinste Menge von Σ'-Formeln, die aus den Elementen des Herbrand Universums und den Relationssymbolen aus Σ'_P sowie $\doteq$ aufgebaut werden können.

(c) Ein **Herbrandsche Σ'-Model** ist ein Σ'-Model mit einem Herbrand Universum als Träger, der kanonischen Interpretation der Funktionssymbole in Σ'_F und einer beliebigen Interpretation der Relationssymbole aus Σ'_P.

13.0.5 Bemerkung. $\mathcal{T}_{\Sigma'}(\varnothing)$ ist natürlich nur dann nicht leer, wenn Σ'_F mindestens eine Konstante enthält. Anderfalls greift man zu einem Trick und erweitert Σ'_F um eine neue Konstante c.

Sofern ein Funktionssymbol der Stelligkeit > 0 in Σ'_F existiert, ist das Herbrand-Universum abzählbar unendlich. Die Herbrand Basis ist sogar immer abzählbar unendlich (warum?).

13.0.6 Satz. *Für jede prädikatenlogische Formel S in Skolemscher Normalform gilt: falls S erfüllbar ist, gilt S in einem Herbrandschen Model über der kleinsten Signatur Σ', in der S formuliert werden kann.*

Beweis. Den Beweis kann der Leser in „Logik für Informatiker“ von U. Schöning finden. □

13.0.7 Beispiel.

(a) Für $\Sigma' = \langle s, 0\rangle$ in Beispiel 13.0.2 gibt es nur ein Herbrandsches Model $\mathcal{A}$: der Träger $\mathcal{A}$ besteht aus allen Iterationen $s^n(0)$, $n \in \mathbb{N}$, des Funktionssymbols, angewendet auf die Konstante 0, und die Operation $s^{\mathcal{A}}$ ordnet jeden Term $s^n(0)$ den Term $s^{n+1}(0)$ zu.

(b) Das Hinzufügen z.B. eines 2-stelligen Prädikatssymbols $<$ zur obigen Signatur liefert vervielfacht die möglichen Herbrandsche Modelle aufgrund der vielen möglichen Interpretationen von $<$. Z.B. kann in $\mathcal{A}$ mit Elementen und Funktionen wie in (a) $<^{\mathcal{A}}$ definiert sein durch

$$\left(s^{\mathcal{A}}\right)^n(0) <^{\mathcal{A}} \left(s^{\mathcal{A}}\right)^m(0) \quad \text{g.d.w} \quad n < m$$

Oder im Model $\mathcal{B}$ mit denselben Elementen und Funktonen wie $\mathcal{A}$ kann $<^{\mathcal{B}}$ als leere Relation oder als Ungleichheit interpretiert werden.

13.0.8 Beispiel.

(a) Wenn wir die Erfüllbarkeit von Formeln in Skolemscher Normalform untersuchen wollen, z.B. von

$$\forall x : \forall y : x + y > a \tag{13.1}$$

können wir uns auf die Betrachtung Herbrandscher Modelle beschränken: falls ein Modell dieser Formel existiert, können wir aufgrund von Satz 13.0.6 annehmen, dass seine Elemente ausschließlich die Form $a, a+a, a+(a+a), (a+a)+a, \ldots$ haben und seine zweistellige Funktion die syntaktische Zusammensetzung $\langle s, t\rangle \longmapsto t + s$ ist. Da in der Formel nur die Konstante a auftritt, genügt es, die minimale Signatur zu betrachten, mit deren Hilfe diese Formel ausgedrückt werden kann, hier also $\Sigma'_F = \langle a, +\rangle$ mit a konstant und $+$ 2-stellig, und $\Sigma'_P = \langle <\rangle$ mit $<$ 2-stellig.

Die Formel ((a)) besagt nun, dass jeder Term t in a mit mindestens einem $+$-Zeichen die Relation $t > a$ erfüllt. Das ist sicherlich erfüllbar, etwa in einem Model $\mathcal{A}$ mit den positiven ganzen Zahlen $\mathbb{N} - \{0\}$ als Trägermenge, der Konstanten 1, der Addition $+$ und der größer-Relation $>$. Dort ist 1 die kleinste Zahl und einzige Zahl, die ohne Verwendung des $+$-Zeichens geschrieben werden kann.

(b) Ist die Formel

$$H := \forall x : \neg(x > a \vee a > x) \wedge \forall x : (x > b \Rightarrow x > a) \wedge a > b$$

erfüllbar? Zunächst benötigen wir H in bereinigter Form, dann in PNF, was in diesem Fall auch schon eine Skolemsche Normalform ist:

$$\begin{aligned} H &\equiv \forall x : \neg(x > a \vee a > x) \wedge \forall y : (y > b \Rightarrow y > a) \wedge a > b \\ &\equiv \forall x : \forall y : \big(\neg(x > a \vee a > x) \wedge (y > b \Rightarrow y > a) \wedge a > b\big) \end{aligned}$$

Mangels nicht-konstanter Funktionssymbole haben alle passenden Herbrand'schen Modelle die Trägermenge $\mathcal{T}_\Sigma(\varnothing) = \{a, b\}$. Gibt es eine Relation $>^{\mathcal{A}}$ auf $\{a, b\}$, die H erfüllt?

Instanziiert man in der zweiten Teilformel $y >^{\mathcal{A}} b \Rightarrow y >^{\mathcal{A}} a$ die Variable y durch a, erhält man $a >^{\mathcal{A}} b \Rightarrow a >^{\mathcal{A}} a$, zusammen mit der dritten Teilformel also $a >^{\mathcal{A}} a$, was aber der ersten Teilformel widerspricht, sofern dort x mit a instanziiert wird. Daraus folgt gemäß Satz 13.0.6, dass F nicht erfüllbar ist.

13.0.9 Bemerkung. Für Signaturen Σ ohne Prädikatensymbole kann man das Σ Model $\mathfrak{T}_\Sigma(\varnothing)$ mit der Gleichheit als Interpretation von $\doteq$ auch einem **abstraktne Datentype** identifizieren: in einem Datentyp nehmen wir an, dass es eine Liste von Operationen gibt, die mit den Daten durchgeführt werden können. Diese bilden dann eine Signatur ohne Prädikatensymbole. Der abstrakte Datentyp ist dann das Model, das durch Anwendung der Operationen auf die Grundterme entsteht, und für das auch keine nichttrivialen Gleichungen gelten. (Mathematisch gesprochen handelt es sich um „freie Σ-Algebren“ über der leeren Menge $\varnothing$.) Ein paar Beispiele:

13.0.10 Beispiel (Abstrakte Datentypen)**.**

(a) Natürliche Zahlen: Wir gehen von einer Konstante 0 und einer einstelligen Operation s aus. Für $\Sigma = \{s, 0\}$ haben wir schon gesehen, dass $\mathfrak{T}_\Sigma(\varnothing)$ die Menge aller Terme $0, s(0), s(s(0)), \ldots$ ist, die die natürlichen Zahlen $0, 1, 2, \ldots$ codieren

(b) Listen. Um die Menge A^* aller Listen in Alphabet A zu formen, brauchen wir eine Konstante nil und für jedes Symbol $a \in A$ die einstellige Operation, die zu jeder Liste a am Anfang hinzufügt. Wir betrachten die Signatur

$$\Sigma = A \cup \{\mathsf{nil}\}$$

in der jedes Symbol $a \in A$ ein einstelliges Funktionssymbol ist. Hier besteht $\mathfrak{T}_\Sigma(\varnothing)$ aus den Termen

$$\mathsf{nil}, a_0(\mathsf{nil}), a_1(a_0(\mathsf{nil})), \ldots (a_i \in A)$$

die die Listen $\mathsf{nil}, a_0, a_0a_0, \ldots$ codieren.

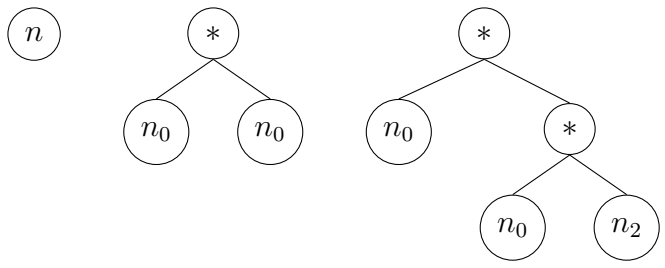
(c) Binäre Bäume: Die Signatur

$$\Sigma = \{*\} \cup \mathbb{N}$$

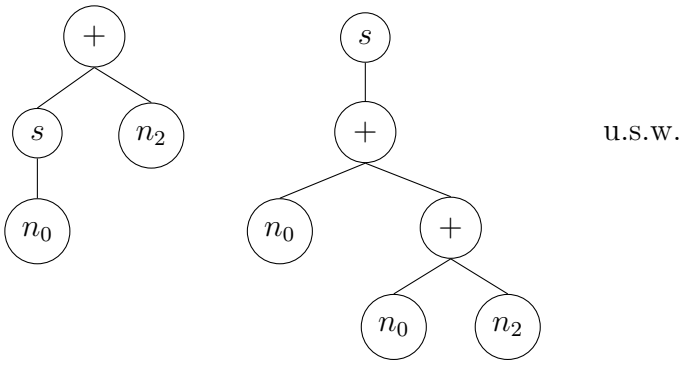
mit einem zweistelligen Operationssymbol $*$ und jeder natürlichen Zahl n als Konstante hat Terme

$$n, *(n_0, n_0), *n_0(*(n_0, n_2)), \ldots$$

die alle binären Syntaxbäume codieren, deren Blätter von links nach rechts durch Listen natürlicher Zahlen markiert werden:



13.0.11 Beispiel. Die natürlichen Zahlen wollen wir jetzt nicht nur mit der Operation s (Nachfolger) und der Konstante 0 versehen, sondern auch mit der Addition + und der Konstante 1. Die ensprechende Signatur $\Sigma = \langle +, s, 0, 1\rangle$ hat als abstrakten Datentyp allerdings **nicht** die natürliche Zahlen, sondern Bäume des Types



u.s.w.

Ähnich wie im Fall der gerichteten/ungerichteten Graphen (Beispiel 10.0.10) müssen wir hier geeignete Axiome hinzufügen, um die möglichen Modelle stärker einzugrenzen. Dazu betrachten wir die Formeln, die fordern, dass + kommutativ und assoziativ ist, dass 0 die Einheit und s die Nachfolgerfunktion ist:

$$\forall x : \forall y : x + y = y + x$$
$$\forall x : \forall y : \forall z : (x + y) + z = (x + y + z)$$
$$\forall x : s(x) = x + 1$$
$$\forall x : x = x + 0$$

Hier betrachten wir den abstrakten Datentyp als ein Model, die (wieder) aus nichts als den Konstanten entsteht und für die nur die Gleichung, die aus den 4 Formeln ableitbar sind, gelten. In unserem Beispiel liefert dies wieder eine Codierung der natürlichen Zahlen $\mathbb{N}$.

14. Resolutionsmethode der Prädikatenlogik

Bei der Frage, ob eine Skolem'sche Normalform S_H der Formel $H = \bigwedge \Gamma \wedge \neg F$ erfüllbar ist, wollen wir nun ausnutzen, dass der quantorenfreie Teil H^* von S_H in KNF vorliegt.

14.0.1 Beispiel. Ist die folgende Formel erfüllbar?

$$H := \neg P(c) \wedge \forall x : \big(P(f(x)) \wedge \neg P(x)\big)$$

Wir bringen sie erst in Skolemsche Normalform:

$$S_H := \forall x : \big(\neg P(c) \wedge P(f(x)) \wedge \neg P(x)\big)$$

Speziell kann von x auch den Wert c annehmen: Falls also F (und S_F) erfüllbar ist, dann gilt dies auch für

$$\neg P(c) \wedge P(f(c)) \wedge \neg P(c) \equiv \neg P(c) \wedge P(f(c)) \tag{14.1}$$

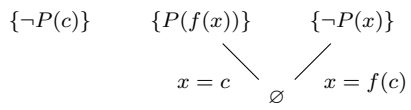
Analog dürfen wir x auch mit $f(c)$ ersetzen: Ist F erfüllbar, so ist es auch die Formel:

$$\neg P(c) \wedge P(f(f(c))) \wedge \neg P(f(c)) \tag{14.2}$$

Beides sind Formeln aus der Herbrand-Basis über dem Herbrand Universum von H. Also folgt aus der Erfüllbarkeit von F auch die der Konjunktion von (14.1) und (14.2)

$$\neg P(c) \wedge P(f(c)) \wedge \neg P(c) \wedge P(f(f(c))) \wedge \neg P(f(c)) \tag{14.3}$$

Nun ist die Antwort aber klar: Diese letzte Formel ist unerfüllbar weil die komplementären Klauseln $P(f(c))$ und $\neg P(f(c))$ die Resolvente $\varnothing$ haben. Ein entsprechender Resolventengraph verzeichnet entlang der Kanten die nötigen Substitutionen:



14.0.2 Bemerkung. Das obige Beispiel ist typisch. Wir haben in der Formel F eine Konstante c benutzt und die durch $\forall$ quantifizierte Variable x haben wir mit c oder $f(c)$ ersetzt. Dies dürfen wir, weil der All-Quantor auch jene Elemente des Trägers erfasst, die für die Konstante oder für $f(c)$ stehen.

Dadurch haben sich immer längere geschlossene Formeln in KNF ergeben, auf die man die klassische Resolutionsmethode der AL anwenden kann.

14.0.3 Bemerkung. Für eine Skolemsche Normalform

$$S_H := \forall y_0 : \ldots \forall y_{n-1} : H^*$$

können wir neue geschlossene Formeln in KNF formen, indem wir in den Klauseln von F jede Variable y_i durch einen Grundterm der Signatur des Herbrand Universums ersetzen (siehe Bemerkung 9.4.3). Wenn H erfüllbar ist, gilt dies auch für jede so geformte KNF, und für deren Konjunktionen. Der folgende Resolutionssatz besagt die Umgekehrung: Wenn jede solche KNF erfüllbar ist, dann ist H erfüllbar.

14.0.4 Beispiel. Ist die Formel

$$\forall x : \exists y : R(x, y)$$

erfüllbar? Wir führen eine 1-stellige Operation f ein und erhalten die Skolemsche Normalform

$$\forall x : R(x, f(x))$$

Diese hat keine Konstanten, also benötigt die relevante Signatur für das Herbrand Universum eine neue Konstante c:

$$\Sigma' = \langle f, c; R \rangle$$

Das Herbrand Universum hat nun die Form

$$\mathfrak{T}_{\Sigma'}(\varnothing) = \{c, f(c), f(f(c)), \ldots\} \cong \mathbb{N}$$

Nun können wir x mit dessen Elementen belegen und erhalten eine Folge von Formeln in KNF

$$\begin{aligned} &R(c, f(c)) \\ &R(c, f(c)) \wedge R(f(c), f(f(c))) \\ &R(c, f(c)) \wedge R(f(c), f(f(c))) \wedge R(f(f(c)), f(f(f(c)))) \\ &\quad\vdots \end{aligned}$$

in der Herbrand Basis. All diese Formeln sind erfüllbar, und ihre Konjunktionen ebenfalls, weil mangels negativer atomarer Formeln keine Resolventen gebildet werden können. Daraus folgt, dass F erfüllbar ist:

14.0.5 Satz (Resolutionssatz der Prädikatenlogik)**.** *Gegeben sei eine Skolemsche Normalform*

$$S_H := \forall y_0 : \ldots \forall y_{n-1} : (K_0 \wedge \ldots \wedge K_{n-1})$$

mit Minimalsignatur Σ, wobei die K_i, $i < n$, quantorenfreie Klauseln sind. Die Formel S_H ist genau dann erfüllbar, wenn jede geschossene KNF, deren Klauseln die Form

$$K_i\left[{}^{t_0}/_{y_0}, \ldots, {}^{t_{n-1}}/_{y_{n-1}}\right] \qquad \text{mit } t_0, \ldots, t_{n-1} \text{ in } \mathfrak{T}_\Sigma(\varnothing)$$

haben, erfüllbar ist. (In der Klausel K_i wird die k-te Variable durch einem geschlossenen Term t_k aus dem Herbrand Universum ersetzt.)

14.0.6 Algorithmus. Die **Resolutionsmethode** der Prädikatenlogik ist eine Methode zur Entscheidung, ob eine geschlossene Formel H erfüllbar ist.

(PR0): Berechne eine Skolemsche Normalform

$$\forall y_0 : \ldots \forall y_{n-1} : H^* \qquad\qquad H^* \text{ ohne Quantoren und in KNF}$$

die genau dann erfüllbar ist, wenn H es ist.

(PR1): Für die zugehörige Minimalsignatur Σ' berechne das Herbrand Universum $\mathcal{T}_{\Sigma'}(\varnothing)$ oder $\mathcal{T}_{\Sigma'}(\{c\})$ zu $\{t_0, t_2, t_3, \ldots\}$.

(PR2): Für $k > 0$ betrachte alle zulässigen Substitutionen der Terme t_i, $i < k$, für die Variablen in H^* und wende die Resolutionsmethode der Aussagenlogik auf die Konjunktion aller Substitutionsergebnisse (geschlossene KNF-Formeln in der Herbrand Basis) an. Ist diese Konjunktion unerfüllbar, so terminiere mit

$$G \text{ unerfüllbar}$$

Andernfalls setze $k = k + 1$.

14.0.7 Beispiel. Ist die Formel

$$G := (\forall x : R(x) \Rightarrow \exists y : S(y)) \Rightarrow \exists x : (R(x) \Rightarrow S(x))$$

allgemeingültig? Wie auch bei der Aussagenlogik ist das genau dann der Fall, wenn die Negation nicht erfüllbar ist:

$$\begin{aligned}
\neg G &:= \neg((\forall x : R(x) \Rightarrow \exists y : S(y)) \Rightarrow \exists x : (R(x) \Rightarrow S(x))) \\
&\equiv \neg(\neg(\forall x : R(x) \Rightarrow \exists y : S(y)) \vee \exists x : (R(x) \Rightarrow S(x))) \\
&\equiv (\forall x : R(x) \Rightarrow \exists y : S(y)) \wedge \neg\exists x : (R(x) \Rightarrow S(x)) \\
&\equiv (\forall x : R(x) \Rightarrow \exists y : S(y)) \wedge \forall x : \neg(R(x) \Rightarrow S(x)) \\
&\equiv (\neg\forall x : R(x) \vee \exists y : S(y)) \wedge \forall x : (R(x) \wedge \neg S(x)) \\
&\equiv (\exists y : S(y) \vee \exists x : \neg R(x)) \wedge \forall z : (R(z) \wedge \neg S(z))
\end{aligned}$$

Dies ist bereits bereinigt, deswegen

$$\neg G \equiv \exists y : \exists x : \forall z : \big((S(y) \vee \neg R(x)) \wedge R(z) \wedge \neg S(z)\big)$$

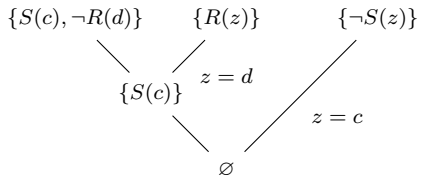
Die Skolemsche Normalform benutzt zwei neue Konstanten c und d, die $\exists y$ und $\exists x$ entsprechen:

$$S_{\neg G} = \forall z : \big((S(c) \vee \neg R(d)) \wedge R(z) \wedge \neg S(z)\big)$$

Wir haben keine Funktionssymbole außer c und d, daher ist das Herbrand Universum endlich:

$$\mathcal{T}_{\Sigma'}(\varnothing) = \{c, d\}$$

Die Resolutionsmethode liefert dann



Dies zeigt, dass $\neg G$ nicht erfüllbar, G also allgemeingültig ist.

14.0.8 Bemerkung. In der Aussagenlogik ist die Resolutionsmethode ein entscheidbarer Algorithmus: Nach endlich vielen Schritten wissen wir, ob die gegebene Formel erfüllbar ist. Der Algorithmus der Prädikatenlogik terminiert jedoch nicht immer: Für erfüllbare Formeln bekommen wir nie in endlich vielen Schritten die Antwort. Dies ist kein Zufall, wie der nächste Satz belegt.

14.0.9 Satz. ***Churchscher Satz***
Die Prädikatenlogik ist nicht entscheidbar: Es gibt keinen Algorithmus, der für jede Formel F in endlich vielen Schritten entscheidet, ob F erfüllbar ist.

Die Unentscheidbarkeit wird intuitiv klar, wenn wir zeigen, dass es Formeln gibt, die

- erfüllbar sind,
- aber in keinem endlichen Model erfüllt werden.

Denn dann benötigen wir ein unendliches Model, um die Erfüllbarkeit zu demonstrieren.

14.0.10 Beispiel. Wir arbeiten mit einem 2-stelligen Relationssymbol $<$. Diese soll formal transitiv $x < y \wedge y < z \Rightarrow x < z$ und antireflexiv $\neg(x < x)$ sein. Zusätzlich verwenden wir ein 1-stelliges Funktionssymbol f, dessen Interpretation echt formal wachsen sollen: erfüllt:

$$F := \forall x : \forall y : \forall z : ((x < y \wedge y < z \Rightarrow x < z) \wedge \neg(x < x) \wedge x < f(x))$$

Diese Formel ist sicher erfüllbar: im Model $\mathcal{N}$ mit dem Träger $\mathbb{N} = \{0, 1, 2, \dots\}$, wobei $<^{\mathcal{N}}$ die übliche Ordnung der natürlichen Zahlen und $f^{\mathcal{N}}$ die Nachfolgerfunktion ist.

Aber F gilt in keinem endlichen Model $\mathcal{A}$. Betrachten wir ein Model, in dem F gilt. Wir wählen ein beliebiges Element a_0 des Trägers und setzen $\wedge_{n+1} = f^{\mathcal{A}}(a_n)$. Für jedes x gilt $x < f(x)$, deswegen $a_0 < a_1 < a_2 \dots$. Dies beweist, dass die Elemente a_n paarweise verschieden sind: aus $a_i = a_j$ folgt $i = j$ denn jedes x erfüllt $\neg(x < x)$ (und die Transitivität ergibt $a_i < a_j$ oder $a_j < a_i$ falls $i \neq j$).

14.0.11 Beispiel. Das mittelalterliche Argument zur Identifkation von Hexen in Monty Pythons "The Holy Grail"(„Die Ritter der Kokosnuß“) verwendet folgende Prämissen:

1. Was machen wir mit Hexen? Wir verbrennen sie.
2. Was verbrennen wir noch? Holz.
3. Was zeichnet Holz sonst noch aus? Es schwimmt.
4. Was schwimmt sonst noch? Eine Ente.
5. (Implizit) Alles, was genausoviel wiegt, wie ein Objekt, das schwimmt, schwimmt ebenfalls.
6. Es wird experiementell festgestellt, dass die junge Dame genausoviel wiegt, wie eine Ente.

Schlußfolgerung: Die junge Dame (Conny Booth) ist eine Hexe!

Ausgangspunkt für die folgende Analyse ist das YouTube-Video von Yannis Haralambous.

Bevor wir die Argumente formalisieren, wählen wir eine Signatur Σ mit einer Konstante, YL ("young lady"), einem 1-stelligen Funktionssymbol w (für das Gewicht), und fünf einstelligen Relationssymbolen Wi (Hexe), Wo (Holz), B (wird verbrannt), F (schwimmt) und D (ist eine Ente).

Die Menge Γ der Prämissen besteht aus

1. $\forall x : \big(Wi(x) \Rightarrow B(x)\big)$;
2. $\forall x : \big(Wo(x) \Rightarrow B(x)\big)$;
3. $\forall x : \big(Wo(x) \Rightarrow F(x)\big)$;
4. $\forall x : \big(D(x) \Rightarrow F(x)\big)$;
5. $\forall x : \forall y : \Big(F(x) \wedge \big(w(x) \doteq w(y)\big) \Rightarrow F(y)\Big)$;
6. $\exists z : D(z) \wedge w(YL) \doteq w(z)$;

und die gewünschte Schlußfolgerung ist $Wi(YL)$; die Konjunktion von deren Negation $\neg Wi(YL)$ mit $\bigwedge \Gamma$ bildet die zu analysierende Formel H.

Im Hinblick auf eine einfache Skolem Normalform bilden wir folgende Pränex Normalform

$$\begin{aligned} H = \exists z : \forall x : \forall y : \Big(& D(z) \wedge w(YL) \doteq w(z) \\ & \wedge \big(\neg Wi(x) \vee B(x)\big) \\ & \wedge \big(\neg Wo(x) \vee B(x)\big) \\ & \wedge \big(\neg Wo(x) \vee F(x)\big) \\ & \wedge \big(\neg D(x) \vee F(x)\big) \\ & \wedge \big(\neg F(x) \vee \neg\big(w(x) \doteq w(y)\big) \vee F(y)\big) \\ & \wedge \neg Wi(YL)\Big) \end{aligned}$$

und daraus die Skolem Normalform mit der neuen Konstante d

$$\begin{aligned} S_H = \forall x : \forall y : \Big(& D(d) \wedge w(YL) \doteq w(d) \\ & \wedge \big(\neg Wi(x) \vee B(x)\big) \\ & \wedge \big(\neg Wo(x) \vee B(x)\big) \\ & \wedge \big(\neg Wo(x) \vee F(x)\big) \\ & \wedge \big(\neg D(x) \vee F(x)\big) \\ & \wedge \big(\neg F(x) \vee \neg\big(w(x) \doteq w(y)\big) \vee F(y)\big) \\ & \wedge \neg Wi(YL)\Big) \end{aligned}$$

Der quantorenfreie Teil H^* besteht also aus acht Klauseln (0) – (7) in der obigen Reihenfolge, während das Herbrandt Universum zwei Folgen von Termen enthält

$$\{\, w^n(YL) \,:\, n \in I\!N \,\} \cup \{\, w^n(d) \,:\, n \in I\!N \,\}$$

Ein Resolventengraph wird leider zu unübersichtlich. Ausgehend von den Klauseln

$$\begin{array}{ll} (0) : D(d) & (4) : \big(\neg Wo(x) \vee F(x)\big) \\ (1) : w(YL) \doteq w(d) & (5) : \big(\neg D(x) \vee F(x)\big) \\ (2) : \neg Wi(x) \vee B(x) & (6) : \big(\neg F(x) \vee \neg\big(w(x) \doteq w(y)\big) \vee F(y)\big) \\ (3) : \neg Wo(x) \vee B(x) & (7) : \neg Wi(YL) \end{array}$$

listen wir die möglichen Resolventen tabellarisch auf:

$$\begin{array}{rl} (0), (5)[{}^{\mathrm{d}}/_{\mathrm{x}}] \longrightarrow & (8) : F(d), \text{ eliminiert } (5)[{}^{\mathrm{d}}/_{\mathrm{x}}] \\ (1), (6)[{}^{\mathrm{d}}/_{\mathrm{x}}, {}^{\mathrm{YL}}/_{\mathrm{y}}] \longrightarrow & (9) : \neg F(d) \vee F(YL), \text{ eliminiert } (6)[{}^{\mathrm{d}}/_{\mathrm{x}}, {}^{\mathrm{YL}}/_{\mathrm{y}}](*) \\ (1), (6)[{}^{\mathrm{YL}}/_{\mathrm{x}}, {}^{\mathrm{d}}/_{\mathrm{y}}] \longrightarrow & (10) : \neg F(YL) \vee F(d), \text{ eliminiert } (6)[{}^{\mathrm{YL}}/_{\mathrm{x}}, {}^{\mathrm{d}}/_{\mathrm{y}}](*) \\ (4), (6)^* \longrightarrow & (11) : \neg Wo(x) \vee \neg(w(x) \doteq w(y)) \vee F(y), \text{ ohne } [{}^{\mathrm{d}}/_{\mathrm{x}}, {}^{\mathrm{YL}}/_{\mathrm{y}}], [{}^{\mathrm{YL}}/_{\mathrm{x}}, {}^{\mathrm{d}}/_{\mathrm{y}}] \\ (5), (6)^* \longrightarrow & (12) : \neg D(x) \vee \neg(w(x) \doteq w(y)) \vee F(y), \text{ ohne } [{}^{\mathrm{d}}/_{\mathrm{x}}, {}^{\mathrm{YL}}/_{\mathrm{y}}], [{}^{\mathrm{YL}}/_{\mathrm{x}}, {}^{\mathrm{d}}/_{\mathrm{y}}] \\ (8), (9) \longrightarrow & (13) : F(YL), \text{ eliminiert } (9) \\ (0), (12)[{}^{\mathrm{d}}/_{\mathrm{x}}] \longrightarrow & (14) : \neg(w(d) \doteq w(y)) \vee F(y), \text{ ohne } [{}^{\mathrm{YL}}/_{\mathrm{y}}] \end{array}$$

Analog zur praktikablen Resolutionsmethode der Aussagenlogik, Algorithmus 6.0.17, haben wir versucht, Klauseln aus der Konstruktion zu entfernen, sobald kleinere Klauseln auftreten. Bei Klauseln mit freien Variablen werden evtl. nur einzelne Substitutionen von der weiteren Berechnung ausgeschlossen, wie bei den Resolventen (8), (9) und (10). Wenn nachfolgend nochmal auf (5) oder (6) zugegriffen werden soll, dann nur mit anderen als den verbotenen Substitutionen. Diese Einschränkungen ziehen sich natürlich auf auf die resultierenden weiteren Resolventen durch, hier (11) und (12). Aus diesem Grund brauchen auch keine Resolventen von (11) und (12) mit (1) gebildet zu werden.

Da keine weiteren Resolventen geformt werden können, tritt insbesondere $\varnothing$ nicht als Resolvente auf, d.h. die Formel H ist erfüllbar, und man kann $Wi(YL)$ *nicht* aus den Prämissen folgern.

A. Mathematische Grundlagen

A.1 Mengen

Achtung: Die folgenden Definitionen und Ergebnisse sind zwangsläufig informell, da wir hier die Sprache der Logik nicht benutzen können, und stattdessen auf die Metasprache angewiesen sind.

A.1.1 Definition. Eine *Menge* ist eine Zusammenfassung unterscheidbarer Objekte zu einer Gesamtheit. Die *Zugehörigkeit* eines Objekts a zu einer Menge A wird mit $a \in A$ ausgedrückt; a heißt dann *Element* von A.

Mengen können spezifiziert werden

- entweder durch Angabe ihrer Elemente (dies funktioniert nur für endliche Mengen, aber die Reihenfolge der Auflistung ist irrelevant),
- oder durch Spezifikation einer Eigenschaft, die alle gewünschten Elemente einer schon bekannten Mente haben sollen; dafür verwendet man die Schreibweise mit den sog. Mengenklammern, etwa:

$$M = \{\, x \,:\, x \in A \text{ und } x \text{ ist rot} \,\}$$

Auf die oben beschriebene Weise lassen sich bisher nur endlichen Mengen bestimmen

Wir sind meist an Mengen abstrakter Objekte interessiert, wie z.B. Mengen von Zahlen, oder Punkten in der Ebene. Bestimmte häufig auftretende Mengen haben besondere Namen:

A.1.2 Definition. Speziell hat die *leere Menge* $\emptyset$ keine Elemente, während $\mathbb{N}$ die Menge der *natürlichen Zahlen* $0, 1, 2, \ldots$ bezeichnet. Ein-elementige Mengen werden auch als *Singletons* bezeichnet. Wir schreiben $|A|$ für die Größe einer (endlichen) Menge A.

Achtung: in dieser Vorlesung ist 0 eine natürliche Zahl!

A.1.3 Definition (Teilmenge, Potenzmenge)**.** Eine Menge A heißt *Teilmenge* von B (geschrieben $A \subseteq B$), falls jedes Element von A auch Element von B ist, halbformal: $x \in A$ impliziert $x \in B$.

Die *Potenzmenge*$P(B)$ besteht aus allen Teilmengen von B, halbformal

$$P(B) = \{\, A \,:\, A \subseteq B \,\}$$

Jede Menge A erfüllt $A \subseteq A$, $\emptyset \subseteq A$, $A \in P(A)$ und $\emptyset \in P(A)$.

A.1.4 Definition (Durchschnitt, Vereinigung)**.**

- Binäre Variante: Der *Durchschnitt* bzw. die *Vereinigung* zweier Mengen A und B sind gegeben durch
$$A \cap B := \{\, x \,:\, x \in A \,\text{und}\, x \in B \,\}$$
$$A \cup B := \{\, x \,:\, x \in A \,\text{oder}\, x \in B \,\}$$

- Allgemeine Variante: Im Falle einer Teilmenge $\mathcal{A} \subseteq P(X)$ erhalten wir
$$\bigcap \mathcal{A} := \{\, x \,:\, x \in A, \text{für alle}\, A \in \mathcal{A} \,\} \subseteq X$$
$$\bigcup \mathcal{A} := \{\, x \,:\, x \in A \text{für mindestens ein}\, A \in \mathcal{A} \,\} \subseteq X$$

Man beachte, dass die Teilmenge $\mathcal{A}$ von $P(X)$ auch leer sein darf, oder mit $p(X)$ übereinstimmen kann. das liefert die Spezialfälle

$$\bigcap \emptyset = \bigcup P(X) = X \quad \text{sowie} \quad \bigcup \emptyset = \bigcap P(X) = \emptyset$$

A.1.5 Definition (cartesisches Produkt)**.**

- Binäre Variante: Das *cartesische Produkt* zweier Mengen A und B ist
$$A \times B := \{\, \langle a, b\rangle \,:\, a \in A \,\text{und}\, b \in B \,\}$$
wobei $\langle a, b\rangle$ ein *geordnetes Paar* bezeichnet. Die Spezifikation für geordnete Paare verlangt
$$\langle a, b\rangle = \langle c, d\rangle \quad \text{genau dann wenn} \quad a = c \,\text{und}\, b = d)$$

- Allgemeine Variante: Im Falle einer Familie A_i, $i \in I$, von Mengen, erhalten wir die Menge der sog. *Auswahlfunktionen*, die für jeden Index $i \in I$ ein Element von A_i auswählt:
$$\prod_{i \in I} A_i := \{\, \langle a_i \,:\, i \in I\rangle \,:\, a_i \in A_i \,\}$$
Im Spezialfall $A_i = A$ für alle $i \in I$ schreiben wir abkürzend A^I für das I-fache cartesische Produkt von A mit sich.

Die Notation für Auswahlfunktionen (oder I-Tupel) ähnelt derjenigen für die Mengenspezifikation mit dem Unterschied, dass hier die Reihenfolge entscheident ist.

Im Unterschied zu Durchschnitt und Vereinigung ist beim cartesischen Produkt die Reihenfolge der Mengen wichtig; i.A. stimmen $A \times B$ und $B \times A$ nicht überein (obwohl es natürlich eine kanonische Bijektion zwischen diesen Mengen gibt, die das geordnete Paar $\langle a, b\rangle$ aud $\langle b, a\rangle$ abbildet (s.u.). Dafür verantwortlich ist natürlich die Spezifikation geordneter Paare, derzufolge die Paare $\langle a, b\rangle$ aud $\langle b, a\rangle$ i.A. nicht übereinstimmen.

Man beachte weiterhin, dass wir noch keine *Realisierung* von geordneten Paaren vorgestellt haben, nur ihre Spezfikation. Man kann verschiedene Realisierungen finden, aber die entsprechenden Mengen $A \times B$ lassen sich alle auf kanonische Weise bijektiv ineinander überführen (sind also canonisch isomorph, s.u.). Insofern ist die Frage der Realisierung irrelevant.

Der einzig pathologische Fall ist der einer leeren Indexmenge:

$$\prod_{i\in\emptyset} A_i := \{\varepsilon\}$$

was das neutrale Element für das cartesische Produkt liefert. Das einzig wichtige hierbei ist, dass es sich beim leeren Produkt um ein Singleton handelt, die Identität des einzigen Elements ist eigentlich irrelevant. Oft wird es aber mit $\boldsymbol{\varepsilon}$ bezeichnet und *leeres Wort* genannt; mehr dazu in Abschnitt A.6.

A.1.6 Definition (disjunkte Vereinigung)**.** Für die *Summe* oder *disjunkte Vereinigung* zweier Mengen A and B geben wir diesmal eine Realisierung an:

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

Jede andere Methode zur "Disjunktifizierung" der Mengen A und B ist ebenfalls zulässig, z.B. folgende von Paul Taylor

$$A + B \cong \{\, \langle U, V\rangle \in P(A) \times P(B) \,:\, |U| + |V| = 1 \,\}$$

Natürlich gibt es auch hier neben der binären eine allgemeine Variante; wir überlassen es der Leserin, sie zu formuieren und den pathologischen Fall der leeren disjunkten Vereinigung zu analysieren.
Die Summenzeichen + bzw. $\sum$ weisen immer darauf hin, dass es sich um eine disjunkte Vereinigung handelt, im Gegensatz zu den Vereinigungsoperatoren $\cup$ bzw. $\bigcup$.

A.2 Relationen und Funktionen

A.2.1 Definition (Relation, partielle Funktion, Funktion)**.** Unter einer *binrären Relation* r von einer Menge A in eine Menge B verstehen wir eine Teilmenge $r \subseteq A \times B$; Schreibweise: $A \xrightarrow{r} B$. Ihr *Definitionsbereich* ist

$$D(r) := \{\, a \in A \,:\, \text{es gibt ein } b \in B \text{ mit } \langle a, b\rangle \in r \,\}$$

Die *Komposition* mit einer Relation $B \xrightarrow{s} C$ definiert man als

$$r; s := \{\langle a, c\rangle \in A \times C \,:\, \text{es gibt ein } b \in B. \text{ mit } \langle a, b\rangle \in r \text{ und } \langle b, c\rangle \in s \,\}$$

Eine Relation $A \xrightarrow{r} B$ heißt

- *total*, falls jedes Element in A mit mindestens einem Element in B in Relation steht, mit anderen Worten, falls $D(r) = A$ gilt;
- *einwertig* oder *partielle Funktion* (Schreibweise: $A \xrightarrow{r} B$), falls jedes Element in A mit höchstens einem Element aus B in Relation steht (traditionelle Schreibweise: $f(a)$);
- *Funktion* (Schreibweise: $A \xrightarrow{r} B$), falls r total und einwertig ist.

Eine Funktion $A \xrightarrow{f} B$ heißt

- *injektiv*, falls aus alle Elemente von A verschiedene Funktionswerte haben, d.h., aus $f(a) = f(b)$ folgt $a = b$;
- *surjektiv*, falls jedes Element von B als Funktionswert auftritt, d.h., zu jedem $b \in B$ existiert ein $a \in A$ mit $f(a) = b$;
- *bijektiv*, falls f injektiv und surjektiv ist.

Funktionen der Form $A^n \xrightarrow{f} B$ heißen auch *n-stellige* Funktionen.

Wie Relationen identifizieren wir auch (partielle) Funktionen von A nach B mit ihren "Graphen", also der bildlichen Darstellung, in der AB-Ebene, in der die A-Werte horizontal und die B-Werte vertikal aufgetragen werden.

Unglücklicherweise unterscheidet sich diese Konvention durch Spiegelung von der Darstellung einer Relation $A \xrightarrow{r} B$ als *binäre $A \times B$-Matrix*, d.h., als Funktion $tmorA \times Br\{0,1\}$, wo die Zeilen durch A und die Spalten durch B indexiert sind. Der Vorteil der Matrix-Darstellung besteht darin, dass Die Komposition von Relationen genau der *Matrix-Multiplikation* entspricht, bei der anstelle der Addition die Infimum-Operation verwendet wird.

Man beachte auch, dass die traditionelle Schreibweise $g \circ f$ für die Komposition zweier Funktionen $A \xrightarrow{f} B \xrightarrow{g} C$ entgegen der Pfeilrichtung erfolgt, aber dasselbe liefert, wie die Komposition als Relationen, d.h., $g \circ f = f; g$.

Anhand der Pfeilspitzen $\longrightarrow$, $\rightarrowtail$ und $\rightarrow$ unterscheiden wir (in dieser VL) Relationen und (partielle) Funktionen.

A.2.2 Definition (spezielle Relationen)**.**

- Für jede Menge A setze $\Delta_A = \{\, \langle a, a\rangle \,:\, a \in A \,\}$.
- Die zu $A \xrightarrow{r} B$ *duale Relation* $B \xrightarrow{r^{\mathrm{op}}} A$ ist durch $\langle b, a\rangle \in r^{\mathrm{op}}$ gdw. $\langle a, b\rangle \in r$ spezifiziert.
- $A \xrightarrow{r} A$ heißt

(r) *reflexiv*, falls $r^0 := \Delta_A \subseteq r$;

(t) *transitiv*, falls $r; r \subseteq r$;

(s) *symmetrisch*, falls $r = r^{\mathrm{op}}$;

(a) *antisymmetrisch*, falls $r \cap r^{\mathrm{op}} \subseteq \Delta_A$;

(i) *idempotent*, falls $r; r = r$;

- $r^* = \bigcup_{n\in\mathbb{N}} r^n$ heißt *reflexive transitive Hülle* von $A \xrightarrow{r} A$.
- Relationen $1 \longrightarrow B$ und $B \longrightarrow 1$ entsprechen Funktionen $1 \times B \longrightarrow 2$ bzw. $B \times 1 \longrightarrow 2$ und somit bijektiv den Teilmengen von B. Wegen $1 \times B \cong B \cong B \times 1$ lassen sich sog. *charakteristische Funktionen* $B \longrightarrow 2 = \{0, 1\}$ mit den Teilmengen von B identifizieren, etwa vermöge des Urbildes der 1.

- Funktionen $1 \longrightarrow B$ lassen sich mit den Elemente von B identifizieren. Folglich kann man für eine Funktion $B \xrightarrow{g} C$ den Funktionswert $g(b)$ an der Stelle $b \in B$ auch mit $b; g$ bezeichnen, wegen $1 \xrightarrow{b} B \xrightarrow{g} C$.

A.3 Geordnete Mengen

Zwar sind wir von den Zahlen her lineare Ordnungen gewohnt, aber es gibt auch andere Ordnungen, etwa die Mengeninklusion auf einer Potenzmenge. Daher muß auch dieser Begriff formalisiert werden.

A.3.1 Definition. Eine binäre Relation $A \xrightarrow{r} A$ heißt

(QO) *Quasi-Ordnung*, sofern (r) und (t) erfüllt sind;

(ÄR) *Äquivalenzrelation*, sofern (r), (t) und (s) erfüllt sind.

(HO) *Halb-* oder *partielle Ordnung*, sofern (r), (t) und (a) erfüllt sind;

(LO) *lineare Ordnung*, sofern r eine Halb-Ordnung ist und je zwei Elemente bzgl. r vergleichbar sind, d.h., für alle $\langle a, b\rangle \in A^2$ gilt $\langle a, b\rangle \in r$ oder $\langle b, a\rangle \in r$.

Auch wenn Äquivalenzrelationen mit (r) und (t) wesentliche Eigenschaften einer Ordnung erfüllen, so sind sie aufgrund von (s) etwas pathologische Vertreter dieser Spezies: zwei Elemente sind entweder in beiden Richtungen vergleichbar, und somit äquivalent, oder gar nicht vergleichbar. Für jede Prä-Ordnung $\sqsubseteq$ ist der Durchschnitt mit $\sqsubseteq^{\mathrm{op}} = \sqsupseteq$ eine Äquivalenzrelation, und $\sqsubseteq$ induziert eine Halb-Ordnung auf der Menge der Äquivalenzklassen (siehe Abschnitt A.5).

Ein generischer Name für eine Quasi-Ordnung ist $\sqsubseteq$, sofern es sich nicht um eine Äquivalenzrelation handelt.

A.3.2 Definition. $\langle P, \sqsubseteq\rangle$ sei eine halb-geordnete Menge und $X \subseteq P$. Ein Element $a \in P$ heißt

- *obere Schranke* von X, geschrieben $X \sqsubseteq a$, falls $x \sqsubseteq a$ für alle $x \in X$ gilt; $X^{\uparrow}$ bezeichnet die Menge aller oberen Schranken von X;
- *untere Schranke* von X, geschrieben $a \sqsubseteq X$, falls $a \sqsubseteq x$ für alle $x \in X$ gilt; $X^{\downarrow}$ bezeichnet die Menge aller oberen Schranken von X.
- Existiert die kleinste obere Schranke für $X \subseteq P$, so wird diese mit $\bigsqcup X$ bezeichnet und *Supremum* von X genannt; besteht X aus zwei Elmenten x und y, so schreibt man auch $x \sqcup y$. Im Falle $X = P$ kürzt man $\bigsqcup P$ häufig als 0 oder $\bot$ ab und spricht dann vom *kleinsten Element* der Halbordnung $\langle P, \sqsubseteq\rangle$.
- Existiert die größte untere Schranke für $X \subseteq P$, so wird diese mit $\bigsqcap X$ bezeichnet und *Infimum* von X genannt; besteht X aus zwei Elmenten x und y, so schreibt man auch $x \sqcap y$. Im Falle $X = P$ kürzt man $\bigsqcap P$ häufig als 1 oder $\top$ ab und spricht dann vom *größten Element* der Halbordnung $\langle P, \sqsubseteq\rangle$.

- Besitzt $\langle P, \sqsubseteq \rangle$ ein kleinstes Element 0 und ein größtes Element 1, so heißen zwei Elemente $x, y \in P$ *Komplemente* voneinander, wenn $x \sqcap y = 0$ und $x \sqcup y = 1$ gilt.

A.3.3 Definition. Besitzen in einer halbgeordneten Menge $\langle P, \sqsubseteq \rangle$

- je zwei Eemente x und y ein Infimum $x \sqcap y$ (Supremum $x \sqcup y$), so spricht man von einem $\sqcap$-($\sqcup$-)*Halbverband*;
- alle Teilmengen ein Infimum (Supremum), so spricht man von einem *vollständigen* $\sqcap$-($\sqcup$-)Halbverband.
- Ein *Verband* besitzt alle binären Infima und Suprema, ein *vollständiger Verband* sämtliche Infima und Suprema.
- Ein Verband heist *distributiv*, wenn immer gilt

$$(x \sqcap y) \sqcup z = (x \sqcup z) \sqcap (y \sqcup z) \quad \text{und} \quad (x \sqcup y) \sqcap z = (x \sqcap z) \sqcup (y \sqcap z)$$

- Hat $\langle P, \sqsubseteq \rangle$ ein kleinstes und ein größtes Element, und besitzt jedes $x \in P$ ein Komplement, so heißt $\langle P, \sqsubseteq \rangle$ *komplementär*.
- Ein komplementärer distributiver Verband heißt auch *Boole'sche Algebra*.

A.3.4 Satz. *Jeder vollständige $\sqcap$-Halbverband ist auch ein $\sqcap$-Halbverband und umgekehrt.*

Beweis. Man überzeugt sich sofort davon, dass das Supremum von $X \subseteq P$ mit dem Infimum der Menge $X^{\uparrow}$ der Oberen Schranken vn X übereinstimmt, und dual. □

A.4 Graphen und Bäume

Binäre Relationen $V \xrightarrow{E} V$ verdienen wegen ihrer besonderen Bedeutung einen speziellen Namen:

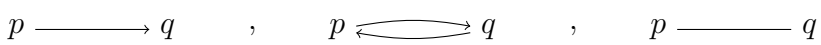
A.4.1 Definition. Unter einem *Graphen* verstehen wir ein Paar $\langle V, E \rangle$ bestehend aus einer Menge V von *Knoten* (english: vertices), und einer Menge $E \subseteq V \times V$ von *gerichteten Kanten* (english: directed edges).

Ein Graph heißt *ungerichtet*, wenn die Relation $E \subseteq V \times V$ symmetrisch ist, und *einfach* oder auch *schlicht*, wenn E zusätzlich *irreflexiv* ist, d.h., $\Delta_V \cap E = \emptyset$.

Graphen lassen sich leicht in der Ebene zeichnen, oder darstellen, mit Punkten als Knoten und ggf. gerichteten Kurven als Kanten. Solche Bilder sollten aber nicht mit den abstrakten Graphen identifiziert werden. Derselbe Graph erlaubt viele verschiedene Darstellungen, und bei Bildern mit derselben Knoten- und Kantenzahl kann man ab einer gewissen Größe nicht mehr leicht feststellen, ob sie denselben Graphen darstellen. Wir verzichten an dieser Stelle auf die mathematische Definition einer Graphen-Darstellungen, denn sie ist überraschend aufwendig.

Unsere Graphen sind grundsätzlich *gerichtet*, d.h., ihre Kanten $\langle u, v \rangle$ haben einen Anfangsknoten u und einen Endknoten v, was man in Darstellungen meist durch

Pfeilspitzen bei v anzeigen kann, oder auch durch eine Konvention, dass z.B. alle Kanten von oben nach unten zeigen. Bei ungerichteten Graphen ersetzt man zumeist zwei entgegengesetzt gerichtete Kurven mit Pfeilspitzen durch eine Kurve ohne Pfeilspitze.



Man kann jeden gerichteten Graphen in einen ungerichteten verwandeln, indem zu jeder gerichteten Kante $\langle u, v\rangle \in E$ die entgegengesetzte Kante $\langle v, u\rangle$ hinzufügt (falls diese schon zu E gehörte, wird sie dadurch *nicht* verdoppelt). Natürlich kann man i.A. den ursprünglichen gerichteten Graphen dann nicht mehr rekonstruieren.

Das mathematische Gebiet der „Graphentheorie“ befaßt sich weitgehend mit schlichten Graphen. Allgemeinere Graphen mit Mehrfachkanten existieren, spielen aber in dieser Vorlesung keine Rolle.

Oben hatten wir die bildliche Darstellung von Relationen bzw. (partiellen) Funktionen von A nach B in der AB-Ebene als „Graphen“ bezeichnet. Um daraus Graphen im obigen Sinne zu machen, kann man z.B. $A \cup B$ als Knotenmenge verwenden. In diesem Fall können auch Schleifen auftreten, wenn A und B einen nichtleeren Durchschnitt haben (insbesondere wenn $A = B$ gilt). Alternativ kann man natürlich auch die disjunkte Vereinigung $A+B$ als Knotenmenge verwenden; dann erhält man sog. bipartite Graphen:

A.4.2 Definition. Ein Graph $G = \langle V, E\rangle$ heißt *bipartit*, wenn die Knotenmenge K eine (disjunkte!) Zerlegung $V = V_0 + V_1$ zuläßt, so dass $E \subseteq V_0 \times V_1$ gilt. Mit anderen Worten, jede Kante startet in einen Knoten in V_0 und endet in einem Knoten in V_1.

Achtung: solch eine Zerlegung braucht nicht eindeutig zu sein!

Eine wichtige Klasse von „Graphen“, die wir gleich zu Beginn der Vorlesung kennenlernen, sind Bäume, genauer, gerichtete geordnete Bäume mit Wurzel.

A.4.3 Definition (naiv)**.** $G = \langle V, E\rangle$ sei ein gerichteter Graph.

(a) Zwei Kanten $\langle u, v\rangle, \langle p, q\rangle \in E$ heißen *komponierbar*, wenn $v = p$ gilt (diese Relation ist nicht symmetrisch!).

(b) Ein *gerichteter Pfad* in G ist eine Folge komponierbarer Kanten.

(c) Ein Knoten $w \in V$ heißt *Wurzel*, wenn jeder Knoten $u \in V$ durch einen gerichteten Pfad von w aus erreicht werden kann.

(d) G heißt *Baum*, wenn G eine Wurzel w hat, von der aus verschiedene gerichtete Pfade verschiedene Knoten erreichen. Knoten eines Baums ohne ausgehende Kanten werden auch als *Blätter* bezeichnet. Unter der *Tiefe* $\tau(G)$ vestehen wir die maximale Länge eines gerichteten Pfades von der Wurzel zu einem Blatt.

Wie Graphen haben auch Bäume verschiedene Darstellungen in der Ebene. Zunächst sollte klar sein, dass man einen Baum so darstellen kann, dass sich keine Kanten überschneiden, insofern handelt es sich um *planare* Graphen. Zusätzlich kann man nun fordern, dass die ausgehenden Kanten eines jeden Knoten von links nach rechts durchnumeriert sind, und es für die Nachfolgeknoten wichtig sein soll, an welcher

Position sie auftreten. In der Arithmetik kennt man dieses Phänomen z.B. von der Subtraktion, wo es auch auf die Reihenfolge der Argumente ankommt. So gelangt man zum Begriff des *geordneten Baums*. Ob zwei geordnete Bäume isomorph sind, d.h., bis auf die Bezeichnung der Knoten übereinstimmen, kann leichter entschieden werden, als die Isomorphie ungeordneter Bäume.

Man beachte, dass in einem Baum mit Wurzel w jeder Knoten außer w genau eine eingehende Kante hat. Im Hinblick auf das Zusammenfügen gegebener Bäume zu größeren Bäumen zeigt dies aber auch einen Nachteil der obigen Definition: es wäre schöner, wenn auch die Wurzel eine eingehende Kante hätte, und so werden wir es auch handhaben: jeder Baum hat eine eingehende und keine ausgehende Kante. Aber solch „offene“ Kanten ohne Startknoten passen nicht in die Standard-Definition eines Graphen als binräre Relation, insofern behelfen wir uns mit obiger naiven Definition von Bäumen.

A.5 Äquivalenzklassen und Partitionen

A.5.1 Definition.

(a) Für eine ÄR $E \subseteq Q \times Q$ ist $[q]_E := \{\, p \in Q \,:\, p\,E\,q \,\}$ die *Äquivalenzklasse* von q bzgl. E.

(b) Eine *Partition* $\mathcal{K} \subseteq P(Q)$ von Q besteht aus nichtleeren paarweise diskunkten Teilmengen von Q mit $\bigcup \mathcal{K} = Q$.

A.5.2 Satz. *Die Äquivalenzrelationen auf einer Menge Q entsprechen bijektiv den Partitionen der Menge Q.*

Beweis. Die Äquivalenzklassen einer ÄR partitionieren Q. Umgekehrt definiert man Elemente von Q als äquivalent, wenn sie zur selben Menge in der Partition gehören. □

A.5.3 Proposition. *Jede Abbildung $Q \xrightarrow{L} B$ induziert eine Äquivalenzrelation $\sim_L$ via*

$$p \sim_L q \quad \textit{genau dann wenn} \quad L(p) = L(q)$$

Das L-Bild $L[Q] = \{\, L(q) \,:\, q \in Q \,\}$ ist in canonischer Weise isomorph zur sogenannten Faktormenge $Q/\!\sim_L := \{\, [q]_{\sim_L} \,:\, q \in Q \,\}$ aller $\sim_L$§- Äquivalenzklassen. □

A.5.4 Beispiel.

- Die Preisfunktion zur Zeit t auf allen Artikeln eines Supermarkts: Toilettenpapier ist äquivalent zu Haarfestiger, ist äquivalent zu Dosen-Kohlrabi...
- Division durch n mit Rest: liefert die Funktion $\mathbb{N} \xrightarrow{\mathrm{mod}n} n = \{\, i \,:\, i < n \,\}$.

A.6 Tupel als Funktionen und die Natur des leeren Worts ε

A.6.1 Definition. B^A bezeichnet die Menge aller Funktionen von A nach B.

Dies imitiert die Schreibweise B^n für die Menge aller n-Tupel über B, also das n-fache cartesische Produkt der Menge B mit sich.
Interpretiert man die Zahl n als Menge ihrer Vorgänger, d.h.,

$$n = \{\, k \in \mathbb{N} : k < n \,\} \quad \text{und speziell} \quad 0 = \emptyset$$

so stimmen n-Tupel über B mit Funktionen $n \longrightarrow B$ überein.

A.6.2 Korollar. *Alle Inklusionsabbildungen $\emptyset \xrightarrow{\varepsilon} X$, X eine Menge, stimmen mit der einzigen Teilmenge $\emptyset \subseteq \emptyset \times X = \emptyset$ überein.* □

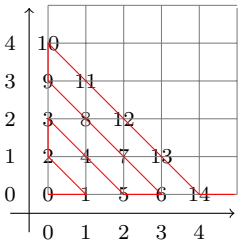
A.7 Abzählbarkeit

A.7.1 Definition. Eine Menge B heißt *abzählbar*, wenn es eine injektive Abbildung $B \longrightarrow \mathbb{N}$ gibt. Anderfalls heißt sie *überabzählbar*.

A.7.2 Satz. *Teilmengen, endliche cartesische Produkte und abzählbare Vereinigungen abzählbarer Mengen sind wieder abzählbar. Dagegen sind Potenzmengen abzählbar unendlicher Mengen überabzählbar.*

Beweis. **Teilmengen.** Für jede Teilmenge C einer abzählbaren Menge B ist die Inklusionsabbildung $C \xrightarrow{i} B$ injektiv, also auch ihre Komposition mit einer injektiven Abbildung $B \xrightarrow{j} \mathbb{N}$. □

Binäre cartesische Produkte. Es genügt, die Abzählbarkeit von $\mathbb{N} \times \mathbb{N}$ zu zeigen:



Abzählbare Vereinigungen.

$$\bigcup\{\, B_i : i \in \mathbb{N} \,\} = \sum\{\, B_i - \bigcup\{\, B_j : j < i \,\} : i \in \mathbb{N} \,\}$$

Also genügt, sich auf abzählbare disjunkte Vereinigungen zu beschränken. Aber $\mathbb{N} \times \mathbb{N}$ ist die disjunkte Vereinigung abzählbar vieler Kopien von $\mathbb{N}$ und abzählbar.

Potenzmengen. Es genügt zu zeigen, daß $P(\mathbb{N})$ überabzählbar ist. Wir nehmen an, $P(\mathbb{N}) \xrightarrow{g} \mathbb{N}$ ist injektiv. Aufgrund der Injektivität erfüllt $K := \{\, g(B) : B \subseteq \mathbb{N} \wedge g(B) \notin B \,\}$ die Bedingung $g(K) \in K$ genau dann wenn $g(K) \notin K$, Widerspruch. (Dieses Argument funktioniert für jede Menge anstelle von $\mathbb{N}$.) □

A.7.3 Korollar. *Für jede nichtleere abzählbare Menge X ist X^* abzählbar unendlich.*

Beweis. $X^* = \sum\{ X^n : n \in \mathbb{N} \}$ ist abzählbare disjunkte Vereinigung endlicher Produkte einer abzählbaren Menge. □

Literaturverzeichnis

[BS57] Friedrich Ludwig Bauer and Klaus Samelson. Verfahren zur automatischen Verarbeitung von kodierten Daten und Rechenmaschine zur Ausübung des Verfahrens. Patent submission, March 30 1957. Patent DE1094019, published December 1, 1960, granted August 12, 1971.

[BWW54] Arthur W. Burks, Don W. Warren, and Jesse B. Wright. An analysis of a logical machine using parenthesis-free notation. *MTAC*, 8(46):53–57, Apr 1954.

[Cal11] James Caldwell. Teaching natural deduction as a subversive activity. http://www.cs.uwyo.edu/ jlc/papers_chronological.html, June 2011. Presented at the Third International Congress on Tools for Teaching Logic, 1-4 June, 2011,Salamanca, Spain.

[D'A05] Marcello D'Agostino. Classical natural deduction. In Sergei N. Artëmov, Howard Barringer, Artur S. d'Avila Garcez, Luís C. Lamb, and John Woods, editors, *We Will Show Them! (1)*, pages 429–468. College Publications, 2005.

[Gen34] Gerhard Gentzen. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, 39(2):176–210, 1934.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen. II. *Mathematische Zeitschrift*, 39(3):405–431, 1935.

[Ham57] Charles Leonard Hamblin. An addressless coding scheme based on mathematical notation. Technical report, N.S.W. University of Technology, May 1957. (typescript).

[Hil1899] David Hilbert. *Grundlagen der Geometrie*. Teubner, 1999. Erstausgabe 1899.

[HR09] Michael Huth and Mark Ryan. *Logic in Computer Science*. Cambridge University Press, 2009.

[Jaś34] Stanisław Jaśkowski. On the rules of suppositions in formal logic. *Studia Logica*, 1:5–32, 1934.

[Kle52] Stephen Cole Kleene. *Introduction to Metamathematics*. Noth Holland, 1952.

Index

$A \times B$-Matrix
    binäre, 122
Σ-Term, 83
$\in$, 38, 119
$\sqcap$-Halbverband, 124
    vollständiger, 124
$\sqcup$-Halbverband, 124
    vollständiger, 124
n-Tupel
    geordnetes, 80
n-stellige Funktion, 80, 122
n-stelliges Prädikat, 80
Äquivalenz, 13, 35
Äquivalenzklasse, 126
Äquivalenzrelation, 19, 123
äquivalent, 98
überabzählbar, 127

Abschlußeigenschaften, 9
abstrahieren, 77
Absurdität, 8
    Elimination der, 56
    Introduktion der, 56
abzählbar, 127
adäquat, 24
Algebra
    Boole'sche, 5, 124
    Boolesche, 21
Algorithmus, 25
All-Quantor, 78
allgemeingültige Formel, 78
Alphabet, 4
    der Aussagenlogik, 8
    der Prädikatenlogik, 82
Annahmen, 46
antisymmetrische Reltion, 122
assoziativ, 22
atomare Formel, 77
atomare Aussage, 8
atomare Formel
    der Prädikatenlogik, 85
Aussage, 8
    atomare, 4, 8
Aussagenlogik
    klassische, 4
Auswahlfunktion, 120
Axiom, 47, 67, 78, 82, 90, 93
Axiomatisierbarkeit
    endliche, 93

Baum, 125
    geordneter, 126
Baum-Version, 8
Baumstruktur
    von Beweisen der ND, 48
Bedeutung
    von Formeln, 5
Belegung, 11
    passende, 13, 33, 108
Beweis, 37, 46
Beweis durch Widersprich, 56
Beweis durch Widerspruch, 21
Beweisbarkeit
    von Aussagen, 5
Beweistheorie, 34
bijektiv, 122
binäre $A \times B$-Matrix, 122
binäre Relation, 121
Bindungskonvention, 10
bipartiter Graph, 125
Blätter, 8, 82
    eines Baumes, 125
Boole'sche Algebra, 5, 124
Booth
    Connie, 117

cartesisches PRodukt, 80
cartesisches Produk
    binäres, 120

cartesisches Produkt
allgemeines, 120
charakteristische Funktion, 76, 122
Churchscher Satz, 116
Connie Booth, 117
Contraposition, 36
Beweisprinzip, 18, 56

Darstellung
eines Graphen, 124
Datentyp
abstrakter, 111
De Morganschen Regeln
Verallgemeinerung, 22
De Morganschen Regeln, 21
Deduktion
natürliche, 47, 48
definite clause, 68
Definitionsbereich, 121
disjunkte Vereinigung, 121
Disjunktion, 8
Elimination der, 54
Introduktion der, 54
distributiver Verband, 124
Distributivgesetz, 29
Distributivgesetze, 22
doppelte Negation
Elimination der, 56
duale Relation, 122
durchmischen, 104
Durchschnitt
allgemeiner, 120
binärer, 120

einfacher Graph, 124
einwertige Relation, 121
einwertige Relation, 80
Element
absorbieredes, 22
größtes, 123
kleinstes, 123
neutrales, 22
Element einer Menge, 119
Elimination der Disjunktion, 54
Elimination der Konjunktion, 49
Elimination der Absurdität, 56
Elimination der Negation, 56
Elimination der doppelten Negation, 56
erfüllbar, 17
erfüllbare Formel, 78
Externalisierung, 19, 35

Faktor-Algebra, 90
Faktormenge, 21
Fernwirkung
Regel mit, 50
Folge-Relation, 33
Form
bereinigte, 110
formal reflexiv, 90
formal symmetrisch, 90
formal transitiv, 90
Formel, 8
allgemeingültige, 78
atomare, 77
atomare der Prädikatenlogik, 85
erfüllbare, 78
formal korrekte, 5
geschlossene, 88
Größe, 11
herleitbare, 63
interpretierte, 92
nicht weiter zerlegbare, 5
quantifizierte, 77
zusammengesetzte, 5
Formel-Schema, 67
Formeln
äquivalente, 19
Frage-Klausel, 68
freie Term-Algebra, 77
Funktion, 121
n-stellige, 80, 122
bijektive, 122
charakteristische, 76, 122
injektive, 122
partielle, 121
surjektive, 122
Funktionssymbol, 81
n-stelliges, 92

gebundene Variable, 78
geordneter Baum, 126
geordnetes n-Tupel, 80
geordnetes Paar, 120

gerichtete Kante, 124
gerichteter Graph, 82, 124
gerichteter Pfad, 125
geschlossene Formel, 88
Gleichheit
    syntaktische, 20
Gleichung, 78, 93
Gleichungslogik, 4
Größe
    einer Formel, 11
größtes Element, 123
Graph
    bipartiter, 125
    einfacher, 124
    gerichteter, 82, 124
    planarer, 125
    schlichter, 124
    ungerichteter, 124
Graphen, 124
Grundterm, 109
Grundterme, 84

Halbordnung, 123
herleitbar, 63
Herleitbarkeit
    syntaktische, 46
Hilbert
    David, 66
Hilberts Programm, 47
Hilfssymbol, 4
Hilfssymbole
    der Prädikatenlogik, 83
Hornformel, 45

idempotent, 22
idempotente Relation, 122
Implikation, 13
    Introduktion der, 51
Induktion
    strukturelle, 5, 11, 27, 29
Infimum, 123
Infimum-Operation, 12
    verallgemeinerte, 12
Infix-Notation, 9
injektiv, 122
Instanziierungen, 92
Intepretation, 77
interpretiert, 77
interpretierte Signatur, 81
Introduktion der Disjunktion, 54
Introduktion der Implikation, 51
Introduktion der Negation, 56
Introduktion der Absurdität, 56
Introduktion der Konjunktion, 49
irreflexive, 124

Junktor, 8
    0-stelliger, 5
    logischer, 4

Kalkül
    korrektes, 46
    logisches, 5, 46
    vollständiges, 46
Kante
    gerichtete, 124
Kanten
    komponierbare, 125
Kasten-Prämisse, 51, 53
Kellerprinzips, 10
Klammern, 4
Klausel, 28, 37
kleinstes Element, 123
KNF, 37
    reine, 44
Knoten, 124
kommutativ, 22
Kompaktheitssatz, 11, 36
komplementär, 124
Komplemente, 124
komponierbare Kanten, 125
Komposition von Relationen, 121
Kongruenzrelation, 90
Konjunktion, 8
    Elimination der, 49
    Introduktion der, 49
Konkatenation, 38
Konsequenz
    logische, 33, 46, 108
Konstante, 80, 81
korrekt, 46
Korrektheit, 64
    eines Kalküls, 46

Länge

einer Infix-Darstellung, 11
leere Menge, 119
leeres Wort, 80
Lichtgeschwindigkeit, 31
lineare Ordnung, 123
Liste von Listen, 37
Literal, 28
negatives, 28
positives, 28
Logik
formale, 4
intuitionistische, 5, 47
logische Konsequenz, 33, 108
allgemeinere, 34

Matrix-Multiplikation, 122
Menge, 119
abzählbare, 8, 82
leere, 119
Mengenlehre
axiomatische, 47
Meta-Sprache, 35, 90
Meta-Variable, 9
Metasprache, 19, 33
Methode
axiomatische, 47
Model
Σ-, 92
Σ-, 93–98, 102, 111
Herbrandsche, 109
Herbrandsches, 110
Herbrandsches Σ-, 109
modulo $\equiv$, 20
Modus Ponens, 67
modus ponens, 50

NAND-Junktor, 24
natürliche Deduktion, 47
natürliche Zahlen, 119
ND, 47
Negation, 8
Elimination der, 56
Introduktion der, 56
nicht erfüllbar, 33, 108
nichtlogische Symbole, 82
Normalform
alternierende Pränex-, 103
disjunktive, 31
konjunktive, 25, 29
Negations-, 25
Pränex-, 103, 104, 106
Skolemsche, 105–107, 109, 110, 114, 115
Notation
polnische, 9
Präfix, 9
umgekehrt polnische, 9

Oder
exklusives, 13
Operation, 77
n-stellige, 92
Ordnung
lineare, 123

Paar
geordnetes, 120
partielle Funktion, 121
partielle ORdnung, 123
Partition, 126
Pfad
gerichteter, 125
planerer Graph, 125
Platzhalter, 77, 81
Potenzemenge, 119
Prädikat, 77
n-stelliges, 80
Prädikatenlogik, 4, 47
Prädikatssymbol, 81
n-stelliges, 92
Prämisse, 49
globale, 47
lokale, 47
Prämissen, 33
Präsentation
von Beweisen, 48
Produkt
cartesisches, 80
Proposition
der Aussagenlogik, 8

quantifizierte Formel, 77
Quantor, 77
Quasi-Ordnung, 123

reflexive Relation, 122

reflexive transitive Hülle, 122
Regel, 68
Regel mit Fernwirkung, 50
Regeln
    DeMorgansche, 21
rein, 44
reine KNF, 44
Rekursion
    strukturelle, 5, 12, 25
Relation
    n-stellige, 92
    antisymmetrische, 122
    binäre, 121
    duale, 122
    einwertige, 80, 121
    idempotente, 122
    irreflexive, 124
    reflexive, 122
    symmetrische, 122
    totale, 80, 121
    transitive, 122
Relationen
    Komposition von, 121
Resolutionsmethode, 37
    naive, 42
    praktikable, 44
Resolvente, 38
Resolventenbildung, 39
Resolventengraph, 38

Satz, 88
schlichter Graph, 124
Schlußregeln, 46
Schlussfolgerungen, 46
Schranke
    größte untere, 12
    obere, 123
    untere, 123
Schritte
    exponentiell viele, 45
scope, 63
selbstinvers, 21
Semantik, 5
    intendierte, 8
    zusammengesetzter Formeln, 5
Sequenz, 46
Sequenzenkalkül, 48
Sheffer-Stroke, 24
Signatur, 77, 81
    interpretierte, 81
Singleton, 119
Stack, 10
Stelligkeit, 8
Stetigkeitsformel, 89
Struktur
    Σ-, 81, 92
Substitution freier Variablen, 100
Summe, 121
Supremum, 123
surjektiv, 122
Symbole
    nichtlogische, 82
symmetrische Relation, 122
Syntax, 5
System
    logisches, 4

Tatsachen, 68
Tautologie, 13, 17, 31, 39, 40
    alternative Charakterisierung, 34
Teilmenge, 119
Term-Algebra
    freie, 77
Theorem, 63
Tiefe
    eines Baumes, 125
totale Relation, 80, 121
Träger, 92
Transformation
    syntaktische, 5
transitive Relation, 122
Tupel
    geordnetes, 80

ungerichteter Graph, 124
uninterpretiert, 77
Update, 94
Urknall, 31

Variable, 4
    der Aussagenlogik, 8
    der Prädikatenlogik, 82
    freie, 78
    gebundene, 78, 89

Verband, 124
    distributiver, 124
    vollständiger, 124
Vereinigung
    allgemeine, 120
    binäre, 120
    disjunkte, 121
vollständig, 46
Vollständigkeit, 66
    eines Kalküls, 46

Wahrheitstabelle, 13
Wahrheitstafel, 16
Wahrheitswert, 5
Wahrheitswerte, 80
Wert
    semantischer, 5
Widerspruch
    Beweis durch, 56
Wort
    leeres, 80
Wurzel
    eines Graphen, 125

Zahlen
    natürliche, 119
Zeitkomplexität, 45
Zugehörigkeit zu einer Menge, 119