Prädikatenlogische Syntax (Forts.)

(2) Außerlogisches Alphabet (Signatur) $\langle PS, KS, FS \rangle$

- Prädikatensymbole (PS): bezeichnen Eigenschaften und Relationen
- Konstantensymbole (KS):
  zur Bezeichnung bestimmter Gegenstände/Individuen
- Funktionssymbole (FS): zur Bezeichnung von Funktionen

Funktions- und Prädikatensymbole haben jeweils eine fixe Stelligkeit (Anzahl der benötigten Argumente)
mehrstellige Prädikatensymbole heißen auch Relationssymbole

FS_n / PS_n ... Menge der n-stelligen Funktions- / Prädikatensymbole

(Konstantensymbole sind auch als 0-stellige Funktionssymbole auffassbar)

Aussagenlogische Variable: als 0-stellige Prädikatensymbole auffassbar!

(3) Hilfssymbole: Klammern '(' und ')'

(4) Individuenvariablensymbole (kurz Variable): $IVS = \{x, y, z, x_1, \dots\}$

Terme

Folgende induktive Definition besagt, dass Terme mit Hilfe von Funktionssymbolen aus Konstanten und Variablen aufgebaut werden:

Definition (Terme über Σ)

Gegeben eine Signatur $\Sigma = \langle PS, KS, FS \rangle$, ist $\mathcal{T}_\Sigma$ (Menge der Terme über Σ) die kleinste Menge, für die gilt:

(T1) $IVS \subseteq \mathcal{T}_\Sigma$ (Variablen sind Terme)

(T2) $KS \subseteq \mathcal{T}_\Sigma$ (Konstantensymbole sind Terme)

(T3) $f(t_1, \dots, t_n) \in \mathcal{T}_\Sigma$ (allgemeiner Term)
wenn $f \in FS_n$ und $t_1, \dots, t_n \in \mathcal{T}_\Sigma$

Beachte: Prädikatensymbole sind für Terme irrelevant!

Terme können als Bäume verstanden werden:

- die Blätter sind Variable oder Konstantensymbole
- an den inneren Knoten stehen Funktionssymbole

Prädikatenlogische Formeln (PL-Formeln)

Definition (PL-Formeln $\mathcal{PF}_\Sigma$ über einer Signatur Σ)

Gegeben eine Signatur $\Sigma = \langle PS, KS, FS \rangle$ ist $\mathcal{PF}_\Sigma$ (Menge der prädikatenlogischen Formeln über Σ) die kleinste Menge, für die gilt:

(P1) $\bot, \top \in \mathcal{PF}_\Sigma$ (*falsum* und *verum* sind atomare Formeln)

(P2) $p(t_1, \dots, t_n) \in \mathcal{PF}_\Sigma$ (allgemeine) atomare Formel
wenn $p \in PS_n$ und $t_1, \dots, t_n \in \mathcal{T}_\Sigma$ (Argumente: Terme über Σ)

(P3) $s = t \in \mathcal{PF}_\Sigma$ wenn $s, t \in \mathcal{T}_\Sigma$ (atomare) Gleichung

(P4) $\neg F \in \mathcal{PF}_\Sigma$ wenn $F \in \mathcal{PF}_\Sigma$ Negation

(P5) $(F \wedge G) \in \mathcal{PF}_\Sigma$ wenn $F, G \in \mathcal{PF}_\Sigma$ Konjunktion

(P6) $(F \vee G) \in \mathcal{PF}_\Sigma$ wenn $F, G \in \mathcal{PF}_\Sigma$ Disjunktion

(P7) $(F \supset G) \in \mathcal{PF}_\Sigma$ wenn $F, G \in \mathcal{PF}_\Sigma$ Implikation

(P8) $\forall v\, F \in \mathcal{PF}_\Sigma$ wenn $v \in IVS$ und $F \in \mathcal{PF}_\Sigma$ All-Formel

(P9) $\exists v\, F \in \mathcal{PF}_\Sigma$ wenn $v \in IVS$ und $F \in \mathcal{PF}_\Sigma$ Existenz-Formel

In $\forall vF / \exists vF$ heißt v die durch $\forall v / \exists v$ in F gebundene Variable

Variablensubstitution

Notation:

$F(x/t)$ bezeichnet die Formel, die aus F entsteht, indem alle freien (und nur die freien) Vorkommen von x in F durch den Term t ersetzt werden.

Beispiele zur Variablensubstitution

Sei $F = \forall x[P(x, y) \supset Q(x, y)] \wedge R(x)$ und $t = f(y)$, dann gilt:

- $F(x/t) = \forall x[P(x, y) \supset Q(x, y)] \wedge R(f(y))$
- $F(y/t) = \forall x[P(x, f(y)) \supset Q(x, f(y))] \wedge R(x)$
- $F(x/t)(y/t) = \forall x[P(x, f(y)) \supset Q(x, f(y))] \wedge R(f(f(y)))$
- $F(y/t)(x/t) = \forall x[P(x, f(y)) \supset Q(x, f(y))] \wedge R(f(y))$

Aussagenlogische Formeln (AL-Formeln)

AL-Formeln als Spezialfall von PL-Formeln:

Definition (AL-Formeln $\mathcal{AF}$)

Gegeben aussagenlogische Variable $AV = PS_0$ ist $\mathcal{AF}$ (Menge der aussagenlogische Formeln) die kleinste Menge, für die gilt:

(P1) $\bot, \top \in \mathcal{AF}$ (*falsum* und *verum* sind atomare Formeln/Atome)

(P2) $p \in \mathcal{AF}$ wenn $p \in AV$ (allgemeine) atomare Formel

(P3)

(P4) $\neg F \in \mathcal{AF}$ wenn $F \in \mathcal{AF}$ Negation

(P5) $(F \wedge G) \in \mathcal{AF}$ wenn $F, G \in \mathcal{AF}$ Konjunktion

(P6) $(F \vee G) \in \mathcal{AF}$ wenn $F, G \in \mathcal{AF}$ Disjunktion

(P7) $(F \supset G) \in \mathcal{AF}$ wenn $F, G \in \mathcal{AF}$ Implikation

(P8)

(P9)

Auswertung von $\underline{(}\underline{(}\underline{2}\underline{+}\underline{1}\underline{)}\underline{+}\underline{3}\underline{)}$:

$$\begin{aligned}\mathcal{M}(\underline{(}\underline{(}\underline{2}\underline{+}\underline{1}\underline{)}\underline{+}\underline{3}\underline{)}) &= \mathcal{M}(\underline{(}\underline{2}\underline{+}\underline{1}\underline{)}) + \mathcal{M}(\underline{3}) \\ &= \mathcal{M}(\underline{2}) + \mathcal{M}(\underline{1}) + \mathcal{M}(\underline{3}) \\ &= 2 + 1 + 3 = 6\end{aligned}$$

Das Beispiel ist in mehrfacher Hinsicht noch zu einfach:

- Die Definition von $\mathcal{M}$ ist nur wegen der Klammern eindeutig. Allerdings bliebe – wegen der speziellen Eigenschaften der Addition – hier auch das Ergebnis einer "nicht-deterministischen Auswertung" eindeutig. Das ist im Allgemeinen nicht so.
  Vergleiche: $\underline{(}\underline{(}\underline{2}\underline{\hat{}}\underline{1}\underline{)}\underline{\hat{}}\underline{3}\underline{)}$ versus $\underline{(}\underline{2}\underline{\hat{}}\underline{(}\underline{1}\underline{\hat{}}\underline{3}\underline{)}\underline{)}$
- Die Erweiterung der Sprache auf mehrstellige Zahlen oder um weitere Operatoren verlangt nach Hierarchisierung. (Diese Methode kann auch das Problem der Mehrdeutigkeit lösen.)
- Brauchbare Definitionen von Termen berücksichtigen auch Variablen.

Behandlung von Variablen

Ergänzung unseres einfachen Beispiels additiver Terme:

Syntax: $T \to \underline{\mathrm{x}} \mid \underline{\mathrm{y}} \mid \underline{\mathrm{z}} \mid \cdots$

Semantik: $\mathcal{M}(\underline{\mathrm{x}}) = 1$ oder $\mathcal{M}(\underline{\mathrm{x}}) = 2$ oder ...?

Speicherbelegung/Variablenbelegung $I : \{\underline{\mathrm{x}}, \underline{\mathrm{y}}, \underline{\mathrm{z}}, \ldots\} \to \omega$

ENV ... Menge aller Speicherbelegungen („Environments")

Erweiterung der *Meaning-Function* (Semantik): $\mathcal{M} : ENV \times \mathcal{T} \to \omega$

$$\begin{aligned}&\mathcal{M}(I, \underline{0}) = 0, \ldots, \mathcal{M}(I, \underline{9}) = 9 \\ &\mathcal{M}(I, v) = I(v) \text{ für } v \in \{\underline{\mathrm{x}}, \underline{\mathrm{y}}, \underline{\mathrm{z}}, \ldots\} \\ &\mathcal{M}(I, \underline{(}\, t_1 \underline{+} t_2 \,\underline{)}) = \mathcal{M}(I, t_1) + \mathcal{M}(I, t_2) \quad \text{für } t_1, t_2 \in \mathcal{T}\end{aligned}$$

- Variablenbelegung (Environments) sind ein universelles Konzept: Egal ob arithmetische Ausdrücke, Terme bezogen auf andere Datentypen, Schleifenbedingungen, prädikatenlogische Terme und Formeln, Anweisungen, ganze Programme oder Korrektheitsaussagen: In jedem Fall ist die Bedeutung dieser Sprachbausteine immer nur relativ zu Variablenbelegungen festgelegt.

Auswertung von $\underline{(}\underline{(}\underline{\mathrm{x}}\underline{+}\underline{2}\underline{)}\underline{+}\underline{\mathrm{y}}\underline{)}$

... für $I(\underline{\mathrm{x}}) = 1$, $I(v) = 3$ sonst, also für alle $v \neq \underline{\mathrm{x}}$:

$$\begin{aligned}\mathcal{M}(I, \underline{(}\underline{(}\underline{\mathrm{x}}\underline{+}\underline{2}\underline{)}\underline{+}\underline{\mathrm{y}}\underline{)}) &= \mathcal{M}(I, \underline{(}\underline{\mathrm{x}}\underline{+}\underline{2}\underline{)}) + \mathcal{M}(I, \underline{\mathrm{y}}) \\ &= \mathcal{M}(I, \underline{\mathrm{x}}) + \mathcal{M}(I, \underline{2}) + \mathcal{M}(I, \underline{\mathrm{y}}) \\ &= I(\underline{\mathrm{x}}) + 2 + I(\underline{\mathrm{y}}) \\ &= 1 + 2 + 3 = 6\end{aligned}$$

... für $J(\underline{\mathrm{x}}) = 5$, $J(\underline{\mathrm{y}}) = 2$, $J(v) = 0$ sonst:

$$\begin{aligned}\mathcal{M}(J, \underline{(}\underline{(}\underline{\mathrm{x}}\underline{+}\underline{2}\underline{)}\underline{+}\underline{\mathrm{y}}\underline{)}) &= \mathcal{M}(J, \underline{(}\underline{\mathrm{x}}\underline{+}\underline{2}\underline{)}) + \mathcal{M}(J, \underline{\mathrm{y}}) \\ &= \mathcal{M}(J, \underline{\mathrm{x}}) + \mathcal{M}(J, \underline{2}) + \mathcal{M}(J, \underline{\mathrm{y}}) \\ &= J(\underline{\mathrm{x}}) + 2 + J(\underline{\mathrm{y}}) \\ &= 5 + 2 + 2 = 9\end{aligned}$$

Hierarchisierung (Forts.)

Semantik:

$$\mathcal{M}_Z(\underline{0}) = 0, \ldots, \mathcal{M}_Z(\underline{9}) = 9$$
$$\mathcal{M}_N(nz) = \mathcal{M}_N(n) \cdot 10 + \mathcal{M}_Z(z) \text{ für } n \in \mathrm{N},\ z \in \mathrm{Z}$$
$$\text{bzw. } \mathcal{M}_N(n) = M_Z(n) \text{ falls } n \in \mathrm{Z}$$
$$\mathcal{M}_T(\underline{(}\, t_1 \underline{+} t_2 \,\underline{)}) = \mathcal{M}_T(t_1) + \mathcal{M}_T(t_2) \text{ für } t_1, t_2 \in \mathrm{T}$$
$$\text{bzw. } \mathcal{M}_T(t) = \mathcal{M}_N(t) \text{ falls } t \in \mathrm{N}$$

Alternativ kann man bei einer einzigen Semantikfunktion $\mathcal{M}$ bleiben, da die Form der Terme die relevante Bedingung eindeutig bestimmt:

$$\mathcal{M}(\underline{0}) = 0, \ldots, \mathcal{M}(\underline{9}) = 9$$
$$\mathcal{M}(nz) = \mathcal{M}(n) \cdot 10 + \mathcal{M}(z) \text{ für } n \in \mathrm{N},\ z \in \mathrm{Z}$$
$$\mathcal{M}(\underline{(}\, t_1 \underline{+} t_2 \,\underline{)}) = \mathcal{M}(t_1) + \mathcal{M}(t_2) \text{ für } t_1, t_2 \in \mathrm{T}$$

Wichtige Beobachtung:
Die Erweiterung mit Variablen ist genau wie oben: $\mathcal{M} : ENV \times \mathcal{T} \to \omega$.

Grammatik vs. Induktive Definition

Grammatik $\mathcal{T} = \mathcal{L}(G)$	Induktive Definition $\mathcal{T}$ ist kl. Menge mit:	Rekursion $\mathcal{M} : ENV \times \mathcal{T} \to \omega$
$T \to \underline{0} \mid \cdots \mid \underline{9}$	$\{\underline{0}, \ldots \underline{9}\} \subseteq \mathcal{T}$	$\mathcal{M}(I, \underline{0}) = 0, \ldots$
$T \to \underline{\mathrm{x}} \mid \underline{\mathrm{y}} \mid \underline{\mathrm{z}}$	$\{\underline{\mathrm{x}}, \underline{\mathrm{y}}, \underline{\mathrm{z}}\} \subseteq \mathcal{T}$	$\mathcal{M}(I, \underline{\mathrm{x}}) = I(\underline{\mathrm{x}}), \ldots$
$T \to \underline{(}\, T \underline{+} T \,\underline{)}$	$t_1, t_2 \in \mathcal{T}$	$\mathcal{M}(I, \underline{(}\, t_1 \underline{+} t_2 \,\underline{)})$
	$\to \underline{(}\, t_1 \underline{+} t_2 \,\underline{)} \in \mathcal{T}$	$= \mathcal{M}(I, t_1) + \mathcal{M}(I, t_2)$

Beachte:
Die verschiedenen Vorkommen von T in $\underline{(}\, T \underline{+} T \,\underline{)}$ entsprechen den verschiedenen (metasprachlichen) Variablen t_1 und t_2!
$t \in \mathcal{T} \to \underline{(}\, t \underline{+} t \,\underline{)} \in \mathcal{T}$ statt $t_1, t_2 \in \mathcal{T} \to \underline{(}\, t_1 \underline{+} t_2 \,\underline{)} \in \mathcal{T}$:
$\mathcal{T}$ ist nicht mehr kontextfrei!

Empfohlene Aufgaben (zur Selbstkontrolle):

1. Beweisen Sie die letzte Behauptung.
2. Was ändert sich, wenn man die Klammern weglässt? ($t_1 + t_2 \in \mathcal{T}$ bzw. $t + t \in \mathcal{T}$)
3. Was ändert sich, wenn man $\underline{+}$ durch $\hat{}$ ersetzt syntaktisch bzw. semantisch? (Intendierte Semantik beachten!)

Grammatik $\mathcal{E} = \mathcal{L}(G)$	Induktive Definition $\mathcal{E}$ ist kl. Menge mit:	Rekursion $\mathcal{M} : \mathcal{E} \to \omega$
$E \to \underline{0} \mid \cdots \mid \underline{9}$	$\{\underline{0}, \ldots \underline{9}\} \subseteq \mathcal{E}$	$\mathcal{M}(\underline{0}) = 0, \ldots$
$E \to E \underline{+} E$	$e_1, e_2 \in \mathcal{E}$	$\mathcal{M}(e_1 \underline{+} e_2)$
	$\to e_1 \underline{+} e_2 \in \mathcal{E}$	$= \mathcal{M}(e_1) + \mathcal{M}(e_2)$
$E \to E \underline{*} E$	$e_1, e_2 \in \mathcal{E}$	$\mathcal{M}(e_1 \underline{*} e_2)$
	$\to e_1 \underline{*} e_2 \in \mathcal{E}$	$= \mathcal{M}(e_1) \cdot \mathcal{M}(e_2)$

Problem:
$\mathcal{M}$ ist keine Funktion mehr! (Vergleiche "+" vs. "^", Folie 6)
$\mathcal{M}(\underline{3} \underline{+} \underline{3} \underline{*} \underline{3}) = \mathcal{M}(\underline{3}) + \mathcal{M}(\underline{3} \underline{*} \underline{3}) = \mathcal{M}(\underline{3}) + \mathcal{M}(\underline{3}) \cdot \mathcal{M}(\underline{3}) = 12$
$\mathcal{M}(\underline{3} \underline{+} \underline{3} \underline{*} \underline{3}) = \mathcal{M}(\underline{3} \underline{+} \underline{3}) \cdot \mathcal{M}(\underline{3}) = (\mathcal{M}(\underline{3}) + \mathcal{M}(\underline{3})) \cdot \mathcal{M}(\underline{3}) = 18$

Grund:
Grammatik für $\mathcal{E}$ ist mehrdeutig; verschiedene Ableitungen führen zu verschiedenen Werten.

Beachte:
Als Sprache ist $\mathcal{E}$ regulär und eindeutig, wie folgende Produktionen zeigen:

$$E \to \underline{0}\, F \mid \cdots \mid \underline{9}\, F \qquad F \to \underline{+}\, E \mid \underline{*}\, E \mid \epsilon$$

Diese Grammatik ist aber in Hinblick auf die Semantik inadäquat!

Hierarchische Definition von $\mathcal{E}$

$\mathcal{E}$ ist kleinste Menge mit:	$\mathcal{H}$ ist kleinste Menge mit:
$\mathcal{H} \subseteq \mathcal{E}$	$\{\underline{0}, \underline{1}, \ldots, \underline{9}\} \subseteq \mathcal{H}$
$e_1, e_2 \in \mathcal{E} \ \to\ e_1 \underline{+} e_2 \in \mathcal{E}$	$e_1, e_2 \in \mathcal{H} \ \to\ e_1 \underline{*} e_2 \in \mathcal{H}$

Daraus lassen sich folgende Definitionen $\mathcal{M}_\mathcal{E}$ und $\mathcal{M}_\mathcal{H}$ ableiten:

$\mathcal{M}_\mathcal{E}(e) = \mathcal{M}_\mathcal{H}(e)$ falls $e \in \mathcal{H}$	$\mathcal{M}_\mathcal{H}(\underline{0}) = 0, \ldots, \mathcal{M}_\mathcal{H}(\underline{9}) = 9$
$\mathcal{M}_\mathcal{E}(e_1 \underline{+} e_2) = \mathcal{M}_\mathcal{E}(e_1) + \mathcal{M}_\mathcal{E}(e_2)$	$\mathcal{M}_\mathcal{H}(e_1 \underline{*} e_2) = \mathcal{M}_\mathcal{H}(e_1) \cdot \mathcal{M}_\mathcal{H}(e_2)$

Als einzigen Wert von $\underline{2} \underline{*} \underline{3} \underline{+} \underline{4}$ erhält man nun:

$$\begin{aligned} \mathcal{M}_\mathcal{E}(\underline{2} \underline{*} \underline{3} \underline{+} \underline{4}) &= \mathcal{M}_\mathcal{E}(\underline{2} \underline{*} \underline{3}) + \mathcal{M}_\mathcal{E}(\underline{4}) \\ &= \mathcal{M}_\mathcal{H}(\underline{2} \underline{*} \underline{3}) + \mathcal{M}_\mathcal{H}(\underline{4}) \\ &= \mathcal{M}_\mathcal{H}(\underline{2}) \cdot \mathcal{M}_\mathcal{H}(\underline{3}) + 4 \\ &= 2 \cdot 3 + 4 = 10 \end{aligned}$$

Modellstrukturen

Motivation:
Wir brauchen ein möglichst allgemeines und einfaches semantisches Konzept, das festlegt worauf sich Termsprachen, Bedingungen, Zusicherungen, Formeln, Programme, etc. beziehen sollen.

Definition (Modellstruktur)

Eine Modellstruktur ist ein Quadruppel $\mathcal{D} = \langle D; P, K, F \rangle$, wobei:

D ...	nicht-leere Menge – Gegenstandsbereich/Domäne)	
P ...	Menge von Prädikaten über D	$D^n \to \{\mathbf{f}, \mathbf{t}\}$
K ...	Menge von Konstanten in D	$K \subseteq D$
F ...	Menge totaler Funktionen über D	$D^n \to D$

f ('false') und **t** ('true') sind Wahrheitswerte

Modellstrukturen können als abstrakte Datentypen aufgefasst werden.

Achtung: Funktionen und Prädikate müssen total sein.
D darf nicht leer sein; aber es muss keine Konstanten geben.

Beispiele von Modellstrukturen

Beachte: Für Zahlenbegriffe identifiziert man oft Menge und Struktur.
Im Folgenden ist aber größere Genauigkeit gefordert!
Daher: Wir verwenden ab nun konsequent die Bezeichnungen $Z = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ und $\omega = \{0, 1, 2, \ldots\}$ für die Mengen und bezeichnen mit $\mathbb{Z}$ bzw. $\mathbb{N}$ die jeweiligen Modellstrukturen.

Ganze Zahlen:

$\mathbb{Z} = \langle Z;\ \{<\}, Z,\ \{+, -, *\} \rangle$

$<$ ... „kleiner"-Relation

$+, -, *$ ... übliche arithmetische Grundfunktionen

Beachte: jede ganze Zahl ist auch als Konstante verfügbar

Natürliche Zahlen:

$\mathbb{N} = \langle \omega;\ \{<\},\ \{0, 1\},\ \{+, \dot{-}, *\} \rangle$

$+, *, <$ wie für $\mathbb{Z}$, aber

$$x \dot{-} y = \begin{cases} x - y & \text{für } x \geq y \\ 0 & \text{für } x < y \end{cases}$$

Beachte: Nur 0 und 1 sind als Konstanten verfügbar!

Beispiele von Modellstrukturen (Forts.)

Binäre Stäcks

$\mathbb{S} = \langle S;\ \{\text{ist0?}, \text{ist1?}, \text{istleer?}\},\ \{\varepsilon\},\ \{\text{push0}, \text{push1}, \text{pop}\} \rangle$
wobei die Stacks durch Worte aus $S = \{\underline{0}, \underline{1}\}^*$ dargestellt werden.

$$\begin{aligned}
\text{push0}(s) &= \underline{0}s \\
\text{push1}(s) &= \underline{1}s \\
\text{pop}(s) &= \begin{cases} \varepsilon & \text{für } s = \varepsilon \\ s' & \text{für } s = \underline{0}s' \text{ oder } s = \underline{1}s' \end{cases} \\
\text{istleer?}(s) &\iff s = \varepsilon \\
\text{ist0?}(s) &\iff \text{es gibt ein } s' \text{ sodass } s = \underline{0}s' \\
\text{ist1?}(s) &\iff \text{es gibt ein } s' \text{ sodass } s = \underline{1}s'
\end{aligned}$$

Beobachtung: Die Domäne (hier $\{\underline{0}, \underline{1}\}^*$) bestimmt nicht die Struktur!
Andere Datentypen/Modellstrukturen über der selben Domäne wären z.B. Binärzahlen oder binäre Strings.

Beispiele von Modellstrukturen (Forts.)

Familie X:

FamX $= \langle D_X; P_X, K_X, F_X \rangle$,
wobei

D_X = Personen$_X$ (Menge der Mitglieder der Familie X)
$P_X = \{$*Geschwister*, *Onkel*, *weiblich*, *männlich* $\}$
$K_X = \{\text{Abdul}, \text{Berta}, \text{Chris}, \text{Dorlan}, \text{Ege}, \ldots\}$
$F_X = \{$*Vater*, *Mutter*$\}$

Zur Erinnerung: Funktionen *Vater* und *Mutter* müssen total sein.
$\Longrightarrow$
(1) Menge Personen$_X$ muss unendlich sein, oder
(2) Personen$_X$ enthält Element "Unbekannt", wobei
Vater(Unbekannt) = *Mutter*(Unbekannt) = Unbekannt

Signaturen zu Modellstrukturen

Definition (Signatur zu $\mathcal{D}$)

Die Signatur $\Sigma(\mathcal{D})$ zur Modellstruktur $\mathcal{D}$ legt das Alphabet einer Sprache fest, die den Bezug auf die Gegenstände in $\mathcal{D}$ zulässt:

$KS(\mathcal{D})$, $FS(\mathcal{D})$, $PS(\mathcal{D})$, ... Konstanten,- Funktions-, Prädikatensymbole
$FS_n(\mathcal{D})$, $PS_n(\mathcal{D})$ ... n-stellige Funktions- bzw. Prädikatensymbole

Ganze Zahlen – Signatur $\Sigma(\mathbb{Z})$:
$PS_2(\mathbb{Z}) = \{\underline{\leq}\}$, $PS_n(\mathbb{Z}) = \{\}$ für $n \neq 2$
$KS(\mathbb{Z}) = \{\ldots, \underline{-2}, \underline{-1}, \underline{0}, \underline{1}, \underline{2}, \ldots\}$
$FS_2(\mathbb{Z}) = \{\underline{+}, \underline{-}, \underline{*}\}$, $FS_n(\mathbb{Z}) = \{\}$ für $n \neq 2$
Beachte: $\underline{-1095}$ ist hier ein einzelnes Zeichen

Natürliche Zahlen – Signatur $\Sigma(\mathbb{N})$:
$PS_2(\mathbb{N}) = \{\underline{\leq}\}$, $PS_n(\mathbb{N}) = \{\}$ für $n \neq 2$ $KS(\mathbb{N}) = \{\underline{0}, \underline{1}\}$
$FS_2(\mathbb{N}) = \{\underline{+}, \underline{\dot{-}}, \underline{*}\}$, $FS_n(\mathbb{N}) = \{\}$ für $n \neq 2$
Beachte: $\underline{2}$ ist kein Zeichen und auch kein Term oder Wort über $\Sigma(\mathbb{N})$!

Signaturen (Forts.)

Binäre Stacks – Signatur $\Sigma(\mathbb{S})$:
$PS_1(\mathbb{S}) = \{\underline{\text{ist0?}}, \underline{\text{ist1?}}, \underline{\text{istleer?}}\}$
$PS_n(\mathbb{S}) = \{\}$ für $n > 1$
$KS(\mathbb{S}) = \{\underline{\varepsilon}\}$
$FS_1(\mathbb{S}) = \{\underline{\text{push0}}, \underline{\text{push1}}, \underline{\text{pop}}\}$,
$FS_n(\mathbb{S}) = \{\}$ für $n > 1$
Beachte: $\underline{\text{push0}}$, $\underline{\text{ist0?}}$ etc. sind einzelne Symbole
Im Gegensatz zu ε (Leerwort) ist $\underline{\varepsilon}$ (Name des leeren Stacks) ein Symbol!

Familie X – Signatur $\Sigma(\mathbf{FamX})$:
$PS_1(\mathbf{FamX}) = \{\underline{\textit{weiblich}}, \underline{\textit{männlich}}\}$,
$PS_2(\mathbf{FamX}) = \{\underline{\textit{Geschwister}}, \underline{\textit{Onkel}}\}$,
$PS_n(\mathbf{FamX}) = \{\}$ für $n > 2$
$KS(\mathbf{FamX}) = \{\underline{\text{Ahmed}}, \underline{\text{Berta}}, \underline{\text{Chris}}, \underline{\text{Dorlan}}, \underline{\text{Ege}}, \ldots\}$
$FS_1(\mathbf{FamX}) = \{\underline{\textit{Vater}}, \underline{\textit{Mutter}}\}$, $FS_n(\mathbf{FamX}) = \{\}$ für $n > 1$

Terme über Modellstrukturen

Syntax:
Alphabet: neben $KS(\mathcal{D})$ und $FS(\mathcal{D})$ auch
Individuenvariablensymbole IVS: $\{\underline{x}, \underline{y}, \underline{z}, \ldots, \underline{x_1}, \ldots\}$ (kurz: Variablen)
Hilfssymbole: "$\underline{(}$" und "$\underline{)}$" und "$\underline{,}$"

Definition (Terme über $\mathcal{D}$)

Die Sprache der Terme $\mathcal{T}(\mathcal{D})$ über $\mathcal{D}$ ist die kleinste Menge, für die gilt:

(T1) $IVS \subseteq \mathcal{T}(\mathcal{D})$
(T2) $KS(\mathcal{D}) \subseteq \mathcal{T}(\mathcal{D})$
(T3) $f'\underline{(}\, t_1 \underline{,} \ldots \underline{,} t_n \,\underline{)} \in \mathcal{T}(\mathcal{D})$
wenn $f' \in FS_n(\mathcal{D})$ und $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{D})$

Empfohlene Übungsaufgaben:
- Warum ist z.B. "$\underline{1} \underline{+} \underline{1}$" kein Term über $\mathbb{N}$ oder $\mathbb{Z}$?
- Geben Sie für konkrete Modellstrukturen $\mathcal{D}$ Grammatiken zu $\mathcal{T}(\mathcal{D})$ an.
- Beweisen Sie, dass $\mathcal{T}(\mathcal{D})$ i.a. nicht regulär ist.
- Unter welchen Bedingungen ist $\mathcal{T}(\mathcal{D})$ regulär?

Terme über Modellstrukturen (Beispiele zur Syntax)

Aufgabe:
Zeigen Sie, dass $\underline{\text{push0}}\underline{(}\underline{\text{push1}}\underline{(}\underline{\text{pop}}\underline{(}\underline{x}\underline{)}\underline{)}\underline{)} \in \mathcal{T}(\mathbb{S})$:

Lösung:
- $\underline{x} \in IVS \overset{T1}{\Longrightarrow} \underline{x} \in \mathcal{T}(\mathbb{S})$
- $\underline{\text{pop}} \in FS_1(\mathbb{S})$, $\underline{x} \in \mathcal{T}(\mathbb{S}) \overset{T3}{\Longrightarrow} \underline{\text{pop}}\underline{(}\underline{x}\underline{)} \in \mathcal{T}(\mathbb{S})$
- $\underline{\text{push1}} \in FS_1(\mathbb{S})$, $\underline{\text{pop}}\underline{(}\underline{x}\underline{)} \in \mathcal{T}(\mathbb{S}) \overset{T3}{\Longrightarrow} \underline{\text{push1}}\underline{(}\underline{\text{pop}}\underline{(}\underline{x}\underline{)}\underline{)} \in \mathcal{T}(\mathbb{S})$
- $\underline{\text{push0}} \in FS_1(\mathbb{S})$, $\underline{\text{push1}}\underline{(}\underline{\text{pop}}\underline{(}\underline{x}\underline{)}\underline{)} \in \mathcal{T}(\mathbb{S}) \overset{T3}{\Longrightarrow}$
$\underline{\text{push0}}\underline{(}\underline{\text{push1}}\underline{(}\underline{\text{pop}}\underline{(}\underline{x}\underline{)}\underline{)}\underline{)} \in \mathcal{T}(\mathbb{S})$

Aufgabe:
Zeigen Sie, dass der Ausdruck $\underline{\text{push0}}\underline{(}\underline{\text{push1}}\underline{(}\underline{1}\,\underline{0}\underline{)}\underline{)}$ kein Element von $\mathcal{T}(\mathbb{S})$ ist, also kein Term über den binären Stacks ist.

Lösung:
$\underline{1}$ und $\underline{0}$ sind weder Konstantensymbole noch Variablen.

Terme über Modellstrukturen (Semantik)

Semantik

$ENV(\mathcal{D})$ ... Menge aller Abbildungen $IVS \to D$

Variablenbelegungen, Environments, auch: Interpretationen

Definition

$\mathcal{M}_\mathcal{T} : ENV(\mathcal{D}) \times \mathcal{T}(\mathcal{D}) \to D$ ist induktiv definiert durch

(MT1) $\mathcal{M}_\mathcal{T}(I, v) = I(v)$ für $v \in IVS$;

(MT2) $\mathcal{M}_\mathcal{T}(I, c') = c$ für $c' \in KS(\mathcal{D})$
(c ... Konstante zu Konstantensymbol c')

(MT3) $\mathcal{M}_\mathcal{T}(I, f'\underline{(}\, t_1 \underline{,} \ldots \underline{,} t_n \,\underline{)}) = f(\mathcal{M}_\mathcal{T}(I, t_1), \ldots, \mathcal{M}_\mathcal{T}(I, t_n))$
(f ... Funktion zum Funktionssymbol f')

Terme über Modellstrukturen (Beispiele zur Semantik)

Aufgabe:

Berechnen Sie den Wert von $t = \underline{+}\underline{(}\underline{-}\underline{(}\underline{\mathrm{x}}\underline{,}\underline{1}\underline{)}\underline{,}\underline{\mathrm{y}}\underline{)}$ über $\mathbb{Z}$ in der Variablenbelegung $I(\underline{\mathrm{x}}) = 0$, $I(\underline{\mathrm{y}}) = 2$.

Lösung:

$$\begin{aligned}
\mathcal{M}_\mathcal{T}(I, t) &\overset{MT3}{=} \mathcal{M}_\mathcal{T}(I, \underline{-}\underline{(}\underline{\mathrm{x}}\underline{,}\underline{1}\underline{)}) + \mathcal{M}_\mathcal{T}(I, \underline{\mathrm{y}}) \\
&\overset{MT1}{=} \mathcal{M}_\mathcal{T}(I, \underline{-}\underline{(}\underline{\mathrm{x}}\underline{,}\underline{1}\underline{)}) + I(\underline{\mathrm{y}}) \\
&\overset{MT3}{=} (\mathcal{M}_\mathcal{T}(I, \underline{\mathrm{x}}) - \mathcal{M}_\mathcal{T}(I, \underline{1})) + 2 \\
&\overset{MT1}{=} (I(\underline{\mathrm{x}}) - \mathcal{M}_\mathcal{T}(I, \underline{1})) + 2 \\
&\overset{MT2}{=} (0 - 1) + 2 \\
&= -1 + 2 = 1
\end{aligned}$$

Terme über Modellstrukturen (Beispiele zur Semantik)

Aufgabe:

Berechnen Sie den Wert von $t = \underline{+}\underline{(}\underline{\dot{-}}\underline{(}\underline{\mathrm{x}}\underline{,}\underline{1}\underline{)}\underline{,}\underline{\mathrm{y}}\underline{)}$ über $\mathbb{N}$ in der Variablenbelegung $I(\underline{\mathrm{x}}) = 0$, $I(\underline{\mathrm{y}}) = 2$.

Lösung:

$$\begin{aligned}
\mathcal{M}_\mathcal{T}(I, t) &\overset{MT3}{=} \mathcal{M}_\mathcal{T}(I, \underline{\dot{-}}\underline{(}\underline{\mathrm{x}}\underline{,}\underline{1}\underline{)}) + \mathcal{M}_\mathcal{T}(I, \underline{\mathrm{y}}) \\
&\overset{MT1}{=} \mathcal{M}_\mathcal{T}(I, \underline{\dot{-}}\underline{(}\underline{\mathrm{x}}\underline{,}\underline{1}\underline{)}) + I(\underline{\mathrm{y}}) \\
&\overset{MT3}{=} (\mathcal{M}_\mathcal{T}(I, \underline{\mathrm{x}}) \dot{-} \mathcal{M}_\mathcal{T}(I, \underline{1})) + 2 \\
&\overset{MT1}{=} (I(\underline{\mathrm{x}}) \dot{-} \mathcal{M}_\mathcal{T}(I, \underline{1})) + 2 \\
&\overset{MT2}{=} (0 \dot{-} 1) + 2 \\
&= 0 + 2 = 2
\end{aligned}$$

Terme über Modellstrukturen (Beispiele zur Semantik)

Aufgabe:

Berechnen Sie den Wert von $t = \underline{\mathrm{push0}}\underline{(}\underline{\mathrm{pop}}\underline{(}\underline{\mathrm{x}}\underline{)}\underline{)}$ in der Speicherbelegung $I(\underline{\mathrm{x}}) = \underline{1}\,\underline{1}$.

Lösung:

$$\begin{aligned}
\mathcal{M}_\mathcal{T}(I, t) &\overset{MT3}{=} \mathrm{push0}(\mathcal{M}_\mathcal{T}(I, \underline{\mathrm{pop}}\underline{(}\underline{\mathrm{x}}\underline{)})) \\
&\overset{MT3}{=} \mathrm{push0}(\mathrm{pop}(\mathcal{M}_\mathcal{T}(I, \underline{\mathrm{x}}))) \\
&\overset{MT1}{=} \mathrm{push0}(\mathrm{pop}(I(\underline{\mathrm{x}}))) \\
&= \mathrm{push0}(\mathrm{pop}(\underline{1}\,\underline{1}) \\
&= \mathrm{push0}(\underline{1}) \\
&= \underline{0}\,\underline{1}
\end{aligned}$$

Terme über Modellstrukturen (Beispiele zur Semantik)

Aufgabe:
Drücken Sie die Funktion $\max(x, y)$ (Maximum zweier Zahlen) als Term über $\mathbb{N}$ aus.

Lösung:
$\max(x, y)$ ist repräsentierbar durch $\underline{+}(\underline{\mathtt{x}},\underline{\dot{-}}(\underline{\mathtt{y}},\underline{\mathtt{x}}))$
(Beweis durch Fallunterscheidung)

Aufgabe:
Drücken Sie die Funktion $\min(x, y)$ (Minimum zweier Zahlen) als Term über $\mathbb{N}$ aus.

Lösung:
$\min(x, y)$ ist repräsentierbar durch $\underline{\dot{-}}(\underline{\mathtt{x}},\underline{\dot{-}}(\underline{\mathtt{x}},\underline{\mathtt{y}}))$

Eine Herausforderung:
Zeigen Sie, dass min und max nicht als Terme über $\mathbb{Z}$ ausdrückbar sind.

Terme über Modellstrukturen (Beispiele zur Semantik)

Aufgabe:
Mit welchem Term über **FamX** kann man die Funktion 'der Großvater mütterlicherseits von ...' ausdrücken?

Lösung:
Um eine Funktion auszudrücken muss der gesuchte Term ein Variablensymbol — sagen wir $\underline{\mathtt{x}}$ — enthalten:
Entsprechend der Definition der Sprache $\mathcal{T}(\mathbf{FamX})$ der Modellstruktur **FamX** hat

$$\underline{\mathit{Vater}}(\,\underline{\mathit{Mutter}}(\,\underline{\mathtt{x}}\,)\,)$$

in jeder Variablenbelegung I als Wert den Vater der Mutter, also den Großvater mütterlicherseits von $I(\underline{\mathtt{x}})$.

Eine weiterführende Überlegungen:
Lässt sich 'der Großvater von ...' ausdrücken?
Warum nicht?

Zusammenfassung: Termsprachen und ihre Grenzen

- Terme über einer Modellstruktur $\mathcal{D} = \langle D; P, K, F\rangle$ beziehen sich auf Elemente des Gegenstandsbereich D
  — Prädikate (Elemente von P) sind dafür irrelevant.
- $\mathcal{D}$ kann man auch als abstrakten Datentyp auffassen
  Funktionen heißen in diesem Zusammenhang auch Constructors
- Neben den Konstanten- und Funktionssymbolen werden auch (Individuen-)Variable benötigt.
  Terme (Syntax) bezeichnen konkrete Elemente von D (Semantik) nur relativ zu Variablenbelegungen (Environments)
- Wir gehen hier immer von totalen Funktionen aus.
  (Verschiedene Möglichkeiten der Behandlung von Partialität ...)
- Mit Termen kann man keine Eigenschaften von oder Beziehungen zwischen Elementen von D auszudrücken.
  Dazu braucht man die Prädikatensymbole von $\mathcal{D}$.

Boolesche Ausdrücke über Modellstrukturen

Definition (Syntax von Booleschen Ausdrücken)

$\mathcal{BA}(\mathcal{D})$ ist die kleinste Menge, für die gilt:

(BA1) $p'\,\underline{(}\,t_1\,\underline{,}\,\ldots\,\underline{,}\,t_n\,\underline{)} \in \mathcal{BA}(\mathcal{D})$, wenn $p' \in PS_n(\mathcal{D})$ und $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{D})$;
$p'\,\underline{(}\,t_1\,\underline{,}\,\ldots\,\underline{,}\,t_n\,\underline{)}$ heißt Atomformel (kurz: Atom);

(BA1') $s \underline{=} t$, wenn $s, t \in \mathcal{T}(\mathcal{D})$ (atomare) Gleichung;

(BA2) $\underline{\neg}\, F \in \mathcal{BA}(\mathcal{D})$, wenn $F \in \mathcal{BA}(\mathcal{D})$;

(BA3) $\underline{(}\,F \underline{\wedge} G\,\underline{)} \in \mathcal{BA}(\mathcal{D})$, wenn $F, G \in \mathcal{BA}(\mathcal{D})$;

(BA4) $\underline{(}\,F \underline{\vee} G\,\underline{)} \in \mathcal{BA}(\mathcal{D})$, wenn $F, G \in \mathcal{BA}(\mathcal{D})$.

Boolesche Ausdrücke über Modellstrukturen

Definition (Semantik von Booleschen Ausdrücken)

Die Semantik von $\mathcal{BA}(\mathcal{D})$ wird durch die Funktion $\mathcal{M}_{\mathcal{BA}} : ENV \times \mathcal{BA}(\mathcal{D}) \to \{\mathbf{t}, \mathbf{f}\}$ festgelegt:

(MBA1) $\mathcal{M}_{\mathcal{BA}}(I, p'\,\underline{(}\, t_1 \underline{,} \ldots \underline{,} t_n \,\underline{)}) = p(\mathcal{M}_{\mathcal{T}}(I, t_1), \ldots, \mathcal{M}_{\mathcal{T}}(I, t_n))$, wobei p das Prädikat zum Symbol $p' \in PS_n(\mathcal{D})$ ist.

(MBA1') $\mathcal{M}_{\mathcal{BA}}(I, s \underline{=} t) = \mathbf{t}$ falls $\mathcal{M}_{\mathcal{T}}(I, s) = \mathcal{M}_{\mathcal{T}}(I, t)$ (sonst $\mathbf{f}$)

(MBA2) $\mathcal{M}_{\mathcal{BA}}(I, \underline{\neg}\, F) = \begin{cases} \mathbf{t} & \text{falls } \mathcal{M}_{\mathcal{BA}}(I, F) = \mathbf{f} \\ \mathbf{f} & \text{falls } \mathcal{M}_{\mathcal{BA}}(I, F) = \mathbf{t} \end{cases}$

(MBA3) $\mathcal{M}_{\mathcal{BA}}(I, \underline{(}\, F \underline{\wedge} G \,\underline{)}) = \begin{cases} \mathbf{t} & \text{falls } \mathcal{M}_{\mathcal{BA}}(I, F) = \mathbf{t} \text{ und } \mathcal{M}_{\mathcal{BA}}(I, G) = \mathbf{t} \\ \mathbf{f} & \text{sonst} \end{cases}$

(MBA4) $\mathcal{M}_{\mathcal{BA}}(I, \underline{(}\, F \underline{\vee} G \,\underline{)}) = \begin{cases} \mathbf{t} & \text{falls } \mathcal{M}_{\mathcal{BA}}(I, F) = \mathbf{t} \text{ oder } \mathcal{M}_{\mathcal{BA}}(I, G) = \mathbf{t} \\ \mathbf{f} & \text{sonst} \end{cases}$

Notationsvereinbarungen

- Infixnotation für Terme über $\mathbb{Z}$ und $\mathbb{N}$: $\underline{(}\, 1 \underline{+} \mathrm{x} \,\underline{)}$ steht für $\underline{+}\,\underline{(}\, 1 \underline{,} \mathrm{x} \,\underline{)}$ etc.
- Infixnotation auch für Atomformeln: $\underline{(}\, t_1 \underline{\leq} t_2 \,\underline{)}$ steht für $\underline{\leq}\,\underline{(}\, t_1 \underline{,} t_2 \,\underline{)}$ etc.
- $t_1 \geq t_2$ steht für $t_2 \leq t_1$
- $t_1 \underline{\neq} t_2$ / $t_1 \underline{\geq} t_2$ steht für $\underline{\neg}\,\underline{(}\, t_1 \underline{=} t_2 \,\underline{)}$ / $\underline{\neg}\,\underline{(}\, t_1 \underline{\leq} t_2 \,\underline{)}$ etc.
- Klammereinsparungen, sofern keine Gefahr der Mehrdeutigkeit besteht

Eine Ankündigung:

- Demnächst werden wir auch die Unterstreichungen weglassen. Vorläufig aber verwenden wir dieses didaktische Hilfsmittel weiterhin, damit es nicht zu Syntax/Semantik-Verwechslungen kommt. (Verwechslung eines Zeichens mit dem wofür das Zeichen steht.)

Beispiel

Wert von $B = \underline{\mathrm{y}} \underline{+} \underline{\mathrm{x}} \underline{\neq} \underline{0} \;\underline{\vee}\; \underline{\mathrm{x}} \underline{\leq} \underline{\mathrm{y}}$ über $\mathbb{Z}$ in einer Variablenbelegung I mit $I(\underline{\mathrm{x}}) = 4$ und $I(\underline{\mathrm{y}}) = -2$.

Erinnerung: B ist 'offiziell' $\underline{(}\, \underline{\neg}\, \underline{+}\, \underline{(}\, \underline{\mathrm{y}} \underline{,} \underline{\mathrm{x}} \,\underline{)} \underline{=} \underline{0} \;\underline{\vee}\; \underline{\leq}\, \underline{(}\, \underline{\mathrm{x}} \underline{,} \underline{\mathrm{y}} \,\underline{)} \,\underline{)}$

$$\mathcal{M}_{\mathcal{BA}}(I, B) \overset{MBA4}{=} \mathbf{t} \quad \text{falls } \mathcal{M}_{\mathcal{BA}}(I, \underline{\mathrm{y}} \underline{+} \underline{\mathrm{x}} \underline{\neq} \underline{0}) = \mathbf{t} \text{ oder } \mathcal{M}_{\mathcal{BA}}(I, \underline{\mathrm{x}} \underline{\leq} \underline{\mathrm{y}}) = \mathbf{t}$$

Wir bestimmen zunächst $\mathcal{M}_{\mathcal{BA}}(I, \underline{\mathrm{y}} \underline{+} \underline{\mathrm{x}} \underline{\neq} \underline{0})$:

$$\mathcal{M}_{\mathcal{BA}}(I, \underline{\mathrm{y}} \underline{+} \underline{\mathrm{x}} \underline{\neq} \underline{0}) \overset{MBA2}{=} \mathbf{t} \text{ falls } \mathcal{M}_{\mathcal{BA}}(I, \underline{\mathrm{y}} \underline{+} \underline{\mathrm{x}} \underline{=} \underline{0}) = \mathbf{f}$$

$$\left[\begin{array}{lll} \mathcal{M}_{\mathcal{BA}}(I, \underline{\mathrm{y}} \underline{+} \underline{\mathrm{x}} \underline{=} \underline{0}) & \overset{MBA1'}{=} & (\mathcal{M}_{\mathcal{T}}(I, \underline{\mathrm{y}} \underline{+} \underline{\mathrm{x}}) = \mathcal{M}_{\mathcal{T}}(I, \underline{0})) \\ & \overset{MT3,MT2}{=} & (\mathcal{M}_{\mathcal{T}}(I, \underline{\mathrm{y}}) + \mathcal{M}_{\mathcal{T}}(I, \underline{\mathrm{x}}) = 0) \\ & \overset{MT1}{=} & (-2 + 4 = 0) = (2 = 0) = \mathbf{f} \end{array}\right]$$

Daher erhalten wir $\mathcal{M}_{\mathcal{BA}}(I, B) = \mathbf{t}$

Beispiel

Aufgabe:
Drücken Sie die Relation (binäres Prädikat) 'x ist Großvater von y' über **FamX** aus.

Lösung: Folgender Boolsche Ausdruck repräsentiert die Großvater-Relation:
$\underline{(}\, \underline{\mathrm{x}} \underline{=} \underline{\mathit{Vater}}\,\underline{(}\, \underline{\mathit{Mutter}}\,\underline{(}\, \underline{\mathrm{y}} \,\underline{)} \,\underline{)} \;\underline{\vee}\; \underline{\mathrm{x}} \underline{=} \underline{\mathit{Vater}}\,\underline{(}\, \underline{\mathit{Vater}}\,\underline{(}\, \underline{\mathrm{y}} \,\underline{)} \,\underline{)} \,\underline{)}$

Aufgabe:
Drücken Sie die Relation'x ist mehr als doppelt so groß wie y' über $\mathbb{N}$ aus.

Lösung: $\underline{\leq}\,\underline{(}\, \underline{+}\,\underline{(}\, \underline{\mathrm{y}} \underline{,} \underline{\mathrm{y}} \,\underline{)} \underline{,} \underline{\mathrm{x}} \,\underline{)}$ oder (gemäß Notationsvereinbarungen): $\underline{\mathrm{x}} \underline{\geq} \underline{\mathrm{y}} \underline{+} \underline{\mathrm{y}}$
Achtung: $\underline{\mathrm{x}} \underline{\geq} \underline{2} \underline{*} \underline{\mathrm{y}}$ ist keine korrekte Lösung! ($\underline{2} \notin KS(\mathbb{N})$.)

Eine Herausforderung:
Zeigen Sie, dass 'x ist ein Vielfaches von y' nicht als Boolescher Ausdruck über $\mathbb{N}$ darstellbar ist. (Warum ist $\underline{\mathrm{x}} \underline{=} \underline{\mathrm{z}} \underline{*} \underline{\mathrm{y}}$ keine Lösung?)

Eine einfache Programmiersprache

genauer: wir definieren eine Familie imperativer Programmiersprachen

$AL(\mathcal{D})$ ... Assignment Language für Datentyp $\mathcal{D}$
(Statements/Programme über $\mathcal{D}$)

Syntax:

Definition

$AL(\mathcal{D})$ ist die kleinste Menge, für die gilt:

(AL1) Ist $v \in IVS$ und $t \in \mathcal{T}(\mathcal{D})$, dann ist $v \underline{\leftarrow} t \in AL(\mathcal{D})$.

(AL2) Sind $\alpha, \beta \in AL(\mathcal{D})$, dann ist $\underline{\texttt{begin}}\ \alpha\underline{;}\ \beta\ \underline{\texttt{end}} \in AL(\mathcal{D})$.

(AL3) Ist $B \in \mathcal{BA}(\mathcal{D})$ und sind $\alpha, \beta \in AL(\mathcal{D})$, dann ist $\underline{\texttt{if}}\ B\ \underline{\texttt{then}}\ \alpha\ \underline{\texttt{else}}\ \beta \in AL(\mathcal{D})$.

(AL4) Ist $B \in \mathcal{BA}(\mathcal{D})$ und $\alpha \in AL(\mathcal{D})$, dann ist $\underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \alpha \in AL(\mathcal{D})$.

Semantik von $AL(\mathcal{D})$

Eine Hilfsdefinition: Für $I, I' \in ENV$ und $v \in IVS$ schreibt man $I \overset{v}{\sim} I'$ wenn $I(w) = I'(w)$ für alle $w \in IVS$ für die $w \neq v$.

Definition

Die Funktion $\mathcal{M}_{AL} : ENV \times AL(\mathcal{D}) \to ENV$ ist definiert durch:

(MAL1) $\mathcal{M}_{AL}(I, v \underline{\leftarrow} t) = I'$, wobei $I'(v) = \mathcal{M}_{\mathcal{T}}(I, t)$ und $I'(w) = I(w)$ für alle $w \in IVS$ mit $w \neq v$, d.h. $I' \overset{v}{\sim} I$.

(MAL2) $\mathcal{M}_{AL}(I, \underline{\texttt{begin}}\ \alpha\underline{;}\ \beta\ \underline{\texttt{end}}) = \mathcal{M}_{AL}(\mathcal{M}_{AL}(I, \alpha), \beta)$

(MAL3) $\mathcal{M}_{AL}(I, \underline{\texttt{if}}\ B\ \underline{\texttt{then}}\ \alpha\ \underline{\texttt{else}}\ \beta) =$
$$\begin{cases} \mathcal{M}_{AL}(I, \alpha) & \text{falls } \mathcal{M}_{\mathcal{BA}}(I, B) = \mathbf{t} \\ \mathcal{M}_{AL}(I, \beta) & \text{falls } \mathcal{M}_{\mathcal{BA}}(I, B) = \mathbf{f} \end{cases}$$

(MAL4) $\mathcal{M}_{AL}(I, \underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \alpha) =$
$$\begin{cases} \mathcal{M}_{AL}(\mathcal{M}_{AL}(I, \alpha), \underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \alpha) & \text{falls } \mathcal{M}_{\mathcal{BA}}(I, B) = \mathbf{t} \\ I & \text{falls } \mathcal{M}_{\mathcal{BA}}(I, B) = \mathbf{f} \end{cases}$$

Was berechnet ein $AL(\mathcal{D})$-Programm?

Beachte:

Relativ zur Auszeichnung von m Variablen als Input- bzw. n (nicht notwendigerweise verschiedenen) Variablen als Output-Variablen drückt jedes Programm aus $AL(\mathcal{D})$ eine Funktion vom Typ $D^m \to D^n$ aus.
Man sagt: Das Programm berechnet die jeweilige Funktion.

Faktum:

Jede durch eine Turingmaschine berechenbare Funktion vom Typ $\omega^m \to \omega^n$ ist durch ein $AL(\mathbb{N})$-Programm berechenbar und umgekehrt.
Mit anderen Worten: $AL(\mathbb{N})$ ist universell!

Anmerkungen:

- Modulo geeigneter Kodierungen sind auch beispielsweise $AL(\mathbb{Z})$ und $AL(\mathbb{S})$ universell.
- In der Praxis sind Programme über mehrere Datenstrukturen definiert.

Beispiel: Multiplikationsprogramm

Abkürzungen:

$\beta = \underline{\texttt{begin}}\ \underline{\mathrm{z}} \underline{\leftarrow} (\underline{\mathrm{z}} \underline{+} \underline{\mathrm{x}})\underline{;}\ \underline{\mathrm{y}} \underline{\leftarrow} (\underline{\mathrm{y}} \underline{-} \underline{1})\ \underline{\texttt{end}}$

$\gamma = \underline{\texttt{while}}\ (\underline{0} \underline{\leq} \underline{\mathrm{y}})\ \underline{\texttt{do}}\ \beta$

Auswertung für $I(\underline{\mathrm{x}}) = 3$, $I(\underline{\mathrm{y}}) = 1$, ($I(\underline{\mathrm{z}})$ beliebig)

$$\begin{aligned}
\mathcal{M}_{AL}(I, \underline{\texttt{begin}}\ \underline{\mathrm{z}} \underline{\leftarrow} \underline{0}\underline{;}\ \gamma\ \underline{\texttt{end}}) &\overset{MAL2}{=} \mathcal{M}_{AL}(\mathcal{M}_{AL}(I, \underline{\mathrm{z}} \underline{\leftarrow} \underline{0}), \gamma) \overset{MAL1}{=} \mathcal{M}_{AL}(I', \gamma) \\
&\quad \text{wobei } I'(\underline{\mathrm{z}}) = \mathcal{M}_{\mathcal{T}}(I, \underline{0}) = 0 \text{ und } I'(v) = I(v) \text{ für } v \neq \underline{\mathrm{z}} \\
&= \mathcal{M}_{AL}(I', \underline{\texttt{while}}\ (\underline{0} \underline{\leq} \underline{\mathrm{y}})\ \underline{\texttt{do}}\ \beta) \\
&\quad \left[\begin{aligned} \mathcal{M}_{\mathcal{BA}}(I', (\underline{0} \underline{\leq} \underline{\mathrm{y}})) &= (\mathcal{M}_{\mathcal{T}}(I', \underline{0}) < \mathcal{M}_{\mathcal{T}}(I', \underline{\mathrm{y}})) = (0 < I'(\underline{\mathrm{y}})) \\ &= (0 < I(\underline{\mathrm{y}})) = (0 < 1) = \mathbf{t} \end{aligned} \right] \\
&\overset{MAL4}{=} \mathcal{M}_{AL}(\mathcal{M}_{AL}(I', \beta), \underline{\texttt{while}}\ (\underline{0} \underline{\leq} \underline{\mathrm{y}})\ \underline{\texttt{do}}\ \beta)
\end{aligned}$$

$$
\begin{aligned}
& \mathcal{M}_{AL}(\mathcal{M}_{AL}(I', \beta), \texttt{while}\ (\underline{0} \leq \underline{y})\ \texttt{do}\ \beta) \\
\overset{MAL2}{=}\ & \mathcal{M}_{AL}(\mathcal{M}_{AL}(\mathcal{M}_{AL}(I', \underline{z} \leftarrow \underline{z} + \underline{x}), \underline{y} \leftarrow \underline{y} - \underline{1}), \texttt{while}\ (\underline{0} \leq \underline{y})\ \texttt{do}\ \beta) \\
\overset{MAL1}{=}\ & \mathcal{M}_{AL}(\mathcal{M}_{AL}(I'', \underline{y} \leftarrow \underline{y} - \underline{1}), \texttt{while}\ (\underline{0} \leq \underline{y})\ \texttt{do}\ \beta) \\
& \text{wobei} \quad I''(\underline{z}) = \mathcal{M}_{\mathcal{T}}(I', \underline{z} + \underline{x}) = \mathcal{M}_{\mathcal{T}}(I', \underline{z}) + \mathcal{M}_{\mathcal{T}}(I', \underline{x}) \\
& \qquad\qquad\quad = I'(\underline{z}) + I'(\underline{x}) = 0 + I(\underline{x}) = 0 + 3 = 3 \\
& \qquad\qquad I''(v) = I'(v) \text{ für } v \neq \underline{z} \\
\overset{MAL1}{=}\ & \mathcal{M}_{AL}(I''', \texttt{while}\ (\underline{0} \leq \underline{y})\ \texttt{do}\ \beta) \\
& \text{wobei} \quad I'''(\underline{y}) = \mathcal{M}_{\mathcal{T}}(I'', \underline{y} - \underline{1}) = \mathcal{M}_{\mathcal{T}}(I'', \underline{y}) - \mathcal{M}_{\mathcal{T}}(I'', \underline{1}) \\
& \qquad\qquad\quad = I''(\underline{y}) - 1 = I'(\underline{y}) - 1 = I(\underline{y}) - 1 = 1 - 1 = 0 \\
& \qquad\qquad I'''(v) = I''(v) \text{ für } v \neq \underline{y} \\
& \left[\begin{aligned} & \mathcal{M}_{\mathcal{BA}}(I''', (\underline{0} \leq \underline{y})) = (\mathcal{M}_{\mathcal{T}}(I''', \underline{0}) < \mathcal{M}_{\mathcal{T}}(I''', \underline{y})) \\ & \qquad\qquad\qquad\qquad = (0 < I'''(\underline{y})) = (0 < 0) = \mathbf{f} \end{aligned} \right] \\
\overset{MAL4}{=}\ & I'''
\end{aligned}
$$

Beispiel: Ein $AL(\mathbb{S})$-Programm

$$\pi = \texttt{if}\ \underline{\text{ist0?}}(\underline{x})\ \texttt{then}\ \underline{x} \leftarrow \underline{\text{push0}}(\underline{x})\ \texttt{else}\ \texttt{while}\ \neg\, \underline{\text{istleer?}}(\underline{x})\ \texttt{do}\ \underline{x} \leftarrow \underline{\text{pop}}(\underline{x})$$

Wir berechnen das Ergebnis von π in der Speicherbelegung $I(\underline{x}) = \underline{1}$.

$$
\begin{aligned}
& \mathcal{M}_{AL}(I, \pi) \\
& \quad [\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, \underline{\text{ist0?}}(\underline{x})) = \text{ist0?}(\mathcal{M}_{\mathcal{T}}(I, \underline{x})) = \text{ist0?}(I(\underline{x})) = \text{ist0?}(\underline{1}) = \mathbf{f}] \\
\overset{MAL3}{=}\ & \mathcal{M}_{AL}(I, \texttt{while}\ \neg\, \underline{\text{istleer?}}(\underline{x})\ \texttt{do}\ \cdots) \\
& \quad \left[\begin{aligned} & \mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, \neg\, \underline{\text{istleer?}}(\underline{x})) = \neg \mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, \underline{\text{istleer?}}(\underline{x})) = \neg\text{istleer?}(\mathcal{M}_{\mathcal{T}}(I, \underline{x})) \\ & \qquad\qquad\qquad\qquad = \neg\text{istleer?}(I(\underline{x})) = \neg\mathbf{f} = \mathbf{t} \end{aligned} \right] \\
\overset{MAL4}{=}\ & \mathcal{M}_{AL}(\mathcal{M}_{AL}(I, \underline{x} \leftarrow \underline{\text{pop}}(\underline{x})), \texttt{while}\ \neg\, \underline{\text{istleer?}}(\underline{x})\ \texttt{do}\ \cdots) \\
\overset{MAL1}{=}\ & \mathcal{M}_{AL}(I', \texttt{while}\ \neg\, \underline{\text{istleer?}}(\underline{x})\ \texttt{do}\ \cdots) \\
& \quad [I'(\underline{x}) = \mathcal{M}_{\mathcal{T}}(I, \underline{\text{pop}}(\underline{x})) = \text{pop}(\mathcal{M}_{\mathcal{T}}(I, \underline{x})) = \text{pop}(I(\underline{x})) = \varepsilon, I' \overset{\underline{x}}{\approx} I] \\
& \quad \left[\begin{aligned} & \mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I', \neg\, \underline{\text{istleer?}}(\underline{x})) = \neg \mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I', \underline{\text{istleer?}}(\underline{x})) = \neg\text{istleer?}(\mathcal{M}_{\mathcal{T}}(I', \underline{x})) \\ & \qquad\qquad\qquad\qquad = \neg\text{istleer?}(I'(\underline{x})) = \neg\mathbf{t} = \mathbf{f} \end{aligned} \right] \\
\overset{MAL4}{=}\ & I'
\end{aligned}
$$

Semantik – Aussagenlogik [WH aus FMod]

$\mathcal{B} = \{\mathbf{f}, \mathbf{t}\}$ (bzw. $\{0, 1\}$) ... Wahrheitswerte
$I : AV \to \mathcal{B}$ ... Wahrheitsbelegung, (AL-)Interpretation
$\mathcal{I} = \{\, I \mid I : AV \to \mathcal{B} \,\}$ ... Menge aller AL-Interpretationen

$I(A) = I(C) = 1$ und $I(v) = 0$ sonst steht für
„Die atomaren Aussagen A und C sind wahr, die übrigen sind falsch."

Semantik aussagenlogischer Formeln

Der Wert einer Formel in einer Interpretation I wird festgelegt durch die Funktion $\text{val} : \mathcal{I} \times \mathcal{AF} \mapsto \mathcal{B}$:

(MA1) $\text{val}_I(A) = I(A)$ für $A \in AV$;
(MA2) $\text{val}_I(\top) = 1/\mathbf{t}$ und $\text{val}_I(\bot) = 0/\mathbf{f}$;
(MA3) $\text{val}_I(\neg F) = \textit{not}\ \text{val}_I(F)$;
(MA4) $\text{val}_I(\,(F * G)\,) = \text{val}_I(F) \circledast \text{val}_I(G)$,
wobei $\circledast$ die logische Funktion zum Operator $*$ ist.

Beispiel einer Formelauswertung

(MA1) $\text{val}_I(A) = I(A)$ für $A \in AV$;
von (MA2) $\text{val}_I(\bot) = 0$;
(MA3) $\text{val}_I(\neg F) = \textit{not}\ \text{val}_I(F)$;
von (MA4) $\text{val}_I(\,(F \wedge G)\,) = \text{val}_I(F)\ \textit{and}\ \text{val}_I(G) = 1$, d.h.:
$\text{val}_I(\,(F \wedge G)\,) = 1$ gdw. $\text{val}_I(F) = \text{val}_I(G) = 1$;
von (MA4) $\text{val}_I(\,(F \supset G)\,) = \text{val}_I(F)\ \textit{implies}\ \text{val}_I(G) = 1$, d.h.:
$\text{val}_I(\,(F \supset G)\,) = 1$ gdw. $\text{val}_I(F) = 0$ oder $\text{val}_I(G) = 1$

Wert von $((A \wedge \neg B) \supset \bot)$ für $I(A) = 1$ und $I(B) = 0$

$$
\begin{aligned}
\text{val}_I(((A \wedge \neg B) \supset \bot)) &= \text{val}_I((A \wedge \neg B))\ \textit{implies}\ \text{val}_I(\bot) \\
&= (\text{val}_I(A)\ \textit{and}\ \text{val}_I(\neg B))\ \textit{implies}\ 0 \\
&= (1\ \textit{and}\ \textit{not}\ \text{val}_I(B))\ \textit{implies}\ 0 \\
&= (1\ \textit{and}\ \textit{not}\ 0)\ \textit{implies}\ 0 \\
&= (1\ \textit{and}\ 1)\ \textit{implies}\ 0 \\
&= 1\ \textit{implies}\ 0 = 0
\end{aligned}
$$

Semantik der PL – Hinweise zur Ausdrucksstärke (1)

Eigenschaften einer Relation

"R ist eine Äquivalenzrelation" kann man — in der Signatur $\langle\{\underline{R}\},\{\},\{\}\rangle$ — als $F_1 \wedge F_2 \wedge F_3$ ausdrücken, wobei:
$F_1 = \forall x\, \underline{R}(x,x)$
$F_2 = \forall x \forall y [\underline{R}(x,y) \supset \underline{R}(y,x)]$
$F_3 = \forall x \forall y \forall z [(\underline{R}(x,y) \wedge \underline{R}(y,z)) \supset \underline{R}(x,z)]$

Spezifikation/Eigenschaften von Modellstrukturen

Modellstruktur $\mathbb{Z}$:

- $\forall x\; x + 0 = x$ (0 ist ein rechtsneutrales Argument bzgl. $+$)
- $\forall x\; x * 1 = x$ (1 ist ein rechtsneutrales Argument bzgl. $*$)
- $\forall x \forall y\; x + y = y + x$ (Addition ist kommutativ)
- $\forall x \forall y \forall z\; (x + y) + z = x + (y + z)$ (Addition ist assoziativ)
- $\forall x \exists y\; x + y = 0$ (Invertierbarkeit der Addition)
- $\neg \forall x \exists y\; x * y = 1$ (Nicht-Invertierbarkeit der Multiplikation)
- $\neg \forall x\; x + x = x$ (Nicht-Idempotenz der Addition)
- $\forall x \forall y (x < y \supset 0 - y < 0 - x)$ ("Minus" ist anti-monoton)

Semantik der PL – Hinweise zur Ausdrucksstärke (2)

Ausdrücken von komplexen Prädikaten

Über der Modellstruktur $\mathbb{N}$:

- $\exists z\; y = x * z$ (x teilt y)
- $\exists y\; x = y * y * y$ (x ist eine Kubikzahl)
- "x ist eine Primzahl" lässt sich wie folgt ausdrücken:
  $\forall y \forall z [x = y * z \supset (y = 1 \vee z = 1)] \wedge 1 < x$

Ausdrucksstärke der leeren Signatur ($\langle\{\},\{\},\{\}\rangle$)

Es lassen sich Kardinalitätsaussagen über die Domäne ausdrücken:

- $\forall x \forall y\; x = y$ (Es gibt genau ein Element)
- $\exists x \exists y\; x \neq y$ (Es gibt mindestens zwei Elemente)
- $\exists x \exists y \forall z (z = x \vee z = y)$ (Es gibt höchstens zwei Elemente)
- "Es gibt genau drei Elemente" lässt sich wie folgt ausdrücken:
  $\exists x \exists y \exists z \forall u ((u = x \vee u = y \vee u = z) \wedge (x \neq y \wedge x \neq z \wedge y \neq z))$

Semantik – PL-Interpretationen

Definition

Eine prädikatenlogische Interpretation $\mathcal{I}$ über der Signatur $\Sigma = \langle PS_\Sigma, KS_\Sigma, FS_\Sigma\rangle$ ist ein Tupel $\langle D, \Phi, \xi\rangle$, wobei

- D: Domäne (Gegenstandsbereich): eine beliebige nicht-leere Menge
- Φ ist eine Signaturinterpretation:
  - $P \in PS_\Sigma$ n-stellig $\Longrightarrow \Phi(P)\colon D^n \to \{\mathbf{f}, \mathbf{t}\}$
  - $c \in KS_\Sigma \Longrightarrow \Phi(c) \in D$
  - $f \in FS_\Sigma$ n-stellig $\Longrightarrow \Phi(f)\colon D^n \to D$
- $\xi : IVS \to D$ ist eine Variablenbelegung

$PINT_\Sigma$ ... Menge aller PL-Interpretationen über der Signatur Σ

Beachte: Jede PL-Interpretation $\mathcal{I}$ bestimmt eine Modellstruktur $\mathcal{D}_\mathcal{I} = \langle D; P_D, K_D, F_D\rangle$, wobei
$P_D = \{\Phi(P) \mid P \in PS_\Sigma\}$, $K_D = \{\Phi(c) \mid c \in KS_\Sigma\}$, $F_D = \{\Phi(f) \mid f \in FS_\Sigma\}$.
D.h.: Jedes $\mathcal{I}$ entspricht einer Modellstruktur + Variablenbelegung
Man spricht vom Interpetieren bzw. Ausdrücken über $\mathcal{D}_\mathcal{I}$ (z.B. $\mathbb{Z}$, $\mathbb{N}$, $\mathbb{S}$).

Semantik – Wahrheit und Falschheit von Formeln

Definition (Auswertungsfunktion $\mathrm{val}_\mathcal{I}$ für PL-Formeln)

$\mathrm{val}_\mathcal{I} : PINT_\Sigma \times \mathcal{PF}_\Sigma \to \{\mathbf{f}, \mathbf{t}\}$ ist induktiv definiert durch

(MP1) $\mathrm{val}_\mathcal{I}(\top) = \mathbf{t}$, $\mathrm{val}_\mathcal{I}(\bot) = \mathbf{f}$

(MP2) $\mathrm{val}_\mathcal{I}(P(t_1,\ldots,t_n)) = \Phi(P)(\mathcal{M}_\mathcal{T}(\xi,t_1),\ldots,\mathcal{M}_\mathcal{T}(\xi,t_n))$

Beachte: $\mathcal{M}_\mathcal{T}$ bereits definiert, hier bezogen auf Modellstruktur $\mathcal{D}_\mathcal{I}$

(MP3) $\mathrm{val}_\mathcal{I}(s = t) = \mathbf{t}$ gdw. $\mathcal{M}_\mathcal{T}(\xi, s) = \mathcal{M}_\mathcal{T}(\xi, t)$ (sonst $= \mathbf{f}$)

(MP4) $\mathrm{val}_\mathcal{I}(\neg F) = \neg\, \mathrm{val}_\mathcal{I}(F)$

(MP5) $\mathrm{val}_\mathcal{I}((F \wedge G)) = \mathrm{val}_\mathcal{I}(F)$ *and* $\mathrm{val}_\mathcal{I}(G)$

(MP6) $\mathrm{val}_\mathcal{I}((F \vee G)) = \mathrm{val}_\mathcal{I}(F)$ *or* $\mathrm{val}_\mathcal{I}(G)$

(MP7) $\mathrm{val}_\mathcal{I}((F \supset G)) = \mathrm{val}_\mathcal{I}(F)$ *implies* $\mathrm{val}_\mathcal{I}(G)$

(MP8) $\mathrm{val}_\mathcal{I}(\forall v\, F) = \mathbf{t} \iff \mathrm{val}_{\mathcal{I}'}(F) = \mathbf{t}$ für alle $\mathcal{I}' \overset{v}{\sim} \mathcal{I}$

(MP9) $\mathrm{val}_\mathcal{I}(\exists v\, F) = \mathbf{t} \iff \mathrm{val}_{\mathcal{I}'}(F) = \mathbf{t}$ für ein $\mathcal{I}' \overset{v}{\sim} \mathcal{I}$

$\mathcal{I}' \overset{v}{\sim} \mathcal{I}$ ... $\mathcal{I} = \langle D, \Phi, \xi\rangle$ und $\mathcal{I}' = \langle D, \Phi, \xi'\rangle$, wobei $\xi \overset{v}{\sim} \xi'$

Erinnerung: $\xi \overset{v}{\sim} \xi'$ ... $\xi(w) = \xi'(w)$ für alle $w \in IVS$ mit $w \neq v$

Semantik — Ein einfaches Beispiel

Auswertung der PL-Formel $F = \forall x \exists y \neg P(x, f(y))$
in der Interpretation $\mathcal{I} = \langle \{3,4\}, \Phi, \xi \rangle$, wobei:
$\Phi(P) =$"$<$" (P bedeudet "kleiner als"), $\Phi(f) = id$ (Identitätsfunktion)
$\xi(v) = 3$ für alle $v \in IVS$ – hier aber irrelevant, da $FV(F) = \{\}$!

$$\begin{aligned}
\mathrm{val}_{\mathcal{I}}(F) = \mathbf{t} &\overset{MP8}{\iff} \mathrm{val}_{\mathcal{I}'}(\exists y \neg P(x, f(y))) = \mathbf{t} \text{ für alle } \mathcal{I}' \overset{x}{\sim} \mathcal{I} \\
&\overset{MP9}{\iff} \mathrm{val}_{\mathcal{I}''}(\neg P(x, f(y))) = \mathbf{t} \text{ für ein } \mathcal{I}'' \overset{y}{\sim} \mathcal{I}' \\
&\overset{MP4}{\iff} \mathrm{val}_{\mathcal{I}''}(P(x, f(y))) = \mathbf{f} \\
&\overset{MP2}{\iff} \mathcal{M}_{\mathcal{T}}(\xi'', x) \geq \mathcal{M}_{\mathcal{T}}(\xi'', f(y)) \quad \text{für } \mathcal{I}'' = \langle D, \Phi, \xi'' \rangle \\
&\overset{MT1,MT3}{\iff} \xi''(x) \geq id(\mathcal{M}_{\mathcal{T}}(\xi'', y)) \iff \xi''(x) \geq \xi''(y)
\end{aligned}$$

$\mathrm{val}_{\mathcal{I}}(F) = \mathbf{t}$, da zu jedem $\mathcal{I}' \overset{x}{\sim} \mathcal{I}$ ein $\mathcal{I}'' \overset{y}{\sim} \mathcal{I}'$ mit $\xi''(x) \geq \xi''(y)$ existiert

Semantische Grundbegriffe der PL (1)

Eine Formel $F \in \mathcal{PF}_{\Sigma}$ heißt

- erfüllbar, wenn $\mathrm{val}_{\mathcal{I}}(F) = \mathbf{t}$ für ein $\mathcal{I} \in PINT_{\Sigma}$,
  $\mathcal{I}$ heißt Modell von F, man sagt auch: $\mathcal{I}$ erfüllt F;
- widerlegbar, wenn $\mathrm{val}_{\mathcal{I}}(F) = \mathbf{f}$ für ein $\mathcal{I} \in PINT_{\Sigma}$,
  $\mathcal{I}$ heißt Gegenbeispiel zu F;
- unerfüllbar, wenn $\mathrm{val}_{\mathcal{I}}(F) = \mathbf{f}$ für alle $\mathcal{I} \in PINT_{\Sigma}$
  (also: alle Interpretationen sind Gegenbeispiele);
- (logisch bzw. allgemein) gültig, wenn $\mathrm{val}_{\mathcal{I}}(F) = \mathbf{t}$ für alle $\mathcal{I} \in PINT_{\Sigma}$
  (also: alle Interpretationen sind Modelle).

Eine Menge $\mathcal{F} \subseteq \mathcal{PF}_{\Sigma}$ heißt (un)erfüllbar, wenn es ein $\mathcal{I} \in PINT_{\Sigma}$ gibt, sodass $\mathrm{val}_{\mathcal{I}}(F) = \mathbf{t}(\mathbf{f})$ für alle $F \in \mathcal{F}$.

Beachte: $\mathcal{F}$ kann unerfüllbar sein, obwohl alle $F \in \mathcal{F}$ erfüllbar sind.
Erfüllbarkeit von $\mathcal{F}$ erfordert ein gemeinsames Modell seiner Elemente.

Beispiele

Folgende Formeln sind gültig:

- $\forall x[P(x) \vee \neg P(x)]$ bzw. auch $P(x) \vee \neg P(x)$
- $\forall x P(x) \vee \neg \forall y P(y)$ und $\forall x[P(x) \supset \exists y P(y)]$
- $\neg \forall x P(x) \supset \exists x \neg P(x)$ und $\neg \exists x P(x) \supset \forall x \neg P(x)$
- $\forall x \forall y Q(x, y) \supset \forall y \forall x Q(x, y)$

Folgende Formeln sind unerfüllbar:

- $\forall x[P(x) \wedge \neg P(x)]$ bzw. auch $P(x) \wedge \neg P(x)$
- $\neg[\forall x P(x) \vee \neg \forall y P(y)]$ und $\exists x P(x) \wedge \forall x \neg P(x)$

Folgende Formeln sind erfüllbar und widerlegbar:

- $\exists x P(x) \supset \forall x P(x)$ und $P(x) \supset P(y)$ und $P(x) \vee \neg P(y)$
- $\forall x P(x) \vee \forall x \neg P(x)$ und $\exists x P(x) \supset \forall x \neg P(x)$
- $\forall x \forall y[Q(x, y) \supset Q(y, x)]$ und $\forall x[\forall y Q(x, y) \supset \forall y Q(y, x)]$

Überlege: Was ist zum Nachweis dieser Behauptungen jeweils nötig?

Übungsaufgabe: Semantischer Status von $\exists x[P(x) \supset \forall y P(y)]$?

Semantische Grundbegriffe der PL (2)

Bezogen auf eine gegebene Modellstruktur $\mathcal{D}$ und eine entsprechende Signaturinterpretation Φ heißt $F \in \mathcal{PF}_{\Sigma}$:

- erfüllbar in $\mathcal{D}$ (bezüglich Φ), wenn $\mathrm{val}_{\mathcal{I}}(F) = \mathbf{t}$ für ein $\mathcal{I}$ mit $\mathcal{D}_{\mathcal{I}} = \mathcal{D}$
  (mit Φ als Signaturinterpretation),
  $\mathcal{I}$ heißt Modell in $\mathcal{D}$ von F;
- widerlegbar in $\mathcal{D}$, wenn $\mathrm{val}_{\mathcal{I}}(F) = \mathbf{f}$ für ein $\mathcal{I}$ mit $\mathcal{D}_{\mathcal{I}} = \mathcal{D}$,
  $\mathcal{I}$ heißt Gegenbeispiel in $\mathcal{D}$ zu F;
- unerfüllbar in $\mathcal{D}$ (bzgl. Φ), wenn $\mathrm{val}_{\mathcal{I}}(F) = \mathbf{f}$ für alle $\mathcal{I}$ mit $\mathcal{D}_{\mathcal{I}} = \mathcal{D}$;
- gültig in $\mathcal{D}$, wenn $\mathrm{val}_{\mathcal{I}}(F) = \mathbf{t}$ für alle $\mathcal{I}$ mit $\mathcal{D}_{\mathcal{I}} = \mathcal{D}$.

Einfache Beispiele:

- $x + y = y + x$ ist gültig in $\mathbb{Z}$ und $\mathbb{N}$ (bezogen auf die Standard-Signaturinterpretation), aber nicht allgemein gültig.
- $1 + x = x$ ist erfüllbar, aber nicht in $\mathbb{Z}$ oder in $\mathbb{N}$.
- $\forall x \exists y\, x + y = z$ ist erfüllbar in $\mathbb{Z}$, aber nicht in $\mathbb{N}$.

Ein interessantes Beispiel

Es sei $F = F_1 \wedge F_2 \wedge F_3$, wobei

$$\begin{aligned} F_1 &= \forall x \neg P(x,x) \\ F_2 &= \forall x \forall y [P(x,y) \supset P(x, f(y,a))] \\ F_3 &= \forall x P(x, f(x,a)) \end{aligned}$$

F ist erfüllbar in $\mathbb{N}$ bzgl. folgendem Φ: $\Phi(P) =$"$<$", $\Phi(f) = +$, $\Phi(a) = 1$.
Mit anderen Worten: $\langle \omega, \Phi, \xi \rangle$ ist ein Modell von F, für alle ξ.

Ein anderes Modell von F: $\mathcal{I}' = \langle D, \Phi', \xi \rangle$, wobei
$D_{\mathcal{I}} = \langle \{\underline{0}, \underline{1}\}^*; \{\prec\}, \{\underline{0}\}, \{concat\} \rangle$ mit
$concat(s,r) = sr$ und $s \prec r =$ "s ist ein echter Teilstring von r".
$\Phi'(P) = \prec$, $\Phi'(f) = concat$, $\Phi'(a) = \underline{0}$, ξ beliebig.

Satz

F hat keine endlichen Modelle:
Für alle Interpretationen $\mathcal{I} = \langle D, \Phi, I \rangle$ folgt aus $\mathrm{val}_{\mathcal{I}}(F) = \mathbf{t}$, dass der Gegenstandsbereich D unendlich ist.

Ein etwas komplexeres Beispiel

Kein Bauer, der ein großes Pferd besitzt, wird seine kleinen Pferde füttern, wenn das Futter nicht für alle Pferde reicht.

Atomare Prädikate (hier bereits als atomare Formeln ausgedrückt):

1-stellig: – *Bauer*(x) – Klein(x) – *Pferd*(x) – *Groß*(x) – *Futter_reicht_für*(x)

2-stellig: – *Besitzt*(x,y) – *Füttert*(x,y)

- x ist ein Bauer, der ein großes Pferd besitzt:
  $\mathrm{BdgPb}(x) = Bauer(x) \wedge \exists y [Besitzt(x,y) \wedge (Pferd(y) \wedge Groß(y))]$
- x füttert seine kleinen Pferde:
  $\mathrm{FskP}(x) = \forall y [((Klein(y) \wedge Pferd(y)) \wedge Besitzt(x,y)) \supset Füttert(x,y)]$
- Das Futter reicht nicht für alle Pferde:
  $\mathrm{DFrnfaP} = \neg \forall y (Pferd(y) \supset Futter_reicht_für(y))$
  oder äquivalent: $\exists y (Pferd(y) \wedge \neg Futter_reicht_für(y))$

Man erhält insgesamt: $\neg \exists x [\mathrm{BdgPb}(x) \wedge (\mathrm{DFrnfaP} \supset \mathrm{FskP}(x))]$

Quantoren und Eigenschaften

Alle vernünftigen Leute verabscheuen Gewalt.

$\forall$ vernünftige x (x verabscheut Gewalt) (Zwischenschritt, ist keine Formel!)

$\forall x\, (Vernünftig(x) \supset Verabscheut(x, gewalt))$

Generell: Eigenschaften universell quantifizierter Variablen werden zu Prämissen einer Implikation.

Es gibt vernünftige Leute, die Gewalt verabscheuen.

$\exists$ vernünftige x (x verabscheut Gewalt) (Zwischenschritt, ist keine Formel!)

$\exists x\, (Vernünftig(x) \wedge Verabscheut(x, gewalt))$

Generell: Eigenschaften existentiell quantifizierter Variablen werden zu Teilen einer Konjunktion.

Ausdrücken formaler Inhalte mit PL-Formeln

Beachte:
F drückt ein n-stelliges Prädikat aus $\Longrightarrow$ F hat n freie Variable
F drückt einen Satz aus $\Longrightarrow$ F ist eine geschlossene Formel

Beispiele zu $\mathbb{Z}$

Wir können in $\mathbb{Z}$ (über der Standardsignatur) z.B. folgende Prädikate und folgende Sätze ausdrücken:

- x ist eine Quadratzahl [unäres Prädikat]: $\exists y\, x = y * y$
- x ist ein echter Teiler von y [binäres Prädikat]:
  $\exists z (x * z = y \wedge z \neq 1 \wedge z \neq y)$

Zur Erinnerung: $s \neq t$ steht für $\neg s = t$

- y ist die (ganzzahlige) Lösung einer quadratischen Gleichung (mit ganzzahligen Koeffizienten) [unäres Prädikat]:
  $\exists u \exists v \exists w\, u * (y * y) + v * y = w$
- Zu jeder Zahl gibt es eine echt kleinere Zahl [Satz]: $\forall x \exists y\, y < x$
- Nicht alle Kubikzahlen sind positiv [Satz]:
  $\neg \forall x (\exists y\, x = y * y * y \supset 0 < x)$

Ausdrücken mit PL-Formeln (Forts.)

Beispiele zu $\mathbb{N}$

In $\mathbb{N}$ lassen sich (über der Standardsignatur) zusätzlich zu den $\mathbb{Z}$-Beispielen auch folgende Prädikate bzw. Sätze ausdrücken:

- x ist ein Primzahl [unäres Prädikat]:
  $\forall y \forall z[x = y * z \supset (y = 1 \vee z = 1)] \wedge 1 < x$
- Addition ist nur für 0 Idempotent [Satz]:
  $0 + 0 = 0 \wedge \forall x(x + x = x \supset x = 0)$
- x ist mindestens dreimal so groß wie y [binäres Prädikat]:
  $(y + y) + y = x \vee (y + y) + y < x$
  Achtung: Weder für "$\leq$" noch für "3" gibt es Zeichen in $\Sigma_{\mathbb{N}}$!

Achtung: Weitere Konstanten lassen sich (nicht als Prädikate oder Funktionen, sondern) als Terme ausdrücken. Z.B., über $\Sigma_{\mathbb{N}}$, "2" als $1 + 1$.

Ausdrücken mit PL-Formeln (Forts.)

Viele (auch nicht-totale) Funktionen lassen sich nur indirekt, nämlich als nach-eindeutige Relationen ausdrücken.

Zur Erinnerung:
$R(x_1, \ldots, x_n, x_{n+1})$ ist nacheindeutig (in $\mathcal{D}$), wenn folgendes (in $\mathcal{D}$) gilt:
$\forall y \forall z \forall x_1 \cdots \forall x_n[(R(x_1, \ldots, x_n, y) \wedge R(x_1, \ldots, x_n, z)) \supset y = z]$

Audrücken weiterer Funktionen über $\mathbb{N}$

- Normale Substraktion ("$-$", nicht verwechseln mit "$\dot{-}$") ist als 3-stellige Relation "x minus y ist z" ausdrückbar: $x = y + z$
  Beachte die Nicht-Totalität, im Unterschied zu "$\dot{-}$"!
- Ganzzahlige Division "$x \div y$" ist ausdrückbar durch $x = y * z$
- Die 1-stellige Funktion "x mod 2" ist ausdrückbar durch
  $(\exists z\, z + z = x \wedge y = 0) \vee (\exists z\, z + z + 1 = x \wedge y = 1)$

Satz (Gödel, Church, Turing)

Jede partiell berechenbare Funktion lässt sich über $\mathbb{N}$ ausdrücken.

Ausdrücken mit PL-Formeln (Forts.)

Wichtige Eigenschaften binärer Relationen

- Reflexivität: $\forall x\, R(x, x)$
- Symmetrie: $\forall x \forall y[R(x, y) \supset R(y, x)]$
- Transitivität: $\forall x \forall y \forall z[(R(x, y) \wedge R(y, z)) \supset R(x, z)]$
- Serialität: $\forall x \exists y\, R(x, y)$
- partielle Funktionalität: $\forall x \forall y \forall z[(R(x, y) \wedge R(x, z)) \supset y = z]$
- totale Funktionalität: $\forall x \exists y[R(x, y) \wedge \forall z(R(x, z) \supset y = z)]$
- Euklidizität: $\forall x \forall y \forall z[(R(x, y) \wedge R(x, z)) \supset R(y, z)]$
- schwache Dichte: $\forall x \forall y[R(x, y) \supset \exists z(R(x, z) \wedge R(z, y))]$
- schwache Gerichtetheit ('Diamant-Eigenschaft'):
  $\forall x \forall y \forall z[(R(x, y) \wedge R(x, z)) \supset \exists u(R(y, u) \wedge R(z, u))]$

Logische Konsequenz

$F_1, \ldots, F_n \models_{\mathcal{I}} G$: Wenn $\mathrm{val}_{\mathcal{I}}(F_1) = \cdots = \mathrm{val}_{\mathcal{I}}(F_n) = \mathbf{t}$, dann $\mathrm{val}_{\mathcal{I}}(G) = \mathbf{t}$.
Bzw: $\mathcal{F} \models_{\mathcal{I}} G$: Wenn $\mathrm{val}_{\mathcal{I}}(F) = \mathbf{t}$ für alle $F \in \mathcal{F}$, dann $\mathrm{val}_{\mathcal{I}}(G) = \mathbf{t}$.

Diese Wenn-Dann-Beziehung kann man auch als Disjunktion schreiben:

$\mathcal{F} \models_{\mathcal{I}} G$: $\mathrm{val}_{\mathcal{I}}(F) = \mathbf{f}$ für mindestens ein $F \in \mathcal{F}$ oder $\mathrm{val}_{\mathcal{I}}(G) = \mathbf{t}$.

Definition (Logische Konsequenz — $F_1, \ldots, F_n \models G$ bzw. $\mathcal{F} \models G$)

Für alle Interpretationen $\mathcal{I}$ gilt: $F_1, \ldots, F_n \models_{\mathcal{I}} G$ bzw. $\mathcal{F} \models_{\mathcal{I}} G$.

In Worten:
„G ist eine logische Konsequenz der Formeln $F_1, \ldots, F_n$ (bzw. in $\mathcal{F}$)"
„G folgt (logisch) aus den Formeln $F_1, \ldots, F_n$ (der Formelmenge $\mathcal{F}$)."
„Der Schluss von $F_1, \ldots, F_n$ ($\mathcal{F}$) auf G ist gültig."

Spezialfall $\mathcal{F} = \{\}$: $\models G$ bedeutet „G ist gültig".

Satz

G folgt aus $\mathcal{F}$ genau dann, wenn $\mathcal{F} \cup \{\neg G\}$ unerfüllbar ist.

Deduktionstheorem

Deduktionstheorem (semantische Version)

G folgt aus $F_1, \ldots, F_n$ genau dann, wenn $F_n \supset G$ aus $F_1, \ldots, F_{n-1}$ folgt:
$F_1, \ldots, F_n \models G$ genau dann, wenn $F_1, \ldots, F_{n-1} \models F_n \supset G$.

Mehrfache Anwendung liefert:

$F_1, \ldots, F_n \models G$ genau dann, wenn $F_1 \supset (F_2 \supset \cdots (F_n \supset G) \cdots)$ gültig.

Wegen $A \supset (B \supset C) \equiv (A \wedge B) \supset C$ erhalten wir weiters:

$F_1, \ldots, F_n \models G$ genau dann, wenn $(F_1 \wedge \cdots \wedge F_n) \supset G$ gültig.

Definition (Logische Äquivalenz)

F und G sind genau dann äquivalent ("$F \equiv G$"), wenn $F \models G$ und $G \models F$.

Aus Deduktionstheorem folgt:
$F \equiv G$ gdw. $\models (F \supset G) \wedge (G \supset F)$ bzw. $\models F \leftrightarrow G$.

Unvollständige Schlüsse

Achtung: "Schlüssige Argumente" werden üblicherweise erst dann zu logisch gültigen Schlüssen, wenn man die Prämissen (= explizite Annahmen) durch implizite Annahmen (versteckte Prämissen) ergänzt.

Beispiel:

"Alfred wurde von einer Frau erschossen. Daher ist Hans nicht sein Mörder." ist informell schlüssig, aber (noch) kein logisch gültiger Schluss.

Implizite Annahmen:
– "sein" bezieht sich hier auf Alfred.
– Hans ist keine Frau.
– Alle, die jemanden erschießen, sind Mörder der erschossenen Person.
– Alfred wurde von genau einer Person ermordet.

Formalisiert:
$\exists x(F(x) \wedge S(x,a)),\ \neg F(h),\ \forall x \forall y(S(x,y) \supset M(x,y)),$
$\exists x[M(x,a) \wedge \forall y(M(y,a) \supset y = x)] \models \neg M(h,a)$

Ist diese Konsequenzbehauptung gültig? Beweis? Widerlegung?

Wichtige logische Äquivalenzen

Wichtige aussagenlogische Äquivalenzen [WH]:

- $A \supset B \equiv \neg A \vee B$
- $A \supset (B \supset C) \equiv (A \wedge B) \supset C$
- $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ [Distribution von $\vee$ über $\wedge$]
- $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ [Distribution von $\wedge$ über $\vee$]
- $\neg(A \wedge B) \equiv \neg A \vee \neg B$, $\neg(A \vee B) \equiv \neg A \wedge \neg B$ [De Morgansche Gesetze]

Wichtige prädikatenlogische Äquivalenzen:

- $\neg\forall x F \equiv \exists x \neg F$, $\neg\exists x F \equiv \forall x \neg F$, $\neg\forall x \neg F \equiv \exists x F$, $\neg\exists x \neg F \equiv \forall x F$
- $\forall x \forall y F \equiv \forall y \forall x F$, $\exists x \exists y F \equiv \exists y \exists x F$
- $\forall x(F \wedge G) \equiv \forall x F \wedge \forall x G \equiv \forall x F \wedge \forall y G(x/y)$
- $\exists x(F \vee G) \equiv \exists x F \vee \exists x G \equiv \exists x F \vee \exists y G(x/y)$
- $\forall x(F^{-x} \supset G) \equiv F \supset \forall x G$, $\exists x(F^{-x} \supset G) \equiv F \supset \exists x G$
  F^{-x} bedeutet: x kommt in F nicht frei vor ($x \notin FV(F)$)
- $\forall x(F \supset G^{-x}) \equiv \exists x F \supset G^{-x}$, $\exists x(F \supset G^{-x}) \equiv \forall x F \supset G^{-x}$,

Beachte: Bislang steht uns zum Nachweis dieser (PL-)Äquivalenzen nur der direkte Bezug auf die Definition der PL-Semantik offen.
Syntaktische Beweise (Herleitungen) in einem Kalkül sind aber einfacher!

Axiomatische Theorien

[Zur Erinnerung:] Eine Theorie ist einfach eine Menge von Formeln.
Eine axiomatische Theorie $\mathcal{T}$ ist gegeben durch eine Mengen von Axiomen $\mathcal{A}$, sodass $\mathcal{T} = \{F \mid \mathcal{A} \models F\}$. Man sagt: $\mathcal{A}$ axiomatisiert $\mathcal{T}$.

Beachte: Man könnte jede Theorie $\mathcal{T}$ einfach durch $\mathcal{T}$ selbst axiomatisieren. Aber man verlangt, dass die Axiommenge $\mathcal{A}$ möglichst endlich oder zumindest entscheidbar ist.

Axiome der Peano-Arithmetik *PA*

Signatur $\Sigma_{PA} = \langle\{\}, \{0\}, \{s, +, *\}, \rangle$ (s steht für 'successor')

- $\forall x\, x + 0 = x$
- $\forall x \forall y\, x + s(y) = s(x + y)$
- $\forall x\, x * 0 = 0$
- $\forall x \forall y\, x * s(y) = x * y + x$
- $\forall x\, 0 \neq s(x)$
- alle Σ_{PA}-Formeln der Form $[F(0) \wedge \forall x(F(x) \supset F(s(x)))] \supset \forall x F(x)$

Peano-Arithmetik ist nicht endlich axiomatisierbar!

Axiomatische Theorien (Forts.)

Gruppentheorie

Signatur $\Sigma_G = \langle\{\}, \{e\}, \{\circ\}\rangle$

- $\forall x \forall y \forall z\, (x \circ y) \circ z = x \circ (y \circ z)$
- $\forall x\, x \circ e = x$
- $\forall x \exists y\, x \circ y = e$

NB: Die Modelle der Gruppentheorie — also diejenigen Interpretationen, in denen diese drei Gruppenaxiome wahr sind — heißen Gruppen.

Axiome der Totalordnung

Signatur $\Sigma_O = \langle\{\preceq\}, \{\}, \{\}\rangle$

- $\forall x \forall y \forall z ((x \preceq y \wedge y \preceq z) \supset x \preceq z)$ [Transitivität]
- $\forall x\, x \preceq x$ [Reflexivität]
- $\forall x \forall y ((x \preceq y \wedge y \preceq x) \supset x = y)$ [Antisymmetrie]
- $\forall x \forall y (x \preceq y \vee y \preceq x)$ [Totalität]

Logische Unabhängigkeit

Definition (Unabhängigkeit einer Formel von anderen Formeln)

Eine Formel G heißt unabhängig von einer Menge von Formeln $\mathcal{F}$ wenn weder $\mathcal{F} \models G$ noch $\mathcal{F} \models \neg G$ gilt.

Überlege: Wie kann man Unabhängigkeit nachweisen?
Allgemein: Wie weist man nach, dass $\mathcal{F} \models G$ nicht gilt ($\mathcal{F} \not\models G$)?

Antwort: Indem man ein Modell von $\mathcal{F}$ angibt in dem G falsch ist.
Damit ist auch die erste Frage beantwortet: Man gibt zwei Modelle $\mathcal{I}$, $\mathcal{I}'$ an, sodass $\mathrm{val}_{\mathcal{I}}(G) = \mathbf{f}$ und $\mathrm{val}_{\mathcal{I}'}(G) = \mathbf{t}$.

Beispiel: Das letztgenannte Axiom von $Th(\mathbb{S})$ — $\forall x\, \underline{\mathrm{push0}}(x) \neq \underline{\mathrm{push1}}(x)$ — ist unabhängig von der Menge aller anderen Axiome.

Betrachte folgende Interpretation $\mathcal{I} = \langle D, \Phi, \xi\rangle$ über $\Sigma(\mathbb{S})$: $D = \omega$;
$\Phi(\underline{\mathrm{istleer?}})$="$= 0$", $\Phi(\underline{\mathrm{ist0?}}) = \Phi(\underline{\mathrm{ist1?}})$="$> 0$"; $\Phi(\underline{\varepsilon}) = 0$;
$\Phi(\underline{\mathrm{push0}})(n) = \Phi(\underline{\mathrm{push1}})(n) = n + 1$, $\Phi(\underline{\mathrm{pop}})(n) = n \dot{-} 1$

Überprüfe: $\mathcal{I}$ erfüllt alle Axiome, außer $\forall x \forall y\, \underline{\mathrm{push0}}(x) \neq \underline{\mathrm{push1}}(y)$.
Hingegen erfüllt $\mathbb{S}$ alle Axiome, inklusive $\forall x \forall y\, \underline{\mathrm{push0}}(x) \neq \underline{\mathrm{push1}}(y)$.

Theorien von konkreten Strukturen

Definition (Theorie einer (Modell-)Struktur)

Für jede Struktur $\mathcal{I}$ heißt $Th(\mathcal{I}) = \{F \mid \mathrm{val}_{\mathcal{I}}(F) = \mathbf{t}\}$ die Theorie von $\mathcal{I}$.

Axiomatisierung von $Th(\mathbb{S})$ – Theorie der binären Stäcks

- $\underline{\mathrm{istleer?}}(\underline{\varepsilon})$, $\forall x (\underline{\mathrm{istleer?}}(x) \supset x = \underline{\varepsilon})$, $\underline{\mathrm{pop}}(\underline{\varepsilon}) = \underline{\varepsilon}$
- $\forall x \forall y (\underline{\mathrm{push0}}(x) = \underline{\mathrm{push0}}(y) \supset x = y)$,
  $\forall x \forall y (\underline{\mathrm{push1}}(x) = \underline{\mathrm{push1}}(y) \supset x = y)$
- $\forall x\, \neg \underline{\mathrm{istleer?}}(\underline{\mathrm{push0}}(x))$, $\forall x\, \neg \underline{\mathrm{istleer?}}(\underline{\mathrm{push1}}(x))$
- $\forall x \exists y (\underline{\mathrm{ist0?}}(x) \supset x = \underline{\mathrm{push0}}(y))$, $\forall x \exists y (\underline{\mathrm{ist1?}}(x) \supset x = \underline{\mathrm{push1}}(y))$
- $\forall x\, \underline{\mathrm{pop}}(\underline{\mathrm{push0}}(x)) = x$, $\forall x\, \underline{\mathrm{pop}}(\underline{\mathrm{push1}}(x)) = x$
- $\forall x (\underline{\mathrm{istleer?}}(x) \vee \underline{\mathrm{ist0?}}(x) \vee \underline{\mathrm{ist1?}}(x))$
- $\forall x\, \underline{\mathrm{push0}}(x) \neq x$, $\forall x\, \underline{\mathrm{push1}}(x) \neq x$
- $\forall x \forall y\, \underline{\mathrm{push0}}(x) \neq \underline{\mathrm{push1}}(y)$

Achtung: Viele konkrete mathematischen Strukturen (z.B. $\mathbb{N}$ und $\mathbb{Z}$) lassen sich nicht vollständig axiomatisieren (Gödels Unvollständigkeitssatz).

Theorien als Wissensbasen

Eine (endliche) Menge von Formeln – also eine Theorie – spielt oft die Rolle einer Knowledge Base (KB) (Wissensbasis).

$F \in KB$ ... der Agent hat direkte Evidenz, dass F gilt
$KB \models F$ ... der Agent kann (potentiell) auf F schließen

Beachte: Wir wissen zwar genau was $KB \models F$ heißt, haben aber noch keine Inferenz-Methoden besprochen. D.h., es bleibt zu klären wie der Agent F aus seiner KB herleiten kann. M.a.W.: Es fehlt noch das syntaktische Gegenstück ("$\vdash$" für Herleitung) zum semantischen "$\models$" (Konsequenz).

Theorien – Zusammenfassung:

(Mengen von) Formeln (also: Theorien) kommen in vielen Rollen vor:

- Gegenstücke (Präzisierungen) natürlichsprachliche Information
- formale Spezifikationen (z.B. Datentypen, Programmeigenschaften)
- Axiome von formalen Theorien
- relationale Datenbanken
- Wissensbasen von Agenten
- ... (und viele mehr!)

Logische Kalküle

Ein (logischer) Kalkül[a] $\mathcal{K}$ ist ein formales Regelssystem, das es erlaubt schrittweise und mechanisch, nur durch Bezugnahme auf die Syntax die Gültigkeit von Formeln bzw. von Konsequenzbehauptungen zu zeigen.
Wir schreiben: $\vdash_{\mathcal{K}} G$ bzw. $F_1, \ldots F_n \vdash_{\mathcal{K}} G$.

[a] Das Wort "Kalkül" ist in der hier verwendeten Bedeutung männlich; im Unterschied zu "Kalkül" in "etwas ins Kalkül ziehen" (sächlich)

Zwei zentrale Eigenschaften von Kalkülen:

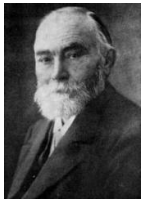
Korrektheit von $\mathcal{K}$:

Wenn $F_1, \ldots F_n \vdash_{\mathcal{K}} G$, dann $F_1, \ldots F_n \models G$.

Vollständigkeit von $\mathcal{K}$:

Wenn $F_1, \ldots F_n \models G$, dann $F_1, \ldots F_n \vdash_{\mathcal{K}} G$.

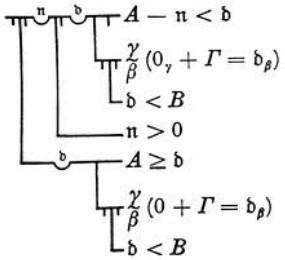
Erinnerung an den "Vater klassischer Logik-Kalküle"



Gottlob Frege (1848 – 1925). Deutscher Mathematiker und Philosoph. Einer der Begründer der modernen Logik. Lang ignoriert; später von Russell, Wittgenstein, Hilbert als wichtiger Wegbereiter erkannt.

Hauptwerke: *Begriffsschrift*, *Grundgesetze der Arithmetik*

Eine Kostprobe zur *Begriffsschrift*:



"Das Verhältnis meiner Begriffsschrift zu der Sprache des Lebens glaube ich am deutlichsten machen zu können, wenn ich es mit dem des Mikroskops zum Auge vergleiche."
(Begriffsschrift, Vorwort, 1879)

Der Tableau-Kalkül – Teil 1: Semantische AL-Tableaux

Der Tableau-Kalkül (E. Beth) ist eine Variante des Sequentialkalküls.

Grundidee:

Indirekter Beweis von Konsequenzbehauptung $F_1, \ldots, F_n \models G$:
Annahme, dass $\mathrm{val}_I(F_1) = \mathbf{t}, \ldots, \mathrm{val}_I(F_n) = \mathbf{t}, \mathrm{val}_I(G) = \mathbf{f}$ für ein I.

Man folgert nun schrittweise und systematisch was aus diesen Annahmen für die unmittelbaren Teilformeln folgt.
Dabei ergeben sich zwei Typen von Folgerungen (auf der Meta-Ebene):
– Konjunktion von neuen (kleineren) Behauptungen
– Disjunktion von neuen (kleineren) Behauptungen.

Tableau-Beweis von $(A \supset B) \wedge A \models B$

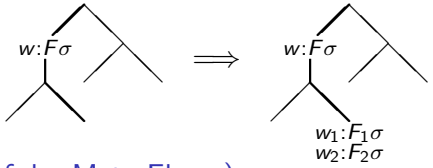
(1) $\mathbf{t} : (A \supset B) \wedge A$		Annahme
(2) $\mathbf{f} : B$		Annahme
(3) $\mathbf{t} : A \supset B$		Zerlegung von (1)
(4) $\mathbf{t} : A$		Zerlegung von (1)
(5) $\mathbf{f} : A$	(6) $\mathbf{t} : B$	Zerlegung von (3)
$\times$	$\times$	Widerspruch zu (4) bzw. (2)

Tableau-Expansion

Tableau-Regeln legen fest, wie ein Baum expandiert werden kann.

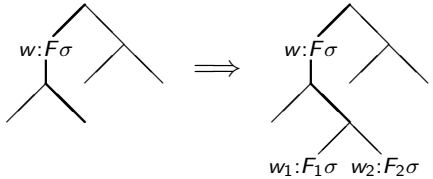
α-Regeln (Konjunktion auf der Meta-Ebene):

$$\frac{w : F}{\begin{array}{c} w_1 : F_1 \\ w_2 : F_2 \end{array}}$$



β-Regeln (Disjunktion auf der Meta-Ebene):

$$\frac{w : F}{w_1 : F_1 \mid w_2 : F_2}$$



w, w_1, w_2	...	Wahrheitswerte
F, F_1, F_2	...	Formeln
$F\sigma$	...	F nach Anwendung der Substitution σ
$w : F$	...	bewertete Formeln (*signed formulas*)

Regeln des (aussagenlogischen) Tableau-Kalküls

Wir arbeiten in einer eingeschränkten Syntax: nur $\vee, \wedge, \supset, \neg$

$$\frac{\mathbf{f}: A \vee B}{\begin{array}{c}\mathbf{f}: A \\ \mathbf{f}: B\end{array}} \qquad \frac{\mathbf{t}: A \vee B}{\mathbf{t}: A \mid \mathbf{t}: B} \qquad \frac{\mathbf{f}: A \wedge B}{\mathbf{f}: A \mid \mathbf{f}: B} \qquad \frac{\mathbf{t}: A \wedge B}{\begin{array}{c}\mathbf{t}: A \\ \mathbf{t}: B\end{array}}$$

$$\frac{\mathbf{f}: A \supset B}{\begin{array}{c}\mathbf{t}: A \\ \mathbf{f}: B\end{array}} \qquad \frac{\mathbf{t}: A \supset B}{\mathbf{f}: A \mid \mathbf{t}: B} \qquad \frac{\mathbf{f}: \neg A}{\mathbf{t}: A} \qquad \frac{\mathbf{t}: \neg A}{\mathbf{f}: A}$$

Beachte:

- Je zwei Regeln: eine für $\mathbf{f}: A * B$, eine für $\mathbf{t}: A * B$ ($\mathbf{f}: \neg A$ / $\mathbf{t}: \neg A$).
- Regeln aus den Wahrheitsfunktionen (*or*, *and*, *implies*, *not*) ablesbar.
- Beweise sind auf den Kopf gestellte Bäume (*downward trees*).
- Jede dieser Expansionsregeln muss auf jedes Vorkommen einer bewerteten Formel nur einmal angewendet werden.
  $\Longrightarrow$ Jeder Beweisversuch terminiert.

Korrektheit und Vollständigkeit des Tableau-Kalküls (1)

- Ein Tableau-Ast heißt geschlossen[a], wenn auf ihm für eine Formel F sowohl $\mathbf{t}{:}F$ als auch $\mathbf{f}{:}F$ vorkommt (Abschlussregel).
- Ein Tableau heißt geschlossen, wenn alle Äste geschlossen sind.
- Ein nicht geschlossener/s Ast/Tableau heißt offen.
- Ein Tableau-Beweis einer Konsequenzbehauptung $F_1, \ldots, F_n \models G$ ist ein geschlossenes Tableau, das mit den Annahmen $\mathbf{t}{:}F_1, \ldots, \mathbf{t}{:}F_n, \mathbf{f}{:}G$ beginnt. Spezialfall: Ein Tableau-Beweis [der Gültigkeit] von F ist ein geschlossenes Tableau mit Wurzel $\mathbf{f}{:}F$.

[a] "geschlossen" hat hier nichts mit geschlossenen Formeln zu tun!

Satz (Korrektheit):

Wenn ein entsprechender Tableau-Beweis existiert, dann $F_1, \ldots, F_n \models G$.

Satz (Vollständigkeit):

Wenn $F_1, \ldots, F_n \models G$, dann existiert ein entsprechender Tableau-Beweis.

Korrektheit und Vollständigkeit des Tableau-Kalküls (2)

Lemma

- Regel für $\mathbf{f}{:}\vee$: $\mathrm{val}_I(A \vee B) = \mathbf{f} \iff \mathrm{val}_I(A) = \mathbf{f}$ und $\mathrm{val}_I(B) = \mathbf{f}$
- Regel für $\mathbf{t}{:}\vee$: $\mathrm{val}_I(A \vee B) = \mathbf{t} \iff \mathrm{val}_I(A) = \mathbf{t}$ oder $I(B) = \mathbf{t}$
- ... (analog für alle anderen Regeln)

Vollständigkeitsbeweis (Skizze):

Indirekt: Wenn ein vollständig expandiertes Tableau mit den Annahmen $\mathbf{t}{:}F_1, \ldots, \mathbf{t}{:}F_n, \mathbf{f}{:}G$ offen ist, so gibt es ein Gegenbeispiel zu $F_1, \ldots, F_n \models G$.

Sei Γ ein vollständig expandierter und offener Ast.
Es folgt: für keine Aussagenvariable A liegen sowohl $\mathbf{f}{:}A$ als auch $\mathbf{t}{:}A$ auf Γ.
Daher gibt es eine Interpretation I, sodass $I(A) = \mathbf{t}\,[\mathbf{f}]$ falls $\mathbf{t}{:}A\,[\mathbf{f}{:}A]$ auf Γ.
Wegen des Lemmas: $\mathrm{val}_I(F) = \mathbf{t}\,[\mathbf{f}]$ für alle $\mathbf{t}{:}F\,[\mathbf{f}{:}F]$ auf Γ. (Induktion!)
Insbesondere $\mathrm{val}_I(F_1) = \mathbf{t}, \ldots, \mathrm{val}_I(F_n) = \mathbf{t}$ und $\mathrm{val}_I(G) = \mathbf{f}$.
Es gibt also ein Gegenbeispiel zu $F_1, \ldots, F_n \models G$, QED.

Korrektheit und Vollständigkeit des Tableau-Kalküls (3)

Korrektheitsbeweis (Skizze):

Die Korrektheit des Tableau-Kalküls ist äquivalent zu:

Wenn ein Tableau mit den Annahmen $\mathbf{t}{:}F_1, \ldots, \mathbf{t}{:}F_n, \mathbf{f}{:}G$ geschlossen ist, dann gibt es kein Gegenbeispiel I zu $F_1, \ldots, F_n \models G$.

Wir gehen wieder indirekt vor und nehmen an, dass es doch so ein I gibt.
Es gelte also $\mathrm{val}_I(F_1) = \mathbf{t}, \ldots, \mathrm{val}_I(F_n) = \mathbf{t}$ und $\mathrm{val}_I(G) = \mathbf{f}$.
Wegen des Lemmas gilt dann aber für jeden Ast Γ des Tableau:
$\mathrm{val}_I(F) = \mathbf{t}\,[\mathbf{f}]$ wenn $\mathbf{t}{:}F\,[\mathbf{f}{:}F]$ auf Γ liegt.
Da das Tableau geschlossen, ist auch jeder Ast Γ geschlossen.
Es gibt also zu jedem Γ ein F, sodass $\mathbf{t}{:}F$ und gleichzeitig $\mathbf{f}{:}F$ auf Γ liegen.
Das bedeutet, dass das angenommene Gegenbeispiel I nicht existiert. QED.

Folgerungen:

- Entscheidungsverfahren:
  Vollständige Expansion liefert einen Beweis oder ein Gegenbeispiel.
- *Don't care*-Indeterminismus:
  Ergebnis ist unabhängig von der Reihenfolge der Regelanwendungen.

AL-Tableaux – Beispiele (1)

Tableau-Beweis von $(A \vee B) \wedge \neg A \models B$

(1)	$\mathbf{t} : (A \vee B) \wedge \neg A$	Annahme
(2)	$\mathbf{f} : B$	Annahme
(3)	$\mathbf{t} : A \vee B$	von 1
(4)	$\mathbf{t} : \neg A$	von 1
(5)	$\mathbf{t} : A$ von 3 | (6) $\mathbf{t} : B$ von 3	
(7)	$\mathbf{f} : A$ von 4 | $\times$ (Widerspruch: 6/2)	
	$\times$ (Widerspruch: 5/7)	

Tableau-Beweis von $\neg(\neg(A \vee C) \supset B) \vee (\neg(A \vee C) \supset B)$

(1)	$\mathbf{f} : \neg(\neg(A \vee C) \supset B) \vee (\neg(A \vee C) \supset B)$	Annahme
(2)	$\mathbf{f} : \neg(\neg(A \vee C) \supset B)$	von 1
(3)	$\mathbf{f} : \neg(A \vee C) \supset B$	von 1
(4)	$\mathbf{t} : \neg(A \vee C) \supset B$	von 2
	$\times$ (Widerspruch: 3/4)	

AL-Tableaux – Beispiele (2)

Tableau für $((A \vee B) \wedge A) \supset B$

(1)	$\mathbf{f} : ((A \vee B) \wedge A) \supset B$	Annahme
(2)	$\mathbf{t} : (A \vee B) \wedge A$	von 1
(3)	$\mathbf{f} : B$	von 1
(4)	$\mathbf{t} : A \vee B$	von 2
(5)	$\mathbf{t} : A$	von 2
(6)	$\mathbf{t} : A$ von 4 | (7) $\mathbf{t} : B$ von 4	
	Ast bleibt offen! | $\times$	

Dem offenen Ast kann man entnehmen:
I mit $I(A) = \mathbf{t}$ und $I(B) = \mathbf{f}$ ist ein Gegenbeispiel zu $((A \vee B) \wedge A) \supset B$.

Generell gilt:
Jede Abfolge von Regelanwendungen führt letztlich

- entweder auf ein geschlossenes Tableau (Beweis)
- oder auf einen offenen Ast (Gegenbeispiel).

AL-Tableaux – Beispiele (3)

Beweis oder Gegenbeispiel finden mit Tableaux:

(1)	$\mathbf{f} : (A \wedge (B \supset C)) \supset ((B \vee C) \wedge (\neg A \vee (D \supset E)))$	Annahme
(2)	$\mathbf{t} : A \wedge (B \supset C)$	von 1
(3)	$\mathbf{f} : (B \vee C) \wedge (\neg A \vee (D \supset E))$	von 1
(4)	$\mathbf{f} : B \vee C$ von 3 | (5) $\mathbf{f} : \neg A \vee (D \supset E)$ von 3	
(6)	$\mathbf{t} : A$ von 2 | dieser Ast muss nicht mehr weiter expandiert werden	
(7)	$\mathbf{t} : B \supset C$ von 2	
(8)	$\mathbf{f} : B$ von 4	
(9)	$\mathbf{f} : C$ von 4	
(10)	$\mathbf{f} : B$ von 7 | (11) $\mathbf{t} : C$ von 7	
	$\Uparrow$ | $\times$ Wid. (9/11)	

Der mit $\Uparrow$ gekennzeichnete Ast kann nicht weiter expandiert werden. Da sich auf diesem Ast kein Widerspruch findet, ist die Interpretation I mit $I(A) = \mathbf{t}, I(B) = \mathbf{f}, I(C) = \mathbf{f}$ ein Gegenbeispiel zur Ausgangsformel ist.

Semantische AL-Tableaux – Zusammenfassung

- Die Regeln des Tableau-Kalküls beziehen sich direkt auf die Semantik der logischen Konnektive. Sie können von den entsprechenden Wahrheitstafeln abgelesen werden kann.
- Tableau: Baum von bewerteten Formeln als Tafel[1] geschrieben
- Geschlossene Tableaux entsprechen indirekten Beweisen.
- Aus vollständig expandierten offenen Tableaux kann man Gegenmodelle zur entsprechenden Konsequenzbehauptung ablesen.
- Tableaux lassen sich als Varianten von Beweisen im Gentzen'schen Sequentialkalül verstehen.
- Vorteile:
  - Einfache Beweise einfacher Konsequenzbeziehungen.
  - Vollständigkeit genau so einfach wie Korrektheit beweisbar.
  - Beweissuche einfach automatisierbar und (relativ) effizient.
  - Liefert ein Entscheidungsverfahren mit expliziter Beweis- bzw. Gegenbeispielskonstruktion.

[1] Tableau/Tableaux (EZ/MZ) heißt Tafel/Tafeln auf Französisch

Ein Frege-Hilbert-Typ-Kalkül für die klassische PL

Axiome:
1. $(A \supset (B \supset A))$
2. $((A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C)))$

(3. – 9. ...)

10. $A(x/t) \supset \exists x A$ ([1])
11. $\forall x A \supset A(x/t)$ ([1])

Regeln:

$$\frac{A \quad A \supset B}{B} \qquad \frac{B \supset A}{B \supset \forall x A}\ (^2) \qquad \frac{A \supset B}{\exists x A \supset B}\ (^2)$$

Seitenbedingungen:

([1]): t darf keine Variable enthalten, die in $\exists x A / \forall x A$. gebunden wird
Man sagt auch: t muss in A frei für x sein

([2]): x darf in B nicht frei vorkommen ($x \notin FV(B)$)

Nachteile: (wie schon für AL)

- Vollständigkeit ist nicht einfach nachzuweisen (Gödels Dissertation!)
- Undurchsichtige Beweise auch von einfachen Formeln
- Ableitungen entsprechen nicht typischen Argumenten
- Keine direkte Beziehung zur Semantik
- Beweissuche ist schwer zu automatisieren

Tableau-Kalkül – Teil 2: Semantische PL-Tableaux ohne Gleichheit

Aufgabenstellung: Erweiterung von Tableaux von AL auf PL.
Wir benötigen geeignete Regeln zur Analyse von Quantoren.
Wichtig: Wir beschränken uns auf geschlossene Formeln ($FV(F) = \{\}$)!

Zur Erinnerung:
$\mathbf{t} : \forall x F$ entspricht $\mathrm{val}_{\mathcal{I}}(\forall x F) = \mathbf{t}$ für ein $\mathcal{I}$. Daher: F ist wahr in allen Interpretation $\mathcal{I}' \overset{x}{\sim} \mathcal{I}$. Daher auch $\mathrm{val}_{\mathcal{I}}(F(x/t)) = \mathbf{t}$ für alle geschlossenen Terme t (sogar für alle t, die frei für x in F sind).
Ähnlich: $\mathbf{f} : \exists x F \implies F(x/t)$ ist für alle geschlossenen t falsch.

Diese Überlegung suggeriert folgende Tableau-Expansionsregeln:

$$\frac{\mathbf{t} : \forall x F}{\mathbf{t} : F(x/t)} \qquad \frac{\mathbf{f} : \exists x F}{\mathbf{f} : F(x/t)}$$

für beliebige geschlossene (also: variablenfreie) Terme t

Problem: Wir können nicht garantieren, dass alle Elemente des Gegenstandsbereichs durch einen Term repräsentiert werden.

Analyse der existentiellen Quantifikation:

$\mathrm{val}_{\mathcal{I}}(\exists x F) = \mathbf{t}$ bedeutet, dass F in mindestens einer Interpretationen $\mathcal{I}' \overset{x}{\sim} \mathcal{I}$ wahr wird. (Analog für $\mathbf{f} : \forall x F$)

Problem:
Wir wissen nicht, welchen Term t wir für x substituieren dürfen.
Noch schlimmer: wir können nicht garantieren, dass es überhaupt einen geschlossenen Term (in der Signatur zu F) gibt, der einen entsprechenden "Zeugen" für $\exists x F$ repräsentiert.

Lösung der Probleme (vgl. Satz von Löwenheim-Skolem)

1. Erweiterung der Sprache um (abzählbar) unendlich viele zusätzliche Konstanten, genannt: Parameter.
2. Bei der Zerlegung von $\mathbf{t} : \exists x F$ und $\mathbf{f} : \forall x F$ verlangt man, dass der jeweilige "Zeuge" für F neu ist, d.h., im Tableau zum Zeitpunkt der Regelanwendung noch nicht vorkommt.
(Vergleiche mit den Seitenbedingungen der Hilbert-Typ-Regeln)

Erinnerung: Wir wenden die Regeln nur auf geschlossene Formeln an.
Daher gibt es in einem PL-Tableau niemals freie Variablen!

Quantoren-Regeln des Tableau-Kalküls

Σ^{par} ... Signatur Σ erweitert um Parameter $c, d, e, \ldots$
$\mathbf{t} : \forall x F$ und $\mathbf{f} : \exists x F$ heißen γ-Formeln
$\mathbf{f} : \forall x F$ und $\mathbf{t} : \exists x F$ heißen δ-Formeln

γ-Regeln:

$$\frac{\mathbf{t} : \forall x F}{\mathbf{t} : F(x/t)} \qquad \frac{\mathbf{f} : \exists x F}{\mathbf{f} : F(x/t)}$$

für beliebige geschlossene (=variablenfreie) Terme t über Σ^{par}

δ-Regeln:

$$\frac{\mathbf{f} : \forall x F}{\mathbf{f} : F(x/c)} \qquad \frac{\mathbf{t} : \exists x F}{\mathbf{t} : F(x/c)}$$

für einen neuen Parameter c in Σ^{par}

Die α- und β-Regeln (AL) bleiben unverändert!

Achtung: γ-Regeln müssen i.A. auf die selbe γ-Formel öfters, mit verschiedenen t angewendet werden. (Im Gegensatz zur δ-Regel.)
Überlege: Warum ist das so?

Beispiele von PL-Formeln [WH]

Folgende Formeln sind gültig:

- $\forall x[P(x) \vee \neg P(x)]$ bzw. auch $P(x) \vee \neg P(x)$
- $\forall x P(x) \vee \neg \forall y P(y)$ und $\forall x[P(x) \supset \exists y P(y)]$
- $\neg \forall x P(x) \supset \exists x \neg P(x)$ und $\neg \exists x P(x) \supset \forall x \neg P(x)$
- $\forall x \forall y Q(x,y) \supset \forall y \forall x Q(x,y)$

Folgende Formeln sind unerfüllbar:

- $\forall x[P(x) \wedge \neg P(x)]$ bzw. auch $P(x) \wedge \neg P(x)$
- $\neg[\forall x P(x) \vee \neg \forall y P(y)]$ und $\exists x P(x) \wedge \forall x \neg P(x)$

Folgende Formeln sind erfüllbar und widerlegbar:

- $\exists x P(x) \supset \forall x P(x)$ und $P(x) \supset P(y)$ und $P(x) \vee \neg P(y)$
- $\forall x P(x) \vee \forall x \neg P(x)$ und $\exists x P(x) \supset \forall x \neg P(x)$
- $\forall x \forall y[Q(x,y) \supset Q(y,x)]$ und $\forall x[\forall y Q(x,y) \supset \forall y Q(y,x)]$

Überlege: Was ist zum Nachweis dieser Behauptungen jeweils nötig?

Übungsaufgabe: Semantischer Status von $\exists x[P(x) \supset \forall y P(y)]$?

PL-Tableaux – Beispiele (1)

Gültigkeit von $\forall x[P(x) \vee \neg P(x)]$

(1)	$\mathbf{f} : \forall x[P(x) \vee \neg P(x)]$	Annahme (δ-Formel)
(2)	$\mathbf{f} : P(a) \vee \neg P(a)$	von 1
(3)	$\mathbf{f} : P(a)$	von 2
(4)	$\mathbf{f} : \neg P(a)$	von 2
(5)	$\mathbf{t} : P(a)$	von 4
	$\times$	Wid.: 3/5

Unerfüllbarkeit von $\exists x P(x) \wedge \forall x \neg P(x)$

(1)	$\mathbf{t} : \exists x P(x) \wedge \forall x \neg P(x)$	Annahme (weder γ noch δ!)
(2)	$\mathbf{t} : \exists x P(x)$	von 1 (δ-Formel)
(3)	$\mathbf{t} : \forall x \neg P(x)$	von 1 (γ-Formel)
(4)	$\mathbf{t} : P(a)$	von 2
(5)	$\mathbf{t} : \neg P(a)$	von 3
(6)	$\mathbf{f} : P(a)$	von 5
	$\times$	Wid.: 4/6

PL-Tableaux – Beispiele (2)

Gültigkeit von $\exists x[P(x) \supset \forall y P(y)]$

(1)	$\mathbf{f} : \exists x[P(x) \supset \forall y P(y)]$	Annahme (γ-Formel)
(2)	$\mathbf{f} : P(a) \supset \forall y P(y)$	von 1
(3)	$\mathbf{t} : P(a)$	von 2
(4)	$\mathbf{f} : \forall y P(y)$	von 2 (δ-Formel)
(5)	$\mathbf{f} : P(b)$	von 4
(6)	$\mathbf{f} : P(b) \supset \forall y P(y)$	von 1
(7)	$\mathbf{t} : P(b)$	von 6
(8)	$\mathbf{f} : \forall y P(y)$	von 6 (δ-Formel)
	$\times$	Wid.: 5/7

Beachte:
In Zeile 5 musste ein neuer Parameter gewählt werden!
Entsprechend muss die γ-Regel ein zweites mal auf 1 angewendet werden um das Tableau schließen zu können.

PL-Tableaux – Beispiele (3)

Gültigkeit von $\forall x P(x) \supset [P(a) \wedge P(f(a))]$

(1)	$\mathbf{f} : \forall x P(x) \supset [P(a) \wedge P(f(a))]$				Annahme
(2)	$\mathbf{t} : \forall x P(x)$				von 1 (γ-Formel)
(3)	$\mathbf{f} : P(a) \wedge P(f(a))$				von 1
(4)	$\mathbf{f} : P(a)$	von 3	(5)	$\mathbf{f} : P(f(a))$	von 3
(6)	$\mathbf{t} : P(a)$	von 2	(7)	$\mathbf{t} : P(f(a))$	von 2
	$\times$	Wid.: 4/6		$\times$	Wid.: 5/7

Beachte:
Die γ-Regel muss auf die Formel $\mathbf{t} : \forall x P(x)$ in jedem der beiden Äste je einmal angewendet werden – und zwar mit jeweils verschiedenen Instanzen.

PL-Tableaux – Beispiele (4)

Gültigkeit von $\forall x[P(x) \vee Q(x)] \supset [\exists x P(x) \vee \forall x Q(x)]$

(1)	$\mathbf{f} : \forall x[P(x) \vee Q(x)] \supset [\exists x P(x) \vee \forall x Q(x)]$				Annahme
(2)	$\mathbf{t} : \forall x[P(x) \vee Q(x)]$				von 1 (γ-Formel)
(3)	$\mathbf{f} : \exists x P(x) \vee \forall x Q(x)$				von 1
(4)	$\mathbf{f} : \exists x P(x)$				von 3 (γ-Formel)
(5)	$\mathbf{f} : \forall x Q(x)$				von 3 (δ-Formel)
(6)	$\mathbf{f} : Q(c)$				von 5
(7)	$\mathbf{t} : P(c) \vee Q(c)$				von 2
(8)	$\mathbf{t} : P(c)$	von 7	(9)	$\mathbf{t} : Q(c)$	von 7
(10)	$\mathbf{f} : P(c)$	von 4		$\times$	Wid.: 6/9
	$\times$	Wid.: 8/10			

Eine sehr nützliche Heuristik:

"δ vor γ": immer wenn möglich zunächst die δ-Regel anwenden, da damit die Wahl geeigneter "Zeugen-Terme" für γ-Formeln unterstützt wird.

PL-Tableaux – Beispiele (5)

Gültigkeit von $\exists x \forall y R(x, y) \supset \forall y \exists x R(x, y)$

(1)	$\mathbf{f} : \exists x \forall y R(x, y) \supset \forall y \exists x R(x, y)$	Annahme
(2)	$\mathbf{t} : \exists x \forall y R(x, y)$	von 1 (δ-Formel)
(3)	$\mathbf{f} : \forall y \exists x R(x, y)$	von 1 (δ-Formel)
(4)	$\mathbf{t} : \forall y R(c, y)$	von 2 (γ-Formel)
(5)	$\mathbf{f} : \exists x R(x, d)$	von 3 (γ-Formel)
(6)	$\mathbf{t} : R(c, d)$	von 4
(7)	$\mathbf{f} : R(c, d)$	von 5
	$\times$	Wid.: 6/7

Beachte: In Zeile 5 musste ein neuer Parameter gewählt werden!

PL-Tableaux – Beispiele (6)

Die Formel $\forall x \exists y R(x, y) \supset \exists y \forall x R(x, y)$ ist widerlegbar.
Entsprechend kann folgendes Tableau auch nicht geschlossen werden:

(1)	$\mathbf{f} : \forall x \exists y R(x, y) \supset \exists y \forall x R(x, y)$	Annahme
(2)	$\mathbf{t} : \forall x \exists y R(x, y)$	von 1 (γ-Formel)
(3)	$\mathbf{f} : \exists y \forall x R(x, y)$	von 1 (γ-Formel)
(4)	$\mathbf{t} : \exists y R(a, y)$	von 2 (δ-Formel)
(5)	$\mathbf{f} : \forall x R(x, a)$	von 3 (δ-Formel)
(6)	$\mathbf{t} : R(a, b)$	von 4
(7)	$\mathbf{f} : R(c, a)$	von 5
(8)	$\mathbf{t} : \exists y R(c, y)$	von 2 (δ-Formel)
(9)	$\mathbf{t} : R(c, d)$	von 8
	$\vdots$	

Beachte: Es musste jeweils ein neuer Parameter gewählt werden.
Der Ast kann unendlich oft weiter expandiert werden, ohne dass er jemals abgeschlossen werden kann.

PL-Tableaux – Beispiele (7)

Beweis von $\forall x[P(x) \vee Q(f(x))], \neg \exists x P(x) \models \exists x Q(x)$

(1)	$\mathbf{t} : \forall x[P(x) \vee Q(f(x))]$				Ann. (γ-Formel)
(2)	$\mathbf{t} : \neg \exists x P(x)$				Ann.
(3)	$\mathbf{f} : \exists x Q(x)$				Ann. (γ-Formel)
(4)	$\mathbf{f} : \exists x P(x)$				von 2 (γ-Formel)
(5)	$\mathbf{t} : P(c) \vee Q(f(c))$				von 1
(6)	$\mathbf{t} : P(c)$	von 5	(7)	$\mathbf{t} : Q(f(c))$	von 5
(8)	$\mathbf{f} : P(c)$	von 4	(9)	$\mathbf{f} : Q(f(c))$	von 3
	$\times$	Wid.: 6/8		$\times$	Wid.: 7/9

Beachte: Da für die γ-Formel (1) in (5) ein geschlossener Term gewählt werden muss, muss hier eine neue Konstante (c) eingeführt werden.

PL-Tableaux – Beispiele (8)

Ein etwas interessantere Aufgabe:

Zeigen Sie mit dem Tableau-Kalkül, dass jede transitive symmetrische und serielle Relation auch reflexiv ist.
Also:

$$\mathit{trans},\ \mathit{sym},\ \mathit{ser} \models \mathit{refl}\ ,$$

wobei

$$\begin{aligned}
\mathit{trans} &= \forall x \forall y \forall z[(R(x,y) \land R(y,z)) \supset R(x,z)] \\
\mathit{sym} &= \forall x \forall y[R(x,y) \supset R(y,x)] \\
\mathit{ser} &= \forall x \exists y R(x,y) \\
\mathit{refl} &= \forall x R(x,x)
\end{aligned}$$

PL-Tableaux – Beispiele (8) [Forts.]

Tableau für *trans*, *sym*, *ser* $\models$ *refl*

(1)	$\mathbf{t} : \forall x \forall y \forall z[(R(x,y) \land R(y,z)) \supset R(x,z)]$	Ann.(γ)
(2)	$\mathbf{t} : \forall x \forall y[R(x,y) \supset R(y,x)]$	Ann.(γ)
(3)	$\mathbf{t} : \forall x \exists y R(x,y)$	Ann.(γ)
(4)	$\mathbf{f} : \forall x R(x,x)$	Ann.(δ)
(5)	$\mathbf{f} : R(c,c)$	von 4
(6)	$\mathbf{t} : \exists y R(c,y)$	von 3(δ)
(7)	$\mathbf{t} : R(c,d)$	von 6
(8)	$\mathbf{t} : \forall y[R(c,y) \supset R(y,c)]$	von 2(γ)
(9)	$\mathbf{t} : R(c,d) \supset R(d,c)$	von 8
(10)	$\mathbf{t} : \forall y \forall z[(R(c,y) \land R(y,z)) \supset R(c,z)]$	von 1 (γ)
(11)	$\mathbf{t} : \forall z[(R(c,d) \land R(d,z)) \supset R(c,z)]$	von 10(γ)
(12)	$\mathbf{t} : (R(c,d) \land R(d,c)) \supset R(c,c)$	von 11
(13)	$\mathbf{f}{:}R(c,d)$ v. 9 $\times$ (7/13) | (14) $\mathbf{t} : R(d,c)$	von 9
(15)	$\mathbf{f} : R(c,d) \land R(d,c)$ v. 12 | (16) $\mathbf{t}{:}R(c,c)$ $\times$ (5/16)	von 12
(17)	$\mathbf{f}{:}R(c,d)$ $\times$ (7/17) | (18) $\mathbf{f}{:}R(d,c)$ $\times$ (14/18)	

Tableau-Kalkül – Teil 3: Gleichheitsregeln für PL-Tableaux

Erinnerung: Wir wenden die Regeln nur auf geschlossene Formeln an.
Daher gibt es in einem PL-Tableau niemals freie Variablen!

Daher können atomare Formeln nur geschlossene (=variablenfreie) Terme enthalten. Für vorkommende "$\mathbf{t}[\mathbf{f}] : s = t$" gilt also $V(s) = V(t) = \{\}$!

Abschlussregel für Gleichheitsatome ($AB^{=}$):
Enthält ein Ast eine negative Formel $\mathbf{f} : t = t$ so wird er geschlossen.

Notation:
$A[s]$ ... A ist ein Atom, s ein bestimmtes Term-Vorkommen darin
$A[s/t]$ ... in $A[s]$ ausgewiesenes Vorkommen von s wurde durch t ersetzt

Substitutionsregeln für Gleichheitsatome ($S^{=}$):

$$\frac{\mathbf{t} : s = t \quad \mathbf{t} : A[s]}{\mathbf{t} : A[s/t]} \qquad \frac{\mathbf{t} : s = t \quad \mathbf{f} : A[s]}{\mathbf{f} : A[s/t]} \qquad \frac{\mathbf{t} : s = t \quad \mathbf{t} : A[t]}{\mathbf{t} : A[t/s]} \qquad \frac{\mathbf{t} : s = t \quad \mathbf{f} : A[t]}{\mathbf{f} : A[t/s]}$$

Achtung: A muss atomar sein, kann aber auch ein Gleichheitsatom sein.
s bzw. t kommen beliebig tief vor (nicht nur als Argument eines Prädikats).

PL-Tableaux mit Gleichheit – Beispiele (1)

Reflexivität, Symmetrie und Transitivität der Gleichheit

(1)	$\mathbf{f} : \forall x\, x = x$	Ann.(γ)
(2)	$\mathbf{f} : a = a$	von 1
	$\times$	$AB^{=}$

(1)	$\mathbf{t} : a = b$	Ann.
(2)	$\mathbf{f} : b = a$	Ann.
(3)	$\mathbf{f} : a = a$	$S^{=}$: 1→2
	$\times$	$AB^{=}$

(1)	$\mathbf{t} : a = b$	Ann.
(2)	$\mathbf{t} : b = c$	Ann.
(3)	$\mathbf{f} : a = c$	Ann.
(4)	$\mathbf{f} : b = c$	$S^{=}$: 1→3
(5)	$\mathbf{f} : c = c$	$S^{=}$: 2→4
	$\times$	$AB^{=}$

Beachte: In den Tableaux für Symmetrie und Transitivität könnten statt den Konstanten a, b, c beliebige Terme stehen.
Alternativ könnte man mit $\mathbf{f} : \forall x \forall y(x = y \supset y = x)$ bzw. mit $\mathbf{f} : \forall x \forall y \forall z[(x = y \land y = z) \supset x = z]$ beginnen.

PL-Tableaux mit Gleichheit – Beispiele (2)

$f(f(a)) = a,\ P(f(f(f(f(a))))) \models P(a)$

(1)	$\mathbf{t} : f(f(a)) = a$	Ann.
(2)	$\mathbf{t} : P(f(f(f(f(a)))))$	Ann.
(3)	$\mathbf{f} : P(a)$	Ann.
(4)	$\mathbf{t} : P(f(f(a)))$	$S^{=}$: 1→2
(5)	$\mathbf{t} : P(a)$	$S^{=}$: 1→4
	$\times$	Wid. 3/5

$a = b,\ Q(f(a), g(a,b)) \models Q(f(b), (g(b,b)))$

(1)	$\mathbf{t} : a = b$	Ann.(γ)
(2)	$\mathbf{t} : Q(f(a), g(a,b))$	Ann.
(3)	$\mathbf{f} : Q(f(b), g(b,b))$	Ann.
(4)	$\mathbf{t} : Q(f(b), g(a,b))$	$S^{=}$: 1→2
(5)	$\mathbf{t} : Q(f(b), (g(b,b)))$	$S^{=}$: 1→4
	$\times$	Wid. 3/5

Beachte: Man kann immer nur ein Vorkommen eines Terms auf einmal ersetzen!

PL-Tableaux mit Gleichheit – Beispiele (3)

$\forall x\, f(x) = g(x),\ Q(f(a), f(f(b))) \models Q(g(a), f(g(b)))$

(1)	$\mathbf{t} : \forall x\, f(x) = g(x)$	Ann.(γ)
(2)	$\mathbf{t} : Q(f(a), f(f(b)))$	Ann.
(3)	$\mathbf{f} : Q(g(a), f(g(b)))$	Ann.
(4)	$\mathbf{t} : f(a) = g(a)$	von 1
(5)	$\mathbf{t} : Q(g(a), f(f(b)))$	$S^{=}$: 4→2
(6)	$\mathbf{t} : f(b) = g(b)$	von 1
(7)	$\mathbf{t} : Q(g(a), f(g(b)))$	$S^{=}$: 6→5
	$\times$	Wid. 3/7

Beachte:
Es werden zwei verschiedene Instanzen von $\forall x\, f(x) = g(x)$ benötigt.

PL-Tableaux mit Gleichheit – Beispiele (4)

$\forall x \forall y [f(x) = y \supset \exists z\, y = g(z)],\ P(f(f(a)) \models \exists x\, P(g(x))$

(1)	$\mathbf{t} : \forall x \forall y [f(x) = y \supset \exists z\, y = g(z)]$		Annahme (γ)
(2)	$\mathbf{t} : P(f(f(a)))$		Annahme
(3)	$\mathbf{f} : \exists x\, P(g(x))$		Annahme (γ)
(4)	$\mathbf{t} : \forall y [f(f(a)) = y \supset \exists z\, y = g(z)]$		von 1 (γ)
(5)	$\mathbf{t} : f(f(a)) = f(f(a)) \supset \exists z\, f(f(a)) = g(z)$		von 4
(6)	$\mathbf{f} : f(f(a)) = f(f(a))$	(7) $\mathbf{t} : \exists z\, f(f(a)) = g(z)$	von 5 (δ)
	$\times\ AB^{=}$	(8) $\mathbf{t} : f(f(a)) = g(b)$	von 7
		(9) $\mathbf{f} : P(g(b))$	von 3
		(10) $\mathbf{f} : P(f(f(a)))$	$S^{=}$:8→9
		$\times$ Wid.: 2/10	

Beachte:

- in 4 wählen wir $x \leftarrow f(a)$ wegen 2: Der relevante Term ist $f(f(a))$
- in 5 wählen wir $y \leftarrow f(f(a))$ um mit 6 abschließen zu können
- in 9 nützen wir 8 um einen Widerspruch mit 2 zu erhalten

Theorien von konkreten Strukturen [WH]

Definition (Theorie einer (Modell-)Struktur)

Für jede Struktur $\mathcal{I}$ heißt $Th(\mathcal{I}) = \{F \mid \mathrm{val}_{\mathcal{I}}(F) = \mathbf{t}\}$ die Theorie von $\mathcal{I}$.

Axiomatisierung von $Th(\mathbb{S})$ – Theorie der binären Stäcks

- $\underline{\text{istleer?}}(\underline{\varepsilon}),\quad \forall x(\underline{\text{istleer?}}(x) \supset x = \underline{\varepsilon}),\quad \underline{\text{pop}}(\underline{\varepsilon}) = \underline{\varepsilon}$
- $\forall x \forall y(\underline{\text{push0}}(x) = \underline{\text{push0}}(y) \supset x = y)$,
  $\forall x \forall y(\underline{\text{push1}}(x) = \underline{\text{push1}}(y) \supset x = y)$
- $\forall x\, \neg\underline{\text{istleer?}}(\underline{\text{push0}}(x)),\quad \forall x\, \neg\underline{\text{istleer?}}(\underline{\text{push1}}(x))$
- $\forall x \exists y(\underline{\text{ist0?}}(x) \supset y = \underline{\text{push0}}(x)),\quad \forall x \exists y(\underline{\text{ist1?}}(x) \supset y = \underline{\text{push1}}(x))$
- $\forall x\, \underline{\text{pop}}(\underline{\text{push0}}(x)) = x,\quad \forall x\, \underline{\text{pop}}(\underline{\text{push1}}(x)) = x$
- $\forall x(\underline{\text{istleer?}}(x) \vee \underline{\text{ist0?}}(x) \vee \underline{\text{ist1?}}(x))$
- $\forall x\, \underline{\text{push0}}(x) \neq x,\quad \forall x\, \underline{\text{push1}}(x) \neq x$
- $\forall x \forall y\, \underline{\text{push0}}(x) \neq \underline{\text{push1}}(y)$

Achtung: Viele konkrete mathematischen Strukturen (z.B. $\mathbb{N}$ und $\mathbb{Z}$) lassen sich nicht vollständig axiomatisieren (Gödels Unvollständigkeitssatz).

PL-Tableaux – Zusammenfassung

- PL-Tableaux sind Bäume von bewerteten geschlossenen Formeln.
- Ein (indirekter) Beweis liegt vor, wenn alle Äste Widersprüche enthalten, d.h. wenn all Äste geschlossen sind.
- γ-Regeln sind im Allgemeinen mehrmals – auch auf dem selben Ast – auf die selbe Formel anzuwenden.
- Im Gegensatz zu AL-Tableaux, sind offene Äste i.A. unendlich lang.
- PL ist unentscheidbar; daher kann es auch keine berechenbare Grenze für die maximale Größe eines Tableau-Beweises (abhängig von der Größe der initialen Annahmen) geben.
- Der PL-Tableau-Kalkül ist vollständig und korrekt:
  Ein Tableau kann genau dann geschlossen werden wenn die entsprechende Konsequenzbehauptung stimmt, also die initialen Annahmen unerfüllbar sind.
- Beweissuche mit Tableau kann (viel besser als für ND) automatisiert werden. In der Praxis verwendet man dazu die Variante der "free variable tableaux", die auf dem Unifikationsprinzip beruhen.

Wichtige Eigenschaften der Prädikatenlogik

- (Abstrakte) Vollständigkeit:
  Es gibt (nicht terminierende) Algorithmen, die genau alle gültigen Formeln aufzählen. M.a.W.: Die Menge der gültigen Formeln ist rekursiv aufzählbar. Gleichbedeutend damit: Es gibt Kalküle für die PL, die korrekt und vollständig sind. (Gödel, 1929)
- Unvollständigkeit der formalen Arithmetik:
  Es gibt keinen Algorithmus, der genau die in $\mathbb{N}$ gültigen Formeln aufzählt. Bzw.: Es gibt keinen Kalkül mit dem man genau die in $\mathbb{N}$ gültigen Formeln beweisen kann. (Gödel, 1931)
- Satz von Löwenheim-Skolem:
  Jede erfüllbare Formel hat ein Modell dessen Gegenstandsbereich aus natürlichen Zahlen besteht. (Löwenheim, 1915; Skolem, 1919)
- Unentscheidbarkeit:
  Es gibt keinen Algorithmus mit dem man die Gültigkeit (bzw. die Erfüllbarkeit) für beliebige PL-Formeln entscheiden könnte. (Turing, Church, 1936)

Das Halteproblem ist unentscheidbar [WH]

Satz [Halteproblem ist unentscheidbar (aber rekursiv aufzählbar)]

Sei $L_u = \{(M, w) \mid M$ ist eine TM die $w \in \{0,1\}^*$ akzeptiert $\}$.
L_u ist rekursiv aufzählbar, aber **nicht entscheidbar**.

Bemerkung: M ist hier auf bestimmte Weise codiert (siehe Folie 43).
Das ist hier aber nicht weiter relevant.

Überlegen Sie:
Was muss man tun um aus der Unentscheidbarkeit von L_u die Unentscheidbarkeit der PL (genauer: der Menge aller gültigen PL-Formeln) schließen zu können? (Was haben Sie in TIL/Teil 1 diesbezüglich gelernt?)

Antwort:
Wir müssen das Halteproblem auf PL-Gültigkeit reduzieren.
Präziser: Wir müssen $L_u \leq L_{PLG}$ zeigen, wobei L_{PLG} die Menge (Sprache) der gültigen PL-Formeln ist.

Prädikatenlogische Repräsentation von Konfigurationen

Syntax:

Signatur $\Sigma_M = \langle PS, KS, FS \rangle$

$PS_2 = \{P_q \mid q \in Q\}$
$FS_1 = \{f_X \mid X \in \Gamma\}$ (zur Erinnerung: $\{B, 0, 1\} \in \Gamma$ daher $f_B, f_0, f_1 \in FS_1$)
$KS = \{b\}$

(Intendierte) Semantik:

Definition (Codierung von Konfigurationen als PL-Atome)

Sei K die Konfiguration $X_1 X_2 ... X_{i-1} q X_i .... X_{n-1} X_n$.
$\xi(K) = P_q(f_{X_{i-1}}(...f_{X_2}(f_{X_1}(b))...), f_{X_i}(...f_{X_{n-1}}(f_{X_n}(b))...))$

$\xi(K)$ drückt aus: "K kommt in der Berechnung vor."

Beachte:
– das erste Argument von P_q codiert das Band von rechts nach links
– das zweite Argument von P_q codiert das Band von links nach rechts
– b steht für "davor/danach kommen nur mehr Leersymbole (B)"

Prädikatenlogische Repräsentation von TM-Befehlen

Übersetzung von Befehlen ($\beta \in \delta_M$) in PL-Formelmengen $\xi(\beta)$

- Kopf bleibt stehen:
  $\beta = (q, X_i; p, Y, S)$, d.h. $\delta(q, X_i) = (p, Y, S)$
  $$X_1X_2...X_{i-1}\mathbf{qX_i}X_{i+1}...X_n \;\Rightarrow\; X_1X_2...X_{i-1}\mathbf{pY}X_{i+1}...X_n$$
  Beachte: $X_iX_{i+1}...X_n$ kann auch leer sein (nur Bs rechts vom Kopf)
  $\xi_1(\beta) = \forall u \forall v[P_{\mathbf{q}}(u, f_{\mathbf{X_i}}(v)) \supset P_{\mathbf{p}}(u, f_{\mathbf{Y}}(v))]$
  $\xi_2(\beta) = \forall u[P_{\mathbf{q}}(u, b) \supset P_{\mathbf{p}}(u, f_{\mathbf{Y}}(b))]$
- Kopf bewegt sich eine Zelle nach rechts:
  $\beta = (q, X_i; p, Y, R)$, d.h. $\delta(q, X_i) = (p, Y, R)$
  $$X_1X_2...X_{i-1}\mathbf{qX_i}X_{i+1}...X_n \;\Rightarrow\; X_1X_2...X_{i-1}\mathbf{Yp}X_{i+1}...X_n$$
  $\xi_1(\beta) = \forall u \forall v[P_{\mathbf{q}}(u, f_{\mathbf{X_i}}(v)) \supset P_{\mathbf{p}}(f_{\mathbf{Y}}(u), v)]$
  $\xi_2(\beta) = \forall u[P_{\mathbf{q}}(u, b) \supset P_{\mathbf{p}}(f_{\mathbf{Y}}(u), b)]$
- Kopf bewegt sich eine Zelle nach links:
  (analog zu oben – Übungsaufgabe!)

Konfigurations-Übergänge entsprechen PL-Inferenzen

$K \Rightarrow_\beta^{[*]} K'$ ... Konfiguration K geht über in Konfiguration K' wegen des Befehls $\beta \in \delta_M$ [eventuell beliebig oft angewendet]

Achtung: Zur Vereinfachung identifizieren wir im Folgenden verschiedene Repräsentationen identischer Konfigurationen (implizite Gleichungen).

Lemma 1 (Rechenschritt $\longrightarrow$ Inferenzschritt):

Wenn $K \Rightarrow_\beta K'$ dann $\xi(K), \xi_1(\beta), \xi_2(\beta) \models \xi(K')$.

Beispiel (Vergleiche Folie 32)

$\beta = (q_0, 0; q_1, X, R)$, Anfangskonfiguration $K_0 = q_0 0011$
Es gilt: $K_0 = q_0 0011 \Rightarrow_\beta Xq_1 011$
$\xi(K_0) = P_{q_0}(b, f_0(f_0(f_1(f_1(b)))))$
$\xi(K_1) = P_{q_1}(f_X(b), f_0(f_1(f_1(b))))$
$\xi_1(\beta) = \forall u \forall v[P_{q_0}(u, f_0(v)) \supset P_{q_1}(f_X(u), v)]$
Es gilt $\xi(K_0), \xi_1(\beta) \models \xi(K_1)$.

Beobachtung: Jede Anfangskonfiguration $K_0(M, w)$ ist durch ein variablenfreies Atom $\xi(K_0(M, w))$ repräsentiert.

Lemma 2 (Inferenzschritte $\longrightarrow$ Rechenschritte):

Wenn $\xi(K), \xi_1(\beta), \xi_2(\beta) \models \xi(K')$ dann $K \Rightarrow_\beta^* K'$.

Akzeptieren (Halten) einer Maschine M wird ausgedrückt durch:

$$Akzept(M) = \exists x \exists y[P_{p_1}(x, y) \vee ... \vee P_{p_n}(x, y)]$$

dabei ist $\{p_1, ..., p_n\}$ die Menge der Endzustände F der Maschine M.

Aus dieser Beobachtungen, sowie Lemma 1 und 2 folgt:

Satz (Reduktion des Halteproblems auf PL-Konsequenz)

Für alle $w \in \{0, 1\}^*$ und Turingmaschinen M ($\delta = \{\beta_1, ..., \beta_n\}$) gilt:
Die Turingmaschine M akzeptiert das Wort w genau dann, wenn
$\xi(K_0(M, w)), \xi_1(\beta_1), \xi_2(\beta_1), ..., \xi_1(\beta_n), \xi_2(\beta_n) \models Akzept(M)$.

Zur Erinnerung:
Aus diesem Satz und dem Deduktionstheorem folgt unmittelbar $L_u \leq L_{PLG}$ und somit wegen der Unentscheidbarkeit von L_u auch die Unentscheidbarkeit der Prädikatenlogik (Gültigkeit und Konsequenz).

Konsequenzen der Unentscheidbarkeit der PL

Konsequenz für Leibniz' bzw. Hilberts Traum:

- Probleme lassen sich tatsächlich meist in PL ausdrücken!
- Wegen Vollständigkeit: Wenn es eine positive Lösung gibt, so lässt sich dieser (als formaler PL-Beweis) prinzipiell auch finden.
- Wegen Unentscheidbarkeit: Es gibt kein Verfahren, dass in jedem Fall feststellen kann, ob eine Lösung existiert.

Satz (Konsequenz für Beweissuche (mit dem Tableau-Kalkül))

Es gibt keine berechenbare Funktion g, derart dass gilt:
Wenn F eine gültige PL-Formel ist, dann gibt es einen Tableau-Beweis von F der Größe $g(|F|)$. ($|F|$ bezeichnet die Länge von F.)

Eine positive Konsequenz:
Aus unserem Reduktionsbeweis wird ersichtlich: Herleiten/Beweisen in einem (korrekten und vollständigen) PL-Kalkül lässt sich als universelles Berechenbarkeitsmodell (wie TM) verstehen ($\Rightarrow$ logisches Programmieren).

Was heißt "Korrektheit"?

Intuitiv: "Das Programm tut was es soll", d.h. es erfüllt eine Spezifikation

Beispiel: Multiplikationsprogramm
Informelle Spezifikation: multipliziere zwei positive Zahlen.

Formale Spezifikation: gegeben durch Precondition und Postcondition in der Sprache der formalen Logik über der Datenstruktur (hier: $\mathbb{Z}$).
Dargestellt durch Assertions:

$$// \ (\!| \ y = y_0 \wedge \neg\, y < 0 \ |\!) \quad \textit{Precondition}$$

```
begin
z ← 0;
while 0 < y do begin
  z ← z + x;
  y ← y − 1
  end
end
```

$$// \ (\!| \ z = x * y_0 \ |\!) \quad \textit{Postcondition}$$

Zwei Formen von Korrektheit I

Partielle Korrektheit:
Wenn die Precondition P gilt, so gilt // $(\!| \ P \ |\!)$
nach Ausführung des Programms α α
die Postcondition Q. // $(\!| \ Q \ |\!)$

Beispiel: Das folgende *ALI*-Programm zur Addition zweier positiver Zahlen ist partiell korrekt bezüglich der gegebenen Spezifikation (Beweis später).

$$// \ (\!| \ x_0 = x \wedge y_0 = y \ |\!)$$

```
while ¬y = 0 do begin
  x ← x + 1;
  y ← y − 1
end
```

$$// \ (\!| \ x = x_0 + y_0 \ |\!)$$

Beachte: Für z.B. $x = x_0 = y = y_0 = -1$ ist die Precondition erfüllt, aber das Programm terminiert nicht - es ist nur partiell korrekt!

Zwei Formen von Korrektheit II

In der Praxis sind wir oft an einer stärkeren Eigenschaft interessiert:

Totale Korrektheit:
Wenn die Precondition P gilt, so terminiert α, und hinterher gilt die Postcondition Q.

Oder äquivalent:
Wenn die Precondition P gilt, so terminiert α,
und α ist partiell korrekt bezüglich Precondition P und Postcondition Q.

Slogan:

Totale Korrektheit = partielle Korrektheit + Termination

Es ist oft sinnvoll, einen Beweis für totale Korrektheit in Beweise für partielle Korrektheit und Termination zu zerlegen.

Im folgenden werden wir hauptsächlich auf partielle Korrektheit eingehen.

(Partielle) Korrektheitsaussagen formal

Erinnerung: $\Sigma(\mathcal{D})$ ... Signatur über dem Datentyp (Modellstruktur) $\mathcal{D}$

Wir schreiben: $\mathcal{D} \models F$, wenn $F \in \mathcal{PF}_{\Sigma(\mathcal{D})}$ in $\mathcal{D}$ gültig ist und $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, F)$ für $\mathrm{val}_{\mathcal{I}}(F)$, wenn I die Variablenbelegung von $\mathcal{I}$ über $\mathcal{D}$ ist.
(Erinnerung: bis auf die Variablenbelegung I ist $\mathcal{I}$ durch $\mathcal{D}$ bereits fixiert.)

Definition

$(\!|P|\!)\ \alpha\ (\!|Q|\!)$ ... partielle Korrektheitsaussage (*"Hoare-Tripel"*)
$P \in \mathcal{PF}_{\Sigma(\mathcal{D})}$ ... Precondition (Vorbedingung)
$Q \in \mathcal{PF}_{\Sigma(\mathcal{D})}$ ... Postcondition (Nachbedingung)
$\alpha \in AL(\mathcal{D})$ ... Programm (mit Variablen, die in P, Q nur frei vorkommen)

$\mathcal{D} \models (\!|P|\!)\ \alpha\ (\!|Q|\!)$ gdw. für alle $I \in ENV$ gilt:

Wenn $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, P) = \mathbf{t}$ *und außerdem* $\mathcal{M}_{AL}(I, \alpha)$ *definiert ist,*
dann $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(\mathcal{M}_{AL}(I, \alpha), Q) = \mathbf{t}$

Man sagt: Die (part.) Korrekheitsaussage ist wahr (über dem Datentyp $\mathcal{D}$).

Partielle vs. totale Korrektheit

$\mathcal{M}_{AL}(I,\alpha)$ ist definiert bedeutet: Das Programm α terminiert (in I).
Daher: Falls α nicht terminiert ist $(\!|P|\!)\ \alpha\ (\!|Q|\!)$ *per definitionem* wahr.

Das Programm $\alpha \in AL(\mathcal{D})$ ist für Precondition P und Postcondition Q total korrekt wenn $\mathcal{D} \models (\!|P|\!)\ \alpha\ (\!|Q|\!)$ und α außerdem in allen Umbegungen I in denen $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I,P) = \mathbf{t}$ terminiert.

Beachte:
Korrektheitsaussagen sind selbst für triviale Pre- und Postconditions oft sehr mächtig. Z.B. kann man mit folgender partiellen Korrektheitsaussage ausdrücken, dass ein Programm nicht terminiert:

$$\mathcal{D} \models (\!|\top|\!)\ \alpha\ (\!|\bot|\!)$$

gilt genau dann, wenn das Programm α in keiner Umgebung terminiert.

Beispiel (cont'd)

Zeige (in $\mathbb{Z}$) die Wahrheit der (partiellen) Korrektheitsaussage

$$(\!|\ \begin{matrix} x_0 = x \\ \wedge y_0 = y \end{matrix}\ |\!)\ \underbrace{\texttt{while}\ \neg y = 0\ \underline{\texttt{do}}\ \underline{\texttt{begin}}\ x \leftarrow x+1;\ y \leftarrow y-1\ \underline{\texttt{end}}}_{\alpha}\ (\!|x = x_0 + y_0|\!)$$

Lösung:
Zu zeigen ist: für alle $I \in ENV$ gilt:

Wenn $\mathcal{M}^{\mathbb{Z}}_{\mathcal{PF}}(I, x_0 = x \wedge y_0 = y) = \mathbf{t}$ *und* $\mathcal{M}_{AL}(I,\alpha)$ *definiert ist,*
dann $\mathcal{M}^{\mathbb{Z}}_{\mathcal{PF}}(\mathcal{M}_{AL}(I,\alpha), x = x_0 + y_0) = \mathbf{t}$

Sei also $I \in ENV$.

1. Fall: Falls $\mathcal{M}^{\mathbb{Z}}_{\mathcal{PF}}(I, x_0 = x \wedge y_0 = y) \neq \mathbf{t}$, so ist die Precondition nicht erfüllt und die Korrektheitsaussage gilt trivialerweise.

2.Fall: $\mathcal{M}^{\mathbb{Z}}_{\mathcal{PF}}(I, x_0 = x \wedge y_0 = y) = \mathbf{t}$.
Somit gilt $I(x) = I(x_0)$ und $I(y) = I(y_0)$.

Beispiel (cont'd)

Zeige (in $\mathbb{Z}$) die Wahrheit der (partiellen) Korrektheitsaussage

$$(\!|\ \begin{matrix} x_0 = x \\ \wedge y_0 = y \end{matrix}\ |\!)\ \underbrace{\texttt{while}\ \neg y = 0\ \underline{\texttt{do}}\ \underline{\texttt{begin}}\ x \leftarrow x+1;\ y \leftarrow y-1\ \underline{\texttt{end}}}_{\alpha}\ (\!|x = x_0 + y_0|\!)$$

Lösung:
Wir schreiben I_k für die Umgebung nach k-maliger Ausführung des Schleifenkörpers α, d.h:

$$I_0 = I \quad \text{und} \quad I_{k+1} = \mathcal{M}_{AL}(I_k, \underline{\texttt{begin}}\ x \leftarrow x+1;\ y \leftarrow y-1\ \underline{\texttt{end}})$$

Wir erhalten die Rekursionen

$$I_{k+1}(x) = I_k(x) + 1 \quad \text{und} \quad I_{k+1}(y) = I_k(y) - 1$$

Auflösen der Rekursion gibt

$$I_k(x) = I(x) + k \quad \text{und} \quad I_k(y) = I(y) - k$$

Beispiel (cont'd)

Zeige (in $\mathbb{Z}$) die Wahrheit der (partiellen) Korrektheitsaussage

$$(\!|\ \begin{matrix} x_0 = x \\ \wedge y_0 = y \end{matrix}\ |\!)\ \underbrace{\texttt{while}\ \neg y = 0\ \underline{\texttt{do}}\ \underline{\texttt{begin}}\ x \leftarrow x+1;\ y \leftarrow y-1\ \underline{\texttt{end}}}_{\alpha}\ (\!|x = x_0 + y_0|\!)$$

Lösung:
Die Zahl der Schleifendurchläufe ist das kleinste $k \geq 0$ so daß

$$0 = I_k(y) = I(y) - k$$

Ein solches k existiert nur falls $I(y) \geq 0$. Somit haben wir

$$\mathcal{M}_{AL}(I,\alpha) = \begin{cases} I_{I(y)} & \text{falls } I(y) \geq 0 \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Für $I(y) < 0$ ist die Korrektheitsaussage somit per definitionem wahr!
Für $I(y) \geq 0$ ist andererseits (mit I' für $\mathcal{M}_{AL}(I,\alpha)$)

$$\mathcal{M}_{\mathcal{T}}(I', x) = I_{I(y)}(x) = I(x) + I(y) = I(x_0) + I(y_0) = \mathcal{M}_{\mathcal{T}}(I', x_0 + y_0)$$

Und somit gilt $\mathcal{M}_{\mathcal{PF}}(I', x = x_0 + y_0) = \mathbf{t}$ QED.

Partielle Korrektheit mittels Assertions

Notation:
$P(v/t)$ ... in P wurden alle freie Vorkommen von v durch t ersetzt

Satz (Assignment)

Sei $v \in IVS$, $t \in \mathcal{T}(\mathcal{D})$, $I \in ENV$ und $I' = \mathcal{M}_{AL}(I, v \underline{\leftarrow} t)$. Dann gilt:

(1) $\mathcal{M}_{\mathcal{T}}(I', s) = \mathcal{M}_{\mathcal{T}}(I, s(v/t))$ *für alle* $s \in \mathcal{T}(\mathcal{D})$.

(2) $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I', A) = \mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, A(v/t))$ *für alle Atome* $A \in \mathcal{PF}_{\Sigma(\mathcal{D})}$.

(3) $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I', F) = \mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, F(v/t))$ *für alle Formeln* $F \in \mathcal{PF}_{\Sigma(\mathcal{D})}$, *in welchen v nur frei vorkommt.*

(4) $\mathcal{D} \models (\!| P(v/t) |\!)\ v \underline{\leftarrow} t\ (\!| P |\!)$ *für alle* $P \in \mathcal{PF}_{\Sigma(\mathcal{D})}$, *in welchen v nur frei vorkommt.*

Anmerkung:
Die Bedingung, dass v in P nur frei vorkommen soll lässt sich durch Umbenennung gebundener Variablen leicht herstellen.

Partielle Korrektheit mittels Assertions

Notation:
$P(v/t)$ ... in P wurden alle freie Vorkommen von v durch t ersetzt

Satz (Assignment)

Sei $v \in IVS$, $t \in \mathcal{T}(\mathcal{D})$, $I \in ENV$ und $I' = \mathcal{M}_{AL}(I, v \underline{\leftarrow} t)$. Dann gilt:

(1) $\mathcal{M}_{\mathcal{T}}(I', s) = \mathcal{M}_{\mathcal{T}}(I, s(v/t))$ *für alle* $s \in \mathcal{T}(\mathcal{D})$.

(2) $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I', A) = \mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, A(v/t))$ *für alle Atome* $A \in \mathcal{PF}_{\Sigma(\mathcal{D})}$.

(3) $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I', F) = \mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, F(v/t))$ *für alle Formeln* $F \in \mathcal{PF}_{\Sigma(\mathcal{D})}$, *in welchen v nur frei vorkommt.*

(4) $\mathcal{D} \models (\!| P(v/t) |\!)\ v \underline{\leftarrow} t\ (\!| P |\!)$ *für alle* $P \in \mathcal{PF}_{\Sigma(\mathcal{D})}$, *in welchen v nur frei vorkommt.*

Dies rechtfertigt die folgenden Assertions (wenn v in P nur frei vorkommt):

$$\begin{array}{lll} & // \ (\!|\ P(v/t)\ |\!) & \\ v \underline{\leftarrow} t & & \\ & // \ (\!|\ P\ |\!) & \text{Ass.} \end{array}$$

Wir notieren Ass. als Begründung

Partielle Korrektheit mittels Assertions

Bei der Konstruktion von Pre- und Postconditions ist es oft vorteilhaft, von der Postcondition rückwärts auf die Precondition zu schließen.

Beispiel (Assignments)

Wir haben eine Programmzeile $y \underline{\leftarrow} y - 1$. Nach deren Ausführung soll $x + y = x_0 + y_0$ gelten. Wir führen die Precondition

$$(x + y = x_0 + y_0)(y/y - 1) \quad = \quad x + y - 1 = x_0 + y_0$$

ein. Partielle Korrektheit ist durch den vorhergehenden Satz garantiert:

$$\begin{array}{lll} & // \ (\!|\ x + y - 1 = x_0 + y_0\ |\!) & \\ y \underline{\leftarrow} y - 1 & & \\ & // \ (\!|\ x + y = x_0 + y_0\ |\!) & \textit{Ass.} \end{array}$$

Partielle Korrektheit mittels Assertions

Beispiel (Begin-End)

Ein Teil unseres Programmes ist $\underline{\texttt{begin}}\ x \underline{\leftarrow} x + 1;\ y \underline{\leftarrow} y - 1\ \underline{\texttt{end}}$.
Nach Ausführung dieses Programmteiles soll wieder $x + y = x_0 + y_0$ gelten.
Zwei Rückwärtsschlüsse auf die Preconditions liefern:

$$\begin{array}{lll} & // \ (\!|\ x + 1 + y - 1 = x_0 + y_0\ |\!) & \\ \underline{\texttt{begin}} & & \\ \quad x \underline{\leftarrow} x + 1\ ; & & \\ & // \ (\!|\ x + y - 1 = x_0 + y_0\ |\!) & \textit{Ass.} \\ \quad y \underline{\leftarrow} y - 1 & & \\ \underline{\texttt{end}} & & \\ & // \ (\!|\ x + y = x_0 + y_0\ |\!) & \textit{Ass.} \end{array}$$

Wieder garantiert der vorherige Satz partielle Korrektheit.

Partielle Korrektheit mittels Assertions

Satz (If-Then)

Wenn $\mathcal{D} \models (\!| P \wedge B |\!)\ \alpha\ (\!| Q |\!)$ *und* $\mathcal{D} \models (\!| P \wedge \neg B |\!)\ \beta\ (\!| Q |\!)$
dann $\mathcal{D} \models (\!| P |\!)\ \underline{\texttt{if}}\ B\ \underline{\texttt{then}}\ \alpha\ \underline{\texttt{else}}\ \beta\ (\!| Q |\!)$.

Dies rechtfertigt die assertions in

$$
\begin{array}{ll}
 & // (\!|\ P\ |\!) \\
\underline{\texttt{if}}\ B\ \underline{\texttt{then}} & \\
 & // (\!|\ P \wedge B\ |\!) \\
\alpha & \\
 & // (\!|\ Q\ |\!) \\
\underline{\texttt{else}} & \\
 & // (\!|\ P \wedge \neg B\ |\!) \\
\beta & \\
 & // (\!|\ Q\ |\!) \\
 & // (\!|\ Q\ |\!) \qquad \textit{If-Then}
\end{array}
$$

Partielle Korrektheit mittels Assertions

Satz (While)

Wenn $\mathcal{D} \models (\!| INV \wedge B |\!)\ \alpha\ (\!| INV |\!)$
dann $\mathcal{D} \models (\!| INV |\!)\ \underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \alpha\ (\!| INV \wedge \neg B |\!)$.

Dies rechtfertigt die Assertions in

$$
\begin{array}{ll}
 & // (\!|\ INV\ |\!) \\
\underline{\texttt{while}}\ B\ \underline{\texttt{do}} & \\
 & // (\!|\ INV \wedge B\ |\!) \\
\alpha & \\
 & // (\!|\ INV\ |\!) \\
 & // (\!|\ INV \wedge \neg B\ |\!) \qquad \textit{While}
\end{array}
$$

Die Formel *INV* nennen wir die Invariante der Schleife.

Bemerkung: Invarianten beschreiben Beziehungen zwischen Variablen, welche durch den Schleifenkörper erhalten bleiben. Ihre Formulierung erfordert oft Kreativität und tiefe Einsicht in den Schleifenkörper.

Partielle Korrektheit mittels Assertions

Satz (Implikation)

Wenn $\mathcal{D} \models R \supset P$ *und* $\mathcal{D} \models (\!| P |\!)\ \alpha\ (\!| Q |\!)$ *und* $\mathcal{D} \models Q \supset S$
dann $\mathcal{D} \models (\!| R |\!)\ \alpha\ (\!| S |\!)$

Zusammen mit Beweisen für $\mathcal{D} \models R \supset P$ und $\mathcal{D} \models Q \supset S$ rechtfertigt dies die Assertions in

$$
\begin{array}{ll}
 & // (\!|\ R\ |\!) \\
 & // (\!|\ P\ |\!) \quad \textit{Imp.} \\
\alpha & \\
 & // (\!|\ Q\ |\!) \\
 & // (\!|\ S\ |\!) \quad \textit{Imp.}
\end{array}
$$

Bemerkung: Dies ist in der Praxis die haupsächliche Schnittstelle zur Logik. Der obige Satz erlaubt weiterhin die modulare Konstruktion von (partiell) korrekten Programmen durch Einsetzung von Programmteilen.

Beispiel: Partielle Korrektheit mittels Assertions

Wir rechtfertigen partielle Korrektheit unseres Additionsprogramms:

$$
\begin{array}{ll}
 & // (\!|\ x = x_0 \wedge y = y_0\ |\!) \\
 & // (\!|\ x + y = x_0 + y_0\ |\!) \qquad \text{Imp., (1)} \\
\underline{\texttt{while}}\ \neg y = 0\ \underline{\texttt{do}} & \\
 & // (\!|\ x + y = x_0 + y_0 \wedge \neg y = 0\ |\!) \\
 & // (\!|\ x + 1 + y - 1 = x_0 + y_0\ |\!) \qquad \text{Imp., (2)} \\
\underline{\texttt{begin}} & \\
\quad x \underline{\leftarrow} x + 1\,\underline{;} & \\
 & // (\!|\ x + y - 1 = x_0 + y_0\ |\!) \qquad \text{Ass.} \\
\quad y \underline{\leftarrow} y - 1 & \\
\underline{\texttt{end}} & \\
 & // (\!|\ x + y = x_0 + y_0\ |\!) \qquad \text{Ass.} \\
 & // (\!|\ x + y = x_0 + y_0 \wedge \neg\neg y = 0\ |\!) \qquad \text{While} \\
 & // (\!|\ x = x_0 + y_0\ |\!) \qquad \text{Imp., (3)}
\end{array}
$$

Beispiel: Partielle Korrektheit mittels Assertions (cont'd)

Es bleibt noch Gültigkeit der folgenden Implikationen über $\mathbb{Z}$ zu zeigen:

(1) $x = x_0 \wedge y = y_0 \supset x + y = x_0 + y_0$:
Sei $I \in ENV$ mit $\mathcal{M}^{\mathbb{Z}}_{\mathcal{PF}}(I, x = x_0 \wedge y = y_0) = \mathbf{t}$. Dann ist $I(x) = I(x_0)$ und $I(y) = I(y_0)$ und somit $\mathcal{M}^{\mathbb{Z}}_{\mathcal{PF}}(I, x + y = x_0 + y_0) = \mathbf{t}$.

(2) $x + y = x_0 + y_0 \wedge \neg y = 0 \supset x + 1 + y - 1 = x_0 + y_0$:
Für jedes $I \in ENV$ gilt:
$\mathcal{M}_{\mathcal{T}}(I, x+y) = I(x)+I(y) = I(x)+1+I(y)-1 = \mathcal{M}_{\mathcal{T}}(I, x+1+y-1)$.
Aber $\mathcal{M}^{\mathbb{Z}}_{\mathcal{PF}}(I, x + y = x_0 + y_0 \wedge \neg y = 0) = \mathbf{t}$ impliziert
$\mathcal{M}^{\mathbb{Z}}_{\mathcal{PF}}(x + y = x_0 + y_0) = \mathbf{t}$ und somit auch
$\mathcal{M}^{\mathbb{Z}}_{\mathcal{PF}}(x + 1 + y - 1 = x_0 + y_0) = \mathbf{t}$.

(3) $x + y = x_0 + y_0 \wedge \neg\neg y = 0 \supset x = x_0 + y_0$:
Für $I \in ENV$ mit $\mathcal{M}^{\mathbb{Z}}_{\mathcal{PF}}(x + y = x_0 + y_0 \wedge \neg\neg y = 0) = \mathbf{t}$ gilt $I(y) = 0$ und somit $\mathcal{M}^{\mathbb{Z}}_{\mathcal{T}}(I, x + y) = \mathcal{M}^{\mathbb{Z}}_{\mathcal{T}}(I, x)$. Da außerdem auch $\mathcal{M}^{\mathbb{Z}}_{PL}(x + y = x_0 + y_0) = \mathbf{t}$, gilt schließlich $\mathcal{M}^{\mathbb{Z}}_{PL}(x = x_0 + y_0) = \mathbf{t}$.

QED.

Beispiel – Verifikation des Divisionsalgorithmus (Forts.)

Rückwärtiges Schließen liefert nun weiter:

$$
\begin{array}{ll}
 & \text{// } (\!| \text{ divisor} \geq \underline{0} \ |\!) \\
 & \text{// } (\!| \text{ dividend} = \text{dividend} + 0 * \text{divisor} \ |\!) \quad \text{Imp., (2)} \\
\underline{\texttt{begin}} & \\
\quad \underline{\texttt{begin}} & \\
\qquad \text{rem} \underline{\leftarrow} \text{dividend}\,\underline{;} & \\
 & \qquad \text{// } (\!| \text{ dividend} = \text{rem} + 0 * \text{divisor} \ |\!) \quad \text{Ass.} \\
\qquad \text{quot} \underline{\leftarrow} 0 & \\
\quad \underline{\texttt{end}}\,\underline{;} & \\
 & \qquad \text{// } (\!| \text{ dividend} = \text{rem} + \text{quot} * \text{divisor} \ |\!) \quad \text{Ass.} \\
\quad \underline{\texttt{while}} \text{ rem} \geq \text{divisor } \underline{\texttt{do}} & \\
\qquad \vdots & \\
\underline{\texttt{end}} & \\
 & \text{// } (\!| \text{ dividend} = \text{rem} + \text{quot} * \text{divisor} \wedge \text{rem} < \text{divisor} \ |\!)
\end{array}
$$

Beispiel – Verifikation des Divisionsalgorithmus (Forts.)

Um den Korrektheitsbeweis abzuschliessen müssen wir nur noch die Anwendung (2) der Implikationsregel rechtfertigen. Dazu benötigen wir Gültigkeit (in $\mathbb{N}$) der Implikation

$$\text{divisor} \geq 0 \supset \text{dividend} = \text{dividend} + 0 * \text{divisor}$$

Die rechte Seite der Implikation ist jedoch in $\mathbb{N}$ immer wahr, daher ist die gesamte Formel gültig in $\mathbb{N}$. QED.

Bemerkung: Der obige Beweis zeigt sogar partielle Korrektheit des Divisionsalgorithmus bezüglich der modifizierten Spezifikation mit der schwächeren Precondition $\top$.

Empfohlene Übungsaufgabe: Erläutern Sie warum die Spezifikation mit Precondition $\text{divisor} > 0$ in der Praxis dennoch relevanter ist.

Der Hoare-Kalkül (1969)

Die obigen Sätze zur Rechtfertigung von Assertions lassen sich als Regeln in einem formalen Kalkül für Hoare-Tripel lesen:

$$\frac{}{(\!| P(v/t) |\!)\ v \underline{\leftarrow} t\ (\!| P |\!)} \ \text{Ass.} \ (v \text{ nur frei in } P)$$

$$\frac{(\!| P |\!)\ \alpha\ (\!| Q |\!) \qquad (\!| Q |\!)\ \beta\ (\!| R |\!)}{(\!| P |\!)\ \underline{\texttt{begin}}\ \alpha\underline{;}\ \beta\ \underline{\texttt{end}}\ (\!| R |\!)} \ \text{Begin-End}$$

$$\frac{(\!| P \wedge B |\!)\ \alpha\ (\!| Q |\!) \qquad (\!| P \wedge \neg B |\!)\ \beta\ (\!| Q |\!)}{(\!| P |\!)\ \underline{\texttt{if}}\ B\ \underline{\texttt{then}}\ \alpha\ \underline{\texttt{else}}\ \beta\ (\!| Q |\!)} \ \text{If-Then}$$

$$\frac{(\!| INV \wedge B |\!)\ \alpha\ (\!| INV |\!)}{(\!| INV |\!)\ \underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \alpha\ (\!| INV \wedge \neg B |\!)} \ \text{While}$$

$$\frac{\mathcal{D} \models R \supset P \qquad (\!| P |\!)\ \alpha\ (\!| Q |\!) \qquad \mathcal{D} \models Q \supset S}{(\!| R |\!)\ \alpha\ (\!| S |\!)} \ \text{Imp.}$$

Der Hoare-Kalkül (1969)

Satz (Korrektheit)

Wenn für ein ALD-Programm α und Formeln $P, Q \in \mathcal{PF}$ das Hoare-Tripel $(\!|P|\!)\ \alpha\ (\!|Q|\!)$ herleitbar ist, so gilt

$$\mathcal{D} \models (\!|P|\!)\ \alpha\ (\!|Q|\!)\ .$$