Logik für Informatiker

Prof. Dr. W. Vogler

Wintersemester 2011/2012

Inhaltsverzeichnis

Einführung

Logik: formale Behandlung des exakten Denkens und Folgerns;
beantwortet die Frage „Was ist ein mathematischer Beweis?“

in der Informatik:

- Korrektheitsbeweise (Verifikation) von Programmen, Schaltkreisen etc.; diese benötigen Rechnerunterstützung; dazu: *formale* Angabe von Aussagen, speziell von
- Spezifikationen (Logik ist *ausdrucksstark* bzw. soll es sein)
- automatisches Beweisen
- logische Programmierung (PROLOG) (Beweis als Programmablauf)
- semantic web: Webseite um Inhaltsbeschreibung/-katalogisierung ergänzen – „Logik pur“

Literatur zum Thema Logik

H.-D. Ebbinghaus, J. Flum, W. Thomas: Einführung in die mathematische Logik. Spektrum Akademischer Verlag, 4. Auflage

M. Huth, M. Ryan: Logic in Computer Science. Modelling and reasoning about systems. Cambridge University Press

M. Kreuzer, S. Kühling: Logik für Informatiker. Pearson Studium 2006

U. Schöning: Logik für Informatiker. BI-Wissenschaftsverlag, Reihe Informatik Bd. 56

Logische Formeln (wie auch Programme) sind Zeichenketten bzw. *syntaktische Objekte*, z.B. $P(x)$. Durch *Interpretation* der Symbole (z.B. $P \widehat{=}$ „ist prim“) geben wir der Formel eine *Bedeutung (Semantik).*

Auch bei Programmen müssen wir nicht nur wissen, was syntaktisch korrekt ist, also vom Compiler akzeptiert wird, sondern auch die Bedeutung kennen; unter Bedeutung kann man die Ausführung bzw. das Ergebnis des Programms verstehen. Auch ein syntaktisch korrektes Programm kann das falsche Ergebnis liefern, also semantisch falsch sein.

Informatiker kommunizieren

- mit Computern, die nichts *verstehen*,
- über Spezifikationen, wobei Uneinigkeit/Unklarheit bzgl. Details besteht

Hier ist eine exakte, also sehr formale Notation wichtig – sogar mehr als in der Mathematik!!

Ein essentieller Begriff ist *Folgerung*. *Aus* A *folgt* B heißt: Ist A unter einer Interpretation wahr, so auch B (semantischer Begriff). Bsp.: aus $P(x)$ folgt $\neg\neg P(x)$

Wichtiges Ziel: B aus A durch Zeichenmanipulation (syntaktisch) *herleiten* (vgl. Chomsky-Grammatiken).

Ein *Kalkül* beschreibt die Herleitbarkeit durch logische *Axiome* und *Schlußregeln*. Ein Kalkül ist *korrekt*, wenn alles Herleitbare gefolgert werden kann (also wahr ist), und *vollständig*, wenn alles Folgerbare (Wahre) hergeleitet werden kann. Ein solcher Kalkül erlaubt die genaue Erfassung der Wahrheit (semantisch) durch Zeichenmanipulationen (syntaktisch)! Dies gilt nur im Rahmen des Formulierbaren, es ist also wichtig, daß die Logik *ausdrucksstark* ist.

Syntaktisches Herleiten bedeutet, dass die Axiome und Schlußregeln sich nur an der syntaktischen Struktur der Aussagen, nicht aber an ihrer Bedeutung (Semantik) orientieren sollen.

Beispiel: Gegeben seien die Annahmen
– „Arbeit macht reich.“
– „Armut macht schön.“
Welche der nachfolgenden Aussagen folgen logisch aus den folgenden Annahmen?
– „Arbeit macht häßlich.“
– „Arme arbeiten nicht.“
– „Schöne können nicht reich sein.“
– „Arbeitende können nicht schön sein.“
– „Durch Arbeit verliert man eine Chance, schön zu werden.“

Eine ganz analoge Struktur haben die Annahmen
– „Aufräumen schafft Ordnung.“
– „Unordnung ist kreativ.“
und die Aussagen
– „Aufräumen tötet die Kreativität.“
– „Unordentliche räumen nicht auf.“
– „Kreative können keine Ordnung erreichen.“
– „Aufräumer können nicht kreativ sein.“
– „Durch Aufräumen verliert man eine Chance, kreativ zu werden.“

Ein brauchbarer logischer Kalkül sollte also in der Lage sein, beide Aufgabengruppen einheitlich zu behandeln.

Später:
- Programmverifikation
- temporale Logik

1 Syntax und Semantik der Prädikatenlogik 1. Stufe

Dies ist eine Logik mit Elementen, Funktionen, Prädikaten und Quantifikation über Elemente ($\forall x\ P(x)$).

Prädikat: Relation wie $<$ (2-stellig) oder auch „ist prim“ (1-stellig).

In der 2. Stufe wird auch die Quantifikation über Prädikate behandelt. (Erwartet hätte man vielleicht Quantifikation über Mengen; dies ist ein Spezialfall: 1-stellige Prädikate entsprechen Mengen, s.u.)

Beispiel: Um in $\mathbb{N}$ (inkl. 0!) $\forall x\ P(x)$ zu zeigen, verwenden wir Induktion, denn es gilt das Induktionsprinzip: $\forall P\ \ P(0) \wedge (\forall x\ \ P(x) \rightarrow P(x+1)) \rightarrow \forall x P(x)$

1.1 Syntax

Gegeben seien Zeichen $x_0, x_1, \dots$ (Variablen); auch: $x, y, z \in \{x_0, x_1, \dots\} =: \mathcal{X}$.

Eine *Signatur* ist ein Paar $(\mathcal{F}, \mathcal{P})$, wobei $\mathcal{F}$ und $\mathcal{P}$ disjunkte, endliche oder abzählbar unendliche (entscheidbare) Zeichenmengen sind, mit $x_i, \forall, \dots \notin \mathcal{F} \cup \mathcal{P}$; implizit sei:

$\mathcal{F} = \mathcal{F}^0 \dot\cup \mathcal{F}^1 \dot\cup \mathcal{F}^2 \dot\cup \dots$

$\mathcal{P} = \mathcal{P}^0 \dot\cup \mathcal{P}^1 \dot\cup \mathcal{P}^2 \dot\cup \dots$

$f, g, h \in \mathcal{F}^n$: *n-stellige Funktionssymbole*

$c \in \mathcal{F}^0$: *Konstante*

$P, Q, R \in \mathcal{P}^n$: *n-stellige Prädikatensymbole* (auch: $p, q, r \in \mathcal{P}^0$)

Die Menge $Term_{\mathcal{F},\mathcal{P}}$ (kurz *Term*) der $\mathcal{F}$,$\mathcal{P}$-*Terme* (kurz: *Terme*) ist die kleinste Menge mit:

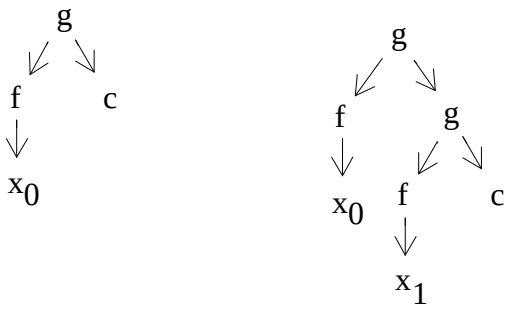
(T1) $x_0, x_1, \dots \in Term$

(T2) Wenn $f \in \mathcal{F}^n$ und $t_1, \dots, t_n \in Term$, dann $f(t_1, \dots, t_n) \in Term$; speziell: $c \in Term$

„kleinste Menge“ heißt: *Term* enthält nichts, was nicht aufgrund dieser Regeln in *Term* sein muß.

Ein Term ist also z.B. $g(f(x_0), c)$; wie üblich stehen die Funktionssymbole vor den Argumenten – *Präfix-Schreibweise*. In arithmetischen Termen haben wir zweistellige Funktionen $+$, $-$ etc., schreiben aber üblicherweise nicht $+(x_0, -(c, d))$ sondern $x_0 + (c - d)$. (*Infix-Schreibweise*; diese werden wir gelegentlich auch verwenden.) Graphische Darstellung von Termen: Syntaxbaum.

Die Auswertung von $g(f(x_0), c)$ kann man sich so vorstellen, dass man sich zunächst die Werte der Blätter (also für x_0 und c) beschafft; kennt man bereits die Werte für die Kinder eines inneren Knotens, so wendet man die entsprechende Funktion auf diese Werte an. Ist also f z.B. die Quadratfunktion und g die Addition und haben x_0 und c die Werte 4 und 1, so kann man bei f den Wert 16 und dann bei g den Wert 17 berechnen. Ist zudem 5 der Wert für x_1, so ergibt sich beim rechten Baum unabhängig für das linke f wieder der Wert 16 und für das rechte g der Wert 26, so dass man schließlich für die Wurzel 42 erhält.



Die Formel gewinnt man aus dem Baum zurück, indem man zunächst die Beschriftung der Wurzel hinschreibt und dann (falls Kinder vorhanden) rekursiv die Äste (die Bäume der Kinder) rekursiv aufschreibt und als geklammerte und durch Kommas getrennte Liste anfügt.

Für den rechten Baum erhalten wir also $g(.,.)$ wobei für die Punkte $f(x_0)$ bzw. $g(f(x_1), c)$ eingesetzt werden müssen, d.h. $g(f(x_0), g(f(x_1), c))$.

Anderes Beispiel für Terme (aus der Theoretischen Informatik): rationale (bzw. reguläre) Ausdrücke über einem Alphabet Σ; dabei ist $\mathcal{F} = \mathcal{F}^0 \dot\cup \mathcal{F}^1 \dot\cup \mathcal{F}^2$, $\mathcal{F}^0 = \Sigma \cup \{\lambda, \emptyset\}$, $\mathcal{F}^1 = \{^*\}$ und $\mathcal{F}^2 = \{+, \cdot\}$; z.B. $(a + b \cdot c)^*$.

Aus der Sicht der funktionalen Programmierung sind die x_i, f und c Konstruktoren.

Wegen oben „kleinste Menge“ sind nur nach den beiden Regeln aufgebaute Objekte Terme. Der Aufbau nach den Regeln wird durch eine *Herleitung* beschrieben, d.h. jeder Term hat eine Herleitung; z.B. sei $f \in \mathcal{F}^1, g \in \mathcal{F}^2$:

(1)	x_0 (d.h. $x_0 \in \mathit{Term}$)	T1 (Begründung)
(2)	$f(x_0)$	T2 (1)
(3)	c	T2
(4)	$g(f(x_0), c)$	T2 (2),(3)

Herleitung: bottom-up (erst Teile herleiten)

Bem.: T1 + T2 entsprechen „unendlicher kontextfreier Grammatik“

$$\text{Term} ::= x \mid c \mid f(\underbrace{\text{Term}, \ldots, \text{Term}}_{n})$$

mit $x \in \{x_0, x_1, \ldots\}$, $c \in \mathcal{F}^0$, $f \in \mathcal{F}^n$, $n \geq 1$.

Ableitung in Grammatik: top-down (im Ableitungsbaum) z.B.: Term $\Rightarrow g$(Term, Term) $\Rightarrow$ $g(f$(Term), Term) $\Rightarrow g(f(x_0)$, Term) $\Rightarrow g(f(x_0), c)$ □

Regeln wie T1, T2 (analog A0 – A5 unten) kann man in folgender Form schreiben:

$$\frac{t_1, \ldots, t_n}{t} \qquad \frac{\mathit{Prämissen}}{\mathit{Konklusion}}$$

Idee: Wenn $t_1, \ldots, t_n$ Terme (oder Formeln (s.u.), gültige Aussagen,...) sind, so auch t.
Speziell für n=0:

$$\frac{}{t} \qquad \mathit{Axiom}$$

Damit sind T1 und T2:

(T1') $\dfrac{}{x}$ $\quad x \in \{x_0, \dots\}$ (Nebenbedingung)

(T2') $\dfrac{t_1, \dots, t_n}{f(t_1, \dots, t_n)}$ $\quad f \in \mathcal{F}^n$ $\quad$ speziell: $\dfrac{}{c}$ $\quad c \in \mathcal{F}^0$

Eine *Herleitung* von t_0 für solche Regeln ist eine Folge $w_1, \dots, w_m$ von Zeichenketten mit $w_m \equiv t_0$ (syntaktisch gleich, d.h. gleiche Zeichenketten), so daß für jedes $i = 1, \dots, m$ eine Regel $\frac{t_1,\dots,t_n}{t}$ mit $w_i \equiv t$ existiert, für die $t_1, \dots, t_n$ unter den $w_1, \dots, w_{i-1}$ auftreten. Wir numerieren die w_i explizit und geben jeweils die angewandte Regel an zusammen mit den Zeilennummern der t_i.

Beispiel: siehe oben; $w_1 \equiv x_0$, $w_2 \equiv f(x_0)$, $w_3 \equiv c$, $w_4 \equiv g(f(x_0), c)$.
Z.B. für w_4 besagt Regel (T2) $\frac{f(x_0),\ c}{g(f(x_0),c)}$, und $f(x_0)$ und c sind w_2 und w_3. □

Man kann nun Definitionen/Beweise für herleitbare Objekte durch *Induktion* (über die Herleitungslänge m) durchführen. Ausführlich:

Objekt t hat mindestens eine Herleitung; sei also eine solche Herleitung der Länge m gegeben. Nach Induktion liegt Definition/Beweis für alle Objekte mit kürzerer Herleitung schon vor (Noethersche Ind.!).

Die zuletzt angewandte Regel zur Herleitung von t sei $\frac{t_1,\dots,t_n}{t}$ – hier muß man jeweils eine Fallunterscheidung nach den gerade behandelten Regeln machen. Die t_i erscheinen in der gegebenen Herleitung vor t; sie haben also kürzere Herleitungen, so daß nach Ind. Definition/Beweis für sie schon vorliegt. Gib basierend darauf Definition/Beweis für t.

Bei einem solchen im Prinzip immer wieder gleichen Vorgehen hat man für jede Regel einen Fall. Man gibt nur die Liste dieser Fälle und schreibt jeweils kurz:

Regel $\frac{t_1,\dots,t_n}{t}$: Definition/Beweis für t basierend auf Definition/Beweis für die t_i

Bei Termen (Formeln s.u.) sind die t_i Teilterme von t; man spricht von *Induktion über den Termaufbau* bzw. *struktureller Induktion*. Vgl. rationale Ausdrücke.

Beispiele:

1) Die Funktionszahl $\#_F t$ von t ist definiert durch:

   (T1) $\#_F x = 0$

   (T2) $\#_F f(t_1, \dots, t_n) =$
   (Spezialfall: $\#_F c = 1$)

   Die Fallunterscheidung nach der Form von t wird deutlicher, wenn man „$t \equiv x \in \mathcal{X}$“ statt (T1) und „$t \equiv f(t_1, \dots, t_n)$ mit $f \in \mathcal{F}^n$, $t_1, \dots, t_n \in \mathit{Term}$“ statt (T2) schreibt.

   Achtung: Definition ist eindeutig, weil Zerlegung in Teilterme (also die letzte Regel) eindeutig ist. Im Allgemeinen kann es sein, dass man ein Objekt auf verschiedene Weise herleiten kann, dass die letzte Regel einer Herleitung also nicht eindeutig ist. Dies entspricht einer Definition, bei der man eine Variante für $x \geq 1$ und eine zweite für $x \leq 1$ hat; was muss man dann beachten?

2) Jeder Term hat eine gerade Anzahl von Klammern.
Beweis durch strukturelle Induktion:

(T1) x hat 0 Klammern.

(T2) t_i habe k_i Klammern, nach Induktionsbeh. sei k_i gerade. Dann hat $f(t_1, \ldots, t_n)$ $2+k_1+k_2+\cdots+k_n$ Klammern, was als Summe gerader Zahlen gerade ist. (Speziell: c hat 0 Klammern) □

Bem.:

- Warum Noethersche Ind. (Schluß von „alle $n < m$“ auf m) statt Schluß von m auf $m+1$?

  Die kürzeren Herleitungen sind in der Regel nicht um 1 kürzer.

- Warum keine Verankerung? bzw.: Sind nicht die Axiome die Verankerung?

  Verankerung nicht nötig. Im Fall $m = 1$ ist die Regel ein Axiom; in diesem Fall machen wir gar keine Annahmen, d.h. wir behandeln die Verankerung implizit.

  Axiome können in einer Herleitung häufiger angewandt werden, also auch beim Schritt auf $m > 1$ auftreten (s.o. (3)); wir müssen sie also auf jeden Fall im Induktionsschritt berücksichtigen und sparen uns daher die Verankerung.

□

Syntax der Formeln

Atomare $\mathcal{F}$,$\mathcal{P}$-Formeln: kleinste Menge $At_{\mathcal{F},\mathcal{P}}$ (At) mit:

(A1) Sind $t_1, t_2 \in Term$, so ist $t_1 = t_2 \in At$.

$$\frac{}{t_1 = t_2} \quad t_1, t_2 \in Term$$

(A2) Sind $P \in \mathcal{P}^n$, $t_1, \ldots, t_n \in Term$, so $P(t_1, \ldots, t_n) \in At$; speziell: $p \in At$

Bem.: ”=“ ist ein spezielles 2-stelliges Prädikat. Man kann auch *Logik ohne Gleichheit* betrachten, dann fällt (A1) weg. □

$\mathcal{F},\mathcal{P}$-*Formeln*: kleinste Menge $For_{\mathcal{F},\mathcal{P}}$ (For) mit:

(A0) $At \subseteq For$

(A3) Ist $A \in For$, so auch $(\neg A) \in For$

(A4) Sind $A, B \in For$, so auch $(A \wedge B), (A \vee B), (A \to B), (A \leftrightarrow B) \in For$

(A5) Ist $A \in For$, so $(\forall x A), (\exists x A) \in For$

Wir führen die Formeln *true* und *false* ein als Abkürzungen für $p_0 \vee \neg p_0$ und $\neg true$ für ein festes $p_0 \in \mathcal{P}^0$. Zur verkürzenden Schreibweise lassen wir Klammern weg:

- Außenklammern
- Klammern bei aufeinanderfolgenden Negationen / Quantoren: $\neg\neg A$ bzw. $\forall x \exists y\, Q(x, y)$
- Linksklammerung bei $A \vee B \vee C, A \wedge B \wedge C$
- Bindungsstärke: $\neg \gg \wedge \gg \vee \gg \to \gg \leftrightarrow \gg \forall, \exists$

  Der Bindungsbereich von $\forall, \exists$ endet so spät wie möglich.

  Also steht $\forall x\ \neg B \to (\forall y\ \ Q(x, y) \to P)$ für ...

Beispiel: Eine Formel, die besagt, daß $R \in \mathcal{P}^2$ eine Äquivalenzrelation ist:
$(\forall x R(x, x)) \wedge (\forall x \forall y\ \ R(x, y) \to R(y, x)) \wedge (\forall x \forall y \forall z\ \ R(x, y) \wedge R(y, z) \to R(x, z))$

Bem.:

- Da die Menge Σ^* der Wörter über einer abzählbaren Menge Σ abzählbar ist, ist auch $For_{\mathcal{F},\mathcal{P}}$ abzählbar.
- Die Zerlegung von Formeln und Termen in Teilformeln und Teilterme, aus denen sie nach (T1)-(T2) und (A0)-(A5) aufgebaut sind, ist eindeutig (und berechenbar).

1.2 Semantik

Gilt $\forall x P(f(x), y)$? Dazu müssen wir natürlich wissen, von welchen x hier die Rede ist – natürlichen oder reellen Zahlen, Vektoren, Wörtern über einem festen Alphabet? Und für welche Eigenschaft dieser Objekte (allgemeiner: welche Relation) steht P, welche Funktion repräsentiert das Symbol f?

Diese in einem Kontext meist festen Informationen gibt eine *Interpretation* I; z.B. könnten die reellen Zahlen die Grundmenge bilden, P ist $\geq$ und f ist die Quadratfunktion. (Unter "=" wird immer die Gleichheit verstanden.)

Die Variablen ändern ihren Wert häufiger; diese Werte liefert eine *Belegung* β, so etwas wie eine Wertetabelle. Ist der Wert von y in unserem Beispiel -1, so gilt die Formel; wir schreiben $I, \beta \models \forall x P(f(x), y)$.

Eine *Interpretation* I einer Signatur $(\mathcal{F}, \mathcal{P})$ stützt sich auf eine nichtleere Menge D (oder D_I) (*Grundbereich*, domain) und ordnet jedem $f \in \mathcal{F}^n$ eine Funktion $f^I : D^n \to D$ (speziell: $c^I \in D$) und jedem $P \in \mathcal{P}^n$ ein Prädikat $P^I : D^n \to \{T, F\}$ (Spezialfall: $p^I \in \{T, F\}$) zu.

Analogie: f ist ein Methoden-Bezeichner, P ein Bezeichner für eine Boolesche Methode; I gibt die Methoden-Deklaration an.

Bem.:

- f, P, p sind Zeichen, f^I, P^I, p^I sind Funktionen/Prädikate.
- Hier haben Werte nur eine Sorte D. Man kann auch mehrere Sorten $(D_1, \ldots, D_n)$ und entsprechend getypte Variablen, Funktionen und Prädikate verwenden.
- Das Prädikat P^I entspricht der Relation $\{(d_1, \ldots, d_n) \mid P^I(d_1, \ldots, d_n)\}$. (Man beachte die Schreibweise.)

  (1-stellige Prädikate entsprechen Mengen: $P^I : D \to \{T, F\}$ entspricht der Menge $\{d \in D \mid P^I(d)\}$; vgl. charakteristische Funktion einer Menge, Darstellung einer Menge als Bitvektor bzw. „Eigenschaft aufzählbarer Sprachen“ .)

□

Die Auswertung von Termen und Formeln wird nun induktiv definiert und folgt dabei den induktiven Definitionen (T1), ..., (A5) s.o.

Eine *Belegung* β zu I ist eine Funktion $\beta : \{x_0, x_1, \ldots\} \to D_I$. Analogie: Der Speicherzustand gibt die aktuellen Werte der Programm-Variablen an, die sich offenbar ändern können. Ändert sich der Wert von x auf d, beschreibt man den neuen „Speicherzustand“ durch β^d_x:

Ist $d \in D_I$, so ist β^d_x die Belegung mit

$$\beta^d_x(y) = \begin{cases} d & \text{für } y \equiv x \\ \beta(y) & \text{für } y \not\equiv x \end{cases}$$

Die Funktionen β und β^d_x unterscheiden sich also (allenfalls) nur an der Stelle x.

Die *Auswertung eines Terms* t unter I und β ist $t_{I,\beta}$ mit

(vgl. T1) $x_{I,\beta} :=$

(vgl. T2) $f(t_1, \ldots, t_n)_{I,\beta} :=$

(Spezialfall: $c_{I,\beta} := \ldots$

Diese Definition entspricht genau der bottom-up Auswertung anhand des Syntaxbaums: Für die Blätter ermittelt man die Werte gemäß (T1) für Variablen und (T2) für Konstanten; für einen f-Knoten wertet man die Äste aus und wendet f in seiner durch I gegebenen Bedeutung an.

Beispiel: $D = \mathbb{R}$, $g^I : (x, y) \to x - y$, $f^I : x \to x^2$, $\beta(x) = 3$.
Dann ist $g(f(x), x)_{I,\beta} = g^I(f(x)_{I,\beta}, x_{I,\beta}) = g^I(f^I(3), 3) = g^I(9, 3) = 6$.

Bem.: Der Mathematiker schreibt einfach $x := 5$ statt $x_{I,\beta} := 5$. Wir untersuchen Terme/Formeln (Zeichenketten) wie x, unterscheiden also x und seinen Wert deutlich. □

Wir definieren nun $I, \beta \models B$, d.h. B *gilt bei* (I, β) bzw. (I, β) *erfüllt* B, durch:

(vgl. A1) $I, \beta \models (t_1 = t_2) :\Leftrightarrow (t_1)_{I,\beta} = (t_2)_{I,\beta}$ ($:\Leftrightarrow$ heißt per Definition äquivalent)

Bemerkung: $\Leftrightarrow$ ist *unsere* Sprache, mit der wir über Formeln sprechen (*Metasprache*); $\leftrightarrow$ gehört zur Sprache der Formeln (*Objektsprache*). Analog: Im Deutschen über Englisch reden.

Beispiel: $D = \mathbb{Z}$, $f^I = +$, $\beta(x) = 1$ und $\beta(y) = 0$; weiter sei $f^{I'}$ die Multiplikation.

Dann ist $f(x, y)_{I,\beta} = 1$ und $f(x, y)_{I',\beta} = 0$; demnach gilt $I, \beta \models f(x, y) = x$, nicht aber $I', \beta \models f(x, y) = x$.

(vgl. A2) $I, \beta \models P(t_1, \ldots, t_n) :\Leftrightarrow$

(vgl. A3) $I, \beta \models \neg A :\Leftrightarrow$ nicht $I, \beta \models A \Leftrightarrow: I, \beta \not\models A$

(vgl. A4) $I, \beta \models A \wedge B \quad :\Leftrightarrow \quad I, \beta \models A$ und $I, \beta \models B$

$I, \beta \models A \vee B \quad :\Leftrightarrow \quad I, \beta \models A$ oder $I, \beta \models B$

$I, \beta \models A \rightarrow B \quad :\Leftrightarrow \quad I, \beta \not\models A$ oder $I, \beta \models B$

$I, \beta \models A \leftrightarrow B \quad :\Leftrightarrow \quad (I, \beta \not\models A$ und $I, \beta \not\models B)$ oder $(I, \beta \models A$ und $I, \beta \models B)$

Beispiel: (Forts.)

Zusätzlich sei $P^I =$ „ist negativ“ und $c^I = -1$. Dann gilt:

$I, \beta \models f(x, c) = y \rightarrow P(f(c, c)) \quad \Leftrightarrow \quad I, \beta \not\models f(x, c) = y$ oder $I, \beta \models P(f(c, c))$
$\Leftrightarrow$ nicht $f^I(\beta(x), c^I) = \beta(y)$ oder $P^I(f^I(c^I, c^I))$ (wie oben)
$\Leftrightarrow$ nicht $1 + (-1) = 0$ oder $(-1) + (-1)$ ist negativ.

Also gilt die Formel unter I und β.

($\forall x A$ heißt: $A(x)$ gilt für alle x-Werte; $I, \beta \models A$ heißt: $A(x)$ gilt für $\beta(x)$; also:)

(vgl. A5) $I, \beta \models \forall x A \quad :\Leftrightarrow \quad$ für alle $d \in D$ gilt $I, \beta^d_x \models A$

$I, \beta \models \exists x A \quad :\Leftrightarrow$

Definition von $\neg A, A \wedge B$, etc. tabellarisch; dabei steht A, $A \wedge B$ etc. für $I, \beta \models A$, $I, \beta \models A \wedge B$ etc.

A	B	$\neg A$	$A \wedge B$	$A \vee B$	A xor B	$A \to B$	$A \leftrightarrow B$
T	T	F	T	T	F	T	T
T	F	F	F	T	T	F	F
F	T	T	F	T	T	T	F
F	F	T	F	F	F	T	T

$\vee$ ist das inklusive, xor das exklusive Oder.

Bem.: $I, \beta \models A$ (bzw. $I, \beta \models A \wedge B$ etc.) ist *keine* $\mathcal{F}, \mathcal{P}$-Formel; daher macht □

Beispiele: 1. p: Augsburg liegt in Thailand.

q: Augsburg liegt in Asien.

r: Augsburg liegt in Afrika.

$p \to r$ *gilt* mit der Realität als Interpretation — $p \to q$ natürlich auch. Wir benutzen $\to$ *extensional*, d.h. nur die Wahrheitswerte sind interessant.

Intentional ist $p \to r$ („Wenn Augsburg in Thailand *läge*, dann läge es in Afrika.“) wohl falsch.

$p \mathrel{\widehat{=}}$ Satz, der wahr oder falsch ist. Also nicht:

- Hau ab !
- Nachts ist es kälter als draußen.
- Der gegenwärtige König von Frankreich hat eine Glatze.
- Diser Saz enthält drei Fehler. (falsch, da nur 2 Schreibfehler; wahr, da ein Sinnfehler dazukommt)

Der letzte Satz zeigt, daß Sätze, die sich auf sich selbst beziehen, problematisch sind; weitere Beispiele:

- Dieser Satz ist falsch.
- Ein Kreter sagt: „Alle Kreter lügen.“

2.

a) Max sagt: „Paul lügt.“

b) Paul sagt: „Max lügt.“

Sei p „Paul lügt“und m „Max lügt“. Dann besagen a) und b) $(\neg m \leftrightarrow p)$ und $(\neg p \leftrightarrow m)$.

3. $D := \mathbb{N}, \qquad P^I :=<$,
$I, \beta \models \exists y : P(y, x) \qquad (\Leftrightarrow$ es gibt $d \in \mathbb{N}$ mit $I, \beta_y^d \models P(y, x)$, also mit $d < \beta(x))$

- ist wahr für $\beta(x) \in \{1, \dots\}$; wähle $d := \beta(x) - 1$ oder $d := 0$.
- ist falsch für $\beta(x) = 0$. (Offenbar ist $\beta(y)$ egal!)

Also ist $I, \beta \models \forall x \exists y : P(y, x)$ falsch.
Beobachtung: Die „Struktur" I ist wichtiger als β. □

Proposition 1.1 *(Hier wie auch sonst bedeutet das folgende: „Für alle Interpretationen I, Belegungen β und Formeln A, B gilt. . . ")*

i) $I, \beta \models A \;\Leftrightarrow\; I, \beta \not\models \neg A \;\Leftrightarrow\; I, \beta \models \neg\neg A$

ii) $I, \beta \models A \wedge B \;\Leftrightarrow\; I, \beta \models \neg(A \to \neg B)$

iii) $I, \beta \models A \vee B \;\Leftrightarrow\; I, \beta \models \neg A \to B$

iv) $I, \beta \models A \leftrightarrow B \;\Leftrightarrow\; I, \beta \models (A \to B) \wedge (B \to A)$

v) $I, \beta \models \exists x A \;\Leftrightarrow\; I, \beta \models \neg\forall x \neg A$

Beweis:

i) $I, \beta \not\models \neg A$ bzw. $I, \beta \models \neg\neg A$
$\Leftrightarrow$ nicht $I, \beta \models \neg A$ (Definition $\not\models$ und $\neg$)
$\Leftrightarrow$ nicht nicht $I, \beta \models A$ (Definition $\neg$)
$\Leftrightarrow I, \beta \models A$ (Metalogik)

iii) $I, \beta \models A \vee B$
$\Leftrightarrow I, \beta \models A$ oder $I, \beta \models B$ (Definition $\vee$)
$\Leftrightarrow I, \beta \not\models \neg A$ oder $I, \beta \models B$ (Teil i))
$\Leftrightarrow I, \beta \models \neg A \to B$ (Definition $\to$)

v) $I, \beta \models \exists x A$
$\Leftrightarrow$ es gibt $d \in D$ mit $I, \beta_x^d \models A$ (Definition $\exists x$)
$\Leftrightarrow$ nicht für alle $d \in D$ gilt nicht $I, \beta_x^d \models A$ (Metalogik)
$\Leftrightarrow$ nicht für alle $d \in D : I, \beta_x^d \models \neg A$ (Definition $\neg$)
$\Leftrightarrow I, \beta \models \neg\forall x \neg A$ (Definition $\forall x, \neg$)

□

Nur üblicher math. Beweis – wir streben größere Präzision an; dazu müssen wir unser formales Vorgehen mit „gesundem (math.) Menschenverstand" absichern.

1.3 Modelle und Folgerbarkeit

Definition 1.2 (*Modell*)
Seien $A \in For$, $M \subseteq For$ und I eine Interpretation. Wir sagen A bzw. M *gilt* in I, I *erfüllt* A bzw. M und I ist ein *Modell* von A bzw. M, wenn $I \models A$ bzw. $I \models M$ gilt gemäß:

- $I \models A :\Leftrightarrow$ für alle Belegungen β gilt $I, \beta \models A$
- $I \models M :\Leftrightarrow$ für alle $A \in M$ gilt $I \models A$
  (Analog: $I, \beta \models M :\Leftrightarrow$ für alle $A \in M$ gilt $I, \beta \models A$

□

Bem.: $I \models \emptyset$ gilt immer □

Beispiele: 3') $D := \mathbb{N}$, $P^I :=<$ (wie vor Proposition 1.1) Q^I: „ist echter Teiler von“ R^I: „ist gerade“

- I ist kein Modell von $\forall x \exists y \;\; P(y,x)$ bzw. $\exists y \;\; P(y,x)$ (vgl. Beispiel 3)
- $I \models P(x,y) \vee P(y,x) \vee x = y$ (Die Formel ist gewissermaßen all-quantifiziert.)
- ACHTUNG: $\underbrace{I \models P(x,y),\ I \models P(y,x) \text{ bzw. } I \models x = y}_{\text{...sind alles falsche Aussagen!}}$
  Die Definition von $I, \beta \models A \vee B$ läßt sich also nicht auf $I \models A \vee B$ übertragen.
  Analog: $\forall x \; R(x) \vee Odd(x)$ ist nicht dasselbe wie $(\forall x \; R(x)) \vee (\forall x \; Odd(x))$.
- Jede gerade Zahl ist kleiner als x. $\forall R(y) \, P(y,x)$?
- Jeder echte Teiler von x ist kleiner als x.
- x ist der größte echte Teiler von 12 (zusätzliche Konstante).
- Es gibt unendlich viele gerade Zahlen.
  Trick:

4) $D := \mathbb{N} \setminus \{0\}$, $f^I := +$, $P^I :=$ „ist prim“
Die sogenannte ... besagt ungefähr, daß I ein Modell ist von: $(\exists x_1 \;\; f(x_1, x_1) = x_0) \rightarrow \exists x_2 \exists x_3 \;\; P(x_2) \wedge P(x_3) \wedge f(x_2, x_3) = x_0$

Definition 1.3 (*Folgerbarkeit und Äquivalenz*)
Seien $A, B \in For$ und $M \subseteq For$; A *folgt aus* M (in Zeichen: $M \models A$), wenn für alle Interpretationen I und Belegungen β gilt: $I, \beta \models M \Rightarrow I, \beta \models A$.
Konvention: Wir schreiben auch $A_1, \ldots, A_n \models A$ für $\{A_1, \ldots, A_n\} \models A$.
A und B sind *logisch äquivalent*, $A \models\!\!\models B$, wenn $A \models B$ und $B \models A$. □

Beispiel: Für eine 2-stellige Funktion $\circ$ in Infix-Schreibweise und eine Konstante e seien

$$M_{gr} = \{\forall x \forall y \forall z \;\; (x \circ y) \circ z = x \circ (y \circ z), \;\; \forall x \;\; x \circ e = x, \;\; \forall x \exists y \;\; x \circ y = e\}$$

$$A :\equiv \forall x \exists y \;\; y \circ x = e$$

$M_{gr} \models A$ bedeutet: A ist ein *Satz der ...*, d.h. in jeder Interpretation / jedem Modell von M_{gr} ... gilt A, d.h. jedes Element .... Belegungen sind hier egal, siehe unten.

Bem.: Mathematik handelt also von Aussagen $M \models A$. □

Nach Proposition 1.1 sind $A \wedge B$ und $\neg(A \to \neg B)$, $A \vee B$ und $\neg A \to B$, $A \leftrightarrow B$ und $(A \to B) \wedge (B \to A)$, sowie $\exists x A$ und $\neg\forall x \neg A$ logisch äquivalent. Wir können jede Formel (auch als Teilformel einer größeren Formel – ohne Beweis; vgl. Satz 2.3) durch eine logisch äquivalente ersetzen; also gilt z.B.

$$\begin{array}{lll} A \vee B \wedge \neg C & =\!|\!\models & \neg A \to B \wedge \neg C \\ & =\!|\!\models & \neg A \to \neg(B \to \neg\neg C) \quad \text{(Teilformel ersetzt)} \\ & =\!|\!\models & \neg A \to \neg(B \to C) \end{array}$$

Damit können wir uns also bei Bedarf auf $\neg, \to$ und $\forall x$ beschränken und $\wedge, \vee, \leftrightarrow, \exists x$ als Abkürzungen ($A \wedge B$ für $\neg(A \to \neg B)$ etc.) auffassen, die wir bei Bedarf zur besseren Lesbarkeit benützen.

Damit haben wir weniger Formeln und weniger Fallunterscheidungen, z.B. beim Beweis von Satz 2.3 oder im Hilbert-Kalkül in Abschnitt 2.2.

Bem.: Zum Vergleich der verschiedenen Äquivalenzsymbole:

- $\leftrightarrow$ steht in Formeln
- $=\!|\!\models$ steht zwischen Formeln; Meta-Sprache
- $\Leftrightarrow$ steht zwischen (meta-sprachlichen) Aussagen, die wahr oder falsch sind; $A \vee B \Leftrightarrow \neg A \to B$ ist syntaktisch falsch, denn $A \vee B$ ist nicht wahr oder falsch – $I, \beta \models A \vee B$ schon.

□

Satz 1.4 *i) Für alle $n \geq 1$ gilt: $I \models \{A_1, \ldots, A_n\} \Leftrightarrow I \models A_1 \wedge \cdots \wedge A_n$*

ii) $M \models A \to B \Leftrightarrow M \cup \{A\} \models B$

Beweis: i) per Induktion über n

ii) "⇒" Gelte $I, \beta \models M \cup \{A\}$ (– z.z. ist $I, \beta \models B$). Dann gilt $I, \beta \models M$ und damit nach Voraussetzung $I, \beta \models A \to B$. Also gilt $I, \beta \not\models A$ oder $I, \beta \models B$. Außerdem gilt $I, \beta \models A$ wegen $I, \beta \models M \cup \{A\}$; zusammen $I, \beta \models B$.

"⇐" Gelte $I, \beta \models M$ (– z.z. ist $I, \beta \models A \to B$). Entweder $I, \beta \models A$ ist falsch (fertig!) oder $I, \beta \models M \cup \{A\}$ und nach Voraussetzung $I, \beta \models B$; fertig nach der Definition von $\to$. □

Definition 1.5 $A \in$ *For* ist *(allgemein)gültig* (eine *Tautologie*), in Zeichen: $\models A$, wenn für alle Interpretationen I gilt $I \models A$. (D.h. für alle I, β: $I, \beta \models A$.) Analog definieren wir $\models M$ für $M \subseteq$ *For*.

A bzw. $M \subseteq$ *For* sind *erfüllbar*, wenn es ein I und ein β gibt mit $I, \beta \models A$ bzw. $I, \beta \models M$, andernfalls *unerfüllbar*. □

$\models A$ bzw. $M \models A$ sind die Wahrheiten, die uns interessieren. Vgl. oben: $M_{gr} \models A$ heißt, daß A aus den Formeln in M_{gr} folgt – unabhängig von einem konkreten I bzw. I, β, d.h. der konkreten Gruppe; analog für Körper, Vektorräume etc. M könnte auch Gesetze für Datenstrukturen enthalten, z.B. $\forall x \forall s \ \mathit{first}(\mathit{prefix}(x, s)) = x$.

Proposition 1.6 *Sei $A \in For$.*

i) A gültig $\Rightarrow$ A erfüllbar, d.h.
A unerfüllbar $\Rightarrow$ A nicht gültig.

ii) A gültig ($\models A$) $\Leftrightarrow \emptyset \models A$

Beweis: ii)

$\emptyset \models A \ \Leftrightarrow$
$\Leftrightarrow$ □
$\Leftrightarrow$ A gültig.

Satz 1.7 *Sei $A \in For$ und $M \subseteq For$. Dann gilt $M \models A$ genau dann, wenn $M \cup \{\neg A\}$ unerfüllbar ist; speziell ist A gültig genau dann, wenn $\neg A$ unerfüllbar ist.*

Beweis: für alle $I, \beta : I, \beta \models M \Rightarrow I, \beta \models A$
genau dann wenn
für alle $I, \beta : I, \beta \models M \Rightarrow I, \beta \not\models \neg A$ (Proposition 1.1 i))
genau dann wenn
$M \cup \{\neg A\}$ ist unerfüllbar. □

Algorithmische Idee: Versuche, erfüllende I, β für $\neg A$ zu konstruieren; gelingt dies, ist A nicht gültig (und wir haben I, β als Beleg dafür), sonst ist es gültig (wenn die Versuche geeignet sind).

1.4 Teil-Interpretationen

Seien I, J zwei Interpretationen zur selben Signatur $(\mathcal{F}, \mathcal{P})$, so dass $D_I \subseteq D_J$ und für alle $f \in \mathcal{F}^n, P \in \mathcal{P}^n$ und $d_1, \ldots, d_n \in D_I$ gilt: $P^I(d_1, \ldots, d_n) \Leftrightarrow P^J(d_1, \ldots, d_n)$ und $f^I(d_1, \ldots, d_n) = f^J(d_1, \ldots, d_n)$ (dazu muss dieser Wert natürlich in D^I liegen). Bsp.: f^I ist die Addition in $\mathbb{N}$, f^J die Addition in $\mathbb{Q}$.

Dann nennen wir I eine *Teil-Interpretation* von J. Sind I und J Modelle zu einer Formelmenge, so nennt man I auch ein *Teil-* oder *Untermodell*.

Dann ist jede Belegung β zu I auch eine zu J, da die Werte von β ja in $D_I \subseteq D_J$ liegen. Man sieht recht leicht: Für jede Belegung β zu I, jeden Term t und jede *quantorenfreie* Formel A gilt: $t_{I,\beta} = t_{J,\beta}$ und damit $I, \beta \models A \Leftrightarrow J, \beta \models A$.

Man kann den Eindruck haben, dass Formeln in I stimmen müssen, wenn sie sogar in dem größeren J stimmen. Dies ist nicht immer wahr, trifft aber für sog. universelle Formeln zu.

Definition 1.8 *Universelle Formeln* sind Formeln, die sich nach folgenden Regeln herleiten lassen:

$i) \ \dfrac{}{A}$ A quantorenfrei $\qquad ii) \ \dfrac{A, B}{A \wedge B} \qquad iii) \ \dfrac{A, B}{A \vee B} \qquad iv) \ \dfrac{A}{\forall x A}$

□

Beispiel: $\forall y Q(y) \wedge \forall x \neg P(x, y)$

Die Formeln in M_{gr} sind nicht alle universell, warum?

Es gibt eine äquivalente Axiomenmenge M'_{gr}, in der alle Formeln universell sind: Ein Modell J dazu ist eine Gruppe, eine Teil-Interpretation ist eine Untergruppe.

Proposition 1.9 *Sei I eine Teil-Interpretation zu J, β eine Belegung zu I und A eine universelle Formel. Dann gilt:*

$$J, \beta \models A \implies I, \beta \models A$$

Beweis: Induktion über die Herleitungslänge:

i) Feststellung s.o.

ii) $J, \beta \models A \wedge B$ bedeutet nach Semantik-Definition $J, \beta \models A$ und $J, \beta \models B$; nach Ind. gilt also $I, \beta \models A$ und $I, \beta \models B$; dies bedeutet $I, \beta \models A \wedge B$

iii) analog

iv) Sei $J, \beta \models \forall x A$.
$\Rightarrow$ für alle $d \in D_J$ gilt $J, \beta^d_x \models A$

$\Rightarrow$ für alle $d \in D_I$ gilt $I, \beta^d_x \models A$ (n. Ind)
$\Rightarrow I, \beta \models \forall x A$ $\square$

2 Aussagenlogik

2.1 Allgemeines, Wahrheitstafeln

In diesem Kapitel sei $\mathcal{P} = \mathcal{P}^0 = \{p_0, p_1, p_2, ...\}$ [1] und "=" ist verboten. Es treten in Formeln also keine Terme auf, demnach auch keine Variablen. $\mathcal{P}$ *ist* demnach die Menge der atomaren Formeln, kurz *Atome*. Wie bei nullstelligen Funktionszeichen schreiben wir bei nullstelligen Prädikatzeichen keine Klammern beim „Aufruf".

Belegungen sind irrelevant, d.h. $I \models A$ genau dann wenn $I, \beta \models A$.
O.E. gibt es keine Variablen, Belegungen, $\forall x$ – und $\mathcal{F} = \emptyset$; es gibt also nur: $\mathcal{P}^0, \rightarrow, \neg$. Interpretation haben demnach die Form $I : \mathcal{P} \rightarrow \{T, F\}$. (Diese werden auch oft Belegungen genannt, wenn man nur Aussagenlogik behandelt – wir werden das also nicht tun.)

Wir können alle Interpretationen (bzw. ihren relevanten Teil) mittels *Wahrheitstafel* durchprobieren; genau dann, wenn alle Einträge einer Formel T sind, ist die Formel gültig. Gültigkeit ist also *entscheidbar*.

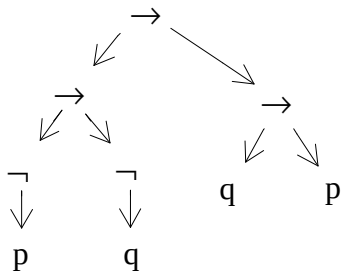
Beispiel 2.1

$$\begin{array}{lcll} A & \equiv & p \rightarrow (q \rightarrow p) & \\ B & \equiv & (\neg p \rightarrow \neg q) \rightarrow (q \rightarrow p) & \text{(Kontraposition)} \\ C & \equiv & (\neg p \rightarrow \neg q) \rightarrow (p \rightarrow q) & \end{array}$$

p	q	$q \rightarrow p$	A	$\neg p \rightarrow \neg q$	B	$p \rightarrow q$	C
T	T	T	T	T	T	T	T
T	F	T	T	T	T	F	F
F	T	F	T	F	T	T	T
F	F	T	T	T	T	T	T

A und B sind gültig, C nicht. □

Wie in der Wahrheitstafel angedeutet werten wir Formeln wieder bottom-up im Syntaxbaum aus; wir betrachten also Formeln wie Terme, wobei die Atome Variablen entsprechen (deren Bedeutung allerdings durch die Interpretation festgelegt wird). Für B haben die Äste in der dritten Zeile oben beide den Wert F, so dass sich an der Wurzel T ergibt.



Eine Formel ist erfüllbar genau dann, wenn mindestens ein Eintrag in der Wahrheitstafel T ist. Also ist Erfüllbarkeit entscheidbar.

Beispiele: 1. C in Beispiel 2.1 ist erfüllbar.

2. Bei vielen Atomen (n) ist dies Verfahren sehr aufwendig (2^n Zeilen); evtl. "gezielte Suche"/geeignete Fallunterscheidung; z.B.:

[1] Menge der nullstelligen Prädikate, s. Kapitel 1

Beim Antrag auf das Vordiplom stehen ein weißes, ein rotes und ein blaues Formular zur Wahl. Die Prüfungsordnung enthält folgende Vorschriften:

a) $w \to \neg b$ (In Worten: Wer ein weißes Formular einreicht, darf kein blaues einreichen.)

b) $w \vee r$

c) $r \to b \wedge w$

Angenommen $r = T$, dann

Also $r = F$, damit c) ✓

wegen

wegen

Ergibt die einzige erfüllende Interpretation. Alternativ: Wahrheitstafel mit 8 Zeilen.

In der Aussagenlogik heißt "$=\!|\!\models$" einfach „haben dieselben Modelle"; auch das können wir mit Wahrheitstafeln durchprobieren.

Beispiel 2.2 $p \to (q \to r) =\!|\!\models (p \wedge q) \to r$

p	q	r	$q \to r$	links	$p \wedge q$	rechts
.	.	.				
T	T	F	F	F	T	F
T	F	T	T	T	F	T
.	.	.				
.	.	.				
F	T	F	F	T	F	T
.	.	.				

Da die Spalten zu links und rechts gleich sind, folgt die Äquivalenz. □

$A \equiv p \to (q \to p)$ aus Beispiel 2.1 ist nur einzelne Formel; "natürlich" gilt auch $\neg p \to (q \wedge r \to \neg p)$. Allgemein:

Satz 2.3 *Sei $B \in For$ gültig mit Atomen $p_0, p_1, ..., p_n$ und seien $A_0, ..., A_n \in For$. B' entstehe aus B, indem man jedes p_i durch A_i ersetzt. Dann ist B' gültig.*

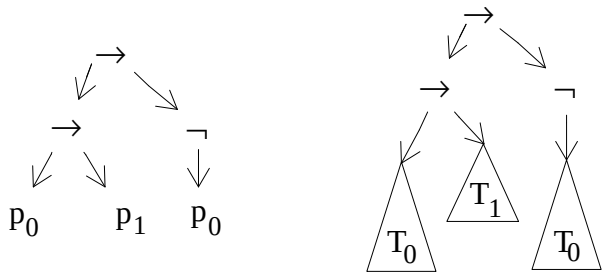
Dies gilt analog auch, wenn $A_0, ..., A_n$ Formeln der allgemeinen *Prädikatenlogik* sind; B' heißt dann *aussagenlogisch gültig.*

Beweis: Werten wir B' unter einer Interpretation I' aus, bestimmen wir, ob $I' \models A_i$ oder nicht. (In der Abbildung unten ist B gerade $(p_0 \to p_1) \to \neg p_0$; A_0 und A_1 entsprechen den Teilbäumen T_0 und T_1.)

$I' \models B'$ hängt nicht von A_i, sondern nur von diesen Ergebnissen ab, die mit $\to$ und $\neg$ kombiniert werden. D.h. $I' \models B'$ genau dann, wenn $I \models B$, wobei $p_i^I = T$ genau dann wenn $I' \models A_i$.

Da $I \models B$ immer gilt, ist B' gültig.

Formaler:
Wir betrachten nur die Ersetzung des Atoms $p_0 \in \mathcal{P}^0 = \{p_0, p_1, \ldots\}$ durch $A \in \mathit{For}$.



Beispiel: Ersetze p_0 durch $p_0 \to p_2$ in $(p_0 \to p_1) \to \neg p_0$.

Strukturelle Definition von *Ersetzung von p_0 durch A*:
Zu $B \in \mathit{For}$ sei B' definiert wie folgt:

i) $B \equiv p_i \in \mathcal{P}$:

a) $p_i \equiv p_0$: $\quad B' :\equiv$

b) $p_i \not\equiv p_0$: $\quad B' :\equiv$

ii) $B \equiv \neg B_1$: $\quad B' :\equiv \neg(B_1')$

iii) $B \equiv B_1 \to B_2$: $\quad B' :\equiv$

Sei I' beliebige Interpretation; wir definieren I, indem wir für alle $p_i \in \mathcal{P}$ setzen:

$$p_i^I := \begin{cases} p_i^{I'} & \text{wenn } i \neq 0 \\ T & \text{wenn } p_i \equiv p_0 \text{ und } I' \models A \\ F & \text{sonst} \end{cases}$$

Beh.: Für alle $B \in \mathit{For}$ gilt: $I \models B \Leftrightarrow I' \models B'$

Beweis: durch strukturelle Induktion über B

i) a) $B \equiv p_0$: $\quad I \models B \Leftrightarrow p_0^I = T \overset{\text{(Def. } I\text{)}}{\Longleftrightarrow} \ldots \overset{\text{(Def. Ersetzung)}}{\Longleftrightarrow} I' \models B'$

b) $B \equiv p_i \not\equiv p_0$: $\quad I \models B \Leftrightarrow p_i^I = T \Leftrightarrow \ldots \Leftrightarrow I' \models B'$

ii) $B \equiv \neg B_1$: $\quad I \models B \Leftrightarrow I \not\models B_1 \ldots \Leftrightarrow I' \models \neg(B_1') \Leftrightarrow I' \models B'$

iii) analog für $\to$

Jetzt sehen wir: Für alle I' gilt $I' \models B'$ genau dann, wenn für alle I' gilt $I \models B$. Ist also B gültig, so auch B'. □

Beispiel 2.1 (Fortsetzung): Nach Satz 2.3 gilt für alle $A, B \in \mathit{For}$: $A \to (B \to A)$

Analog zu 2.3 folgt aus $B_1 \models\!\models B_2$ auch $B_1' \models\!\models B_2'$. Nach Beispiel 2.2 ist also für alle $A, B, C \in \mathit{For}$: $A \to (B \to C) \models\!\models A \wedge B \to C$; die zweite Formel ist wohl besser zu verstehen.

Hier eine Liste mit bekannten aussagenlogischen Äquivalenzen (d.h. die folgenden Aussagen gelten sogar für alle prädikatenlogischen Formeln A, B und C); vgl. auch Proposition 1.1:

Kommutativität, Assoziativität $\vee$ und $\wedge$ sind kommutativ und assoziativ, d.h. $A \vee B \models\!\models B \vee A$ und $(A \vee B) \vee C \models\!\models A \vee (B \vee C)$ etc.

Idempotenz $A \vee A \models\!\models A$ und $A \wedge A \models\!\models A$

Distributivität $A \vee (B \wedge C) \models\!\models (A \vee B) \wedge (A \vee C)$ und $A \wedge (B \vee C) \models\!\models (A \wedge B) \vee (A \wedge C)$

De Morgansche Gesetze $\neg(A \vee B) \models\!\models \neg A \wedge \neg B$ und $\neg(A \wedge B) \models\!\models \neg A \vee \neg B$

Doppelte Negation (vgl. 1.1 i)) $\neg\neg A \models\!\models A$

Absorption $A \vee (A \wedge B) \models\!\models A$ und $A \wedge (A \vee B) \models\!\models A$

Neutrale Elemente $A \vee \mathit{false} \models\!\models A$ und $A \wedge \mathit{true} \models\!\models A$

Inverse Elemente $A \vee \neg A \models\!\models \mathit{true}$ (tertium non datur) und $A \wedge \neg A \models\!\models \mathit{false}$
(Achtung: die Verknüpfung mit $\vee$ ergibt das neutrale Element von $\wedge$ und vice versa!)

Null-Elemente $A \vee \mathit{true} \models\!\models \mathit{true}$ und $A \wedge \mathit{false} \models\!\models \mathit{false}$
(Sehen wir $\vee$ als Addition, das neutrale Element *false* also als 0 und $\wedge$ als Multiplikation, so entspricht die letzte Äquivalenz gerade der Aussage $x \cdot 0 = 0$.)

Wenn wir aus diesen Gesetzen weitere herleiten, verwenden wir meist unter der Hand Kommutativität und Assoziativität sowie die Symmetrie von $\models\!\models$.

Beispiele: Als Anwendung wollen wir zeigen, wie man im Gegenzug zu unseren obigen Überlegungen $\rightarrow$ aus Formeln entfernen kann:
$A \rightarrow B \models\!\models \neg\neg A \rightarrow B$ (Doppelte Negation) $\models\!\models \neg A \vee B$ (vgl. 1.1 iii)). (Hier haben wir zweimal die Symmetrie von $\models\!\models$ verwendet.)
Damit und mit 1.1 iv): $A \leftrightarrow B \models\!\models (\neg A \vee B) \wedge (\neg B \vee A)$

Ferner ist *false* nach Definition $\neg\mathit{true}$; also ist $\neg\mathit{false}$ äquivalent zu $\neg\neg\mathit{true}$ und zu *true* (Doppelte Negation).

Ähnlich gilt: $\mathit{false} \rightarrow A \models\!\models \neg\mathit{false} \vee A$ (s.o.) $\models\!\models \mathit{true} \vee A$ (gerade gesehen) $\models\!\models \mathit{true}$ (Null-Element; genaugenommen auch Kommutativität).
Die Aussage $\mathit{false} \rightarrow A \models\!\models \mathit{true}$ heißt ex falso quodlibet.

Exkurs:

Literale : $p, \neg p$ mit $p \in \mathcal{P}^0$

Klausel : $l_1 \vee \cdots \vee l_n$ mit Literalen l_i.

Eine Formel ist in bzw. eine *konjunktive(r) Normalform* (konj. NF, KNF), wenn sie eine Konjunktion von Klauseln ist; z.B. $(p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_3) \wedge (p_2 \vee \neg p_3)$. (Erfüllbarkeit ist schwer zu entscheiden, s.u.)

Analog: *disjunktive Normalform* (DNF): Disjunktion von Konjunktionen von Literalen

Erfüllbarkeit leicht:
(Verwendet zur Beschreibung von Schaltkreisen; es gibt Verfahren zu ihrer Minimierung).

(*) Zu jeder Formel gibt es eine logisch äquivalente disjunktive Normalform.

Wir lesen ihre Konjunktionen aus den Zeilen der Wahrheitstafel mit T, den T-Zeilen, ab:

Beispiel: $p \vee q \to \neg q \mathrel{=\!|\!\models}$??

p	q	$p \vee q$	$\neg q$	$\bullet$
T	T	T	F	F
T	F	T	T	T
F	T	T	F	F
F	F	F	T	T

(*) gilt auch für konjunktive Normalformen:

Man erhält eine konjunktive Normalform von A aus der disjunktiven Normalform D von $\neg A$ (betrachte F-Zeilen von A), indem man die De Morganschen Gesetze auf $\neg D$ anwendet. Das Ergebnis ist nämlich äquivalent zu $\neg D$; da D äquivalent ist zu $\neg A$, ist das Ergebnis also äquivalent zu $\neg\neg A$ und daher zu A (Idempotenz).
Im Bsp.: $D \mathrel{=\!|\!\models} (p \wedge q) \vee (\neg p \wedge q)$ und $\neg D \mathrel{=\!|\!\models}$??

Problem **SAT** (*satisfiability*): Geg. sei A in konj. NF; ist A erfüllbar ?

Dies ist (vermutlich) nur mit hohem Aufwand entscheidbar! (*NP-vollständig*) – Mit Wahrheitstafel entscheidbar; die hat aber bei n Atomen 2^n Zeilen!

$\Longrightarrow$ logische Formeln können

- sehr kompakt sein
- den Schwierigkeitsgrad von Problemen charakterisieren. (z.B. *NP-vollständig*) $\rightsquigarrow$ Informatik III; Komplexitätstheorie.

Da die Aufstellung einer Wahrheitstafel schnell sehr aufwendig wird, wollen wir noch ein weiteres Verfahren zur Bestimmung der KNF behandeln. Zuvor wollen wir kurz überlegen, wie man einer Formel in KNF ansieht, ob sie gültig ist. Offenbar bedeutet Gültigkeit der Formel, dass...

Gegeben sei also eine aussagenlogische Formel.

1) Entferne $\rightarrow$ and $\leftrightarrow$ mit obigen Äquivalenzen ($A \rightarrow B \;=\!|\!\models\; \neg A \vee B$, $A \leftrightarrow B \;=\!|\!\models\; (\neg A \vee B) \wedge (\neg B \vee A)$) am besten von innen nach außen, d.h. bottom-up im Syntaxbaum der Formel.

Beispiel: Gegeben $(\neg p \rightarrow \neg q) \leftrightarrow (q \rightarrow p)$. Ersetzen der beiden $\rightarrow$ ergibt $(\neg\neg p \vee \neg q) \leftrightarrow (\neg q \vee p)$. Ersetzung von $\leftrightarrow$ ergibt $((\neg(\neg\neg p \vee \neg q) \vee (\neg q \vee p)) \wedge (\neg(\neg q \vee p) \vee (\neg\neg p \vee \neg q))$.

Dies waren 3 Ersetzungen; wieviele benötigt man, wenn man top-down vorgeht und $\leftrightarrow$ zuerst ersetzt?

Für den nächsten Schritt: eine aussagenlogische Formel ist in *Negations-NF*, wenn Negationen nur unmittelbar vor Atomen stehen.

2) Bringe die Formel aus 1), die nur aus Atomen, $\wedge$, $\vee$ und $\neg$ aufgebaut ist, mit dem Verfahren NNF in Negations-NF. $NNF(A)$ ist

- A, wenn A ein Literal ist
- $NNF(B)$, wenn $A \equiv \neg\neg B$ (vgl. doppelte Negation)
- $NNF(\neg B \vee \neg C)$, wenn $A \equiv \neg(B \wedge C)$ (De Morgan)
- $NNF(\neg B \wedge \neg C)$, wenn $A \equiv \neg(B \vee C)$ (De Morgan)
- $NNF(B) \wedge NNF(C)$, wenn $A \equiv B \wedge C$
- $NNF(B) \vee NNF(C)$, wenn $A \equiv B \vee C$

Beispiel: (Forts.)

Ein interessanter Teilfall ist $NNF(\neg(\neg\neg p \vee \neg q)$, das auf $NNF(\neg\neg\neg p) \wedge NNF(\neg\neg q)$ und schließlich zu $\neg p \wedge q$ führt. Insgesamt ergibt sich

$$((\neg p \wedge q) \vee (\neg q \vee p)) \wedge ((q \wedge \neg p) \vee (p \vee \neg q))$$

3) Bringe die Formel A aus 2) mit dem Verfahren KNF in KNF. Dabei ist $KNF(A)$ $KNF(B) \wedge KNF(C)$, wenn $A \equiv B \wedge C$. Ansonsten lässt sich A als mehrstellige („große“) Disjunktion (evtl. auch einstellig) schreiben ($\vee$ ist kommutativ und assoz.); entweder ist dies eine Klausel – das Ergebnis ist A – oder A hat als ein Disjunkt eine Konjunktion und lässt sich als $B \vee C \wedge D$ schreiben – das Ergebnis ist $KNF(B \vee C) \wedge KNF(B \vee D)$ (Distributivität).

Der letzte Fall ergibt z.B.
$KNF(p \vee q \wedge \neg r \vee r) = KNF((p \vee r) \vee q \wedge \neg r) = KNF(p \vee r \vee q) \wedge KNF(p \vee r \vee \neg r)$.

Beispiel: (Forts.)

Mit zweimal Distribuieren ergibt sich

$$(\neg p \vee \neg q \vee p) \wedge (q \vee \neg q \vee p) \wedge (q \vee p \vee \neg q) \wedge (\neg p \vee p \vee \neg q)$$

Mit obigem Verfahren sehen wir sofort, dass diese Formel gültig ist, damit auch $(\neg p \rightarrow \neg q) \leftrightarrow (q \rightarrow p)$ und wegen Satz 2.3 auch $(\neg A \rightarrow \neg B) \leftrightarrow (B \rightarrow A)$.

Oft kann man die KNF vereinfachen, indem man gleiche Literale in einer Klausel oder auch gleiche Klauseln (als Mengen gesehen) zusammenfasst. Ferner kann man Klauseln mit einem Atom und seiner Negation weglassen. Weitere Vereinfachungen können sich durch Absorption ergeben.

2.2 Der Hilbert-Kalkül

Ziel: *Herleitung* aller gültigen Formeln durch Zeichenmanipulation aus *Axiomen* mittels *Schlußregeln*. Trotz Wahrheitstafeln ist dies aus folgenden Gründen interessant:

- Herleitung ist evtl. schneller (vgl. Exkurs und gezielte Suche)
- Wiederverwendbarkeit von alten Ergebnissen bzw. Zwischenergebnissen (Sätze, Lemmata)
- Wahrheitstafeln existieren nicht für Prädikatenlogik.

$M \vdash A$: A kann aus M *hergeleitet* werden. ($\vdash$ = turnstile, Drehkreuz)

Ziel:

- $M \vdash A \Rightarrow M \models A$ Kalkül *korrekt*, alles Herleitbare ist wahr.
- $M \models A \Rightarrow M \vdash A$ Kalkül *vollständig*, alles Wahre kann hergeleitet werden.

Bei diesem Vorgehen werden wir uns auf Formeln beschränken, die nur aus Atomen, Implikationen und Negationen bestehen.

Definition 2.4 Der *Hilbert-Kalkül* besteht aus den Axiomen (Regeln ohne Prämissen)

$$Ax1 := \{A \to (B \to A) \mid A, B \in For\}$$

$$Ax2 := \{(A \to (B \to C)) \to ((A \to B) \to (A \to C)) \mid A, B, C \in For\}$$

$$Ax3 := \{(\neg A \to \neg B) \to (B \to A) \mid A, B \in For\}$$

(endlich viele Axiomenschemata, s. Bem. u.)

und der Schlußregel *Modus Ponens MP*

$$\frac{A,\ A \to B}{B}$$

□

Bedeutung dieser Notation: Wenn wir A und $A \to B$ (*Prämissen* von MP) bewiesen haben, beweist dies B (*Konklusion* von MP).

Bem.: $Ax1$ und $Ax2$ sind besser einsichtig in der Form $A \wedge B \to A$ und $(A \wedge B \to C) \wedge (A \to B) \to (A \to C) \dashv\models ((A \wedge B \to C) \wedge (A \to B) \wedge A) \to C$; s. Satz 2.3. □

Bem.: $Ax1$ ist eine Menge von Axiomen. Man kann z.B. $A \to (B \to A)$ als *ein* Axiom mit "Meta-Variablen" A und B (also kein x) auffassen, die für Formeln stehen, und hat dann endl. viele Axiome. Man erhält jede Formel aus $Ax1$ (die man ja in den Herleitungen verwenden will), indem man jedes A durch dieselbe Formel ersetzt und analog für B etc., d.h. als sogenannte *Instantiierung* des Axioms.
(vgl. Satz 2.3) □

Was eine Herleitung zu Def. 2.4 ist, ist eigentlich schon klar; neu: Herleitung aus M.

Definition 2.5 $A \in For$ ist *aus* $M \subseteq For$ *H-herleitbar*, $M \vdash_H A$, (oder *herleitbar aus*, $M \vdash A$, in Abschnitt 2.2 und 2.3), wenn es eine Folge $A_1, \dots A_n$ in *For* gibt (ein "Beweis") mit:

(i) $A_n \equiv A$

(ii) Für $i = 1, \dots n$ gilt:

- $A_i \in Ax1 \cup Ax2 \cup Ax3 \cup M$ oder
- (Modus Ponens) Es gibt $j, k < i$ mit $A_j \equiv A_k \to A_i$
  (Beweis enthält A_k und $A_k \to A_i$; nach MP ist auch A_i bewiesen).

Die Folge heißt *Herleitung von A aus M* (vgl. Abschnitt 1.1)
$\vdash A$ steht für $\emptyset \vdash A$: A ist *herleitbar*. □

Beobachtung:

i) Für $1 \leq j < n$ ist auch $A_1, \dots A_j$ eine Herleitung (von A_j).

ii) Sind $A_1, \dots A_n$ und $B_1, \dots B_m$ Herleitungen, so auch $A_1, \dots A_n, B_1, \dots B_m$ (von B_m).

Beispiel 2.6 a) Für alle $A, B \in For$ gilt $\neg A \vdash A \to B$. Zum Beweis geben wir eine Herleitung aus $\neg A$ an – wieder mit einer Begründung für jede Zeile:

(1) $\neg A \to (\neg B \to \neg A)$	Ax1
(2) $\neg A$	trivial ($\in M$)
(3) $\neg B \to \neg A$	MP (2),(1)
(4) $(\neg B \to \neg A) \to (A \to B)$	Ax3
(5) $A \to B$	MP (3),(4)

Konstruktion (hier und meistens) rückwärts: $A \to B$ ist kein Axiom, also müssen wir es aus einem C und $C \to (A \to B)$ mit MP herleiten. $C \to (A \to B)$ ist in Ax1 für $C \equiv B$, aber B können wir sicher nicht aus $\neg A$ herleiten. $C \to (A \to B)$ ist in Ax3 für $C \equiv \neg B \to \neg A$; also suchen wir jetzt analog ein C für $\neg B \to \neg A$. $\neg A \to (\neg B \to \neg A)$ ist in Ax1 und diesmal liegt $\neg A$ einfach in M.

b) Für alle $A \in For$: $\vdash A \to A$.
Beweis:

(1) $(A \to (\underbrace{(A \to A)}_{B} \to \underbrace{A}_{C})) \to ((A \to (\underbrace{A \to A}_{B}) \to (A \to \underbrace{A}_{C}))$	Ax2
(2) $A \to (\underbrace{(A \to A)}_{B} \to A)$	Ax1
(3) $(A \to (A \to A)) \to (A \to A)$	MP (2),(1)
(4) $A \to (A \to A)$	Ax1
(5) $A \to A$	MP (4),(3)

□

- $A \to A$ ist klar (semantisch, d.h. im Sinne von $\models$), seine Herleitbarkeit aber nicht. Man könnte mehr Axiome nehmen, aber Ziel ist Minimalität: Möglichst wenige Fälle („Eleganz“) und starkes Vollständigkeitsresultat.

  Für praktische Herleitungen scheint der Kalkül also weniger geeignet; tatsächlich ist die Situation aber nicht so unbequem wie man denken könnte, denn jetzt können wir einfach $A \to A$ (wie ein Axiom!) in jeder Herleitung direkt (als Lemma) verwenden. Streng genommen müßten wir jeweils obige Herleitung einbauen, aber das können wir ja immer, so daß wir einfach darauf verzichten wollen.

- Wie kommt man auf den Beweis?
  $A \to A$ ist kein Axiom, z.z. sind also C und $C \to (A \to A)$.
  $C \equiv A$ unmöglich herleitbar, wie auch $C \equiv \neg A$.
  $C \equiv A \to A$ wollen wir ja gerade zeigen.
  $C \equiv A \to (A \to A)$: Axiom, relativ einfach, hängt mit $A \to A$ zusammen.
  $(A \to (A \to A)) \to (A \to A)$ sieht aus wie rechte Seite von Ax2 $\rightsquigarrow$ (1).
  Beweisen ist nicht einfach, aber endlich eine klare Def. – für Aussagenlogik.

Um das Beweisen zu vereinfachen, benötigen wir Regeln zum ”Zusammenstricken“ von Beweisen, d.h. zum Verwenden alter Ergebnisse wie $A \to A$. Proposition 2.7 i) ist fast dasselbe wie MP ; wir werden den Unterschied in Zukunft nicht immer machen. Proposition 2.7 ii) ist eine *abgeleitete Regel*.

Proposition 2.7 *i) Wenn $M \vdash A$ und $M \vdash A \to B$, dann $M \vdash B$.*

ii) Wenn $M \vdash \neg A \to \neg B$, dann $M \vdash B \to A$.

$$\frac{\neg A \to \neg B}{B \to A}$$

Beweis:

1. Herleitung für A und $A \to B$ hintereinanderschreiben – das ist eine Herleitung! Dann $MP \rightsquigarrow B$.

2. Folgt aus $M \vdash (\neg A \to \neg B) \to (B \to A)$ gemäß Ax3 und i).

□

Haben wir also in einer Herleitung $\neg A \to \neg B$ erreicht ($\neg A \to \neg B$ ist also aus dem jeweiligen M herleitbar), können wir in einer folgenden Zeile unmittelbar $B \to A$ schreiben, da wir nun wissen, daß wir eine Herleitung von $B \to A$ aus M einfügen können. Wir können also im Beispiel 2.6 a) Zeile (4) weglassen und (5) mit 2.7 ii) (3) begründen.

Der folgende Satz ist ein wichtiges Hilfsmittel zur Vereinfachung von Beweisen.

Satz 2.8 *(**Deduktionstheorem**) (vgl. Satz 1.4)*
$M \vdash A \to B \Leftrightarrow M \cup \{A\} \vdash B$

Bem.: „$\Leftarrow$“ wird implizit in Situationen wie der folgenden angewandt: Um einen Satz der Gruppentheorie zu beweisen, leiten wir ihn aus der Menge M_{gr} von Voraussetzungen her (s. Bsp. nach 1.3). Hat der Satz die Form $A \to B$, beginnen wir den Beweis of „Gelte also A ...“; dann leiten wir B her, wobei wir A als weitere Voraussetzung hinzunehmen, die wir bei Bedarf also jederzeit in die Herleitung einflechten können – wie ein Axiom. Wir zeigen also $M_{gr} \cup \{A\} \vdash B$; das bedarf einer Rechtfertigung, die nicht offensichtlich ist. □

Beweis:

$\Rightarrow$	n. Vor.	$M \cup \{A\} \vdash A \to B$	(selbe Herleitung)
	ferner	$M \cup \{A\} \vdash A$	(triviale Herleitung)
	also	M $\cup\{A\} \vdash B$	(2.7)

$\Leftarrow$ (wichtig) Es sei $A_1, ..., A_n$ eine Herleitung von B aus $M \cup \{A\}$, d.h. $A_n \equiv B$. Wir zeigen allgemeiner durch Induktion über i, daß für $i = 1, ..., n$ gilt $M \vdash A \to A_i$. Sei dies also wahr für alle $j < i$. (Noethersche Induktion)

1. $A_i \in M \cup Ax1 \cup Ax2 \cup Ax3$

$M \vdash A_i$	trivial
$M \vdash A_i \to (A \to A_i)$.	Ax1
$M \vdash A \to A_i$	(2.7)

2. $A_i \equiv A$

$M \vdash A \to A_i$	(2.6 b))

3. Es gibt $k, j < i$ mit $A_j \equiv A_k \to A_i$. Nach Induktion gilt also $M \vdash A \to A_k$ und $M \vdash A \to (A_k \to A_i)$. Damit

$M \vdash (A \to (A_k \to A_i)) \to ((A \to A_k) \to (A \to A_i))$	Ax2
$M \vdash A \to A_i$	(2 x 2.7)

Bem.: Noethersche Induktion, denn Schluß von i auf $i + 1$ funktioniert nicht in 3.; in einer Verankerung würde 1. und 2. stehen, die dann im Ind.schritt wiederholt werden müssten. □

Der Beweis gibt eine Konstruktion, wie aus einer Herleitung von A_n eine von $A \to A_n$ wird; dabei wird jeder Schritt der ersten Herleitung durch einen (abgeleiteten) oder durch drei Schritte ersetzt. Beachte: In den Herleitungen steht nicht „$M \vdash$“.)

Anwendungen:

Satz 2.9 *Für alle $A, B, C \in For$:*

i) $\vdash (A \to B) \to ((B \to C) \to (A \to C))$ *(Transitivität von $\to$)*

ii) $\vdash \neg A \to (A \to B)$ *($\neg A \wedge A \to B$)(ex falso quod libet)*

iii) $\vdash \neg\neg A \to A$

iv) $\vdash A \to \neg\neg A$

v) $\vdash (\neg A \to A) \to A$ *(ein Widerspruchsbeweis)*

Beweis: Die folgenden Beweise sind *keine Herleitungen* aus einer festen Menge M; trotzdem sind sie wie Herleitungen aufgebaut, wobei wir zusätzlich in jeder Zeile angeben, aus welcher Menge die jeweilige Formel herleitbar ist.

i) Herleitung von hinten konstruieren. Sei $M = \{A \to B, B \to C, A\}$

Naheliegend: Deduktion. Die führt zu (5), und jetzt muss man sich nur unabhängig vom Kalkül überlegen, warum C gilt, wenn man die Formeln in M schon weiß.

(1) $M \vdash A$	trivial
(2) $M \vdash A \to B$	trivial
(3) $M \vdash B$	MP (1),(2) – dasselbe M!! 2.7 i)
(4) $M \vdash B \to C$	trivial
(5) $M \vdash C$	MP (3),(4)
(6)	Deduktion (5)
(7)	
(8) $\vdash (A \to B) \to ((B \to C) \to (A \to C))$	

ii) Wieder Deduktion, aber nur einmal! Dann muss man Negation ins Spiel bringen, also naheliegenderweise Ax3 oder 2.7 ii) (2) anwenden. Um eine Implikation zu beweisen, bietet es sich immer an, sie als rechte Seite von Ax1 zu sehen. Deduktion „rückwärts“ spart dann eine Zeile im Vergleich zu „trivial, Ax1, MP“.

(1) $\vdash \neg A \to (\neg B \to \neg A)$	Ax1
(2) $\neg A \vdash \neg B \to \neg A$	Deduktion (1)
(3) $\neg A \vdash A \to B$	2.7 ii) (2) (vgl. obiges Bsp. – jetzt kürzer)
(4) $\vdash \neg A \to (A \to B)$	Deduktion (3)

iii) Wieder beginnen wir mit Deduktion und müssen nun wohl A mit MP beweisen. Wir suchen wieder C, so dass wir C und $C \to A$ herleiten können. $C \equiv A$ bringt nichts, $C \equiv \neg A$ wird sich aus $\neg\neg A$ nicht herleiten lassen; $C \equiv \neg\neg A$ scheint in (5) ziemlich dasselbe zu sein wie (7), aber hier haben wir $\neg\neg A$ als Hilfe, s. (1). Jetzt...

(1) $\neg\neg A \vdash \neg\neg A$	trivial
(2) $\neg\neg A \vdash$	
(3) $\neg\neg A \vdash$	

(4) $\neg\neg A \vdash$
(5) $\neg\neg A \vdash \neg\neg A \to A$
(6) $\neg\neg A \vdash A$ MP (1),(5)
(7) $\vdash \neg\neg A \to A$ Deduktion (6)

iv) (1) $\vdash$
(2) $\vdash A \to \neg\neg A$

□

abgeleitete Regeln:

Proposition 2.10 *i)* $\dfrac{A \to B, \quad B \to C}{A \to C}$ *ii)* $\dfrac{\neg\neg A}{A}$

Beweis: i) Wenn $M \vdash A \to B$ und $M \vdash B \to C$ (mit *derselben* Menge M!), dann: $M \vdash (A \to B) \to ((B \to C) \to (A \to C))$ wg. Satz 2.9 i), also $M \vdash A \to C$ mit 2 x MP (bzw. 2.7 i))
ii) Übung

□

2.3 Korrektheit und Vollständigkeit

Satz 2.11 *(Korrektheitssatz)*
Wenn $M \vdash A$ für $A \in For$ und $M \subseteq For$, dann gilt $M \models A$. Speziell: Wenn $\vdash A$, dann ist A gültig.

Beweis: Sei $A_1, ..., A_n$ eine Herleitung von A aus M, und sei I eine Interpretation mit $I \models M$. Wir zeigen durch Induktion über i: $I \models A_i$ für $i = 1, \ldots, n$.

Dann gilt $I \models A$ wegen
Sei also $I \models A_j$ für alle $j < i$.

i) $A_i \in M$: $I \models A_i$ nach Voraussetzung.

ii) $A_i \in Ax1 \cup Ax2 \cup Ax3$: $I \models A_i$, denn A_i ist gültig nach Satz 2.3 und Beispiel 2.1 bzw. Übung.

iii) $j, k < i$ mit $A_j \equiv A_k \to A_i$. Nach Induktion ist $I \models A_k$ und $I \models A_k \to A_i$, nach Def. von $\models$ also $I \models A_i$.

□

Schritte zur Vollständigkeit:

- Konsistenz (= keine widersprüchlichen Herleitungen)
- Modell-Lemma
- Vollständigkeitssatz

Definition 2.12 $M \subseteq For$ heißt *konsistent*, wenn für kein $A \in For$ zugleich $M \vdash A$ und $M \vdash \neg A$ gilt. □

Lemma 2.13 *i) Ist M inkonsistent, so $M \vdash B$ für alle $B \in For$.*

ii) $M \nvdash A \Rightarrow M \cup \{\neg A\}$ ist konsistent.

Beweis:

i) Sei $M \vdash A$ und $M \vdash \neg A$; wg. Satz 2.9 ii) ist $M \vdash \neg A \to (A \to B)$. Also $M \vdash B$ mit 2x MP.

ii) Sei $M \cup \{\neg A\}$ inkonsistent. Dann gilt $M \cup \{\neg A\} \vdash A$ nach i), d.h. $M \vdash \neg A \to A$ nach Satz 2.8. Also $M \vdash A$ nach Satz 2.9 v) und MP.

□

Mit Lemma 2.13 ii) kann man jede konsistente Menge M erweitern zu einer konsistenten Menge M', so daß für alle $A \in For$ entweder $M' \vdash A$ oder $M' \vdash \neg A$. (Entweder es gilt bereits $M \vdash A$, oder man nimmt $\neg A$ hinzu; wegen 2.13 ii) ist auch $M \cup \{\neg A\}$ konsistent.)

Dann erhält man ein Modell I von M durch $p^I := T$ gdw. $M' \vdash p$ (p beliebiges Atom).

Lemma 2.14 *(Modell-Lemma)*
Jede konsistente Menge ist erfüllbar, d.h. (in der Aussagenlogik): Sie besitzt ein Modell.

Satz 2.15 *(Vollständigkeitssatz)*
Für alle $A \in For$ und $M \subseteq For$ gilt $M \models A \Rightarrow M \vdash A$.

Beweis: Angenommen $M \nvdash A$. Dann ist $M \cup \{\neg A\}$ konsistent (Lemma 2.13 ii)) und hat ein Modell I (Lemma 2.14) mit $I \models M$ und $I \models \neg A$, d.h. $I \not\models A$. Also $M \not\models A$. □

Zusammenfassung 2.16 *i) $M \models A \Leftrightarrow M \vdash A$ (Sätze 2.11 und 2.15)*

ii) Tautologie $\widehat{=}$ herleitbar (Spezialfall von i))

iii) M erfüllbar $\Leftrightarrow$ M konsistent

Beweis: zu iii):

$\Rightarrow$ Wenn $I \models M$ gilt und $M \vdash A$ und $M \vdash \neg A$, dann gilt wg. Satz 2.11 auch $M \models A$ und $M \models \neg A$ und damit $I \models A$ und $I \models \neg A$ – Widerspruch!

$\Leftarrow$ Lemma 2.14

□

Satz 2.17 *(Endlichkeits- bzw. Kompaktheitssatz)*

i) $M \models A \Leftrightarrow$ es gibt eine endliche Teilmenge $M' \subseteq M$ mit $M' \models A$.

ii) M erfüllbar $\Leftrightarrow$ jede endliche Teilmenge von M ist erfüllbar.

Beweis: i) $\Leftarrow$ ist klar. Da eine Herleitung von A aus M nur endlich viele Formeln ($\widehat{=}M'$) von M benutzt, folgt $\Rightarrow$ aus 2.16 i).

ii) Wegen 2.16 iii) genügt es, zu zeigen:
M ist inkonsistent $\Leftrightarrow$ eine endliche Teilmenge $M' \subseteq M$ ist inkonsistent.

$\Rightarrow$ da Herleitungen von $M \vdash A$ und $M \vdash \neg A$ nur endlich viele Formeln ($\widehat{=}M'$) brauchen.

$\Leftarrow$ klar.

□

Es kann also nicht sein,

- daß man unendlich viele Formeln braucht, die in einem Modell gelten müssen, damit A gilt;
- daß jedes endliche $M' \subseteq M$ ein Modell hat, das man für größere M' aber immer wieder ändern muß, so daß letzlich M kein Modell hat.

3 Hilbert-Kalkül für Prädikatenlogik

3.1 Vorbereitung

Jetzt ist wieder β wichtig; z.B. gilt nach Definition: $M \models A \Leftrightarrow$ für alle $I, \beta : (I, \beta \models M \Rightarrow I, \beta \models A)$. In Kapitel 2 galt $M \models A \Leftrightarrow$ für alle I: $(I \models M \Rightarrow I \models A)$; jetzt haben wir:

Proposition 3.1 *$M \models A \Rightarrow$ für alle I: $(I \models M \Rightarrow I \models A)$, aber nicht umgekehrt.*

Beweis:
nur "aber": Sei $M = \{P(x)\}$ und $A \equiv P(y)$. Z.B. gilt für $P \mathrel{\widehat{=}}$ "ist prim", $\beta(x) = 2$ und $\beta(y) = 4$ sowohl $I, \beta \models M$ als auch $I, \beta \not\models A$. Also $M \not\models A$.

Andererseits gilt für alle I: $I \models M \Rightarrow I \models A$: Sei I beliebig mit $I \models P(x)$, d.h. für alle γ ist $I, \gamma \models P(x)$. Sei β eine beliebige Belegung; dann gilt auch $I, \beta_x^{\beta(y)} \models P(x)$, also $P^I(\beta(y))$ und $I, \beta \models P(y)$. Demnach gilt $I, \beta \models A$ für alle β, d.h. $I \models A$.
Bem.: $I \models P(x)$ – implizites $\forall x$ □

Wichtig ist auch wieder „$\forall x$". Zur Erinnerung:
$I, \beta \models A$ heißt: $A(x)$ gilt für $\beta(x)$. $\forall x A$ heißt: $A(x)$ gilt für alle x-Werte, d.h. für alle $d \in D$ gilt $I, \beta_x^d \models A$. Zum Eingewöhnen:

Proposition 3.2 *i)* $\forall x A \models A$

ii) i.a. nicht $A \models \forall x A$

Beweis:

i) Wenn $I, \beta \models \forall x A$, so speziell $I, \beta_x^{\beta(x)} \models A$ und $\beta_x^{\beta(x)} = \beta$, d.h. $I, \beta \models A$.

   Bsp.: Sei I $\mathbb{N}$ mit le als $\leq$; $\beta(x) = 5$, $\beta(y) = 0$. Dann gilt $I, \beta \models \forall x\, le(y, x)$, d.h. für alle $d \in \mathbb{N}$ gilt $I, \beta_x^d \models le(y, x)$ (informell: $le(y, d)$).

   Wir sehen: $\beta(x)$ ist egal; es gilt immer $0 \leq d$, speziell natürlich auch $0 \leq 5 = \beta(x)$ – und daher $I, \beta \models le(y, x)$. Aber: β (nämlich $\beta(y)$) ist wichtig; $3 \leq d$ wäre ja nicht immer richtig.

ii) $A \equiv P(x)$, $P \mathrel{\widehat{=}}$ "ist prim", $\beta(x) = 2$. Dann gilt in $\mathbb{N}$ $I, \beta \models P(x)$, nicht aber $I, \beta \models \forall x P(x)$.

□

Man sagt, dass x in der Formel $P(x)$ (für sich betrachtet) *frei* auftritt, weil über seinen Wert weiter nichts bekannt ist; darüber müssen wir eine Belegung nach $\beta(x)$ fragen. Dagegen ist das zweite Auftreten von x in $\forall x P(x)$ durch den Allquantor *gebunden*; es müssen alle möglichen Werte von x durchprobiert werden.

Vergleiche dazu die Blockstruktur in Programmiersprachen:

```
x=5; {int x; ... x=7;... if (x==5)... } if (x==5)...
```

`x == 5` hat innen den Wert `false`, außen `true`.

Gebunden $\mathrel{\widehat{=}}$ (neu) vereinbart; das Auftreten von `x` ist im inneren Block gebunden, Belegung außerhalb ist irrelevant.

Definition 3.3 FV bzw. BV, die Menge der *freien* bzw. *gebundenen Variablen*, ist wie folgt definiert:

i) Für $t \in Term$ ist $FV(t)$ die Menge der in t auftretenden Variablen, $BV(t) = \emptyset$.

ii)
- $FV(t_1 = t_2) := FV(t_1) \cup FV(t_2)$
  $BV(t_1 = t_2) := \emptyset$
- $FV(P(t_1, ..., t_n)) := \bigcup_{i=1}^{n} FV(t_i)$ (P bewirkt keine Bindung.) $BV(\ldots) = \emptyset$
- $FV(\neg A) := FV(A)$ ($\neg$ bewirkt ebenfalls keine Bindung)
  $BV(\neg A) := BV(A)$ (wegen evtl. Quantoren)
- $FV(A \to B) :=$
  $BV(A \to B) :=$
- $FV(\forall x A) := FV(A) \backslash \{x\}$ (vgl. Bew. zu 3.2 ii): $\beta(x)$ ist egal für $\forall x P(x)$)
  $BV(\forall x A) := BV(A) \cup \{x\}$

$A \in For$ heißt *geschlossen*, wenn $FV(A) = \emptyset$. □

Beispiele:

- $A \equiv (\forall x P(x)) \to P(x)$ (verschiedene *Auftreten* von x)
- $A \equiv \forall x P(y)$ $FV(A) = \{y\}$ $BV(A) = \{x\}$
  Wir unterstellen $x \not\equiv y$. Wäre $x \equiv x_0 \equiv y$, so wäre $FV(A) = \emptyset$.

Definition 3.4 $\forall x A$ heißt *Generalisierung* von A. Ist $FV(A) = \{y_1, \ldots, y_n\}$, so heißt jede der $n!$ Formeln $\forall y_1 \forall y_2 \ldots \forall y_n \; A$ ein *universeller Abschluß* von A. □

Beispiel: Kommutativgesetz: Wir schreiben oft $x+y = y+x$ und meinen $\forall x \forall y \; x+y = y+x$.

Funktionsrestriktion: $f : \mathbb{R} \to \mathbb{R}_0^+, \; x \to x^2$ ist nicht bijektiv, da z.B. $f(2) = f(-2)$. Die Einschränkung $f\big|_{\mathbb{R}_0^+} : \mathbb{R}_0^+ \to \mathbb{R}_0^+, \; x \to x^2$ ist bijektiv mit Umkehrfunktion $\sqrt{x}$.

Satz 3.5 *(Koinzidenzlemma)*
Seien $A \in For$ und β_1, β_2 Belegungen mit $\beta_1\big|_{FV(A)} = \beta_2\,|_{FV(A)}$. ($\beta_1$ und β_2 stimmen auf den freien Variablen von A überein.) Dann $I, \beta_1 \models A \Leftrightarrow I, \beta_2 \models A$.

Beweis: strukturelle Induktion □

Korollar 3.6 *Seien A, M geschlossen und β_1, β_2 Belegungen.*

i) $I, \beta_1 \models M \Leftrightarrow I, \beta_2 \models M$

ii) $I, \beta_1 \models M \Leftrightarrow I \models M$ *(mit i))*

iii) M *erfüllbar* $\Leftrightarrow M$ *hat ein Modell (mit ii))*

iv) $M \models A \Leftrightarrow$ *für alle* I*:* $(I \models M \Rightarrow I \models A)$

Beweis: zu iv):

" $\Rightarrow$ "

" $\Leftarrow$ "

□

Bem.: zu iv):
Beachte Unterschied zu 3.1.
Vgl. M_{gr} im Beispiel nach 1.3; dort sind Belegungen also in der Tat irrelevant. □

Proposition 3.7 *i)* $I \models A \Leftrightarrow I \models \forall x A$

ii) $\models A \Leftrightarrow \models \forall x A$

Bem.: Beachte den Kontrast zu 3.2; man muß mit offenen Formeln also vorsichtig sein. □

Beweis: Idee: A bzw. $\forall x A$ gilt jeweils für alle β, also für alle Werte $\beta(x)$; daher ist $\forall x$ egal.

i) für alle $\beta : I, \beta \models A$

$\Leftrightarrow_*$ für alle β und alle $d \in D : I, \beta^d_A \models A$

$\Leftrightarrow_{\text{Def.}}$ für alle $\beta : I, \beta \models \forall x A$

"$\Rightarrow_*$" β^d_x ist eine spezielle Belegung, die also auch A erfüllt.

"$\Leftarrow_*$" wähle $d = \beta(x)$; dann $\beta^d_x = \beta$.

ii) i) gilt für alle I.

□

Aus $\forall x P(x)$ wollen wir $P(c)$ oder $P(f(x, y))$ herleiten (z.B. wenn jede natürliche Zahl einen Nachfolger hat, so auch $3 + 2$); dazu ersetzen wir in $\underline{P(x)}$ x durch beliebige Terme, z.B. auch durch y. Allgemein: in $\forall x A$ redet A über x, wir wollen x in A substituieren. Substitution evtl. problematisch, wenn Formel A selbst Quantoren enthält:

- Enthält A $\forall x Q(x)$, wollen wir x dort nicht substituieren; $\forall x Q(x)$ redet (im Gegensatz zu $Q(x)$) gar nicht über x; genausogut könnten wir $\forall y Q(y)$ schreiben – sog. *α-Konversion.* (Analog zum Umbenennen von Variablen in Programmen: Enthält das Programm einen Block {int x; ...}, so können wir alle x in diesem Block durch „frisches" y ersetzen, ohne das Verhalten des Programms zu ändern.)

  Also: nur freie x ersetzen.

- $\forall y P(x, y)$ sagt etwas über x (z.B. "x ist kleinstes Element"). Ersetzt man darin x durch y, so entsteht die Formel $\forall y P(y, y)$; sie sagt nichts über y, sondern daß P reflexiv ist; beim Ersetzen des freien x durch y wurde dies "gefangen", als es in den *Bindungsbereich* von $\forall y$ geriet (vgl. Programmiersprachen).

  Das müssen wir vermeiden, in dem wir zuvor $\forall y P(x, y)$ zu $\forall z P(x, z)$ umschreiben.

Definition 3.8 Wir definieren die *Substitution* $[t/x]$ von x durch $t \in Term$ induktiv (vgl. Beweis zu 2.3):

T1) $x[t/x] :\equiv$

$y[t/x] :\equiv$

T2) $f(t_1, \ldots, t_n)\ [t/x] :\equiv f(t_1[t/x], \ldots, t_n[t/x])$

speziell: $c[t/x] :\equiv c \quad c \in \mathcal{F}^0$

Die Zeichenkette „$x[f(y)/x]$“ (9 Zeichen) ist *kein* Term! Sie beschreibt einen Term: $f(y)$
Analog: „der Term, der aus x entsteht, wenn man x durch $f(y)$ ersetzt“ ist kein Term.

A1, A2) $P(t_1, \ldots, t_n)\ [t/x] :\equiv$

„$=$ “ analog

A3) $(\neg A)\ [t/x] :\equiv \neg A[t/x]$ (steht für $\neg(A[t/x])$, d.h. $[t/x]$ bindet stärker als $\neg$)

A4) $(A \to B)[t/x] :\equiv$

A5) $(\forall x A)[t/x] :\equiv \forall x A$

$$(\forall y A)[t/x] :\equiv \begin{cases} \forall y\ A & \text{für } x \notin FV(\forall y A) \\ \forall y\ A[t/x] & \text{sonst, und } y \notin FV(t) \\ \forall z\ A[z/y][t/x] & \text{sonst, wobei } z \text{ eine } \textit{frische} \text{ Variable ist,} \\ & \text{d.h. } z \notin FV(t) \cup FV(A) \end{cases}$$

□

Beispiel:

$$\begin{aligned}
&\Big(\big(\forall z\ P(x,z) \to \forall x\ Q(x,z)\big) \to \forall y\ R(y,x)\Big)\ [f(y,x)/x] \\
&\equiv \Big(\forall z\ \big(P(x,z) \to \forall x\ Q(x,z)\big)\Big)[f(y,x)/x] \to \big(\forall y\ R(y,x)\big)[f(y,x)/x] \\
&\equiv \Big(\forall z\ \big(P(x,z) \to \forall x\ Q(x,z)\big)[f(y,x)/x]\Big) \to \forall z\ R(z,x)[f(y,x)/x] \\
&\equiv \Big(\forall z\ P(x,z)[f(y,x)/x] \to \big(\forall x\ Q(x,z)\big)[f(y,x)/x]\Big) \to \forall z\ R(z,x)[f(y,x)/x] \\
&\equiv \big(\forall z\ P(f(y,x),z) \to \forall x\ Q(x,z)\big) \to \forall z\ R(z,f(y,x))
\end{aligned}$$

Es gibt genau ein x mit $P(x)$ formalisiert man: $\exists x\ P(x) \wedge (\forall y\, P(y) \to x = y)$ – Klammern überflüssig. So kann man statt für $P(x)$ für jede Formel A mit freiem x vorgehen: $\exists_1 x\, A$ steht für $\exists x\ A \wedge \forall y\, A[y/x] \to x = y$.

Bei gegebener Signatur ist es oft nützlich ein *abgeleitetes*, z.B. 1-stelliges Prädikat $Q(x)$ einzuführen: Q steht einfach für eine Formel mit $FV = \{x\}$; die Zeichenkette $Q(t)$ ist jetzt eine (bei längerer Formel) viel übersichtlichere Abkürzung für $[t/x]$ angewandt auf die Formel.

Bsp.: $Q \equiv \exists z\ x = mult(z, z)$; $Q(y)$ steht für $\exists z\ y = mult(z, z)$, besagt also, dass y eine Quadratzahl ist – und auch $Q(z)$ funktioniert!

3.2 Der Kalkül

Definition 3.9 Der *Hilbert-Kalkül* hat als *Axiome* für alle $A, B, C \in For$ und $t \in Term$ alle (auch iterierte) *Generalisierungen* von:

Ax1: $A \to (B \to A)$ (wie in Kapitel 2)

Ax2: $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$ (wie in Kapitel 2)

Ax3: $(\neg B \to \neg A) \to (A \to B)$ (wie in Kapitel 2)

Ax4: $(\forall x\ A) \to (A[t/x])$ SP (Spezialisierung)
(z.B. Wenn jede natürliche Zahl einen Nachfolger hat, dann auch $3 + 2$; s.o.)

Ax5: $A \to \forall x\ A$, falls $x \notin FV(A)$ GE (Generalisierung)

Ax6: $(\forall x\ A \to B) \to \big((\forall x\ A) \to (\forall x\ B)\big)$ DA (Distributivität Allquantor)
(Wenn A für ein x gilt, dann auch B; zudem gilt A für jedes x, also auch B.)

Ax7: $x = x$ RE (Reflexivität)

Ax8: $x = y \to (A \to A')$, wobei A quantorenfrei ist und A' aus A entsteht, indem einige (oder kein) x durch y ersetzt werden. GL (Gleichheit)

und als *Schlußregel* nur Modus Ponens.

Die Definition einer Herleitung und von $M \vdash A$ wird analog aus Definition 2.5 übernommen. Für eine aussagenlogische Formel B heißt $\vdash_H B$, daß man B gemäß Definition 2.5 (mit dem "Aussagen-Hilbert-Kalkül") herleiten kann. □

Satz 3.10 *Sei B eine aussagenlogische Formel mit Atomen $p_0, \ldots, p_n$ und seien $A_0, \ldots, A_n \in For$. B' entsteht aus B wie in Satz 2.3. Gilt $\vdash_H B$ (also $\models B$), so auch $\vdash B'$ und $\models B'$. (syntaktisches Gegenstück zu Satz 2.3)*

Beweis: $\vdash_H B \Leftrightarrow \models B$ wegen 2.16; $\models B'$ wegen Satz 2.3.

Man kann in einer Herleitung $B_1, \ldots, B_m$ von B gemäß $\vdash_H B$ jedes B_i durch B_i' ersetzen; denn:

- Ist B_i Axiom in Definition 2.5, so B_i' Axiom in Definition 3.9.

- Wird B_j aus B_i und $B_i \to B_j$ hergeleitet, so auch B_j' aus B_i' und $(B_i \to B_j)' \equiv B_i' \to B_j'$.

□

Damit sind $A \to A$ und Satz 2.9 i) – v) auch im Prädikaten-Kalkül herleitbar. Die Propositionen 2.7 und 2.10 lassen sich hier auch genau wie in Kapitel 2 beweisen.

Beispiel: $\vdash A[t/x] \to \exists x A$ (Gilt A für t, so gibt es einen x-Wert, der A erfüllt.)

Bemerkung: $\exists x\ A \equiv \neg\ \forall x\ \neg\ A$; wir haben folgende Herleitung:

(1) $(\forall x \neg A) \to (\neg A)[t/x]$ Ax4 SP (beachte: $(\neg A)[t/x] \equiv \neg A[t/x]$)

(2) $\big((\underbrace{\forall x \neg A}_{C}) \to (\neg \underbrace{A[t/x]}_{D})\big) \to (\underbrace{A[t/x]}_{D} \to \neg \underbrace{\forall x \neg A}_{C})$ aussagenlogisch gültig

(ist ein B' mit $\models B$, Satz 3.10)

(3) $A[t/x] \to \neg\forall x \neg A$ MP (1),(2)

Wir haben hier benutzt, daß (2) aussagenlogisch gültig ist; wir haben uns also auf eine semantische Argumentation gestützt und damit die Herleitung deutlich verkürzt. Das ist gerechtfertigt, da wir vor allem Aussagen herleiten wollen, die man nicht so einfach überblicken kann wie (2).

Beispiel: $\vdash (\forall x \forall y \;\; P(x,y)) \to \forall x \;\; P(x,x)$

Im ersten Schritt der Herleitung verwenden wir eine Generalisierung von Ax4!

(1) $\forall x \;\; (\forall y \;\; P(x,y)) \to P(x,x)$	Ax4 SP
(2) $\big(\forall x \;\; (\forall y \;\; P(x,y)) \to P(x,x)\big) \to \big((\forall x \forall y \;\; P(x,y)) \to \forall x \;\; P(x,x)\big)$	Ax6 DA
(3) $(\forall x \forall y \;\; P(x,y)) \to \forall x \;\; P(x,x)$	MP (1), (2)

Satz 3.11 *(Deduktionstheorem)*
$M \vdash A \to B \;\Leftrightarrow M \cup \{A\} \vdash B$

Beweis: praktisch derselbe wie zu Satz 2.8

□

Satz 3.12 *(Generalisierungstheorem)*
Aus $M \vdash A$ und $x \notin FV(B)$ für alle $B \in M$ folgt $M \vdash \forall x A$.

Beweis: Sei A_1,...,A_n Herleitung von A aus M. Wir zeigen, daß man $\forall x A_1, \ldots, \forall x A_n$ zu einer Herleitung "auffüllen" kann. (Konstruktion einer Herleitung!)

i) Ist A_i Axiom, so auch $\forall x A_i$ ("Generalisierung" in Definition 3.9)

ii) Ist $A_i \in M$, fügen wir ein:

(1) A_i	Vor.	
(2) $A_i \to \forall x A_i$	Ax5 GE	**!!**
(3) $\forall x \; A_i$	MP (1),(2)	

iii) Wird A_i aus A_j und $A_j \to A_i$ hergeleitet, leiten wir aus $\forall x \; (A_j \to A_i)$ und Ax6 mit MP $(\forall x A_j) \to \forall x A_i$ her, daraus mit $\forall x \; A_j$ und MP $\forall x \; A_i$.

□

Korollar 3.13 $\vdash A \;\Rightarrow \vdash \forall x A$ *(vgl. Prop. 3.7 und 3.2)*

Proposition 3.14 *Sei $y \notin FV(\forall x A)$. Dann gilt $\vdash (\forall x A) \to \forall y (A[y/x])$ (Umbenennen von Hilfsvariablen (dummy variables) – α-Konversion)*

Beweis: $\vdash (\forall x A) \to (A[y/x])$ wegen Ax4 SP, also $\forall x A \vdash A[y/x]$ nach Satz 3.11. Nach Satz 3.12 gilt $\forall x A \vdash \forall y A[y/x]$ und fertig mit Satz 3.11. □

Beispiel: $\vdash (\forall x \;\; P(x) \to Q(f(x), z)) \to \forall y \;\; P(y) \to Q(f(y), z)$

Wenn man t mit $x \in FV(t)$ für z in $\forall x A$ substituieren will, nimmt man oft an, daß die Einsetzung eines frischen y "automatisch" geschieht, d.h. o.E. geschehen ist; also: Wenn wir "$A[t/z]$" schreiben und $x \in FV(t)$, so gilt immer $x \notin BV(A)$. Der schwierige 3. Fall von Definition 3.8 A5 tritt unter dieser Annahme (*Barendregt-Konvention*) nicht mehr auf; daß wir bei diesem 3. Fall ein beliebiges frisches z zugelassen und uns daher nicht auf genau einen Term festgelegt haben, ist jetzt durch 3.14 gerechtfertigt.

3.3 Korrektheit und Vollständigkeit

Satz 3.15 *(Korrektheit)*
$M \vdash A \Rightarrow M \models A$ *(speziell:* $\vdash A \Rightarrow \models A$*)*

Beweis: wie zu Satz 2.11. Wir müssen "nur noch" die Gültigkeit von Ax1 bis Ax8 beweisen. Wegen Proposition 3.7 ii) können wir das Wort "Generalisierung" in Definition 3.9 ignorieren;

damit sind Ax1 – Ax3 nach Satz 2.3 erledigt. Ax7 und Ax8 sind nicht schwer.

Ax4 SP Ansatz: Induktion über die Länge der Herleitung von A; langwierig, besonders für den Fall $A \equiv \forall y B$

Ax5 GE Es genügt zu zeigen: $A \models \forall x A$ (wegen Satz 1.4 ii)). Sei also $I, \beta \models A$ und $d \in D$ beliebig. Da $x \notin FV(A)$, sind β^d_x und β auf $FV(A)$ gleich. Demnach $I, \beta^d_x \models A$ nach Satz 3.5 und daher $I, \beta \models \forall x A$.

Ax6 DA einfacher (mit 2x 1.4 ii))

□

Zum Beweis der Vollständigkeit verwendet man wie in Kapitel 2 die Konsistenz (vgl. Definition 2.12) und zeigt analog zu Lemma 2.13:

Lemma 3.16 *i)* $M \nvdash A \Rightarrow M \cup \{\neg A\}$ *ist konsistent.*

ii) $M \nvdash \forall x A \Rightarrow M \cup \{\neg \forall x A, \neg A[c/x]\}$ *ist konsistent für jede Konstante* c*, die nicht in* M *und* A *vorkommt.*

Die Vereinigung in ii) enthält nicht nur $\neg \forall x A$, sondern auch einen "Zeugen" dafür.

Damit kann man zum Beweis des Modell-Lemmas wieder jede konsistente Menge in gewissem Sinne vervollständigen; aus der resultierenden Formelmenge liest man für Logik *ohne Gleichheit* erfüllende I_0 und β_0 mit $D_0 \subseteq Term(!)$ ab. Sind $P(c)$ und $P(f(c))$ in der Menge, nehmen wir c und $f(c)$ selbst als Elemente von D_0 und lassen P für diese Elemente wahr sein; s.a.u.

Für Logik mit Gleichheit ist D eine Menge von Äquivalenzklassen von D_0. (Terme, die gleich sein müssen – z.B. $c = c \circ e$ ist in M – sind äquivalent; als Terme, also als Elemente von D_0, sind c und $c \circ e$ verschieden – $[c]$ und $[c \circ e]$ sind aber dieselbe Äquivalenzklasse, also dasselbe Element von D.) Da *Term* abzählbar unendlich ist, ergibt sich:

Lemma 3.17 *(Modell-Lemma)*
konsistent $\Leftrightarrow$ *erfüllbar* *(vgl. Lemma 2.14 und 2.16)*

Satz 3.18 *(Löwenheim-Skolem)*
Jede erfüllbare Menge M *geschlossener Formeln hat ein höchstens abzählbares Modell bzw. im Falle von Logik ohne Gleichheit ein abzählbar unendliches Modell.*

Anwendung: Daher kann kein M $\mathbb{R}$ charakterisieren; neben $\mathbb{R}$ hätte M auch ein abzählbares, also anderes Modell.

Der Beweis zu 3.18 folgt aus obigen Überlegungen; beachte: Aus „geschlossen “ folgt $I, \beta \models M$ (M erfüllbar) $\Leftrightarrow I \models M$ (M hat ein Modell). Der Satz besagt natürlich nur, daß es jeweils zumindest ein solches Modell gibt – evtl. gibt es natürlich auch andere.

Jetzt folgt Vollständigkeit wie in Kapitel 2 ($I \rightsquigarrow I, \beta$):

Satz 3.19 *(Vollständigkeit)*
$M \models A \Rightarrow M \vdash A$

Damit gilt auch Satz 2.17:

Satz 3.20 *(Endlichkeits- bzw. Kompaktheitssatz der Prädikatenlogik)*

i) $M \models A \Leftrightarrow$ *es gibt eine endliche Teilmenge* $M' \subseteq M$ *mit* $M' \models A$.

ii) M *erfüllbar* $\Leftrightarrow$ *jede endliche Teilmenge von* M *ist erfüllbar.*

Anwendungen

Anwendung 3.21 *Die Menge der gültigen Formeln ist aufzählbar bzw. semi-entscheidbar. (Verfahren (z.B. eine Turingmaschine) sagt "ja" (akzeptiert), wenn Formel A gültig; sonst: "nein" oder keine Terminierung)*

Beweis: Mit Sorgfalt kann man alle Herleitungen nach und nach generieren, ggf. bis A hergeleitet wurde; 3.15 & 3.19. Gilt nicht $\models A$, terminiert dies Verfahren natürlich nicht. □

Bem.: Analog für $M \models A$ statt $\models A$, wenn M aufzählbar. □

Satz (Church) *Das Gültigkeitsproblem der Prädikatenlogik erster Stufe (also die Frage, ob* $\models A$*) ist unentscheidbar.*

Anwendung 3.22 *Jeder Satz der Gruppentheorie* ($M_{gr} \models A$, *vgl. Beispiel nach Definition 1.3) hat einen gruppentheoretischen Beweis, d.h. eine Herleitung aus* M_{gr} ($M_{gr} \vdash A$).

Ein Satz könnte ja auch in verschiedenen Gruppen aus verschiedenen Gründen gelten; der Beweis gibt *einen* einheitlichen Grund. Analog für andere in Prädikatenlogik formulierte Theorien.

Logik ist wichtig, um Sachverhalte exakt auszudrücken, d.h. formal zu definieren; dies hat jedoch auch seine Grenzen! Dies zeigen wir jetzt.

Eine Klasse $\mathcal{C}$ von Interpretationen bzw. Modellen (d.h. eine Eigenschaft) ist *definierbar*, wenn es $M \subseteq For$ gibt mit $I \models M \Leftrightarrow I \in \mathcal{C}$, *elementar definierbar*, wenn $|M| = 1$. Z.B. ist für eine geeignete Signatur $(\mathcal{F}, \mathcal{P})$ die Eigenschaft „I ist eine Gruppe“ definierbar – durch M_{gr}.

Anwendung 3.23 *Es gibt kein* $A \in For$ *mit:*

i) $I \models A \Leftrightarrow D$ *ist endlich*

(Endlichkeit ist nicht elementar definierbar, wohl aber $|D| = 3$, *siehe Übung)*

ii) (ohne Gleichheit) $I \models A \Leftrightarrow |D| = n$, *wobei* $n \in \mathbb{N}$ *fest*

(in Logik ohne Gleichheit ist n-Elementigkeit nicht elementar definierbar)

Beweis: Angenommen A existiert.

i) Setze $C_i :\equiv \exists x_1 \ldots \exists x_i \quad ((x_1 \neq x_2 \wedge \cdots \wedge x_1 \neq x_i) \wedge (x_2 \neq x_3 \wedge \cdots \wedge x_2 \neq x_i) \wedge \cdots \wedge (x_{i-1} \neq x_i))$; also: $C_i \Leftrightarrow$ es gibt $\geq i$ Elemente.

$M := \{A\} \cup \{C_i \mid i \geq 2\}$. Jede endliche Menge $M' \subseteq M$ ist erfüllbar – wähle D' mit $max\{i \mid C_i \in M'\}$ Elementen (hier sei ausnahmsweise $max\ \emptyset = 1$); damit sind alle $C_i \in M'$ erfüllt und auch A, das ja in M' sein könnte. Also ist auch M nach Satz 3.20 durch ein D erfüllt; $|D|$ ist endlich nach Annahme und $C_{|D|+1}$ ist verletzt. Widerspruch!!

ii) Wegen Proposition 3.7 i) ist A o.E. geschlossen. Wende Satz 3.18 auf {A} an, das natürlich erfüllbar ist. Widerspruch!!

□

i) und ii) gelten auch für "definierbar".

Modelle, die auf Termen basieren, interessieren bei Datenbanken (logische Programmierung, Prolog); speziell: Ein *Herbrand-Modell* I von $M \subseteq For$ erfüllt:

i) D ist die Menge der Terme, die aus Konstanten in M (falls nicht existent, Konstante c hinzunehmen) und Funktionssymbolen in M gebildet werden

ii) Für $f \in \mathcal{F}^n$ und $t_1, \ldots, t_n \in D$ gilt $f^I(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ speziell: $c^I = c$.

Jeder Term ist also seine eigene Interpretation. Prädikate sind nicht festgelegt.

Beispiel: $M = \{\forall x\ P(f(x), c),\ \forall x \forall y\ Q(g(x, c),\ y)\}$. Dann $D \ni c,\ f(c),\ g(c, c),\ f(f(c)),\ g(f(c), c), \ldots$.

g^I bildet $(f(c), g(c, c))$ auf $g(f(c), g(c, c))$ ab.

Satz: Eine erfüllbare Menge universeller, geschlossener Formeln ohne Gleichheit hat ein Herbrand-Modell.

Wir können uns bei der Suche nach erfüllenden Modellen also auf Herbrand-Modelle beschränken.

Dabei ist eine Formel *universell* (vgl. Abschnitt 1.4), wenn sie aus quantorfreien Formeln mittels Konjunktion, Disjunktion und All-Quantifizierung aufgebaut ist.

4 Weitere Beweisverfahren

Während der Hilbert-Kalkül mehrere Axiome und nur eine Regel hat, hat der Sequenzenkalkül nur ein Axiom und viele Regeln. Diese Regeln orientieren sich an der Struktur der involvierten Aussagen, was insbesondere das automatische Beweisen sehr erleichtert. Ein ganz anderes Beweisverfahren ist die Resolution, die sich auf Formeln in KNF anwenden lässt und die Grundlage für die Logikprogrammierung darstellt.

Im folgenden werden diese beiden Verfahren für die Aussagenlogik skizziert.

4.1 Sequenzenkalkül

Der Sequenzen- oder Gentzen-Kalkül verwendet vorwiegend *Regeln* zur Herleitung von Aussagen. Diese Aussagen heißen *Sequenzen*; sie haben die Form $M \vdash_G A$, wobei $M \subseteq For$ endlich ist, und bedeuten, dass sich A syntaktisch aus M „ergibt“. Die Voraussetzungsmenge ist also in die Aussage integriert, und es werden „normale“ Herleitungen ohne weitere Voraussetzungsmengen betrachtet.

Der Kalkül ist korrekt und vollständig, d.h. wieder: $M \vdash_G A \Leftrightarrow M \models A$ (wobei $M \vdash_G A$ hier eben etwas anders zu verstehen ist.)

In folgender Variante des Gentzen-Kalküls finden Sie je zwei Regeln für jeden logischen Operator, abhängig davon, ob der Operator in der zu zeigenden Formel (... *rechts*) oder in einer vorausgesetzten Formel (... *links*) auftritt:

$$\frac{M \cup \{A\} \vdash_G B}{M \vdash_G A \to B} \text{ Imp rechts} \qquad \frac{M \cup \{\neg C\} \vdash_G A \quad M \cup \{B\} \vdash_G C}{M \cup \{A \to B\} \vdash_G C} \text{ Imp links}$$

$$\frac{M \cup \{A\} \vdash_G \neg B}{M \cup \{B\} \vdash_G \neg A} \text{ Neg rechts} \qquad \frac{M \cup \{\neg B\} \vdash_G A}{M \cup \{\neg A\} \vdash_G B} \text{ Neg links}$$

$$\frac{M \vdash_G A \quad M \vdash_G B}{M \vdash_G A \wedge B} \text{ Kon rechts} \qquad \frac{M \cup \{A, B\} \vdash_G C}{M \cup \{A \wedge B\} \vdash_G C} \text{ Kon links}$$

$$\frac{M \cup \{\neg B\} \vdash_G A}{M \vdash_G A \vee B} \text{ Dis rechts} \qquad \frac{M \cup \{A\} \vdash_G C \quad M \cup \{B\} \vdash_G C}{M \cup \{A \vee B\} \vdash_G C} \text{ Dis links}$$

$$\frac{}{M \cup \{A\} \vdash_G A} \text{ Axiom}$$

Hier ist also das Deduktionstheorem eine Regel, nämlich *Imp rechts*. Durch die Brille des Deduktionstheorems ist z.B. *Neg links* eine Form der Kontraposition.

Korrektheit des Kalküls beweist man wieder durch Induktion über die Herleitungslänge. Das Axiom ist offenbar korrekt, da $M \cup \{A\} \models A$. Für die Korrektheit von *Imp rechts* nehmen wir nach Induktion an, dass $M \cup \{A\} \models B$; dann gilt auch $M \models A \to B$ nach Satz 1.4 ii).

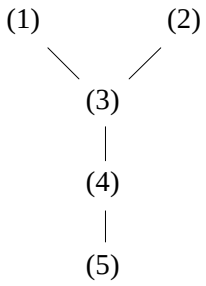
Ferner ist *Neg links* korrekt: Wenn $M \cup \{\neg B\} \models A$, so auch $M \cup \{\neg A\} \models B$ wegen Satz 1.4 ii) und $\models (\neg B \to A) \to (\neg A \to B)$.

Da wir beim Konstruieren von Herleitungen wie gewohnt von unten nach oben vorgehen werden, betrachten wir zuerst auch die Regeln von unten nach oben: Regel *Kon rechts* ist z.B. anwendbar, wenn die zu zeigende Formel eine Konjunktion $A \wedge B$ ist. In diesem Fall müssen wir zwei Prämissen beweisen; also zuerst die eine Hälfte der Konjunktion A, dann die andere B. Haben wir eine Konjunktion als Voraussetzung, so können wir diese mit Regel *Kon links* eliminieren und beide Teilformeln A und B als Voraussetzung betrachten.

Herleitungen sind wie üblich aufgebaut, wobei wir im weiteren die *Mengenklammern auf der linken Seite weglassen*, so dass wir es im Prinzip mit Sequenzen von aussagenlogischen Formeln zu tun haben.

Ein Beispiel:

(1)	$A, B \vdash_G A$	Axiom
(2)	$A, B \vdash_G B$	Axiom
(3)	$A, B \vdash_G B \wedge A$	Kon rechts (2) (1)
(4)	$A \wedge B \vdash_G B \wedge A$	Kon links (3)
(5)	$\vdash_G (A \wedge B) \rightarrow (B \wedge A)$	Imp rechts (4)



Die zu zeigende Formel (5) ist eine Implikation. Also müssen wir die Regel *Imp rechts* verwenden. Nun haben wir in Schritt (4) je eine Konjunktion als Voraussetzung und als zu zeigende Formel. Es ist hier egal, mit welchem Operator wir uns als nächstes befassen. Praktischerweise wählen wir die Regel *Kon links*, da diese weniger Prämissen aufweist. Die anschließende Regel *Kon rechts* in Schritt (3) spaltet den Beweis in zwei Fälle, wobei wir beide Fälle (1) und (2) mit dem *Axiom* abschließen können. Die Struktur der Herleitung kann man auch sehr gut als Baum darstellen.

Jetzt wollen wir noch eine Herleitung für $\vdash_G (\neg A \vee B) \rightarrow (A \rightarrow B)$ konstruieren und wieder die Struktur der Herleitung mit einem Baum illustrieren.

4.2 Resolution

Zur Erinnerung: Eine Formel in KNF ist eine Konjunktion von Klauseln; eine Klausel ist eine Disjunktion von Literalen; ein Literal ist ein (aussagenlogisches) Atom oder seine Negation. Bsp.: $(p_1 \vee \neg p_2 \vee p_3) \wedge (\neg p_1 \vee p_3) \wedge (p_2 \vee \neg p_3)$

Wir wollen für ein Literal l sein *negiertes Literal* mit $\bar{l}$ bezeichnen, also z.B. $\overline{\neg p} \equiv p$. (In anderen Worten, $\bar{l}$ enthält höchstens ein $\neg$.) Ferner wollen wir Klauseln als Mengen schreiben, so dass wiederholte Literale zusammenfallen und die Reihenfolge der Literale keine Rolle spielt. Ferner wollen wir eine KNF-Formel als Menge solcher Klauseln schreiben; das hat u.a. zur Folge, dass es zu einer endlichen Menge von Atomen (sagen wir n viele) nur endlich viele Klauseln (2^{2n}) und daher nur endlich viele KNF-Formeln gibt ($2^{2^{2n}}$).

Eine Formel in KNF ist unter einer Interpretation genau dann wahr, wenn in jeder Klausel mindestens ein Literal wahr ist. Wir wollen auch die leere Menge als Klausel zulassen, die entsprechende KNF-Formel ist dann offenbar unerfüllbar.

Sei A eine Formel in KNF mit Klauseln K und K', so dass ein Literal l existiert mit $l \in K$ und $\bar{l} \in K'$. Dann heißt $R = (K - \{l\}) \cup (K' - \{\bar{l}\})$ *Resolvente* von K und K'. Eine solche Resolvente ist genau dann leer, wenn $K = \{l\}$ und $K' = \{\bar{l}\}$.

Beispiel: Sei $A = \{\{p_1, p_3\},\ \{\{p_2, \neg p_3\}\}$; dann ist die einzige Resolvente $\{p_1, p_2\}$. (Idee: Ist p_3 falsch, so gilt p_1, sonst p_2; also gilt auch $p_1 \vee p_2$.)

Ein *Resolutionsschritt* fügt A eine neue Resolvente R zweier Klauseln von A hinzu. Ist $R = \emptyset$, so ist die neue Formel also unerfüllbar; dies war aber bereits für A der Fall.

Fügen wir wiederholt neue Klauseln als Resolventen hinzu, terminiert dieses Vorgehen nach obigen Betrachtungen. Das Ergebnis hängt auch nicht von der Reihenfolge der Hinzufügungen ab – man sagt, das Verfahren ist *determiniert*: Kann man R hinzufügen, wählt aber stattdessen R', dann kann man R hinterher immer noch hinzufügen und muss es auch schließlich. Wir bezeichnen das Ergebnis mit $Res^*(A)$.

Beispiel: Sei $A = \{\{p_1, p_3\},\ \{\{p_2, \neg p_3\}, \{\neg p_2\}\}$; dann gibt es zunächst zwei Resolventen $\{p_1, p_2\}$ und $\{\neg p_3\}$. Fügen wir diese hinzu, ergibt sich auf zwei verschiedene Weisen auch noch die Resolvente $\{p_1\}$. Alle diese Klauseln bilden $Res^*(A)$.

Lemma 4.1 *Sei A eine Formel in KNF mit Klauseln K und K' und einer Resolvente $R = (K - \{l\}) \cup (K' - \{\bar{l}\})$. Dann ist A genau dann erfüllbar, wenn $A \cup \{R\}$ es ist.*

Beweis: Eine $A \cup \{R\}$ erfüllende Interpretation erfüllt natürlich auch A. Gilt umgekehrt $I \models A$, so ist o.E. l^I wahr; wegen der Klausel K' muss dann I ein Literal in $K' - \{\bar{l}\}$ wahr machen, also auch eins in R. Damit gilt auch $I \models A \cup \{R\}$. □

Das Ziel des Resolutionsverfahrens ist es zu entscheiden, ob eine KNF-Formel A unerfüllbar ist. Dass die Bestimmung von $Res^*(A)$ dafür immer eine korrekte Antwort liefert, das Verfahren also vollständig und korrekt ist, zeigt der folgende Satz.

Satz 4.2 Resolutionssatz

Eine KNF-Formel A ist genau dann unerfüllbar, wenn $\emptyset \in Res^*(A)$.

Beweis: Korrektheit ($\Leftarrow$): Ist $\emptyset \in Res^*(A)$, so ist $Res^*(A)$ unerfüllbar; ferner ist mit wiederholter Anwendung von Lemma 4.1 jede Formel auf dem Weg zu $Res^*(A)$ (rückwärts betrachtet) unerfüllbar, also auch A.

Vollständigkeit ($\Rightarrow$): Sei A unerfüllbar. Wir zeigen durch Induktion über die Anzahl n der Atome in A: $\emptyset \in Res^*(A)$. Dies ist für $n = 0$ klar, da A wegen der Unerfüllbarkeit mindestens eine Klausel enthalten muss, und diese ist $\emptyset$.

Ind.ann.: Die Aussage gilt für jede KNF-Formel B mit Atomen $p_1, \ldots, p_n$. A habe die Atome $p_1, \ldots, p_n, p_{n+1}$. Wir gewinnen aus A zwei neue Klauselmengen A_0 und A_1.

A_0 entsteht aus A durch Streichen aller Klauseln mit p_{n+1} und entfernen aller Literale $\neg p_{n+1}$ aus den verbleibenden Klauseln. A_1 entsteht analog aus A durch Streichen aller Klauseln mit $\neg p_{n+1}$ und entfernen aller Literale p_{n+1} aus den verbleibenden Klauseln.

Wäre A_0 erfüllbar durch I, so können wir $p_{n+1}^I := T$ definieren (bzw. annehmen, dass es so definiert ist). Dann wären die gestrichenen Klauseln von A wegen des Literals p_{n+1} wahr; die anderen wären wahr, weil I ein Literal wahr macht, das in der entsprechenden (kleineren) Klausel in A_0 ist.

Dies ist ein Widerspruch zur Voraussetzung „A unerfüllbar“. Also ist A_0 unerfüllbar und analog A_1; nach Induktion gilt $\emptyset \in Res^*(A_0)$ und $\emptyset \in Res^*(A_1)$.

Wenn wir die Resolventen, die zu $\emptyset \in Res^*(A_0)$ führen, analog in A bilden, so erhalten wir entweder $\emptyset \in Res^*(A)$ und sind fertig, oder eine verwendete Klausel hat in A zusätzlich das Literal $\neg p_{n+1}$. Es ergibt sich also $\{\neg p_{n+1}\} \in Res^*(A)$ und analog $\{p_{n+1}\} \in Res^*(A)$. Ein weiterer Resolutionsschritt führt nun auf $\emptyset \in Res^*(A)$. □

Die Anwendung besteht oft darin, dass zu prüfen ist, ob sich aus einer Reihe von Voraussetzungen eine Folgerung ergibt. Dazu fügt man die negierte Folgerung den Voraussetzungen hinzu und prüft nun die Unerfüllbarkeit. Trifft diese zu, so ist die Folgerung korrekt.

Beispiel: Gegeben sei z.B. $p \wedge q \to r$, $q \wedge r \to r'$ und $p \wedge q$. Gilt dann r'?

Wir negieren r' und bringen alles in KNF. Hier werden $p \wedge q \to r$ und $q \wedge r \to r'$ zu Klauseln (1) $\neg p \vee \neg q \vee r$ und (2) $\neg q \vee \neg r \vee r'$; $p \wedge q$ zerlegen wir in (3) p und (4) q; und schließlich haben wir (5) $\neg r'$.

Die Eingabe ist also $\{\{\neg p, \neg q, r\}, \{\neg q, \neg r, r'\}, \{p\}, \{q\}, \{\neg r'\}\}$. Wir resolvieren (1) und (2) zu (6) $\{\neg p, \neg q, r'\}$, dies mit (5) zu (7) $\{\neg p, \neg q\}$, dies mit (4) zu (8) $\{\neg p\}$ und dies mit (3) zu $\emptyset$. Also ist r' in der Tat eine Folgerung.

Das Resolutionsverfahren kann i.A. sehr aufwendig sein; für die sogenannten *Horn-Klauseln* ist es aber effizient. Das sind Klauseln der Form $\{\neg p_1, \ldots, \neg p_n, p\}$ (entsprechend $p_1 \wedge \ldots \wedge p_n \to p$) bzw. $\{\neg p_1, \ldots, \neg p_n\}$ bzw. $\{p\}$, also mit höchstems einem nicht negierte Atom. Die Eingabe-Klauseln (1) bis (5) im letzten Beispiel sind Horn-Klauseln.

5 Korrektheit von Programmen – der Zusicherungskalkül

Ziele:
– formale Spezifikation von Programmen (Vor- und Nachbedingungen);
– Nachweis der Korrektheit, d.h. dass Programme ihre Spezifikation erfüllen.
Varianten des Kalküls gibt es allg. für imperative Programme; hier: C-Programmstücke mit Zuweisungen, `if` und `while` .

Man verwendet als Aussagen *Hoare-Tripel* der Form $\{A\}\ S\ \{B\}$, wobei

A, B: prädikatenlogische Formeln – sogenannte *Zusicherungen* –, die u.a. die Konstanten, Funktionen und Prädikate der Grundtypen (int, char, ...) und Programmvariablen verwenden (1.Verwendung von Logik).

A: *Vorbedingung*

B: *Nachbedingung*

S: Anweisung, d.h. ein Programmstück

A und B sind Kommentare, die man ins Programm schreiben kann; in C muß man dann schreiben /*A*/ S /*B*/ .

Hoare-Tripel sind selbst Aussagen, mit denen man folgern und herleiten kann.
(2.Verwendung von Logik)

5.1 Semantik

Hoare-Tripel kann man zwei verschiedene Bedeutungen geben.

schwache Semantik: Gilt A vor Ausführung von S, so gilt B danach, *falls* S ohne Fehlerabbruch terminiert. (Beachte: S ändert die Variablenbelegung β, d.h. sowohl $x = 1$ als auch $x > 1$ können *jeweils zu gegebener Zeit* wahr sein.) Man sagt dann, daß S *partiell korrekt* bzgl. A und B ist.

Beispiele:

i) $\{x \geq 1\}\ x = x - 1;\ \{x \geq 0\}$

ii) $\{true\}\ x = 2;\ \{x > 0\}$

iii) $\{x > 0\}\ x = x/0;\ \{x = 25\}$ endlicher Fehler (= in $\{\ldots\}$ ist Gleichheit)

iv) $\{true\}$ `while` () $x = x;\ \{x = 42\}$ unendlicher Fehler

strenge Semantik: Gilt A vor S, so terminiert S ohne Fehlerabbruch und dann gilt B. Man sagt dann, daß S *total korrekt* bzgl. A und B ist.

Beispiele: i), ii); nicht: iii), iv)

5.2 Zusicherungskalkül

(Floyd, Hoare (bekannt für Quicksort, TCSP), Dijkstra (Kürzeste-Wege-Algorithmus)):
Regeln, die es gestatten, gültige Aussagen der Form $\{A\}\ \ S\ \ \{B\}$ herzuleiten. Da wir zwei verwandte Semantiken haben, gibt es auch zwei Versionen von Gültigkeit; also haben wir genaugenommen auch zwei (sehr ähnliche) Kalküle.

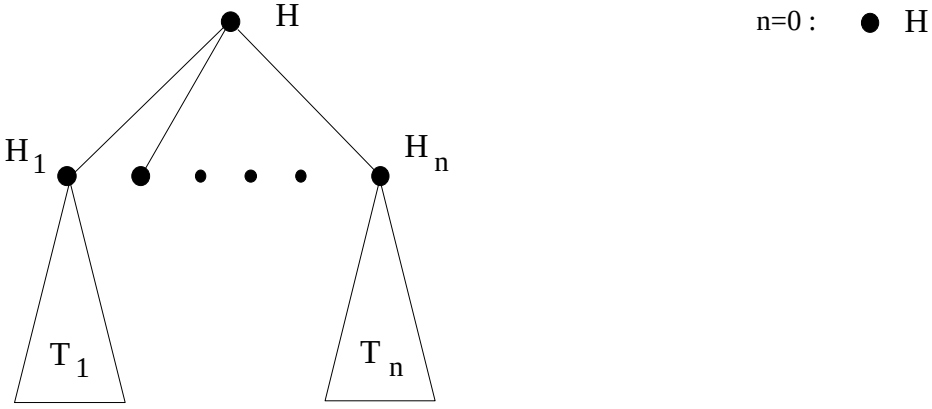
Ein Kalkül soll *korrekt* sein – und möglichst *vollständig*, d.h. die Herleitung aller gültigen Hoare-Tripel erlauben.

Regeln haben wieder die Form $\frac{H_1,\ldots,H_n}{H_{n+1}}$, wobei $n \geq 0$ und die H_i Hoare-Tripel sind. Ein *Axiom* liegt wieder vor bei $n = 0$, also $\frac{}{H_1}$.

Aus Regeln kann man wieder Herleitungen gewinnen; das sind Folgen $K_1,\ldots,\ K_m$, wobei für alle i eine Regel $\frac{H_1,\ldots,H_n}{H_{n+1}}$ existiert mit $K_i \equiv H_{n+1}$ und $H_1,\ldots,\ H_n$ erscheinen in $K_1,\ \ldots,\ K_{i-1}$.

Alternativ kann man (hier und in der Prädikatenlogik) eine Herleitung auch als Baum auffassen: Eine *Herleitung* (von H) ist (auch) ein Baum, dessen Knoten mit Hoare-Tripeln (und die Wurzel mit H) beschriftet sind, gemäß folgender induktiver Definition:

Sind $T_1,\ \ldots,\ T_n$ Herleitungen von $H_1,\ \ldots,\ H_n$ und $\frac{H_1,\ldots,H_n}{H}$ eine Regel, so ist T eine Herleitung von H:



Der Baum T besteht also aus einer mit H beschrifteten Wurzel und ggf. den Ästen (den durch die Kinder der Wurzel bestimmten Teilbäumen) $T_1,\ \ldots,\ T_n$.

Beispiele und Notation, s.u. (dort: Wurzel unten!)
Längere Beweise werden wie bisher zeilenweise angegeben, s. Bsp. am Ende.

Der Kalkül

In einer Zuweisung "$x = E$;" ist E ein Ausdruck. Sei D_E eine Formel, die sicherstellt, daß die Auswertung von E ohne Fehler terminiert. (Definiertheitsbedingung)

Beispiel: Zu $E \equiv x/y$ ist $D_E \equiv y \neq 0$.

5.2.1 Zuweisungsaxiom

$$\frac{}{\{B[^E/_x]\}\quad x = E;\quad \{B\}} \quad (= p)$$

$$\frac{}{\{D_E \wedge B[^E/_x]\} \quad x = E; \quad \{B\}} \quad (= t)$$

(=p) ist das Axiom für partielle, (=t) Axiom für totale Korrektheit. Diese Axiome sind jeweils korrekt, denn:

Gilt B, falls wir x mit dem Wert des Ausdrucks E belegen, so gilt nach der Zuweisung B mit dem aktuellen Wert von x. D_E garantiert zusätzlich Terminierung; ist E stets definiert, so ist $D_E \equiv \mathit{true}$ und kann wegfallen.

Beobachtung: Zu verschiedenen Zeitpunkten gelten verschiedene Formeln – anders als in Kapitel 1-3!! Man sollte Substitution beherrschen!!

Beispiel: 1. a) $\{2 > 0\}\ x = 2;\ \{x > 0\}$ ($D_E \equiv \mathit{true}$)

5.2.2 Konsequenzregel

$$\frac{A \Rightarrow B, \quad \{B\}\ S\ \{C\}, \quad C \Rightarrow D}{\{A\}\ S\ \{D\}} \quad (K)$$

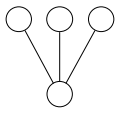
- Beide Varianten unseres Kalküls haben dieselbe Regel – wie auch in den nächsten beiden Fällen.
- Regel korrekt
- $A \Rightarrow B$ heißt: gilt A für unsere Grundtypen, so auch B. Werden die Grundtypen durch Axiome M beschrieben, so heißt $A \Rightarrow B$ also $M \cup \{A\} \models B$. Dies können wir evtl. durch Herleitung $M \cup \{A\} \vdash B$ nach Kapitel 3 zeigen. (3. Verwendung von Logik)
- Ist $A \Rightarrow B$ ein Hoare-Tripel?
- Ist $A \equiv B$ (und analog für $C \equiv D$), so gilt offenbar $A \Rightarrow B$; wir können die Regel anwenden, ohne diese Prämisse aufzuführen. (abgeleitete Regel)

Beispiel: 1. b) $\{\mathit{true}\}\ \ x = 2;\ \ \{x > 0\}$, denn $\mathit{true} \Rightarrow 2 > 0$ und 1.a) (und natürlich $x > 0 \Rightarrow x > 0$)

2. $\{x \geq 1\}\ \ x = x - 1;\ \ \{x \geq 0\}$,

Herleitung (Wurzel unten) zu diesem Beispiel – dabei kann $x \geq 0 \Rightarrow x \geq 0$ entfallen (Begründung zu $x \geq 0 \Rightarrow x \geq 0$: *Logik*; zu $x \geq 1 \Rightarrow x - 1 \geq 0$: *Logik/Arithmetik*):

$$\frac{x \geq 1 \Rightarrow x - 1 \geq 0, \quad \dfrac{}{\{x - 1 \geq 0\}\ \ x = x - 1;\ \ \{x \geq 0\}}\ (=), \quad x \geq 0 \Rightarrow x \geq 0}{\{x \geq 1\}\ \ x = x - 1;\ \ \{x \geq 0\}}\ (\mathrm{K})$$



Beherrscht man (=), argumentiert man nur: $x \geq 1$ (oder auch $x > 0$ für int x !) impliziert $x - 1 \geq 0$.

Beachte: x tritt in $E \equiv x - 1$ auf!

3. $\{x \geq 1\}\ \ x = x + y;\ \ \{x \geq y\}$, denn $x \geq 1 \Rightarrow x \geq 0 \Rightarrow x + y \geq y$

4. $\{x \geq 0\}\ \ x = -x;\ \ \{x \leq 0\}$, denn $x \geq 0 \Rightarrow -x \leq 0$

Bem.: $B[^E/_x]$ ist die allgemeinste bzw. schwächste Vorbedingung, die die Nachbedingung B garantiert. (weakest precondition, wp-Kalkül, Dijkstra) □

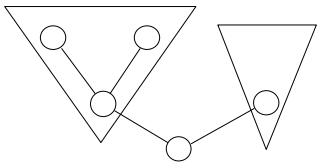
5.2.3 sequentielle Komposition

$$\frac{\{A\}\ \ S\ \ \{B\},\quad \{B\}\ \ T\ \ \{C\}}{\{A\}\ \ S;T\ \ \{C\}}\ (sK)$$

korrekt

Beispiel: Herleitung für int x

$$\dfrac{\dfrac{x > 0 \Rightarrow x - 1 \geq 0,\quad \dfrac{}{\{x - 1 \geq 0\}\ \ x = x - 1;\ \ \{x \geq 0\}}\ (=)}{\{x > 0\}\ \ x = x - 1;\ \ \{x \geq 0\}}\ (K),\quad \dfrac{\textit{siehe Beispiel 4}}{\{x \geq 0\}\ \ x = -x;\ \ \{x \leq 0\}}}{\{x > 0\}\ \ x = x - 1; x = -x;\ \ \{x \leq 0\}}\ (sK)$$



2 Bäume werden mit (sK) zusammengefügt.

5.2.4 bedingte Anweisung

$$\frac{\{A \wedge B\}\ \ S\ \ \{C\},\quad \{A \wedge \neg B\}\ \ T\ \ \{C\}}{\{A\}\ \ \texttt{if}\ (B)\ S\ \texttt{else}\ T\ \ \{C\}}\quad (if)$$

korrekt Vor.: B hat keine Seiteneffekte ($2 < + + x$) – dann wäre $A \wedge B$ keine Formel.

Beispiel: In diesem Beispiel wird (K) „wirklich“ gebraucht (bei der ersten Anwendung); wir geben zunächst zwei Teilbäume $T1$ und $T2$ an, denen die Wurzel folgt.

$$\dfrac{\textit{true} \wedge x > 3 \Rightarrow -x \leq 0,\quad \dfrac{}{\{-x \leq 0\}\ \ x = -x;\ \ \{x \leq 0\}}\ (=)}{\{\textit{true} \wedge x > 3\}\ \ x = -x;\ \ \{x \leq 0\}}\ (K)$$

$$\frac{true \wedge x \leq 3 \Rightarrow x - 3 \leq 0, \quad \overline{\{x - 3 \leq 0\}\ \ x = x - 3;\ \ \{x \leq 0\}}\ (=)}{\{true \wedge x \leq 3\}\ \ x = x - 3;\ \ \{x \leq 0\}}\ (K)$$

$$\frac{T1 \quad T2}{\{true\}\ \texttt{if}\ (x > 3)\ x = -x;\ \texttt{else}\ x = x - 3;\ \ \{x \leq 0\}}\ (if)$$

5.2.5 while-Schleife

Zwei Begriffe, die auch unabhängig vom Kalkül wichtig sind:

- *Schleifeninvariante*: Formel A, die bei jedem Schleifendurchlauf wahr bleibt. Man könnte denken, daß dies $\{A\}\ \ S\ \ \{A\}$ entspricht und $\{A\}\ \ \texttt{while}\ (B)\ S\ \ \{A\}$ impliziert; die nächste Regel ist genauer – und dadurch eher zu verwenden.
- *Terminierungsgröße* t: ein **int**-wertiger Term in den Programmvariablen, der jeweils kleiner wird, aber nach unten beschränkt ist.

$$\left.\frac{\{A \wedge B\}\ \ S\ \ \{A\}}{\{A\}\ \ \texttt{while}\ (B)\ S\ \ \{A \wedge \neg B\}}\right\}\ \text{(Wp)}$$

$$\left.\begin{array}{ll} (1) & \forall z \in \mathbb{Z} :\ \{A \wedge B \wedge t = z\}\ \ S\ \ \{A \wedge t < z\} \\ (2) & A \wedge B \Rightarrow t \geq 0 \\ \hline & \{A\}\ \ \texttt{while}\ (B)\ S\ \ \{A \wedge \neg B\} \end{array}\right\}\ \text{(Wt)}$$

In (1) werden Schleifeninvariante und Terminierungsgröße gemeinsam behandelt; die Verwendung von z ist ein „Trick“, um den alten und den neuen Wert von t zu vergleichen.

Korrektheit

Informell sehen wir die Korrektheit im partiellen Fall wie folgt. Gegeben ist $\{A \wedge B\}\ \ S\ \ \{A\}$, und zu Beginn gelte A. Ist die Überprüfung der while-Bedingung B positiv, gilt also bei Eintritt in den Schleifenrumpf $A \wedge B$. Am Ende des Schleifenrumpfes, d.h. bei der nächsten Überprüfung der while-Bedingung, gilt nach Voraussetzung also wieder A. Die Überlegung gilt genauso für jeden weiteren Schleifendurchlauf, bis schließlich A am Ende von S gilt, B aber falsch ist. Bei Verlassen der Schleife gilt also $A \wedge \neg B$.

$\{A\}$
`while` (B)
{
  $\{A \wedge B\}$
  S
  $\{A\}$
}
$\{A \wedge \neg B\}$

Ein formaler Beweis erfordert eine Induktion über die Anzahl der Schleifendurchläufe; zudem wollen wir natürlich auch die totale Korrektheit behandeln.

Es gelte A. Wir zeigen zunächst für (Wt), daß `while` $(B)\ S$ terminiert.

Ist B falsch, so Termination; sei also B wahr, so daß wegen (2) $t = z \geq 0$. Wir zeigen durch Induktion über $z \in \mathbb{N}$:
Falls $A \wedge B \wedge t = z \in \mathbb{N}$, so terminiert `while` (B) S.

Werde also Termination erreicht, falls $A \wedge B \wedge t = z' < z$, $z' \in \mathbb{N}$. Führen wir in der gegebenen Situation (wo $t = z$) S einmal aus, so gelten A und $t < z$ wegen (1). Ist nun B wahr und damit wegen (2) $t = z' \in \mathbb{N}$, wenden wir Induktion an; sonst sind wir wie oben fertig.

Wir können nun annehmen, daß A gilt und `while` (B) S terminiert, und zeigen für (Wp) und (Wt), daß danach $A \wedge \neg B$ gilt. Beweis durch Induktion über die Anzahl der Schleifendurchläufe; ist die Anzahl 0, so ist B von Anfang an falsch, und wir sind fertig.

Sei die Aussage für $n \geq 0$ Durchläufe wahr, und in der gegebenen Situation werde S $n+1$-mal ausgeführt. Es ist also zunächst B wahr; nach Prämisse gilt dann A nach dem ersten Durchlauf. Jetzt terminiert die Schleife nach n weiteren Durchläufen, nach Induktion fertig. □

5.2.6 Abgeleitete Schlußregel für Schleifen

In Anwendungen stehen Schleifen natürlich nicht isoliert, sondern meist nach einer initalisierenden Anweisung *init*.

Lemma 5.1 *Folgende Schlußregeln sind korrekt:*

$$\left.\begin{array}{ll} (1) & \{C\}\ \mathit{init}\ \{A\} \qquad \text{(Invariante muß bei Erreichen der Schleife gelten)} \\ (2) & \{A \wedge B\}\ S\ \{A\} \\ (3) & A \wedge \neg B \Rightarrow D \quad \text{(Invariante ist stark genug zum Beweis der Spezifikation } D\text{)} \\ \hline & \{C\}\ \mathit{init}\ \texttt{while}\ (B)\ S\ \{D\} \end{array}\right\} \text{(Sp)}$$

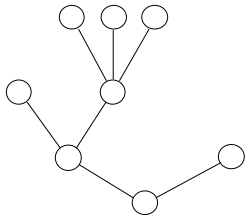
$$\left.\begin{array}{ll} (1) & \{C\}\ \mathit{init}\ \{A\} \qquad \text{(Invariante muß bei Erreichen der Schleife gelten)} \\ (2) & \forall z \in \mathbb{Z}:\ \{A \wedge B \wedge t = z\}\ S\ \{A \wedge t < z\} \\ (3) & A \wedge B \Rightarrow t \geq 0 \\ (4) & A \wedge \neg B \Rightarrow D \quad \text{(Invariante ist stark genug zum Beweis der Spezifikation } D\text{)} \\ \hline & \{C\}\ \mathit{init}\ \texttt{while}\ (B)\ S\ \{D\} \end{array}\right\} \text{(St)}$$

Bem.: In Anwendungen hat man C und D und muß A finden – Problem!! □

Beweis: für (St):

$$\dfrac{(1) \quad \dfrac{\dfrac{(2) \quad (3)}{\{A\}\ \texttt{while}\ (B)\ S\ \{A \wedge \neg B\}}\ (Wt)}{\{C\}\ \mathit{init}\ \texttt{while}\ (B)\ S\ \{A \wedge \neg B\}}\ (sK) \qquad (4)}{\{C\}\ \mathit{init}\ \texttt{while}\ (B)\ S\ \{D\}}\ (K)$$

Baumstruktur: □



Beispiel: Seien $m_0, n_0 > 0$ und g kurz für $\text{ggt}(m_0, n_0)$.

int m, n
$A \equiv \text{ggt}(m,n) = g \quad$ (impliziert $m, n > 0$) $\quad t \equiv m + n$
$GGT \equiv init\ loop$
$init \equiv m = m_0;\ n = n_0;$
$loop \equiv$ `while` $(m \neq n)\ body$
$body \equiv$ `if` $(m > n)\ m = m - n;$ `else` $n = n - m;$

Behauptung: In der starken Semantik gilt $\{true\}\ GGT\ \{m = g\}$

Beweis: (abgekürzte) Herleitung als Liste. (L) $\widehat{=}$ Logik

(1)	$true \Rightarrow \text{ggt}(m_0, n_0) = g$	(L)
(2)	$\{\text{ggt}(m_0, n_0) = g\}\ m = m_0;\ \{\text{ggt}(m, n_0) = g\}$	(=t)
(3)	$\{\text{ggt}(m, n_0) = g\}\ n = n_0;\ \{A\}$	(=t)
(4)	$\{\text{ggt}(m_0, n_0) = g\}\ init\ \{A\}$	(sK) (2),(3)
(5)	$\{true\}\ init\ \{A\}$	(K) (1),(4)
(6)	$A \wedge m \neq n \wedge m > n \wedge m + n = z \Rightarrow$ $\text{ggt}(m-n, n)$ ist definiert und $= \text{ggt}(m,n) = g$ und $m - n + n < z$	(Arithmetisch: $(m-n, n)$ und (m, n) haben dieselben gemeinsamen Teiler und $A \Rightarrow n > 0$)
(7)	$\{\text{ggt}(m-n, n) = g \wedge m - n + n < z\}\ m = m - n;$ $\{A \wedge m + n < z\}$	(=t)
(8)	$\{A \wedge m \neq n \wedge m > n \wedge m + n = z\}\ m = m - n;$ $\{A \wedge m + n < z\}$	(K)(6),(7)
(9)	$\{A \wedge m \neq n \wedge \neg(m > n) \wedge m + n = z\}\ n = n - m;$ $\{A \wedge m + n < z\}$	analog zu (8)
(10)	$\{A \wedge m \neq n \wedge m + n = z\}\ body\ \{A \wedge m + n < z\}$	(if) (8),(9)
(11)	$A \wedge m \neq n \Rightarrow m, n > 0 \Rightarrow m + n \geq 0$	(Arithmetik)
(12)	$A \wedge m = n \Rightarrow m = \text{ggt}(m,n) = g$	(Arithmetik)
(13)	$\{true\}\ GGT\ \{m = g\}$	(St) (5),(10),(11),(12)

Graphisch (L: Logik A: Arithmetik) :

$$\dfrac{\dfrac{\dfrac{(L)}{(1)} \quad \dfrac{\overline{(2)}\ (=t) \quad \overline{(3)}\ (=t)}{(4)}\ (sK)}{(5)}\ (K) \qquad \dfrac{\dfrac{\dfrac{(A)}{(6)} \quad \overline{(7)}\ (=t)}{(8)}\ (K) \qquad \dfrac{analog}{(9)}}{(10)}\ (\texttt{if}) \quad \dfrac{(A)}{(11)} \quad \dfrac{(A)}{(12)}}{(13)}\ (St)$$

5.3 Verwendungsmöglichkeiten des Zusicherungskalküls

- Verifikation eines gegebenen Programms S mit gegebener Nachbedingung (Spezifikation) B:

  - zeige $\{A\}\ S\ \{B\}$ für gegebene Vorbedingung A

    oder

  - bestimme A und zeige $\{A\}\ S\ \{B\}$

  Schlußweise eher "bottom-up", d.h. von einfachen zu zusammengesetzten Programmstücken

- Die oder einige Zusicherungen des Beweises (als Kommentar) ins Programm schreiben $\Rightarrow$ Dokumentation

- Konstruktion von Programmen mit gleichzeitigem Korrektheits- und Terminierungsbeweis: Gegeben B und eventuell A, suche S mit $\{A\}\ S\ \{B\}$.

  Schlußweise "top-down": Aus Anforderungen an einen größeren, noch nicht ausprogrammierten Programmteil werden Anforderungen an kleinere Teile und eine Struktur, gemäß der jene zum größeren Teil zusammengesetzt werden sollen. Vgl. Lemma oben: Sind C und D gegeben und man hat die Idee, daß das Programm die Form $init$+ Schleife haben soll, so ergeben sich Anforderungen an diese Teile.

Bem.:

1. Für ernsthafte Anwendung ist Rechnerunterstützung nötig

   - prüfen der Regelanwendung
   - halbautomatisches Beweisen: einige Beweisschritte automatisch, aber der Mensch gibt z.B. Schleifeninvarianten und Terminierungsgröße an.
   - oder (einfacher): nur einige Zusicherungen als (vermutete) Beweisskizze ins Programm schreiben (Dokumentation!); diese werden zur Laufzeit für den konkreten Fall geprüft; ggf. Stop mit Fehlermeldung. Prüfung nur im debugging-Modus , da diese

2. Trotz des formalen Beweises kann es selbts beim obigen GGT-Verfahren in der Praxis Probleme geben. Sind z.B. $m = 2^{100}$ (Gleitkommadarstellung) und $n = 4$, so ist nach $m = m - n$ m unverändert und das Verfahren terminiert nicht.

3. Beide Varianten sind nur anwendbar, wenn Programm terminieren soll; sonst (z.B. Betriebssystem): temporale Logik für Zusicherungen und Spezifikation.

□

6 Temporale Logik

6.1 LTL

In Systemabläufen ändern Formeln ihren Wahrheitswert, vgl. Kapitel 5. (Ist $x = 1$, so nicht mehr nach der Zuweisung $x = x + 1$.) Der Zusicherungskalkül ist nur geeignet für terminierende Systeme (z.B. Programme, die *ein* Ergebnis liefern). Viele Systeme terminieren nicht, sondern sollen immer wieder auf Anforderungen reagieren (sogenannte *reaktive* Systeme): Betriebssysteme, Schaltkreise, Kommunikationsprotokolle ... (oft parallel / verteilt)

Beispiel 6.1 Spezifikationen, die über Abläufe reden:

i) Es greifen *nie* zwei Prozesse gleichzeitig auf denselben Datensatz zu. (wechselseitiger Ausschluß, mutual exclusion, MUTEX).

ii) Es erscheint *immer wieder* eine Eingabeaufforderung (prompt).

iii) *Immer, wenn* eine Anforderung (request, r) kommt, wird diese *nach einer Weile* erfüllt (granted, g).

Formulierung in Prädikatenlogik: $\forall t_1 \;\; r(t_1) \rightarrow \exists t_2 \;\; t_2 \geq t_1 \wedge g(t_2)$

□

Idee: keine explizite Modellierung der Zeit (Axiome dafür sind kompliziert, Formeln unübersichtlich); besonders günstig, wenn wir sonst nur Aussagenlogik brauchen.

$\mathbf{G}$: (von jetzt an) immer, always (globally); $\Box$

$\mathbf{F}$: irgendwann (ab jetzt), eventually (finally); $\Diamond$

Damit 6.1 iii) formal als $\mathbf{G}\,(r \rightarrow \mathbf{F}\,g)$; hier soll g jeweils irgendwann „ab jetzt“, also nach r gelten.

Eigenschaften i) - iii) aus 6.1 sollen für alle Abläufe gelten, die typischerweise unendliche Folgen von Zuständen sind. (Systeme, die wir betrachten, machen unendlich viele Schritte.) Zeit ist hier also diskret:

$\mathbf{X}$: im nächsten Zustand, next(time), $\circ$

Im folgenden sei wie in Kapitel 2 eine Menge $\mathcal{P} = \mathcal{P}^0$ von *Atomen* (atomare Formeln) gegeben. Systemzustände sind evtl. sehr detailliert; wir können von diesen Details abstrahieren – uns interessiert nur, welche Atome wahr sind. (L in 6.2 leistet gerade diese Abstraktion.) Anhand dieser Information können wir z.B. prüfen, ob jedem Zustand, in dem r wahr ist, ein Zustand folgt, in dem g wahr ist.

Definition 6.2 Ein *Ablauf* $\pi = s_0, s_1, \ldots$ ist eine unendliche Folge von Zuständen s_i aus einer Menge S von *Zuständen* mit einer *Bewertung* $L: \; S \rightarrow \mathcal{P}(\mathcal{P})$ (Potenzmenge von $\mathcal{P}$).
$\pi^j = s_j, s_{j+1}, \ldots$ ist das j-te *Suffix* von π. □

Die Idee ist, daß p im Zustand s gilt, falls $p \in L(s)$; $L(s)$ ist also im Sinne von Kapitel 2 eine Interpretation (Modell) – *hier* sind Abläufe die Modelle, siehe Definition 6.4. Wir schreiben in Abläufen auch $L(s_i)$ statt s_i; dann ist $\pi = \{p\},\ \{p,r\},\ \{p\},\ \{p,r\} \ldots$ ein Ablauf, den wir formaler schreiben als $\pi = (\{p\},\ \{p,r\})^\omega$; ähnlich wie $*$ in regulären Ausdrücken für beliebige endliche Wiederholung steht, steht ω für unendliche Wiederholung.

Die folgende Logik redet über Abläufe, eine lineare Struktur; daher heißt sie Linear Time Logic, LTL – oder auch genauer PLTL, da wir ansonsten nur Aussagenlogik verwenden:

Definition 6.3 (Syntax von LTL/PLTL)
Formeln von PLTL (Propositional Linear Time Logic): $TFor_{\mathcal{P}}(TFor)$ ist die kleinste Menge mit:

(T1) $p \in \mathcal{P} \Rightarrow p \in TFor$

(T2) $A, B \in TFor \Rightarrow \neg A,\ A \wedge B,\ A \vee B,\ A \rightarrow B,\ A \leftrightarrow B\ \in TFor$
(Klammerung wie üblich; $\neg$, $\mathbf{G}$, $\mathbf{F}$ und $\mathbf{X}$ (T3) binden am stärksten, $\mathbf{U}$ (T4) stärker als die restlichen Operatoren)

(T3) $A \in TFor \Rightarrow \mathbf{G}\,A,\ \mathbf{F}\,A,\ \mathbf{X}\,A\ \in TFor$

(T4) $A, B \in TFor \Rightarrow A\,\mathbf{U}\,B \in TFor$ (until)

□

Definition 6.4 (Semantik von LTL)
Sei $\pi = s_0, s_1, \ldots$ ein Ablauf. $A \in TFor$ *gilt* für π (π *erfüllt* A, $\pi \models A$) ist wie folgt definiert:

(T1) $\pi \models p$, wenn $p \in L(s_0)$ (p gilt im 1. Zustand)

(T2) • $\pi \models \neg A$, wenn nicht $\pi \models A$.

• $\pi \models A \vee B$, wenn $\pi \models A$ oder $\pi \models B$ etc.

(T3) • $\pi \models \mathbf{X}\,A$, wenn $\pi^1 \models A$

Mit $\pi = (\{p\},\ \{p,r\},\ \emptyset,\ \{p,r'\})^\omega$ gilt also $\pi \models \mathbf{X}\,r$, nicht aber $\pi \models r$ oder $\pi \models \mathbf{X}\,\mathbf{X}\,r$.

Warum brauchen wir $\pi^1 = \{p,r\},\ \emptyset,\ \{p,r'\}(\{p\},\ldots)^\omega$? Können wir nicht einfach sagen, daß A im Zustand s_1 gelten muß?

• $\pi \models \mathbf{G}\,A$, wenn

Bei obigem π gilt $\pi \models \mathbf{G}\,(r \rightarrow p)$, aber nicht $\pi \models \mathbf{G}\,p$

• $\pi \models \mathbf{F}\,A$, wenn

Es würde auch Sinn machen zu verlangen, daß A irgendwann in der Zukunft gelten soll – wie ändert das die Def?

Hier ist also eine formale Semantik-Def. nötig, da es verschiedene sinnvolle Möglichkeiten gibt. Variante bei uns als

Oben gilt $\pi \models \mathbf{F}\,(r' \wedge \mathbf{X}\,p)$.

(T4) $\pi \models A \,\mathbf{U}\, B$, wenn es ein $j \geq 0$ gibt mit: $\pi^j \models B$ und für alle $0 \leq i < j:\ \pi^i \models A$.

A gilt also, bis schließlich B gilt – starkes until; weitere Design-Entscheidung: A muß hier bis, aber nicht im Zustand gelten, in dem B gilt

Oben gilt $\pi \models p \,\mathbf{U}\, r$, aber nicht $\pi \models p \,\mathbf{U}\, r'$

A ist *gültig / erfüllbar*, wenn alle π / ein π A erfüllen.
Wie bisher bedeutet *logisch äquivalent*, $=\!|\!\models$: dieselben π erfüllen links wie rechts; entsprechend bedeutet $M \models A$: "A folgt aus M." □

Beispiel: Sei $\pi = (\{p\},\ \{p,\ r\},\ \{r,\ q\},\ \{r,\ q'\})^\omega$.

$\pi \models p \quad \ldots \qquad \pi \models \mathbf{G}\, p \quad \ldots \qquad \pi \models \mathbf{X}\, p \quad \ldots \qquad \pi \models r \quad \ldots$
$\pi \models p \,\mathbf{U}\, q \quad \ldots \qquad \pi \models \mathbf{G}\,(p \,\mathbf{U}\, q) \quad \ldots \qquad \pi \models p \,\mathbf{U}\, q' \quad \ldots$
$\pi \models \mathbf{G}\,(p \rightarrow \mathbf{F}\, q') \quad \ldots \qquad \pi \models \mathbf{G}\,(r \rightarrow \mathbf{F}\, p) \quad \ldots \qquad \pi \models \mathbf{G}\,(p \rightarrow \mathbf{X}\, r) \quad \ldots$

Es ist wieder $A \vee B =\!|\!\models \neg A \rightarrow B$ etc., d.h. man kann $\vee$, $\wedge$, $\leftrightarrow$ als Abkürzungen auffassen.

Proposition 6.5 *i)* $\mathbf{G}\, A =\!|\!\models \neg \mathbf{F} \neg A$

ii) $\mathbf{F}\, A =\!|\!\models \mathit{true} \,\mathbf{U}\, A$

iii) $A \,\mathbf{U}\, B \ =\!|\!\models\ \neg(\neg B \,\mathbf{U}\, (\neg A \wedge \neg B)) \wedge \mathbf{F}\, B$

Beweis:

i) $\pi^i \models A$ gilt für alle i gdw. es für kein i falsch ist. $\quad (\forall x A =\!|\!\models \neg \exists x \neg A)$

ii) Übung

iii) $\neg B \,\mathbf{U}\, (\neg A \wedge \neg B)$ besagt, dass B falsch bleibt bis auch noch A falsch ist; mit anderen Worten: $(*)$ bevor B wahr wird (wenn überhaupt), ist A einmal falsch.

Gilt nun $A \,\mathbf{U}\, B$ für einen Ablauf π, ist $(*)$ offenbar falsch und $\mathbf{F}\, B$ gilt. Gelte umgekehrt $\neg(\neg B \,\mathbf{U}\, (\neg A \wedge \neg B)) \wedge \mathbf{F}\, B$, und sei j minimal mit $\pi^j \models B$; da $(*)$ falsch ist, gilt für alle $0 \leq i < j:\ \pi^i \models A$.

□

Man kann also bei Bedarf auch $\mathbf{G}$ und $\mathbf{F}$ als Abkürzungen auffassen. Auf Teil iii) werden wir im nächsten Abschnitt zurückkommen. Man kann wieder Axiome, Regeln, Herleitungen und vollständige Kalküle untersuchen. Meist steht aber das sogenannte *model-checking* im Vordergrund (s.u.); wir beschränken uns deshalb auf semantische Überlegungen.

Satz 6.6 *i)* $\models \mathbf{G}\,(A \to B) \to (\mathbf{G}\,A \to \mathbf{G}\,B)$,

ii) $\models \mathbf{G}\,\mathbf{X}\,A \leftrightarrow \mathbf{X}\,\mathbf{G}\,A$

iii) $\models (A \wedge \mathbf{G}\,(A \to \mathbf{X}\,A)) \to$

iv) $\models \mathbf{X}\,\mathbf{F}\,A \to \mathbf{F}\,A$

Beweis:

i) Bemerkung: $C \to (D \to E) \dashv\models C \wedge D \to E$

Erfüllt jedes π^i $A \to B$ und A, so auch B; d.h. $\pi \models \mathbf{G}\,B$. (vgl. Ax6: $(\forall x A \to B) \to ((\forall x A) \to (\forall x B))$.

ii) Für alle $i \geq 0$ gilt: $\pi^i \models \mathbf{X}\,A$ gdw. für alle $i \geq 0$: $(\pi^i)^1 = \pi^{i+1} = (\pi^1)^i \models A$ gdw. $\pi^1 \models \mathbf{G}\,A$

iii) $\pi \models A \wedge \mathbf{G}\,(A \to \mathbf{X}\,A)$ heißt $\pi^0 \models A$ und für alle $i \in \mathbb{N}$ $(\pi^i \models A \Rightarrow \pi^{i+1} \models A)$. Nach vollständiger Induktion gilt also $\pi^i \models A$ für alle $i \in \mathbb{N}$.

iv) Übung

□

Beispiel: MUTEX: Zwei (oder mehr) Prozesse fordern eine Ressource an (z.B. Zugriff auf Datensatz oder Drucker); wenn sie das tun, gilt (ggf. wiederholt) r_1 bzw. r_2 (vgl. 6.1 iii)). Wenn sie die Ressource bekommen / haben (man sagt: "sie sind in ihrem *kritischen Bereich*"), „erlischt" r_1 bzw. r_2 und es gilt g_1 bzw. g_2, bis sie freigegeben wird.
Anforderungen für Lösung / Scheduler:

α) Nur einer zur Zeit hat die Ressource (MUTEX-Eigenschaft; vgl. 6.1 i))

$\mathbf{G}\,\neg(g_1 \wedge g_2)$

β) Jede Anforderung wird erfüllt (vgl. 6.1 iii))

$\mathbf{G}\,(r_1 \to \mathbf{F}\,g_1)$, $\quad \mathbf{G}\,(r_2 \to \mathbf{F}\,g_2)$

α ist eine typische *Sicherheitseigenschaft* ("Etwas Schlechtes geschieht nie."), wie alle Formeln $\mathbf{G}$ A mit $A \in$ *For* (nicht *TFor*). β ist eine *Lebendigkeitseigenschaft* ("Etwas Gutes geschieht schließlich."); weitere Beispiele sind 6.1 ii) ($\mathbf{G}\,\mathbf{F}\,p$) bzw. „Programm hält (h) schließlich endgültig" $\mathbf{F}\,\mathbf{G}\,h$.

Typisch: Ist α bzw. β zu einem Zeitpunkt nicht wahr, so wird α auch in keiner Fortsetzung wahr; β kann immer noch erfüllt werden.

Satz: *Jede Eigenschaft ist Konjunktion einer Sicherheits- und einer Lebendigkeitseigenschaft.*

Einfache Lösung: Ressource immer wieder abwechselnd vergeben; funktioniert aber nur wenn jeder immer wieder anfordert, also $\mathbf{G}\,\mathbf{F}\,r_1 \wedge \mathbf{G}\,\mathbf{F}\,r_2$; das können wir *nicht* annehmen.

Damit haben wir einige *nützliche Schemata* kennengelernt, d.h. Formeln dieser Bauart sind oft nützlich.

- $\mathbf{G}\,\neg b$: Etwas Schlechtes b geschieht nie.
- $\mathbf{G}\,\mathbf{F}\,p$: Immer wieder geschieht p, was man auch als „unendlich oft p“ lesen kann.
- $\mathbf{F}\,\mathbf{G}\,h$: Schließlich gilt h endgültig, z.B. „das Programm hält (h) schließlich endgültig“.
- $\mathbf{G}\,(r \to \mathbf{F}\,g)$: Jedesmal folgt der Anforderung r schließlich eine positive Antwort g.

Manchmal wird im Gegensatz dazu eine Anforderung auch ignoriert; nur wenn man immer wieder anfordert, wird die Anforderung erfüllt. Wenn dies gewährleistet ist, können wir schreiben: ...

Man kann dies so verstehen, dass man unendlich oft anfordern muss, um eine positive Antwort zu bekommen – was man aber realistisch nicht kann! Tatsächlich muss die Antwort aber nach einiger Zeit kommen, wenn das System die formulierte Eigenschaft hat; dann kann man die Wiederholung der Anforderung einstellen – das *ist* realistisch.

Unterscheidet sich die letzte Formel von $\mathbf{G}\,\mathbf{F}\,r \to \mathbf{F}\,g$?

Und wie ist es mit $\ldots\,\mathbf{G}\,\mathbf{F}\,r \to \mathbf{G}\,\mathbf{F}\,g$?

Anwendung temporaler Logik oft als

Model-Checking

Spezifikation ist eine LTL-Formel A. Man beschreibt alle System-Abläufe der (evtl. geplanten) Implementierung durch eine Struktur K und prüft, ob *alle* Abläufe von K A erfüllen (d.h. *Modelle* von A sind).

Definition 6.7 Eine *Kripke-Struktur* $K = (S, \to, L, s_0)$ besteht aus einer Menge S von *Zuständen* mit *Startzustand* s_0, einer Bewertung $L : S \to \mathfrak{P}(\mathcal{P})$ und einer *Transitionsrelation* $\to \subseteq S \times S$, so daß für alle $s \in S$ ein s' mit $s \to s'$ (d.h. $(s, s') \in \to$) existiert. (vgl. endliche Automaten)

K modelliert z.B. die Abläufe eines interaktiven Programms, wobei ein Zustand durch die Variablenwerte und den Programmzähler (das Restprogramm) gegeben ist.

Ein Ablauf π von K ist eine unendliche Folge von Zuständen beginnend mit s_0 (in der ein s ohne $s \to s'$ nicht auftreten könnte), also $\pi = s_0, s_1, s_2, \ldots$ mit $s_i \to s_{i+1}$ für alle $i \geq 0$. Die Zustände eines solchen Ablaufs heißen *erreichbar*.

K *erfüllt* ein $A \in TFor$, falls für alle Abläufe π von K gilt $\pi \models A$. □

Es gibt viele Ideen (und tools) wie man auch für sehr große (z.T. sogar unendliche) K automatisch prüfen kann, ob K ein A erfüllt.

Als Beispiel für Definition 6.7 diskutieren wir die Modellierung einer MUTEX-Lösung: Eine solche Lösung ist eine Kripke-Struktur, die die Formeln unter α und β (s.o.) erfüllt.

Bem.: Tatsächlich müssen noch weitere Eigenschaften gelten; ist z.B. $\rightarrow = \{(s_0, s_0)\}$ und $L(s_0)$ leer, gelten die gewünschten Formeln – die Prozesse können gar nicht erst anfordern. □

Graphische Konventionen:

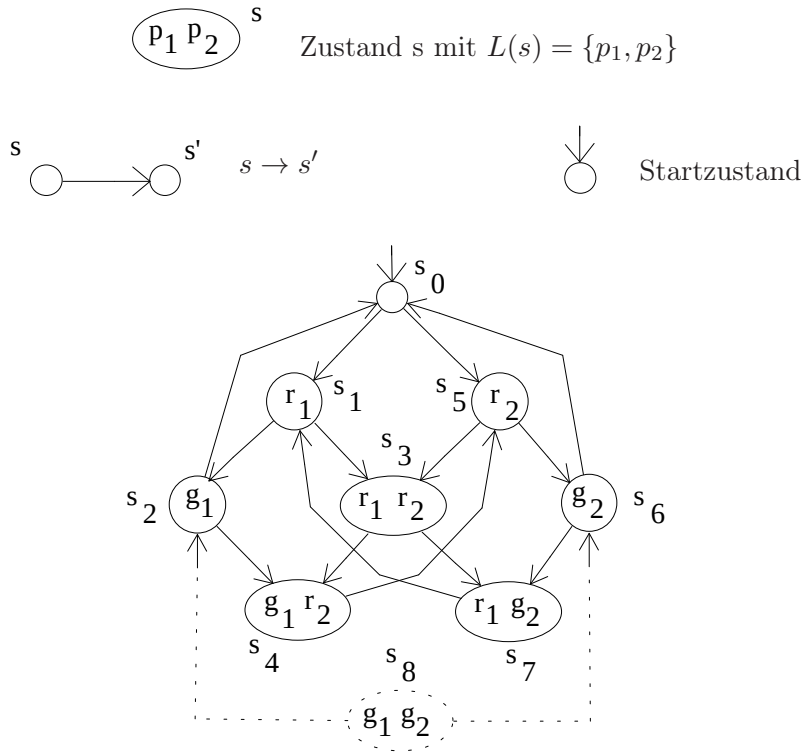


Abbildung 1:

Idee für K_1 (vgl. Abb. 1): Jeder Prozeß kann anfordern, erhält die Ressource – soweit frei – und gibt sie wieder zurück; falls beide Prozesse anfordern (Zustand s_3) würfelt der Scheduler.

In jedem Ablauf von K_1 wird angefordert ($\mathbf{F}\,(r_1 \vee r_2)$); das ist in Wirklichkeit nicht so, also ergänzen wir bei s_0 eine Schleife (vgl. Abb. 2). Die Schleife repräsentiert Aktivitäten der Prozesse, die nichts mit der Ressource zu tun haben. (Bzw.: Schleife als technischer Trick, um im Prinzip doch endliche Abläufe einzubeziehen.)

Inspektion von K_1 zeigt, daß K_1 $\mathbf{G}\,\neg(g_1 \wedge g_2)$ erfüllt.

Bem.: Wird K_1 aus einem Programm erzeugt, liegt es zunächst nicht explizit vor. In einer ersten Phase wird K_1 also systematisch erzeugt – BFS/DFS auf einem noch nicht existenten Graphen. Diese Erreichbarkeitsanalyse zeigt, daß s_8 von s_0 nicht erreichbar ist, also in Abläufen gar nicht auftreten kann. □

Aber: $\pi = s_0, (s_5, s_3, s_4)^\omega \not\models \mathbf{G}\,(r_2 \to \mathbf{F}\,g_2)$, denn $\pi^1 = (s_5, s_3, s_4)^\omega \models r_2$, jedoch $\pi^1 \not\models \mathbf{F}\,g_2$.

Verbesserung (K_2 in Abb. 2): Wer zuerst kommt, mahlt zuerst – aber dann kommt der andere dran.

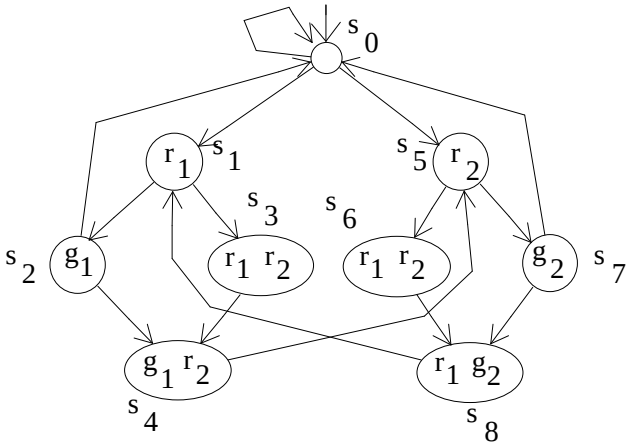


Abbildung 2:

Beachte, daß $L(s_3) = L(s_6)$; L abstrahiert also von (für diese Lösung wesentlichen!) Details, in denen s_3 und s_6 sich unterscheiden. Offenbar erfüllt K_2 die Formel $\mathbf{G}\,\neg(g_1 \wedge g_2)$. Wir stellen fest:

- $K_2[s_2]$ (K_2 mit s_2 als Startzustand) und $K_2[s_4]$ erfüllen $\mathbf{F}\,g_1$.
- Erfüllt $K_2[s']$ $\mathbf{F}\,g_1$ für alle s' mit $s \to s'$, so auch $K_2[s]$; vgl. Satz 6.6 iv).
- Also erfüllen $K_2[s_3]$, damit $K_2[s_1]$, damit $K_2[s_8]$ und schließlich $K_2[s_6]$ $\mathbf{F}\,g_1$.
- Erfüllt ein Ablaufsuffix $\pi'\,r_1$, so beginnt π' mit s_1, s_3, s_6 oder s_8. Daher auch $\pi' \models \mathbf{F}\,g_1$.
- Damit gilt für jeden Ablauf π die Formel $\mathbf{G}\,(r_1 \to \mathbf{F}\,g_1)$.

Also: K_2 erfüllt $\mathbf{G}\,(r_1 \to \mathbf{F}\,g_1)$ und mit Symmetrie auch $\mathbf{G}\,(r_2 \to \mathbf{F}\,g_2)$; K_2 ist also eine Lösung für das MUTEX-Problem.

Typisches Vorgehen: Untersuchung, welche Zustände Teilformeln von A erfüllen. (vgl. Wahrheitstafeln)

Tatsächlich geht man beim LTL-Model-Checking aber anders vor; die Überlegungen hier sind eher eine Vorbereitung für Abschnitt 6.3.

K_2 is stark vereinfacht: Genauso, wie der erste Prozeß im Zustand s_0 beliebig „für sich“ arbeiten kann, kann er das in s_5. Wir müssen also bei s_5 eine Schleife einfügen – und analog in s_7, s_1 und s_2. Dann ergibt sich aber $s_0, (s_5)^\omega \not\models \mathbf{G}\,(r_2 \to \mathbf{F}\,g_2)$. Ist unsere Lösung falsch?

Argument: Dieser Ablauf ist ein Artefakt; 2 *unabhängige* Prozesse (Prozess 1 vs. Scheduler/Prozess 2) wollen stets einen Schritt machen – in der Realität wird schließlich jeder einen machen, so daß s_5 verlassen wird.

Wir wollen also nur Abläufe betrachten, die die *schwache Fairness* (*progress assumption*) erfüllen: Ist ein Prozeß ständig aktiviert (enabled) (z.B. *nicht* der erste Prozeß in s_6, wohl aber der Scheduler in s_5), so macht er einen Schritt (last step von Prozess i). Um dies formal zu erreichen, gibt es 2 Möglichkeiten:

1. Struktur mit Fairness-Bedingung $\mathcal{F} \subseteq S$; (Analogon zu Endzuständen in endlichen Automaten, hier für *unendliche* Abläufe; vgl. Büchi-Automaten)

   Eine unendliche Folge wie bisher ist nur dann ein Ablauf, wenn ein $s \in \mathcal{F}$ unendlich oft durchlaufen wird. Wenn ein fairer Ablauf in den Schlingen bei s_1, s_2, s_5 und s_7 nicht hängenbleiben soll, so wählen wir z.B. $\mathcal{F} = \{s_0, s_4, s_8\}$. Jetzt ist s_0, s_5^ω kein Ablauf mehr, wohl aber $s_0, (s_1, s_2, s_4, s_5, s_6, s_8)^\omega$ und s_0^ω.

2. Fairness als Teil der Spezifikation

   Spezifikation besagt: wenn Ablauf fair, dann $\mathbf{G}\,(r_1 \to \mathbf{F}\,g_1)$ etc.
   Dazu müssen wir in der Kripke-Struktur beschreiben, wer aktiviert ist und wer einen Schritt macht bzw. gemacht hat.

   Wir fügen Atome en_1, en_2 (1 bzw. 2 aktiviert/enabled) und $last_1$, $last_2$ (letzter Schritt kam von 1 bzw. 2) hinzu und verlangen $\mathbf{G}\,(\mathbf{G}\,en_i \to \mathbf{F}\,last_i)$, $i = 1, 2, \ldots$, d.h. die Lebendigkeitseigenschaften heißen jetzt:

   $$\mathbf{G}\,((\mathbf{G}\,en_1 \to \mathbf{F}\,last_1) \wedge (\mathbf{G}\,en_2 \to \mathbf{F}\,last_2)) \to \mathbf{G}\,(r_i \to \mathbf{F}\,g_i), \quad i \in \{1, 2\}$$

   $\Rightarrow$ Aufblähung der Kripke-Struktur, um die Idee von K_2 mit den neuen zusätzlichen Atomen zu modellieren, vgl. Abb. 3.

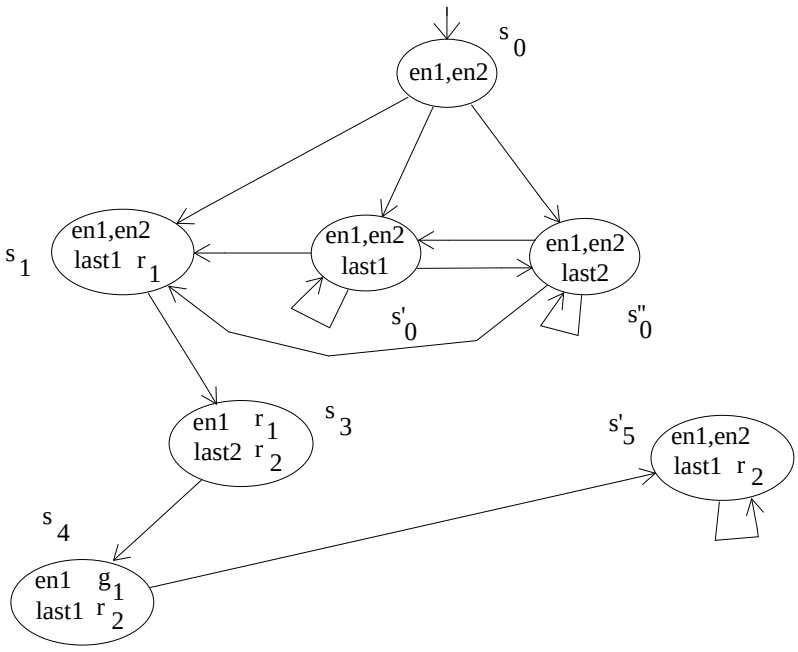


Abbildung 3:

Wir können also Fairness in LTL spezifizieren. Eine interessante Eigenschaft, die sich nicht beschreiben lässt, ist: Terminierung (t) ist stets möglich. Ein System kann diese Eigenschaft

haben, aber trotzdem einen Ablauf, indem t nie eintritt; eine LTL-Formel kann mit dieser Situation wohl nicht umgehen. Genauer: die Kripke-Strukturen in Abb. 4 haben im Prinzip dieselben Abläufe (in der „zustandsfreien“ Notation: $\emptyset^*\{t\}^\omega + \emptyset^\omega$), erfüllen also dieselben LTL-Formeln; nur die erste Kripke-Struktur hat die Eigenschaft.

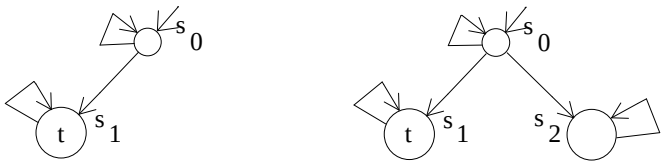


Abbildung 4:

6.2 CTL

Um uns die Grenzen von LTL zu verdeutlichen, können wir eine Kripke-Struktur wie z.B. K_1 (Abbildung 1) zu einem unendlichen Baum (s. Abbildung 5) entfalten. Die Abläufe / Berechnungen von K_1 sind gerade die unendlichen Wege, die von der Wurzel ausgehen; man nennt den Baum daher Berechnungsbaum (computation tree).

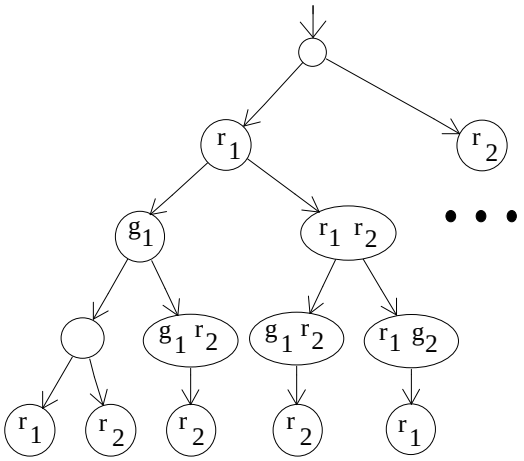


Abbildung 5:

LTL betrachtet alle Wege, nicht aber die Verzweigungsmöglichkeiten; auf dem linkesten Pfad tritt z.B. nie r_2 auf, aber es ist immer unmittelbar möglich (wir können nach rechts abbiegen); auch g_2 ist immer möglich, wenn auch nicht mit nur einem Schritt. LTL kann also nicht über die Existenz von Wegen oder alternativen Fortsetzungen reden.

Beispiel: Eine weitere, beim MUTEX-Problem wichtige Eigenschaft wird als *nicht-blockierend* bezeichnet und lässt sich auch nicht ausdrücken:
Wann immer der erste bzw. zweite Prozess nicht die Ressource anfordert oder hat, kann er sie sofort anfordern; mit anderen Worten: es gibt in solchen Zuständen immer einen Ablauf mit $\mathbf{X}\, r_1$ bzw. $\mathbf{X}\, r_2$.

Nebenbemerkung: In einfachen Fällen lässt sich etwas erreichen: Die Aussage, dass der erste Prozess sofort anfordern kann (r_1 ist initial sofort möglich), lässt sich zwar nicht ausdrücken; das Gegenteil dazu ist aber $\mathbf{X}\neg r_1$. (Auf jedem Weg fordert der erste Prozess nicht sofort an.) Dies ist für K_1 falsch, also gilt die ursprüngliche Aussage. Man beachte: $K_1 \not\models \mathbf{X}\neg r_1$ ist nicht dasselbe wie $K_1 \models \neg\mathbf{X}\neg r_1$!

Auswertung einer LTL-Formel für eine Kripke-Struktur beinhaltet *eine* All-Quantifikation über die Wege, daher können wir indirekt auch *eine* Existenz-Quantifikation behandeln. □

In CTL (Computation Tree Logic) wollen wir über die Existenz von Abläufen reden und auch geschachtelte Quantifikationen erlauben. Dabei wird jede dieser Quantifikationen (**A** bzw. **E**) mit einem der bisherigen temporalen Operatoren **G**, **F**, **X** bzw. **U** kombiniert.

Wie aus Abbildung 5 ersichtlich gilt für K_1 also z.B. $\mathbf{EF}\,(g_1 \wedge r_2)$. Wegen des linkesten Wegs gilt $\mathbf{EG}\,(\neg r_2 \wedge \neg g_2)$, und wie der Baum andeutet gilt $\mathbf{AG}\,\neg(g_1 \wedge g_2)$ (MUTEX-Eigenschaft). Dass immer g_2 möglich ist, kann mit $\mathbf{AG}\,\mathbf{EF}\,g_2$ ausgedrückt werden, nicht-blockierend mit $\mathbf{AG}\,(\neg r_1 \wedge \neg g_1 \rightarrow \mathbf{EX}\, r_1) \wedge \mathbf{AG}\,(\neg r_2 \wedge \neg g_2 \rightarrow \mathbf{EX}\, r_2)$.

Definition 6.8 (Syntax von CTL)
Formeln von CTL (Computation Tree Logic): $CTFor_{\mathcal{P}}(CTFor)$ ist die kleinste Menge mit:

(CT1) $p \in \mathcal{P} \Rightarrow p \in CTFor$

(CT2) $A, B \in CTFor \Rightarrow \neg A,\ A \wedge B,\ A \vee B,\ A \to B,\ A \leftrightarrow B\ \in CTFor$
(Klammerung wie üblich; $\neg$, $\mathbf{A}$, $\mathbf{E}$, $\mathbf{AG}$, $\mathbf{EG}$, $\mathbf{AF}$, $\mathbf{EF}$, und $\mathbf{AX}$, $\mathbf{EX}$ (T3) binden am stärksten, $\mathbf{U}$ (CT4) stärker als die restlichen Operatoren)

(CT3) $B \in CTFor \Rightarrow \mathbf{AG}\,B,\ \mathbf{EG}\,B,\ \mathbf{AF}\,B,\ \mathbf{EF}\,B,\ \mathbf{AX}\,B,\ \mathbf{EX}\,B\ \in CTFor$

(CT4) $B, C \in CTFor \Rightarrow \mathbf{A}\,(B\,\mathbf{U}\,C),\ \mathbf{E}\,(B\,\mathbf{U}\,C) \in CTFor$

(2 Operatoren $\mathbf{A}\,(.\,\mathbf{U}\,.)$ und $\mathbf{E}\,(.\,\mathbf{U}\,.)$) □

Bei CTL sind die Modelle (Interpretationen) direkt Kripke-Strukturen, wobei wir aber die Formel bei einem Zustand auswerten wollen. (vgl. $K[s]$ im letzten Abschnitt)

Definition 6.9 (Semantik von CTL)
Sei K eine Kripke-Struktur, s ein Zustand. $A \in CTFor$ *gilt* für (K, s) ((K, s) *erfüllt* A, $K, s \models A$) ist wie folgt definiert:

(CT1) $K, s \models p$, wenn $p \in L(s)$ (p gilt im Ausgangszustand)

(CT2)
- $K, s \models \neg A$, wenn nicht $K, s \models A$.
- $K, s \models A \vee B$, wenn $K, s \models A$ oder $K, s \models B$ etc.

(CT3)
- $K, s \models \mathbf{AX}\,B$ ($K, s \models \mathbf{EX}\,B$), wenn für alle Abläufe (einen Ablauf) $s'_0(= s), s'_1, \ldots$ gilt $K, s'_1 \models B$.

  Z.B. gilt $K_1, s_0 \models \mathbf{EX}\,r_1$, nicht aber $K_1, s_2 \models \mathbf{EX}\,r_1$ oder $K_1, s_0 \models \mathbf{AX}\,r_1$.

- $K, s \models \mathbf{AG}\,B$ ($K, s \models \mathbf{EG}\,B$), wenn

  Wie oben schon angekündigt gilt also tatsächlich $K_1, s_0 \models \mathbf{AG}\,\neg(g_1 \wedge g_2)$ und $K_1, s_0 \models \mathbf{EG}\,(\neg r_2 \wedge \neg g_2)$, es gilt aber nicht $K_1, s_0 \models \mathbf{AG}\,(\neg r_2 \wedge \neg g_2)$. $\mathbf{AG}$ kann man lesen als „für jeden erreichbaren Zustand gilt“.

  Ferner gilt $K_1, s_0 \models \mathbf{AG}\,(\neg r_1 \wedge \neg g_1 \to \mathbf{EX}\,r_1)$.

  Auf dem linkesten Weg gilt immer: Wenn der erste Prozess anfordert, so erhält er *auf jeden Fall* im nächsten Schritt Zugriff – auch wenn man s_6 entfernt ($\to K'_1$); man könnte daher erwarten, dass $K'_1, s_0 \models \mathbf{EG}\,(r_1 \to \mathbf{AX}\,g_1)$ gilt; dies ist aber falsch, denn jeder r-Zustand hat einen Folgezustand, in dem g_1 nicht gilt.

  Beachte: Beim Auswerten von $r_1 \to \mathbf{AX}\,g_1$ vergessen wir den ursprünglichen Pfad.

- $K, s \models \mathbf{AF}\,B$ ($K, s \models \mathbf{EF}\,B$), wenn

  Wie oben schon angekündigt gilt also tatsächlich $K_1, s_0 \models \mathbf{AG}\,\mathbf{EF}\,g_2$, d.h. für jeden erreichbaren Zustand gilt, dass ein g_2-Zustand erreichbar, also g_2 möglich ist. Ferner gilt (fälschlicherweise) $K_1, s_0 \models \mathbf{AF}\,(r_1 \vee r_2)$ (zumindest ein Prozess wird die Ressource anfordern).

(CT4) $K, s \models \mathbf{A}\,(B\,\mathbf{U}\,C)$ ($K, s \models \mathbf{E}\,(B\,\mathbf{U}\,C)$), wenn

Es gilt $K_1, s_0 \models \mathbf{AG}\,(g_1 \rightarrow \mathbf{A}\,(g_1\,\mathbf{U}\,(\neg g_1 \wedge \neg g_2)))$, d.h. wann immer der erste Prozess die Ressource hat, gilt dies bis die Ressource schießlich ganz frei wird. Achtung: eine Formel wie $\mathbf{A}\,(g_1 \rightarrow (g_1\,\mathbf{U}\,(\neg g_1 \wedge \neg g_2)))$ scheint dasselbe zu besagen, ist aber nicht einmal zulässig. Zudem wird hier nur der Fall behandelt, dass der erste Zustand g_1 erfüllt.

Schließlich schreiben wir $K \models A$ wenn $K, s_0 \models A$ für den Startzustand s_0.

A ist *gültig / erfüllbar*, wenn alle K / ein K A erfüllen. Sollte man hier besser K, s statt K schreiben?
Wie bisher bedeutet *logisch äquivalent*, $=\!|\!\models$: dieselben K erfüllen links wie rechts; entsprechend bedeutet $M \models A$: "A folgt aus M." □

Neben der MUTEX- und der Non-Blocking-Eigenschaft lässt sich auch die entsprechende Lebendigkeitseigenschaft in CTL ausdrücken: $\mathbf{AG}\,(r_1 \rightarrow \mathbf{AF}\,g_1)$. Im Vergleich zu $\mathbf{G}\,(r_1 \rightarrow \mathbf{F}\,g_1$ haben wir einfach ein $\mathbf{A}$ vor die LTL-Operatoren gesetzt.

Dies funktioniert z.B. auch bei $K \models \mathbf{G}\,\mathbf{F}\,p$ (auf allen Wegen gilt p unendlich oft): Wenn $K \models \mathbf{AG}\,\mathbf{AF}\,p$, wird man von jedem erreichbaren Zustand aus ($\mathbf{AG}$) zwangsläufig wieder einen p-Zustand erreichen; in anderen Worten: Egal wie man durch K läuft, man erreicht immer wieder p. Dies ist genau die Bedeutung von $\mathbf{G}\,\mathbf{F}\,p$, beide Formeln werden von denselben K erfüllt. Ferner sind $\mathbf{X}\,\mathbf{F}\,p$ und $\mathbf{AX}\,\mathbf{AF}\,p$ äquivalent. (Egal wie man einen Schritt macht, bei beliebigem Weiterlaufen trifft man p.)

Dies Vorgehen schlägt fehl für $\mathbf{G}\,\mathbf{F}\,p \rightarrow \mathbf{G}\,\mathbf{F}\,q$. Dies sagt, dass q unendlich oft auf einem Weg gilt, der unendlich oft p erfüllt; $\mathbf{AG}\,\mathbf{AF}\,p \rightarrow \mathbf{AG}\,\mathbf{AF}\,q$ hingegen stellt nur eine Forderung, wenn *jeder* Weg unendlich oft p erfüllt. Tatsächlich lässt sich die LTL-Formel gar nicht in CTL ausdrücken. Auch die starke Fairness-Eigenschaft $\mathbf{G}\,\mathbf{F}\,en \rightarrow \mathbf{F}\,done$ lässt sich nicht ausdrücken.

Satz 6.10 *i) In CTL kann die MUTEX- und die entsprechende Lebendigkeits-Eigenschaft ausgedrückt werden.*

ii) Schließliche Terminierung ($\mathbf{AG}\,\mathbf{EF}\,t$) und die Non-Blocking-Eigenschaft des MUTEX-Problems (für beide Prozesse gilt $\mathbf{AG}\,(\neg r \wedge \neg g \rightarrow \mathbf{EX}\,r)$) lassen sich in CTL, nicht aber in LTL ausdrücken.

iii) Zu den LTL-Formeln $\mathbf{F}\,\mathbf{G}\,q$, $\mathbf{G}\,\mathbf{F}\,p \rightarrow \mathbf{G}\,\mathbf{F}\,q$ und $\mathbf{G}\,\mathbf{F}\,en \rightarrow \mathbf{F}\,done$ gibt es keine äquivalenten CTL-Formeln.

Beweis:

i) s.o.

ii) $\mathbf{AG}\,\mathbf{EF}\,t$ haben wir bereits mit Abbildung 4 behandelt. Für $\mathbf{AG}\,(\neg r \wedge \neg g \rightarrow \mathbf{EX}\,r)$ ersetzen wir in der Abbildung t durch r: Dann gilt die Eigenschaft links, aber nicht rechts; LTL-Formeln können die Kripke-Strukturen aber nicht unterscheiden.

iii) Zu $\mathbf{F}\,\mathbf{G}\,q$ wird der Beweis in Baier, Katoen: Principles of Model Checking auf S. 337 skizziert; der Beweis lässt sich für $\mathbf{G}\,\mathbf{F}\,p \rightarrow \mathbf{G}\,\mathbf{F}\,q$ modifizieren, wenn man in jedem Zustand zusätzlich p wahr sein lässt, so dass $\mathbf{G}\,\mathbf{F}\,p$ in jedem Fall gilt.

□

Wir zeigen jetzt u.a. ein Analogon zu Satz 6.6 iii).

Satz 6.11 *i)* $\models (B \wedge \mathbf{AG}\,(B \rightarrow \mathbf{AX}\,B)) \rightarrow \mathbf{AG}\,B$ *(induktiver Beweis von* $\mathbf{AG}\,B$*)*

ii) $\models \mathbf{AX}\,(B \rightarrow C) \wedge \mathbf{AX}\,B \rightarrow \mathbf{AX}\,C$

Beweis: i) Angenommen, $K \models B$ und $K \models \mathbf{AG}\,(B \rightarrow \mathbf{AX}\,B)$, d.h. für jeden erreichbaren Zustand s gilt $K, s \models B \rightarrow \mathbf{AX}\,B$. Sei ferner $s_0, s_1, \ldots$ ein Ablauf. Dann gilt $K, s_0 \models B$ und für alle $i \geq 0$ gilt $K, s_i \models B \rightarrow \mathbf{AX}\,B$, d.h. insbesondere $K, s_i \models B \implies K, s_{i+1} \models B$. Mit Induktion fertig.

ii) Angenommen, $K, s_0 \models \mathbf{AX}\,(B \rightarrow C) \wedge \mathbf{AX}\,B$; dann gilt in jedem Folgezustand $B \rightarrow C$ und B, also auch C mit Modus Ponens. □

Als nächstes wollen wir noch untersuchen, wie man CTL-Operatoren durch andere ausdrücken kann.

Satz 6.12 *i)* $\mathbf{AG}\,B \not\models \neg\mathbf{EF}\,\neg B$; $\mathbf{EG}\,B \not\models \neg\mathbf{AF}\,\neg B$

ii) $\mathbf{EF}\,B \not\models \mathbf{E}\,(\mathit{true}\,\mathbf{U}\,B)$ $\quad$ $\mathbf{AF}\,B \not\models \mathbf{A}\,(\mathit{true}\,\mathbf{U}\,B)$

iii) $\mathbf{AX}\,B \not\models \neg\mathbf{EX}\,\neg B$

iv) $\mathbf{A}\,(B\,\mathbf{U}\,C) \not\models \neg\mathbf{E}\,(\neg C\,\mathbf{U}\,(\neg B \wedge \neg C)) \wedge \mathbf{AF}\,C$

vgl. Proposition 6.5 iii): $B\,\mathbf{U}\,C \not\models \neg(\neg C\,\mathbf{U}\,(\neg B \wedge \neg C)) \wedge \mathbf{F}\,C$

Beweis:

i) B gilt genau dann in allen erreichbaren Zuständen, wenn es nicht zutrifft, dass B in einem erreichbaren Zustand falsch ist. Es gibt genau dann einen Ablauf, in dem B in allen Zuständen gilt, wenn es nicht zutrifft, dass B in jedem Ablauf einmal falsch ist.

ii) Proposition 6.5 ii) gilt für Abläufe: Ein Ablauf (oder alle) erfüllt genau dann irgendwann B, wenn er (oder alle) $\mathit{true}\,\mathbf{U}\,B$ erfüllt.

iii) ähnlich zu i)

iv) Zunächst: (*) Alle Abläufe erfüllen $\neg(\neg C\,\mathbf{U}\,(\neg B \wedge \neg C))$ genau dann, wenn es keinen Ablauf gibt, der $\neg C\,\mathbf{U}\,(\neg B \wedge \neg C)$ erfüllt.

Proposition 6.5 iii) besagt: Wenn $B\,\mathbf{U}\,C$ für alle Abläufe gilt, dann gilt für alle Abläufe $\mathbf{F}\,C$ und auch $\neg(\neg C\,\mathbf{U}\,(\neg B \wedge \neg C))$. Das heißt aber: Es gilt $\mathbf{AF}\,C$ und wegen (*) $\neg\mathbf{E}\,(\neg C\,\mathbf{U}\,(\neg B \wedge \neg C)$.

Andererseits: Wenn es keinen Ablauf gibt, der $\neg C\,\mathbf{U}\,(\neg B \wedge \neg C)$ erfüllt, und wenn jeder Ablauf $\mathbf{F}\,C$ erfüllt, so erfüllt wegen (*) jeder Ablauf $\neg(\neg C\,\mathbf{U}\,(\neg B \wedge \neg C)) \wedge \mathbf{F}\,C$, also $B\,\mathbf{U}\,C$ nach Proposition 6.5 iii).

□

Diese Aussagen zeigen: Wir können wieder $\mathbf{AX}\,B$ als Abkürzung auffassen, d.h. in unseren Untersuchungen auf **AX** verzichten, solange wir **EX** haben (– dies gilt auch umgekehrt).

Wenn wir **E U** haben, können wir einerseits auf **EF** und damit auf **AG** verzichten; andererseits kann man **EG** als auch **A U** mit **AF** ausdrücken (Teil i) und iv)), und jeweils umgekehrt (Teil i) und ii)).

Wir können uns also bei den temporalen Operatoren z.B. auf $\{\mathbf{E\,U}, \mathbf{EX}, \mathbf{AF}\}$ oder auf $\{\mathbf{E\,U}, \mathbf{EX}, \mathbf{EG}\}$ beschränken, d.h. dieses sind Mengen von Basis-Temporaloperatoren.

6.3 CTL-Model-Checking

Gegeben sei eine *endliche Kripke-Struktur* K und eine CTL-Formel B; wir wollen prüfen, ob $K \models B$. Dazu *beschriften* wir die Zustände von K bottom-up *mit Teilformeln* von B, die in den jeweiligen Zuständen gelten.

Zunächst ist jeder Zustand s mit den in B auftretenden Atomen p beschriftet, soweit $p \in L(s)$. Für die Teilformel $C \to D$ nehmen wir an, dass die Beschriftungen für C und D schon erfolgt sind, und beschriften s mit $C \to D$, falls s mit D oder nicht mit C beschriftet ist. Analog prüfen wir für $\neg C$, ob s nicht mit C beschriftet ist.

Diese Beschriftungsphasen benötigen Zeit linear in $|S|$. Wir werden sicher auch die Kanten des Graphen K berücksichtigen müssen. Unser Ziel ist, jede Phase in linearer Zeit in der Größe des Graphen (Zahl der Kanten plus Zustände, also $|\ S \cup \to\ |$) durchzuführen.

Am Ende wissen wir dann für jeden Zustand s, also insbesondere für s_0, ob $K, s \models B$, wobei der *Gesamtaufwand linear* in der Größe des Graphen und der Formel ($|B|$) war, d.h. $O(|\ S \cup \to\ | \cdot |B|)$.

Es fehlt uns noch die Behandlung der temporalen Operatoren. Wir verwenden zunächst nur $\{\mathbf{E\,U}, \mathbf{EX}, \mathbf{AF}\}$, ersetzen dann aber $\mathbf{AF}$ durch $\mathbf{EG}$. Da die Ersetzung von anderen Operatoren die Formel vergrößern, gilt unsere Aufwandsanalyse zunächst nur für Formeln mit den Operatoren $\{\mathbf{E\,U}, \mathbf{EX}, \mathbf{EG}\}$; für eine Verallgemeinerung müssen dann doch alle temporalen Operatoren behandelt werden.

$\mathbf{EX}\,C$: Wir beschriften s mit $\mathbf{EX}\,C$, wenn ein Folgezustand s' (also $s \to s'$) mit C beschriftet ist.

Dazu müssen wir von jedem Zustand aus schlimmstenfalls alle ausgehenden Kanten durchmustern, also linear.

Bsp.: Mit $\mathbf{EX}\,r_1$ werden in Abb. 8 s_0, s_1, s_5, s_6, s_7 und s_8 beschriftet. Daher werden die Zustände mit $\mathbf{EX\,EX}\,r_1$ beschriftet, die eine Kante zu diesen Zuständen haben, also s_0, s_2, s_4, s_5, s_6, s_7 und s_8. K_2 erfüllt also beide Formeln.

Alle neuen Beschriftungen sind korrekt; alle nötigen neuen Beschriftungen werden gemacht (Vollständigkeit).

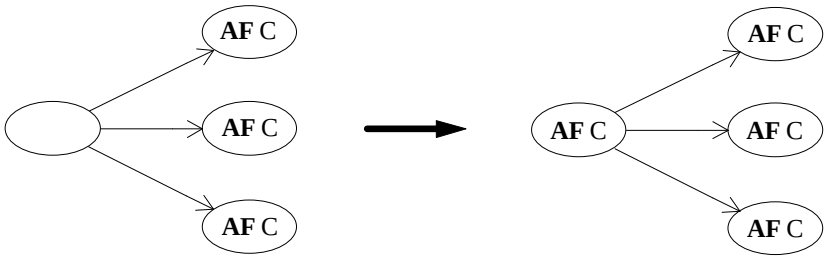


Abbildung 6:

$\mathbf{AF}\,C$: Beschrifte zunächst alle s mit $\mathbf{AF}\,C$, wenn s mit C beschriftet.

Beschrifte dann *iterativ* jedes s mit $\mathbf{AF}\,C$, falls alle Folgezustände s' mit $\mathbf{AF}\,C$ beschriftet; stop, wenn kein solches s mehr existiert. s.Abb. 6

Bps.: Für $\mathbf{AF}\,g_1$ und Abb. 8 haben wir dies schon in Abschnitt 6.1 gemacht: Zunächst

werden s_2 und s_4 beschriftet, dann in dieser Reihenfolge s_3, s_1, s_8 und s_6. K_2 erfüllt also diese Formel nicht.

Man kann die Menge der neu beschrifteten Zustände als kleinsten Fixpunkt sehen. Hier ist der Aufwand nicht offensichtlich.

Alle neuen Beschriftungen sind korrekt; für die Vollständigkeit (mit Induktion): gilt $\mathbf{AF}\, C$, so ist die maximale Weglänge bis zu einem C-Zustand $< |S|$. Initial werden alle Zustände mit maximaler Weglänge 0 beschriftet. Dann zeigt man den Induktionsschritt: Sind alle Zustände mit maximaler Weglänge n beschriftet, so werden auch alle Zustände mit maximaler Weglänge $n+1$ beschriftet; ist nämlich die maximale Weglänge $n+1$, so haben alle Folgezustände maximale Weglänge $\leq n$, sind also schon beschriftet.

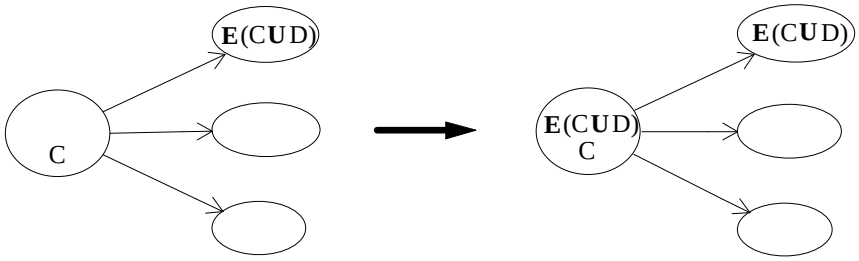


Abbildung 7:

$\mathbf{E}\,(C\,\mathbf{U}\,D)$: Beschrifte zunächst alle s mit $\mathbf{E}\,(C\,\mathbf{U}\,D)$, wenn s mit D beschriftet.

Beschrifte dann iterativ jedes s mit $\mathbf{E}\,(C\,\mathbf{U}\,D)$, falls es mit C und mindestens ein Folgezustand s' mit $\mathbf{E}\,(C\,\mathbf{U}\,D)$ beschriftet; stop, wenn kein solches s mehr existiert.

Dies kann man als Breiten(- oder Tiefen)suche anlegen, wenn man zunächst alle Kanten umdreht, bei den direkt mit D beschrifteten Zuständen beginnt (d.h. diese kommen in die Warteschlange oder auf den Stack) und Zustände, die nicht mit C beschriftet sind, als bereits besucht behandelt, d.h. ignoriert. Also linearer Aufwand.

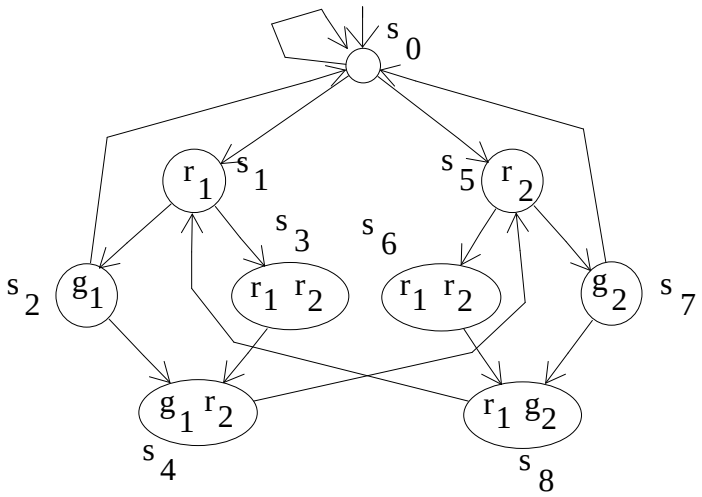


Abbildung 8:

Bsp.: Für $\mathbf{E}\,(\neg g_2\,\mathbf{U}\,g_1)$ und Abb. 8 müsste man zunächst noch $\neg g_2$ eintragen; dann werden initial s_2 und s_4 beschriftet. Von diesen können wir rückwärts s_1 und s_3 (in beliebiger Rei-

henfolge) erreichen, die $\neg g_2$ tragen und daher auch beschriftet werden. Von diesen können wir rückwärts als neue Zustände s_0 (wird beschriftet) und s_8 (gilt wegen g_2 als schon besucht) erreichen; schließlich können wir von s_0 rückwärts noch s_7 erreichen, das wieder wegen g_2 als schon besucht gilt. K_2 erfüllt diese Formel.

Alle neuen Beschriftungen sind korrekt; für die Vollständigkeit (mit Induktion): gilt $\mathbf{E}\,(C\,\mathbf{U}\,D)$, so ist die kürzeste Weglänge via C-Zuständen bis zu einem D-Zustand $< |S|$. Initial werden alle Zustände mit Weglänge 0 beschriftet. Dann zeigt man den Induktionsschritt: Sind alle Zustände mit Weglänge n beschriftet, so werden auch alle Zustände mit Weglänge $n+1$ beschriftet; ist nämlich die Weglänge $n+1$, so hat ein Folgezustand Weglänge $\leq n$, ist also schon beschriftet.

EG C: Dies ist dual zu **AF** C; daher neue Idee: beschrifte zunächst alle Zustände mit **EG** C. Dann lösche diese Beschriftung, wenn keine Beschriftung C vorliegt.

Lösche dann iterativ **EG** C bei s, wenn kein Folgezustand mit **EG** C beschriftet.

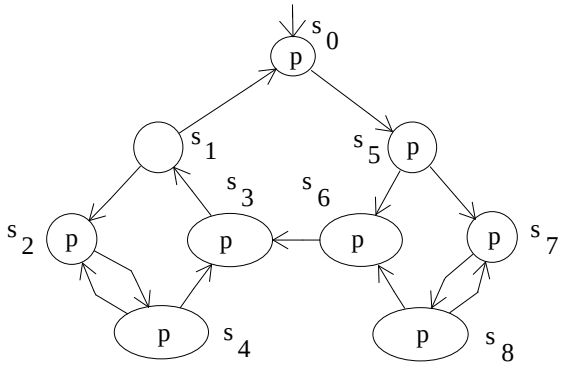


Abbildung 9:

Bsp.: Für K in Abbildung 9 und **EG** p beschriftet man zunächst alle Zustände und löscht dann die Beschriftung von s_1. Jetzt hat s_3 keinen beschrifteten Folgezustand; löschen; dann dasselbe für s_6; dies ist ein stabiler Zustand. K erfüllt diese Formel.

Man kann die Menge der neu beschrifteten Zustände als größten Fixpunkt sehen. Aufwand wieder nicht offensichtlich; wir brauchen also noch eine neue Idee:

Ein unendlicher Weg zu $\mathbf{EG}\,C$ muss sich wiederholen; wichtig sind also Kreise, auf denen jeder Zustand C erfüllt – wie $s_2 s_4 s_2$ und $s_7 s_8 s_7$ im obigen Beispiel.

Kreise (evtl. mehrere zusammen) bilden sogenannte *starke Zusammenhangskomponenten (SCC)*; darunter verstehen wir eine maximale Menge von Ecken in einem Graphen (bzw. Zustände in einer Kripke-Struktur), bei der es von jeder Ecke zu jeder Ecke einen Weg gibt. (*Hier* soll eine Ecke allein nur dann eine SCC bilden, wenn sie eine Schlinge hat.) Alle SCCs können in linearer Zeit mit einer modifizierten Tiefensuche bestimmt werden.

Jeder Ablauf zu $\mathbf{EG}\,C$ endet in einer C-SCC; daher: Man *entfernt* in K zunächst alle Zustände, die nicht mit C beschriftet sind. Dann bestimmt man die SCCs; in Abb. 9:

Jeder Zustand in einer SCC kann offensichtlich innerhalb der SCC einen Ablauf verfolgen, der $\mathbf{EG}\,C$ wahr macht. Ausgehend von diesen SCC bestimmt man mit Breitensuche und umgedrehten Kanten alle erreichbaren Zustände, d.h. gerade die, die mit einem Weg aus C-beschrifteten Zuständen eine SCC erreichen können; vgl. Abb. 9. Die Zustände der SCCs und die durch die Breitensuche erreichten Zustände werden in linearer Zeit mit $\mathbf{EG}\,C$ beschriftet.

7 Einführung in die modale Logik

Wie wir schon im letzten Kapitel gesehen haben, können Aussagen nicht nur einfach wahr oder falsch sein, sie können auf verschiedene Arten (Modi) wahr sein wie jetzt, immer oder irgendwann. In der modalen Logik betrachtet man darüber hinaus, dass man *weiß* oder *glaubt*, dass eine Aussage wahr ist, oder dass eine Aussage wahr sein *sollte* oder *muss*. Temporale Logik ist damit ein detailliert ausgearbeiteter Zweig der modalen Logik.

Um verschiedene temporale Formen von Wahrheit auszudrücken, haben wir in Kapitel 6 verschiedene temporale Operatoren mit verschiedenen Semantiken eingeführt und diese sinnvollerweise auch unterschiedlich bezeichnet; so bedeuten $K \models \mathbf{G}\,(B \rightarrow \mathbf{F}\,C)$, $K \models \mathbf{AG}\,(B \rightarrow \mathbf{EF}\,C)$ und $K \models \mathbf{EG}\,(B \rightarrow \mathbf{EF}\,C)$ verschiedenes.

In der modalen Logik gibt es hingegen (im Prinzip) nur zwei modale Operatoren $\Box$ und $\Diamond$, die zueinander dual sind (d.h. $\Diamond A$ kann durch $\neg\Box\neg A$ definiert werden, dann sind auch $\Box A$ und $\neg\Diamond\neg A$ äquivalent) und auch in LTL anstelle von $\mathbf{G}$ und $\mathbf{F}$ verwendet werden. Diese Operatoren sollen verschiedenes bedeuten, ihre formale Semantik (wieder auf Kripke-Strukturen gestützt) ist aber einheitlich: Um die verschiedenen Bedeutungen sinnvoll widerzuspiegeln, werden stattdessen die zulässigen Kripke-Strukturen geeignet eingeschränkt. Mögliche Bedeutungen sind u.a.:

$\Box A$	$\Diamond A$
Agent Q weiß, dass A	soweit Q weiß, könnte A wahr sein (Q weiß nicht, dass A falsch ist)
A muss wahr sein	A könnte wahr sein (A ist nicht notwendigerweise falsch)
Agent Q glaubt, dass A	A ist glaubhaft (A widerspricht nicht Q's Überzeugungen)
A sollte wahr sein	A ist zulässig

Wissen und Notwendigkeit (z.B. aufgrund bekannter physikalischer Gesetze) werden in gleicher Weise behandelt. Der Unterschied zwischen Glauben und Wissen ist, dass man nur zutreffende Aussagen wissen kann. Bei „sollte“ geht es um moralische oder politische Überzeugungen.

In allen Fällen geht es darum, dass es Aussagen gibt (z.B. „Der griechische Philosoph Zeno hat die Geschichte von Achill und der Schildkröte erfunden“), deren Wahrheit oder Falschheit man nicht weiß, glaubt oder für geboten hält; formal: es gibt A, so dass

$$\Box A \vee \Box\neg A$$

nicht zutrifft. Wir nehmen aber idealisierend an, dass alle Agenten und Moralisten perfekte Logiker sind: Agent Q weiß alle Tautologien, er weiß auch alle logischen Konsequenzen aus seinem Wissen. Z.B. soll auf jeden Fall gelten $\Box(A \rightarrow B) \wedge \Box A \rightarrow \Box B$.

Definition 7.1 (Syntax und Semantik der modalen Logik)
Formeln bilden die kleinste Menge $MFor_{\mathcal{P}}(MFor)$ mit:

(M1) $p \in \mathcal{P} \Rightarrow p \in MFor$

(M2) $A, B \in \mathit{MFor} \Rightarrow \neg A,\ A \wedge B,\ A \vee B,\ A \to B,\ A \leftrightarrow B\ \in \mathit{MFor}$
(Klammerung wie üblich; $\neg$, $\Box$ und $\Diamond$ (T3) binden am stärksten)

(M3) $A \in \mathit{MFor} \Rightarrow \Box A,\ \Diamond A\ \in \mathit{MFor}$

Für eine Kripke-Struktur K mit Zustand s und $A \in \mathit{MFor}$ wird $K, s \models A$ wie in Definition 6.9 definiert, wobei $\Box$ wie **AX** und $\Diamond$ wie **EX** behandelt wird. Abweichend wird aber $K \models A$ definiert als $K, s \models A$ für alle Zustände s.

Die Zustände werden in diesem Kontext mögliche Welten (engl.: possible worlds) genannt; Folgezustände von s werden von s zugängliche oder (hier) auch für s *nächste Welten* genannt. □

Zunächst sehen wir, dass in der Tat immer gilt (s. Satz 6.11 ii))

$$\Box(A \to B) \wedge \Box A \to \Box B$$

Wir haben ferner:

$$\models \Box(A \wedge B) \leftrightarrow \Box A \wedge \Box B \qquad \text{und} \qquad \models \Diamond(A \vee B) \leftrightarrow \Diamond A \vee \Diamond B$$

Die Idee einer Kripke-Struktur bei der Modellierung von Agenten-Wissen ist hier: Die nächsten Welten zu s reflektieren genau, was Agent Q glaubt oder weiß. Es sind genau die Welten die mittels der Bewertung L alles wahr machen, was Q glaubt oder weiß. Weiß Q z.B., dass $p \vee q$ für die einzigen Atome p und q gilt, so gibt es drei nächste Welten mit Bewertungen $\{p, q\}$, $\{p\}$ und $\{q\}$.

Analog sind bei der „Moral-Modellierung“ die nächsten Welten die besseren Welten, die alle wünschenswerten Aussagen wahr machen. Sollte also in einer Welt $p \vee q$ gelten, gibt es wieder dieselben drei nächsten oder besseren Welten.

Oft wird in der modalen Logik nicht wie in Kapitel 6 verlangt, dass die Kripke-Strukturen „verklemmungsfrei“ sind, d.h. es darf Welten ohne eine nächste Welt geben. Dies ist z.B. angemessen, wenn die nächsten Welten zu s den Ergebnissen eines nicht-deterministischen Programms nach dessen Terminierung entsprechen: Hat s keine nächsten Welten, so heißt das gerade, dass das Programm im Zustand s gestartet auf keinen Fall terminiert. Eine solche Situation hat aber überraschende Effekte:

$\Diamond \mathit{true}$ kann falsch sein,

weil es einfach keine nächste Welt gibt, also auch keine, die *true* wahr macht. Das heißt aber gerade, dass *true* möglicherweise nicht wahr ist. Analog kann $\Box \mathit{false}$ wahr sein (es gibt keine nächste Welt, die dies widerlegt); dies besagt gewissermaßen, dass man weiß, dass *false* gilt. Übrigens:

$$K, s \models \Diamond \mathit{true} \Leftrightarrow K, s \models \Box A \to \Diamond A \text{ für alle } A \in \mathit{MFor}$$

Es ist klar, dass man nicht weiß oder glaubt, dass *false* gilt, und *false* sollte auch nicht wahr sein. In diesen drei Modi müssen die Kripke-Strukturen also „verklemmungsfrei“ sein; dies ist bereits ein Fall, in dem die Wünsche bei der Modellierung Konsequenzen für die betrachteten Strukturen haben.

Logic Engineering

Wir wollen nun anhand von drei Formeln untersuchen, was in den drei oben erwähnten Modi gelten sollte: Die Fragen sind jeweils, ob für alle Formeln A gelten soll $\Box A \to A$, $\Box A \to \Box\Box A$ und $\Diamond A \to \Box\Diamond A$. $\Box\Box A$ erscheint evtl. merkwürdig, aber zumindest bei verschiedenen Agenten ist so etwas im praktischen Leben schon wichtig; z.B. kann es in Verhandlungen evtl. entscheidend sein zu wissen, was die Gegenseite weiß.

$\Box$ heißt	Q weiß	Q glaubt	sollte gelten
$\Box A \to A$	$\surd$	–	–
$\Box A \to \Box\Box A$	$\surd$	$\surd$	–
$\Diamond A \to \Box\Diamond A$	$\surd$	$\surd$	–

Die erste Formel besagt, dass auch in s gilt, was in allen nächsten Welten gilt. Wenn man etwas weiß, so gilt es in der Tat auch in der Realität; im Fall von „glauben“ ist das nicht so. (Vielleicht glauben Sie, die Geschichte von Achill und der Schildkröte sei von Heraklit ; logisch ist dagegen nichts einzuwenden.) Und was wahr sein sollte, ist es deswegen noch lange nicht in der Wirklichkeit.

Angesichts der großen Geisteskräfte, die wir Q zubilligen, erscheint plausibel: Wenn Q etwas weiß/glaubt oder für möglich hält, dann weiß/glaubt er auch, dass er dies weiß/glaubt oder für möglich hält. Dies wird positive bzw. negative Introspektion genannt. ($\Diamond A$ besagt ja, dass Q *nicht* weiß/glaubt, dass A falsch ist.) Für „sollen“ erscheint dies weniger plausibel. Wenn A zulässig ist ($\Diamond A$), erscheint eine Welt, in der A verboten ist, zumindest eine Möglichkeit zu sein ($\neg\Box\Diamond A$, was ja $\Diamond\Box\neg A$ bedeutet).

Übrigens wurde schon argumentiert, die Modellierung des Sollens mittels besserer Welten sei widersprüchlich. Folgende Aussagen scheinen durchaus zusammen Sinn zu machen:

(FK) Fred ist eine Krimineller.

(FnK) Fred sollte kein Krimineller sein.

(KS) Kriminelle sollen bestraft werden.

(KnS) Nicht-Kriminelle sollen nicht bestraft werden.

(1) Wegen (FK) und (KS) soll Fred bestraft werden, d.h. in jeder besseren Welt wird Fred bestraft. (2) In jeder besseren Welt ist Fred aber gar kein Krimineller; (3) dass er dort bestraft wird ist also falsch, die vier Aussagen sind laut Modellierung widersprüchlich.

Als eine Anwendung von modaler Logik wollen wir die Situation analysieren. Für (1) formalisieren wir (FK) als fk und (KS) als $fk \to \Box fs$, was in der Tat zu $\Box fs$ führt. Die beiden anderen Sätze (angewandt auf Fred) formalisieren wir als (2) $\Box\neg fk$ und $\Box(\neg fk \to \neg fs)$, d.h. $\Box\neg fk \wedge (\neg fk \to \neg fs)$) und daher (3) $\Box\neg fs$. Also $\Box fs$ und $\Box\neg fs$, d.h. $\Box(fs \wedge \neg fs)$; in jeder nächsten Welt gilt *false*, was nicht sein kann.

Bei genauer Beobachtung sieht man aber: (KS) und (KnS) haben dieselbe Form, nämlich „Wenn etwas gilt, soll etwas anderes gelten“. Das wird im ersten Fall als $. \to \Box.$, im zweiten

als $\Box(. \to .)$. Das ist inkonsistent, ein rhetorischer Trick, um die „die schlechtere Sache zur besseren zu machen “.

Wenn man das gleiche Schema für die analogen Sätze nimmt, verschwindet der Widerspruch. Plausibel ist wohl: In jeder besseren Welt gilt, dass ein (Nicht-)Krimineller dort (nicht) bestraft wird. (Alternativ: Wenn jemand in der Realität (nicht) kriminell ist, wird er in jeder besseren Welt (nicht) bestraft – egal, ob er dort kriminell ist oder nicht.

Damit formalisieren wir (KS) durch $\Box(fk \to fs)$. Dies und die Formeln fk, $\Box\neg fk$ und $\Box(\neg fk \to \neg fs)$ gelten für die Realität s, wenn dort fk gilt und in der einzigen besseren Welt zu s $\neg fk$ und $\neg fs$ (und damit eben auch $fk \to fs$).

Nun wollen wir untersuchen, welche Konsequenzen sich aus obiger Tabelle ergeben.

Definition 7.2 Ein *Rahmen* $F = (S, \to)$ besteht aus einer Menge von Welten und einer Transitionsrelation.

Er erfüllt eine modale Formel A genau dann, wenn jede Kripke-Struktur $K = (S, \to, L, s_0)$ mit beliebigem $L : S \to \mathfrak{P}(\mathcal{P})$ und (irrelevant) $s_0 \in S$ A erfüllt.

Eine Relation $\to$ heißt *euklidisch*, wenn aus $s \to s'$ und $s \to s''$ stets $s' \to s''$ folgt. Diese Eigenschaft hat sehr starke Konsequenzen. □

Beispiel: Der Rahmen in Abbildung 10 erfüllt $\Box A \to A$ für jede modale Formel A: Wenn $\Box A$ in einer Welt s gilt, so gilt A in jeder nächsten Welt; s ist wegen der Schlinge aber selbst eine nächste Welt; also gilt A auch in s.

Entscheidend ist also, dass die Transitionsrelation dieses Rahmens reflexiv ist. Sie ist nicht transitiv und auch nicht euklidisch, denn $s_1 \to s_2$ und $s_1 \to s_1$, aber nicht $s_2 \to s_1$. Man sieht: Ist die Relation reflexiv und euklidisch, muss sie auch symmetrisch sein.

Der Rahmen erfüllt nicht $\Box p \to \Box\Box p$: Bewerten wir genau s_1 und s_2 mit p, so gilt $\Box p$ in s_1, aber nicht in der nächsten Welt s_2; also gilt in s_1 nicht $\Box\Box p$.

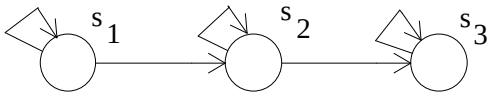


Abbildung 10:

Satz 7.3 *Für jeden Rahmen F und Atom p sind bei den folgenden Punkten jeweils die drei Aussagen äquivlent:*

i) – $\to$ *ist reflexiv.*
  – F *erfüllt* $\Box A \to A$ *für jede modale Formel* A.
  – F *erfüllt* $\Box p \to p$.

ii) – $\to$ *ist transitiv.*
  – F *erfüllt* $\Box A \to \Box\Box A$ *für jede modale Formel* A.
  – F *erfüllt* $\Box p \to \Box\Box p$.

iii) – $\rightarrow$ *ist euklidisch.*

– F *erfüllt* $\Diamond A \rightarrow \Box\Diamond A$ *für jede modale Formel* A.

– F *erfüllt* $\Diamond p \rightarrow \Box\Diamond p$.

Beweis: i) Die erste Folgerung haben wir gerade schon gesehen, die zweite ist in allen Fällen offensichtlich.

Für die dritte nehmen wir an, dass nicht $s \rightarrow s$. Wir bewerten alle Welten außer s mit p , so dass offenbar $K, s \models \Box p$ für das resultierende K; dann müsste aber $K, s \models p$ gelten, ein Widerspruch.

ii) Die erste Folgerung: Sei K eine Kripke-Struktur zu F und s eine Welt mit $\Box A$; sei s' eine beliebige nächste Welt und s'' eine beliebige nächste Welt zu s'. Nach Transitivität ist s'' auch eine nächste Welt zu s und erfüllt daher A. Damit erfüllt das beliebige s' auch $\Box A$ und s erfüllt $\Box\Box A$.

Die dritte Folgerung: Angenommen $s \rightarrow s'$ und $s' \rightarrow s''$, aber nicht $s \rightarrow s''$. Wir bewerten alle Welten außer s'' mit p, so dass offenbar $K, s \models \Box p$ für das resultierende K, nicht aber $K, s' \models \Box p$; damit gilt auch nicht $K, s \models \Box\Box p$, ein Widerspruch.

iii) Die erste Folgerung: Sei K eine Kripke-Struktur zu F und s eine Welt mit $\Diamond A$; es gibt also eine nächste Welt s'', die A erfüllt. Sei s' eine beliebige nächste Welt zu s; nach Voraussetzung ist dann s'' eine nächste Welt zu s'. Also gilt $K, s' \models \Diamond A$ und daher $K, s \models \Box\Diamond A$.

Die dritte Folgerung: Üb. □

Unsere Design-Entscheidungen in obiger Tabelle haben für die Moral-Modellierung keine weiteren Einschränkungen an die Kripke-Strukturen zur Folge. Bei der Glaubensmodellierung müssen alle Kripke-Strukturen einen transitiven und euklidischen Rahmen haben.

Bei der Wissensmodellierung müssen alle Kripke-Strukturen einen reflexiven, transitiven und euklidischen Rahmen haben. Wie wir bereits gesehen haben, impliziert reflexiv und euklidisch symmetrisch, die Rahmen müssen also Äquivalenzrelationen sein, wobei solche auch immer euklidisch sind.

Weiß Q z.B., dass $p \vee q$ für die einzigen Atome p und q gilt, so gibt es tatsächlich drei voll verbundene Welten mit Bewertungen $\{p, q\}$, $\{p\}$ und $\{q\}$, wobei also jede Welt für jede Welt eine mögliche nächste Welt ist. Da nicht in allen Welten p gilt, gilt überall $\neg\Box p$, also auch $\Box\neg\Box p$.

Tatsächlich kann man bei der Wissensmodellierung jede Folge von Negationen und modalen Operatoren äquivalent reduzieren auf die leere Folge, $\Box$, $\neg\Box$, $\Diamond$, $\neg\Diamond$ oder $\neg$. Z.B. bedeutet $K, s \models \Box\neg\Box p$, dass jede nächste Welt von s wiederum eine nächste Welt hat, die p verletzt; da diese auch eine nächste Welt von s ist, folgt $K, s \models \neg\Box p$, mit obigem bedeuten also $\neg\Box p$ und $\Box\neg\Box p$ dasselbe.

Im richtigen Leben ist es natürlich wichtig, dass sich Wissen verändern kann – darum sind Sie an der Universität! Wenn Q oben erfährt, dass p wahr ist, fällt die $\{q\}$-Welt weg; auch in der Struktur sehen wir jetzt, wie gewünscht, $\Box p$. Man muss jetzt vorsichtig sein: positive Einsichten wie „es gilt $\Box(p \vee q)$“ bleiben erhalten, negierte Einsichten wie „es gilt $\neg\Box p$“ können natürlich wegfallen – aber eben auch eine nicht-negierte Wissensaussage wie $\Box\neg\Box p$.

Anwendung auf Wissen in Multi-Agenten-Systemen

Kripke-Strukturen, die Äquivalenz-Relationen entsprechen, sind eigentlich recht langweilig. Interessanter wird die Sache, wenn wir mehrere Agenten haben: Während die Frage, ob Agent Q weiß, dass er weiß, dass A gilt, etwas esoterisch erscheint, ist es praktisch sehr relevant, ob Agentin R weiß, das Q weiß, dass A. (Anwendung beim Handeln, Kryptographie)

Entsprechend müssen jetzt die Pfeile in Kripke-Strukturen durchnumeriert werden. $\Box_1 A$ bedeutet, dass alle 1-nächsten (über einen 1-Pfeil erreichbaren) Welten, also alle für Agent 1 vorstellbaren Welten A erfüllen; $\Box_2\Box_1 A$ heißt, dass alle 2-nächsten Welten $\Box_1 A$ erfüllen, d.h. Agent 2 weiß, dass Agent 1 A weiß. Die i-Pfeile müssen bei der Wissensmodellierung jeweils eine Äquivalenzrelation bilden. Hier ein Anwendungsbeispiel.

3 weise Männer stehen mit je einem gelben oder roten Hut auf dem Kopf im Kreis; sie können also die Hutfarben der anderen, nicht aber ihre eigene sehen – (allerdings könnte der 3. Mann sogar blind sein). Man teilt ihnen mit, dass mindestens ein Hut rot ist. Der 1. wird gefragt, ob er die Farbe seines Huts weiß; er verneint. Der 2. wird gefragt, ob er die Farbe seines Huts weiß; er verneint. Jetzt weiß der 3. seine Farbe; wieso und welche ist es?

Ohne die Mitteilung wäre die Erkenntnis nicht möglich. Verblüffend ist dabei, dass es sehr wohl sein kann, dass ...

Dann weiß also auch ohne Mitteilung jeder, dass mindestens ein Hut rot ist (wenn der 3. nicht blind ist); er weiß sogar, dass jeder andere das weiß. Entscheidend ist, dass der 3. weiß, dass der 2. weiß, dass der 1. weiß, dass mindestens ein Hut rot ist!

Wir wollen diese Schlussfolgerung in unserer Wissenslogik semantisch nachvollziehen (für ein formales, syntaktisches Vorgehen s. Huth, Ryan), der Einfachheit halber mit nur 2 Männern; r_1 soll bedeuten, dass der 1. Hut, r_2, dass der 2. Hut rot ist. Aufgrund der Mitteilung ist $r_1 \vee r_2$ allgemein bekannt, insbesondere gilt $\Box_2\Box_1(r_1 \vee r_2)$. Ferner kann der 1. die andere Hutfarbe sehen, insbesondere gilt $\neg r_2 \to \Box_1\neg r_2$, und der 2. weiß dies: $\Box_2(\neg r_2 \to \Box_1\neg r_2)$. Nach der ersten Frage wird zudem offensichtlich, dass $\neg\Box_1 r_1$, also auch $\Box_2\neg\Box_1 r_1$. Dies bedeutet, wie wir wissen:

$$\Box_2(\Box_1(r_1 \vee r_2) \wedge (\neg r_2 \to \Box_1\neg r_2) \wedge \neg\Box_1 r_1)$$

In jeder 2-nächsten Welt, in der $\neg r_2$ gilt, wäre also $\Box_1(r_1 \vee r_2) \wedge \Box_1\neg r_2$, d.h. $\Box_1((r_1 \vee r_2) \wedge \neg r_2)$ und $\Box_1 r_1$, aber auch $\neg\Box_1 r_1$, was ein Widerspruch wäre. Da also in jeder 2-nächsten Welt r_2 gilt, haben wir $\Box_2 r_2$; zudem gilt r_2 in der Realität, welche immer das ist.