4.0 VU Theoretische Informatik und Logik (Logik-Teil)

Christian Fermüller

unter Mitwirkung von Marion Oswald, Rudi Freund,
Alex Leitsch, Gernot Salzer
sowie 15 TutorInnen

Institut für Computersprachen
Arbeitsbereich Theoretische Informatik und Logik

Sommersemester 2014

Bisheriger Lehrinhalt (in Stichworten):

- Berechenbarkeit:
  Turingmaschinen, Problem/Sprache vs. Algorithmus, Problemreduktion, (Un-)Entscheidbarkeit, rekursive Aufzählbarkeit ('Semi-Entscheidbarkeit'), Church-Turing These, Universalität
- Sprachen/Grammatiken/Automaten:
  Regularität, Kontextfreiheit, Kontextsensitivität, Eindeutigkeit/Mehrdeutigkeit, Pumping-Lemmas, Abschlusseigenschaften, Automaten (endliche, gsm, ..., deterministisch/interdeterministisch) , Normalformen von Grammatiken, Chomsky-Hierarchie, ...
- Komplexität:
  polynomiell vs. exponentiell, Zeit- und Speicherplatzhierarchien, polynomielle Problemreduktion, P vs. NP, NP-Vollständigkeit, weitere wichtige Komplexitätsklassen...

Stichworte für den zweiten Abschnitt von TIL:

1. Klassische Prädikaten- und Aussagenlogik: Woher? Warum? Wie?
2. Unterscheidung syntaktischer und semantischer Aspekte:
   systematische Zuordnung von Bedeutung / strukturelle Induktion
3. Modellierung und Formalisierung mit klassischer Logik:
   Formalisierung natürlicher Sprache, Logik als Spezifikationssprache,
   Axiomatisierung von Strukturen und Theorien
4. Verständnis des Konzepts logischer Kalküle:
   Syntax / Semantik — Korrektheit und Vollständigkeit
5. Computerorientierte Konstruktion formaler Beweise:
   "Semantische Tableaux" (aussagen- und prädikatenlogisch)
6. Kenntnis wichtiger Eigenschaften der klassischen Logik
   insbesondere Unentscheidbarkeit der Prädikatenlogik
7. Verifikation von Programmen: Der "Hoare-Kalkül"

Was Sie diese Woche erwartet

1. Logik: Woher? Warum? Wie?
   1.1. Ein Paradies mit Schlangen
   1.2. Alles ist Logik! Ist alles Logik?
   1.3. Ein Programm für das verbleibende Semester
   1.4. Was hat das alles mit Informatik zu tun?
2. Die Sprache der Prädikaten (und Aussagen-)Logik (Syntax)
   2.1. Was heißt "klassische Prädikatenlogik erster Stufe"
   2.2. Logische Symbole / außerlogische Signaturen
   2.3. Terme
   2.4. PL-Formeln
   2.5. Schreibkonventionen
   2.6. Variablenvorkommen / Variablensubstitutionen
   2.7. Wo ist die Aussagenlogik geblieben?
3. Von Syntax zu Semantik
4. Die Semantik der Prädikaten- (und Aussagen-)Logik

Logik: Woher? Warum? Wie?

Gute Geschichten beginnen im Paradies ...

"Aus dem Paradies, das Cantor uns geschaffen hat,
soll uns niemand vertreiben können."
(David Hilbert, 1925)

David Hilbert (1862 – 1943). Einer der einflussreichsten Mathematiker aller Zeiten. (Z.B. durch die berühmte Problemliste von 1900.) Der letzte große Universalist: prägende Beiträge zu Geometrie, Topologie, Algebra, Zahlentheorie, Funktionalanalysis, Logik (Beweistheorie!), aber auch zur Physik.

Georg Cantor (1845 – 1918). Deutscher Mathematiker. Begründer der Mengenlehre. Schöpfer vieler Begriffe und Werkzeuge, die heute jeder Mathematiker und Ingenieur verwendet: Potenzmenge, Mächtigkeit, Diagonalargument, Kodierung von Zahlenpaaren als Zahlen, ...

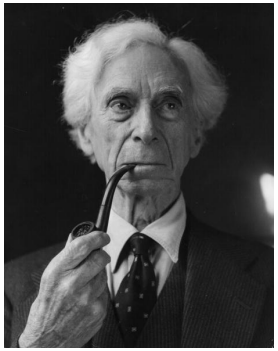
Was ist "Cantors Paradies"?

Alle wichtigen mathematischen Konzepte können als Mengen verstanden werden: Zahlen, Paare, Tupel, Relationen, Funktionen, ...

Daher eignet sich die Mengenlehre als Fundament der Mathematik — und damit auch der Informatik und anderer Ingenieurdisziplinen.

Aber ist das Fundament sicher?

All-Menge: Gibt es eine Menge, die alles (alle Mengen) enthält?



Bertrand Russell (1872 – 1970). Britischer Philosoph, Mathematiker und Logiker. "*Principia Mathematica*" (1910-13, mit A.N. Whithead). Strikter Pazifist, Atheist und Rationalist. Literaturnobelpreis 1950.

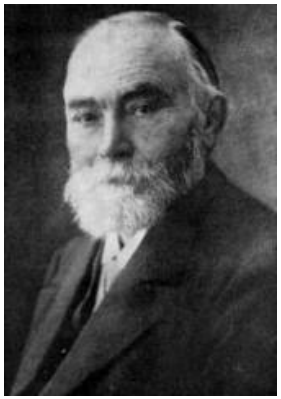
Russell (in einem Brief an Frege): Was ist mit der Menge aller Mengen, die sich nicht selbst enthalten? ("Russell-Klasse" $\{x \mid x \notin x\}$)

Lassen sich diese "Schlangen" aus dem Paradies bannen?
Trägt das Fundament der Mengenlehre oder gibt es Widersprüche?

Frege: Alles ist Logik!

Freges Programm:

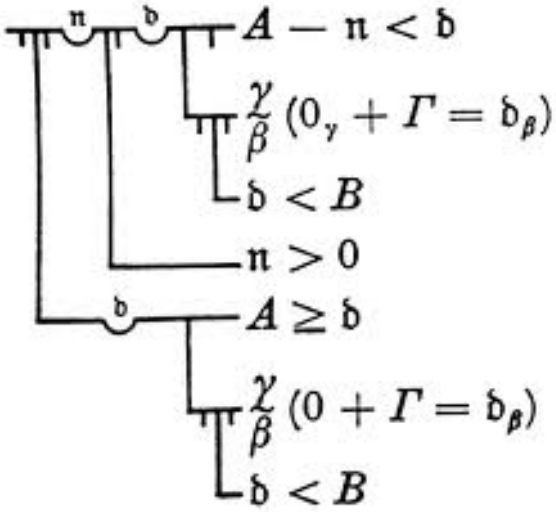
Alles was sich präzise aussagen lässt, kann man in einer "Begriffsschrift" (d.h., formalen Sprache) ausdrücken. Mathematik ist Logik.



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

Hauptwerke: *Begriffsschrift* (1879), *Grundlagen der Arithmetik* (1884), *Grundgesetze der Arithmetik* (1893/1903)

Eine Kostprobe zur *Begriffsschrift*:



Noble Träume (der Aufklärung und des 20. Jahrhunderts)

Ein wichtiger "Vorfahre" von Frege, Russell, Hilbert:

Gottfried Wilhelm Leibniz (1646 – 1716). Deutscher Philosoph, Wissenschaftler, Mathematiker, Diplomat, Physiker, Theologe, Historiker, Politiker, Bibliothekar.
Informatisch bedeutsam: Binärsystem, Rechenmaschinen, Ideen zur Logik und zur Automatischen Deduktion.

Leibniz' Traum (bezogen auf seine "Characteristica Universalis"):

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

Hilberts Traum (etwas bescheidener, weil nur auf Mathematik bezogen):

> *Da ist das Problem, suche die Lösung. Du kannst sie durch reines Denken finden; denn in der Mathematik gibt es kein Ignorabimus.*

Damit verbunden ist "Hilberts Programm":
Formalisieren, Axiomatisieren, Widerspruchsfreiheit zeigen!

Ist alles Logik?

Freges *"Begriffsschrift"* ist einer der zentralen Quellen moderner Logik. (Wichtig sind aber, z.B., auch George Booles' *"Laws of Thought"*.)

Logizismus: Aufbauend auf der *"Begriffsschrift"* wollte Frege in den *"Grundgesetzen der Arithmetik"* zeigen, dass die gesamte Mathematik aus nur wenigen, rein logischen Begriffen und Regeln herleitbar ist.

> *Und nur das würde ich als Widerlegung anerkennen können, wenn jemand durch die That zeigte, dass auf andern Grundüberlegungen ein besseres, haltbareres Gebäude errichtet werden könnte, oder wenn mir jemand nachwiese, dass meine Grundsätze zu offensichtlich falschen Folgesätzen führten. Aber das wird Keinem gelingen. (Vorwort, 1893)*

Dieses Programm ist gescheitert (zumindest in Freges Version):

> *Einem wissenschaftlichen Schriftsteller kann kaum etwas Unerwünschteres begegnen, als dass ihm nach Vollendung einer Arbeit eine der Grundlagen seines Baues erschüttert wird. In diese Lage wurde ich durch einen Brief des Herrn Bertrand Russell versetzt, als der Druck dieses Bandes sich seinem Ende näherte. (Nachwort, Band 2, 1903)*

Die Vertreibung aus dem Paradies!?

Frege hat sich nie vom "Schlangenbiss" Russells (Brief von 1902) erholt.

Eine moderne Diagnose: Frege wollte zu viel auf einmal:

- logische Begriffe beliebiger Ordnung
- "uneingeschränkte Komprehension":
  für alle Formeln ϕ gibt es die Menge $\{x \mid \phi(x)\}$.
- Es fehlt (auch noch in Russells "*Principia Mathematica*") eine klare Unterscheidung von Zeichen/Regeln (Syntax) und deren Bedeutung
  $\Longrightarrow$ Korrektheit und Vollständigkeit nicht nachweisbar

Eine moderne Lösung:

- Axiomatische Mengenlehre als Theorie der Prädikatenlogik erster Stufe
  (Zermelo, Fraenkel, Gödel, Bernays, von Neumann, ...)
- Korrektheit und Vollständigkeit von Kalkülen (Gödels Diss, 1929)
- Saubere Syntax/Semantik-Trennung:
  "gültig" versus "beweisbar" ("ableitbar"/"herleitbar")
  "wahr in einem Modell" versus "logisch gültig"
  darauf aufbauend: Def. von "logische Konsequenz" (Tarski, ...)

Ein Programm für das verbleibende Semester

Es bleiben nun folgende Fragen zu klären:

- Was genau sind "Formeln der Prädikatenlogik erster Stufe"?
- Was ist eine "axiomatische Theorie" in dieser Sprache (PL)?
- Wie kann man die Bedeutung unserer Sprache präzise festlegen?
  Erinnerung: Frege und Russell sind letztlich auch deshalb gescheitert, weil sie keine klare Syntax/Semantik-Trennung kannten!
- Wie drückt man mathematische – und andere – Konzepte in PL aus?
- Was genau heißt "ein Satz folgt (logisch) aus Annahmen/Axiomen"?
- Wie, also mit welchen Regeln, kann bzw. soll man "herleiten"?
- Kann man wirklich alles herleiten, was logisch folgt?
- Kann man Leibniz' bzw. Hilberts Traum verwirklichen?
- Was hat das alles mit beweisbar korrekten Programmen zu tun?

NB: Einige Fragen wurden bereits in "Formale Modellierung" beantwortet!

Was hat das alles mit Informatik zu tun?

"The aim of logic in computer science is to develop languages to model the situations we encounter as computer science professionals, in such a way that we can reason about them formally. Reasoning about situations means constructing arguments about them; we want to do this formally, so that the arguments are valid and can be defended rigorously, or executed on a machine."

Aus:
Huth, Ryan: *Logic in Computer Science*, Cambridge UP, 2004/06

Die klassische Prädikatenlogik erster Stufe ist das wohl wichtigste allgemeine Werkzeug zur Modellierung und zur Formalisierung von Argumenten und Beweisen, dabei (prinzipiell) auch automatisierbar.

Ist Informatik ohne Prädikatenlogik denkbar?

Prädikatenlogische (und aussagenlogische) Formeln kommen in der Informatik in unterschiedlichsten Varianten nahezu überall, zumindest in folgenden Formen vor:

- Schleifen- oder If-Bedingungen in Programmen
- Zusicherungen in Programmen ("Assertions")
- Spezifikationen von Daten, Programmeigenschaften, Protokollen, ...
- Formale Gegenstücke natürlichsprachlicher Information
- Abfragen in Datenbanken, Web-Repositories, etc.
- Datenconstraints
- Klauseln in logischen Programmen
- Input für "Theorem Provers" (automatische Beweiser)
- ...— Übungsaufgabe: ergänzen Sie die Liste selbst!

Achtung: Oft werden nichtklassische Logiken bzw. Fragmente benötigt!

Was heißt "klassische Prädikatenlogik erster Stufe"

Was bedeutet "klassisch"?

"Klassisch" bezieht sich auf die besprochene Geschichte der Logik.
Frege: Es gibt nur wahre und falsche Aussagen – *tertium non datur*.

Was bedeutet "erster Stufe"? (Man sagt auch: "erster Ordnung")

- Stufe 0: Aussagenlogik, keine Quantifikation $[(A \wedge B) \supset C, \ldots]$
- Stufe 1: Quantifikation über Elemente des Gegenstandbereichs (Domäne) $[\forall x \exists y (P(x, y) \supset Q(c, f(x, y))), \ldots]$
- Stufe 2: Quantifikation auch über Prädikate (Relation) und Funktionen $[\forall R \exists x \exists f (R(x, f(x)) \supset P(f(x))), \ldots]$
- höhere Stufen: Quantifikation auch über Funktionale, Relationen zwischen Relationen, etc. $[\forall F \exists P \exists f \forall x \, (P(F(f, y)) \supset g(x) = y), \ldots]$

Aussagenlogik ist oft zu ausdrucksschwach (grob); zweit- und höherstufige Logik oft zu mächtig und komplex um gut automatisierbar zu sein.

Wir sprechen im Folgenden einfach von "Prädikatenlogik", abgekürzt: PL.

Die Sprache (Syntax) der Prädikatenlogik (PL)

Wir benötigen verschiedene Kategorien von Zeichen (Symbolen):

(1) Logische Symbole:

- (aussagenlogische) Konnektive (Junktoren, Operatoren):
  1-stellig: $\neg$ (nicht, non);
  2-stellig: $\wedge$ (und), $\vee$ (oder), $\supset$ (impliziert, wenn ... dann ...);
  eventuell weitere, wie $\uparrow$ (NAND), $\downarrow$ (NOR), $\leftrightarrow$ (genau dann wenn)
- aussagenlogische Konstanten (Wahrheitskonstanten):
  $\bot$ (*falsum*), $\top$ (*verum*)
  Achtung: nicht mit Wahrheitswerten verwechseln!

  Alternativ kann man aussagenlogische Konstanten auch als 0-stellige Konnektive auffassen
- Quantoren:
  All-Quantor: $\forall$ (für alle, jedes, jede, jeder)
  Existenz-Quantor: $\exists$ (es gibt ein, es existiert ein)
- Gleichheit (Identität): $=$ (hier als logisches Symbol aufgefasst!)

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): $IVS = \{x, y, z, x_1, \ldots\}$

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

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) $p(t_1, \ldots, t_n) \in \mathcal{PF}_\Sigma$ (allgemeine) atomare Formel
wenn $p \in PS_n$ und $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$ (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

Anmerkungen zur PL-Syntax

- Die Definition bezieht sich nur auf Zeichen (Signaturen) und nicht auf Modelle oder andere semantische (d.h. bedeutungsbezogene) Begriffe.
  $\Longrightarrow$ saubere Trennung von Syntax und Semantik (vgl. Frege!)
- Die Zeichensequenz $\forall y$ bzw. $\exists x$ bildet ein Quantoren-Vorkommen.
- Wir verwenden hier nur diese klassische (eingeschränkte) Syntax.
  Die Erweiterung auf andere Konnektive ($\uparrow$, $\leftrightarrow$, ...) ist naheliegend.

(Unsere) Klammereinsparungsregeln:

- äußere Klammern können weggelassen werden
- $\neg$ bindet stärker als $\wedge$ und $\vee$
- $\wedge$ und $\vee$ binden gleich stark, aber stärker als $\supset$
- Assoziativität von $\wedge$ und $\vee$ berücksichtigen

- z.B. ist $\neg F_1 \wedge F_2 \wedge F_3 \supset G_1 \vee \neg G_2$ wohlgeformt
  aber $F_1 \vee F_2 \wedge F_3$ und $F_1 \supset F_2 \supset F_3$ sind nicht wohlgeformt
- Achtung bei der Klammerung von quantifizierten Formeln:
  $\forall x(P(x) \supset Q(x))$ besagt etwas ganz anderes als $(\forall x P(x) \supset Q(x))$

Schreibkonventionen

a, b, c, d, e	...	Konstantensymbole
f, g, h	...	Funktionssymbole
P, Q, R	...	Prädikatensymbole
x, y, z, u, v, w	...	Variable (Variablensymbole)

Um die Lesbarkeit zu erhöhen verwenden wir auch '[' / ']' für '(' / ')'

Für Signaturen zu bekannten Strukturen (wie $\mathbb{N}$ und $\mathbb{Z}$) verwenden wir daneben auch traditionelle Symbole, wie '0', '1' (KS), '$+$', '$-$' (FS) oder '$<$', '$\leq$' (PS), in üblicher Infix-Notation.

Beispiele wohlgeformter Formeln

- $F_1 = [\neg P(f(x), a) \wedge P(g(a, y), b)]$
- $F_2 = \forall x[\neg P(f(x), a) \wedge P(g(a, x), b)]$
- $F_3 = [\forall x\, \neg P(f(x), a) \wedge P(g(a, x), b)]$
- $F_4 = \forall x \exists y \exists z (f(y) < a \vee x + 1 = b \vee z > f(1))$
- $F_5 = \forall x\, P(y, z)$
- $F_6 = \forall x \exists x\, P(x, x)$

Variablenvorkommen

$V(t), V(F)$ ... Variablen, die in Term t bzw. Formel F vorkommen

Definition (Menge $FV(F)$ der freien (ungebundenen) Variablen in F)

- $FV(\bot) = FV(\top) = \{\}$
- $FV(p(t_1, \ldots, t_n)) = V(p(t_1, \ldots, t_n))$ $(= V(t_1) \cup \ldots \cup V(t_n))$
- $FV(t = s) = V(t = s)$ $(= V(t) \cup V(s))$
- $FV(\neg F) = FV(F)$
- $FV(F \circ G) = FV(F) \cup FV(G)$ für $\circ \in \{\wedge, \vee, \supset\}$
- $FV(\mathrm{Q}v\, F) = FV(F) - \{v\}$ für $\mathrm{Q} \in \{\forall, \exists\}$
  "$\mathrm{Q}v$ bindet die freien Vorkommen von v in F"
  "F ist der Bindungsbereich von $\mathrm{Q}v$ in $\mathrm{Q}v\, F$"

Achtung: v kann gleichzeitig frei und gebunden vorkommen!

F bzw. t ist geschlossen bedeutet: $FV(F) = \{\}$ bzw. $FV(t) = V(t) = \{\}$.

Beispiele zu Variablenvorkommen

Variablen in unseren Beispielen wohlgeformter Formeln

- $F_1 = [\neg P(f(x), a) \wedge P(g(a, y), b)]$
  $FV(F_1) = \{x, y\}$ — zur Erinnerung: a, b sind Konstantensymbole
- $F_2 = \forall x[\neg P(f(x), a) \wedge P(g(a, x), b)]$
  $FV(F_2) = \{\}$, F_2 ist also geschlossen
- $F_3 = [\forall x\, \neg P(f(x), a) \wedge P(g(a, x), b)]$
  $FV(F_3) = \{x\}$, x kommt frei und gebunden vor
  Klammereinsparungsregeln: '[' und ']' könnten weggelassen werden (auch in F_1, nicht aber in F_2)
- $F_4 = \forall x \exists y \exists z (f(y) < a \vee x + 1 = b \vee z > f(1))$
  $FV(F_4) = \{\}$, F_2 ist also geschlossen
- $F_5 = \forall x\, P(y, z) \quad FV(F) = \{y, z\}$
- $F_6 = \forall x \exists x\, P(x, x) \quad FV(F) = \{\}$

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:

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

Wo ist die Aussagenlogik (AL) geblieben?

AL-Formeln sind Spezialfälle von PL-Formeln, für die gilt:

- keine Quantoren
- keine Individuenvariablensymbole
- keine Konstanten- und Funktionssymbole
- kein Gleichheitssymbol ('$=$')
- Alle Prädikatensymbole sind 0-stellig (keine Argumentstellen)
  Diese 0-stelligen Prädikatensymbole heißen Aussagenvariable
- man braucht keine Signatur angeben:
  wir nehmen an, dass immer genügend viele (offiziell: abzählbar unendlich viele) Aussagenvariable vorhanden sind

Anmerkung:
Neben den Aussagenvariablen P, Q, R, sind $\bot$ und $\top$ atomare AL-Formeln

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)

Aussagenlogische Formeln (AL-Formeln)

zur Erinnerung:

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/Atome)

(P2) $p(t_1, \ldots, t_n) \in \mathcal{PF}_\Sigma$ (allgemeine) atomare Formel/Atom)
wenn $p \in PS_n$ und $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$ (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

Zwei wichtige syntaktische Konzepte

1. Kalkül:
   Ein Kalkül[1] ist ein Regelsystem mit dem sich Formeln aus anderen Formeln durch systematische Regelanwendungen herleiten lassen.

   Solche Herleitungen nennt man "formale Beweise". (Wir werden konkrete Kalküle behandeln, insbesondere den *Tableau-Kalkül*.)
2. Theorie:
   (In der Logik) ist eine Theorie einfach eine Menge von Formeln.

   Theorien sind oft definiert als Mengen von Formeln, die aus einer gegebenen Menge $\mathcal{A}$ von Formeln folgen bzw. herleitbar sind.
   Die Elemente von $\mathcal{A}$ heißen Axiome — man spricht dann von einer axiomatischen Theorie. (Z.B. Peano-Arithmetik, Mengenlehre *ZFC*, Theorie der Äquivalenz- bzw. der Kongruenzrelationen, ...)

Beachte: Wir haben bisher noch nicht auf semantische Konzepte wie Wahrheit, Gültigkeit, Interpretation, Modell, etc. Bezug genommen!

[1] "Der Kalkül" (*m.*) ist etwas anderes als "das Kalkül" (*n.*)!

Ein Programm für das verbleibende Semester

- Was genau sind "Formeln der Prädikatenlogik erster Stufe"?
- Was ist eine "axiomatische Theorie" in dieser Sprache (PL)?
- Wie kann man die Bedeutung unserer Sprache präzise festlegen?
  Erinnerung: Frege und Russell sind letztlich auch deshalb gescheitert, weil sie keine klare Syntax/Semantik-Trennung kannten!
- Wie drückt man mathematische – und andere – Konzepte in PL aus?
- Was genau heißt "ein Satz folgt (logisch) aus Annahmen/Axiomen"?
- Wie, also mit welchen Regeln, kann bzw. soll man "herleiten"?
- Kann man wirklich alles herleiten, was logisch folgt?
- Kann man Leibniz' bzw. Hilberts Traum verwirklichen?
- Was hat das alles mit beweisbar korrekten Programmen zu tun?

Was bisher geschah:

1. Logik: Woher? Warum? Wie?
   1.1. Ein Paradies mit Schlangen
   1.2. Alles ist Logik! Ist alles Logik?
   1.3. Ein Programm für das verbleibende Semester
   1.4. Was hat das alles mit Informatik zu tun?
2. Die Sprache der Prädikaten (und Aussagen-)Logik (Syntax)
   2.1. Was heißt "klassische Prädikatenlogik erster Stufe"
   2.2. Logische Symbole / außerlogische Signaturen
   2.3. Terme
   2.4. PL-Formeln
   2.5. Schreibkonventionen
   2.6. Variablenvorkommen / Variablensubstitutionen
   2.7. Wo ist die Aussagenlogik geblieben?

Ein Programm für das ganze Semester

- Was genau sind "Formeln der Prädikatenlogik erster Stufe"?
- Was ist eine "axiomatische Theorie" in dieser Sprache (PL)?
- Wie kann man die Bedeutung unserer Sprache präzise festlegen?
  Erinnerung: Frege und Russell sind letztlich auch deshalb gescheitert, weil sie keine klare Syntax/Semantik-Trennung kannten!
- Wie drückt man mathematische – und andere – Konzepte in PL aus?
- Was genau heißt "ein Satz folgt (logisch) aus Annahmen/Axiomen"?
- Wie, also mit welchen Regeln, kann bzw. soll man "herleiten"?
- Kann man wirklich alles herleiten, was logisch folgt?
- Kann man Leibniz' bzw. Hilberts Traum verwirklichen?
- Was hat das alles mit beweisbar korrekten Programmen zu tun?

Was Sie heute und nächste mal erwartet

3. Von Syntax zu Semantik:
   - 3.0. Was bedeutet das? Wie kommen Zeichen dazu etwas zu bezeichnen? (Behandlung einfacher arithmetische Ausdrücke als "running example")
   - 3.1. Terme mit Variablen
   - 3.2. Grammatik vs. induktive Definition
   - 3.3. Datentypen als Modellstrukturen
   - 3.4. Terme über Modellstrukturen
   - 3.5. Boolesche Ausdrücke über Modellstrukturen
   - 3.6. Eine einfache Programmiersprache: AL($\mathcal{D}$)

* *Einschub:* Beispiele zur strukturellen Induktion

4. Die Semantik der Prädikaten- (und Aussagen-)Logik

Von Syntax zu Semantik

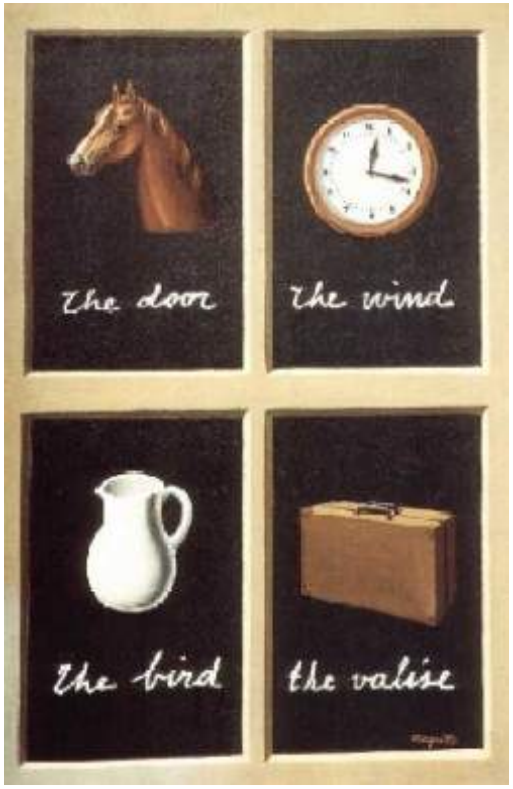
Zwei **Merkhilfen** zu Syntax vs. Semantik

1. Ein alter Informatiker-Schmäh:

   Es gibt 10 Arten von Leuten:
   Die, die Binärzahlen verstehen, und die, die sie nicht verstehen!

2. Für visuell Orientierte:

René Magritte "Der Verrat der Bilder"



"Schlüssel der Träume"

Eine Anmerkung

Zur Verdeutlichung des Zeichencharakters werden wir Elemente eines Alphabets bzw. einer Signatur in diesem Abschnitt immer unterstreichen.

Zum Beispiel meinen wir mit $\underline{0}$ ebendieses Zeichen (die Ziffer) selbst und nicht "die kleinste natürliche Zahl".

Mathematiker müssen selten explizit zwischen Zeichen (Syntax) und deren Bedeutung (Semantik) unterscheiden.
Für Informatiker hingegen gehört die Unterscheidung zum "Kerngeschäft"!

Ein einfaches Beispiel: Additive Terme

Sei $\mathcal{T} = \mathcal{L}(G)$ mit $G = \langle \{T\}, \{\underline{(}, \underline{)}, \underline{+}, \underline{0}, \ldots, \underline{9}\}, P, T \rangle$ wobei

$$P \quad = \quad \{\, T \to \underline{0} \mid \cdots \mid \underline{9} \mid \underline{(}\, T \underline{+} T \,\underline{)} \,\}$$

Zunächst ein paar Überlegungen zur Syntax:

- Die Grammatik ist kontextfrei, (leider?) nicht regulär.
- Wo in der Chomsky-Hierarchie steht die Sprache?
- Was, wenn wir die Klammersymbole weglassen?
- Wie kann man die Grammatik auf mehrstellige Zahlen ergänzen?

Nun zur Semantik: ("intendierte Semantik")
$\mathcal{M} : \mathcal{T} \to \omega \ldots$ „Meaning", interpretiert $\mathcal{T}$ als additive Ausdrücke
Achtung: Ab nun unterscheiden wir zwischen der Menge ω und der Struktur $\mathbb{N}$ der natürlichen Zahlen. (Mehr dazu auf Folie .)

Induktive Definition:

$$\mathcal{M}(\underline{0}) = 0, \ldots, \mathcal{M}(\underline{9}) = 9$$
$$\mathcal{M}(\underline{(}\, t_1 \underline{+} t_2 \,\underline{)}) = \mathcal{M}(t_1) + \mathcal{M}(t_2) \quad \text{für } t_1, t_2 \in \mathcal{T}$$

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: $T \to \underline{\mathtt{x}} \mid \underline{\mathtt{y}} \mid \underline{\mathtt{z}} \mid \cdots$
Semantik: $\mathcal{M}(\underline{\mathtt{x}}) = 1$ oder $\mathcal{M}(\underline{\mathtt{x}}) = 2$ oder ...?
Speicherbelegung/Variablenbelegung $I : \{\underline{\mathtt{x}}, \underline{\mathtt{y}}, \underline{\mathtt{z}}, \ldots\} \to \omega$

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

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

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

- 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 $\underline{(}\underline{(}\underline{\mathtt{x}}\underline{+}\underline{2}\underline{)}\underline{+}\underline{\mathtt{y}}\underline{)}$

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

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

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

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

Hierarchisierung

Einfaches Beispiel: Additive Ausdrücke mit mehrstellige Zahlen

Wir unterscheiden drei Ebenen:

1. Ziffern (Z)
2. Zahlenausdrücke ("Numerale") für natürliche Zahlen (N)
3. additive Terme (T)

Syntax:

$\mathcal{T} = \mathcal{L}(G)$ mit $G = \langle \{T, Z, N\}, \{\underline{(}, \underline{)}, \underline{+}, \underline{0}, \ldots, \underline{9}\}, P, T \rangle$ wobei

$$\begin{aligned} P = \{ & T \to \underline{(}\ T\ \underline{+}\ T\ \underline{)} \mid N \\ & N \to NZ \mid Z \\ & Z \to \underline{0} \mid \cdots \mid \underline{9} \} \end{aligned}$$

Beachte:

Es gilt Z $\subseteq$ N $\subseteq$ T.

Das spiegelt sich direkt in den Produktionen für Z, N und T wieder.

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: $\mathcal{M} : ENV \times \mathcal{T} \to \omega$.

Auswertung eines additiven Terms mit mehrstelligen Zahlen und Variablen

... für $I(\underline{\mathtt{x}}) = 10$, $I(v) = 0$ für alle $v \neq \underline{\mathtt{x}}$:

$$
\begin{aligned}
\mathcal{M}(I, \underline{(}\underline{\mathtt{3}}\underline{\mathtt{0}}\underline{+}\underline{(}\underline{\mathtt{x}}\underline{+}\underline{\mathtt{0}}\underline{\mathtt{2}}\underline{)}\underline{)}) &= \mathcal{M}(I, \underline{\mathtt{3}}\underline{\mathtt{0}}) + \mathcal{M}(I, \underline{\mathtt{x}}\underline{+}\underline{\mathtt{0}}\underline{\mathtt{2}}) \\
&= (\mathcal{M}(I, \underline{\mathtt{3}}) \cdot 10 + \mathcal{M}(I, \underline{\mathtt{0}})) + \mathcal{M}(I, \underline{\mathtt{x}}\underline{+}\underline{\mathtt{0}}\underline{\mathtt{2}}) \\
&= (3 \cdot 10 + 0) + \mathcal{M}(I, \underline{\mathtt{x}}\underline{+}\underline{\mathtt{0}}\underline{\mathtt{2}}) \\
&= 30 + \mathcal{M}(I, \underline{\mathtt{x}}) + \mathcal{M}(I, \underline{\mathtt{0}}\underline{\mathtt{2}}) \\
&= 30 + 10 + \mathcal{M}(I, \underline{\mathtt{0}}\underline{\mathtt{2}}) \\
&= 30 + 10 + (\mathcal{M}(I, \underline{\mathtt{0}}) \cdot 10 + \mathcal{M}(I, \underline{\mathtt{2}})) \\
&= 30 + 10 + (0 + 2) \\
&= 42
\end{aligned}
$$

Grammatik vs. Induktive Definition

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

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 $\underline{\hat{}}$ ersetzt syntaktisch bzw. semantisch? (Intendierte Semantik beachten!)

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

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:

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

Diese Grammatik ist aber in Hinblick auf die Semantik inadäquat!

Auflösung der Mehrdeutigkeit

Das Problem ist nicht, dass $\mathcal{E}$ kontextfrei spezifiziert ist sondern:
Die Ableitungen entsprechen nicht der intendierten Semantik.
Die übliche Prioritätsregel "$\cdot$" vor "$+$" sollte berücksichtigt werden.

Zwei Möglichkeiten:

1. Prioritätsregel (nur) in die Definition von $\mathcal{M}$ einbauen
2. Prioritätsregel gleich in die Definition von $\mathcal{E}$ einbauen:
   $\Longrightarrow$ Definition von $\mathcal{E}$ hierarchisch gestalten

Nur die zweite Lösung unterstützt eine systematische Anbindung der Semantik an die Syntax!

Hierarchische Definition von $\mathcal{E}$

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

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

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

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

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

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

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:

$\mathbf{FamX} = \langle D_X; P_X, K_X, F_X \rangle$,

wobei

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

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: $\{\underline{\mathrm{x}}, \underline{\mathrm{y}}, \underline{\mathrm{z}}, \ldots, \underline{\mathrm{x}_1}, \ldots\}$ (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 $\underline{\mathrm{push0}}\underline{(}\underline{\mathrm{push1}}\underline{(}\underline{\mathrm{pop}}\underline{(}\underline{\mathrm{x}}\underline{)}\underline{)}\underline{)} \in \mathcal{T}(\mathbb{S})$:

Lösung:

- $\underline{\mathrm{x}} \in \mathit{IVS} \overset{T1}{\Longrightarrow} \underline{\mathrm{x}} \in \mathcal{T}(\mathbb{S})$
- $\underline{\mathrm{pop}} \in \mathit{FS}_1(\mathbb{S}),\ \underline{\mathrm{x}} \in \mathcal{T}(\mathbb{S}) \overset{T3}{\Longrightarrow} \underline{\mathrm{pop}}\underline{(}\underline{\mathrm{x}}\underline{)} \in \mathcal{T}(\mathbb{S})$
- $\underline{\mathrm{push1}} \in \mathit{FS}_1(\mathbb{S}),\ \underline{\mathrm{pop}}\underline{(}\underline{\mathrm{x}}\underline{)} \in \mathcal{T}(\mathbb{S}) \overset{T3}{\Longrightarrow} \underline{\mathrm{push1}}\underline{(}\underline{\mathrm{pop}}\underline{(}\underline{\mathrm{x}}\underline{)}\underline{)} \in \mathcal{T}(\mathbb{S})$
- $\underline{\mathrm{push0}} \in \mathit{FS}_1(\mathbb{S}),\ \underline{\mathrm{push1}}\underline{(}\underline{\mathrm{pop}}\underline{(}\underline{\mathrm{x}}\underline{)}\underline{)} \in \mathcal{T}(\mathbb{S}) \overset{T3}{\Longrightarrow}$
  $\underline{\mathrm{push0}}\underline{(}\underline{\mathrm{push1}}\underline{(}\underline{\mathrm{pop}}\underline{(}\underline{\mathrm{x}}\underline{)}\underline{)}\underline{)} \in \mathcal{T}(\mathbb{S})$

Aufgabe:

Zeigen Sie, dass der Ausdruck $\underline{\mathrm{push0}}\underline{(}\underline{\mathrm{push1}}\underline{(}\underline{1}\,\underline{0}\underline{)}\underline{)}$ 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 $\underline{\mathrm{x}}$ — enthalten:
Entsprechend der Definition der Sprache $\mathcal{T}(\mathbf{FamX})$ der
Modellstruktur **FamX** hat

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

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

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 $\mathbb{N}$:
  $\underline{(}\ \underline{1}\ \underline{+}\ \underline{\mathrm{x}}\ \underline{)}$ steht für $\underline{+}\ \underline{(}\ \underline{1}\ \underline{,}\ \underline{\mathrm{x}}\ \underline{)}$ 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.
- $t_1 \underline{\geq} t_2$ steht für $t_2 \underline{\leq} t_1$
- $t_1 \underline{\neq} t_2$ / $t_1 \underline{\geq} t_2$ 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 $v \leftarrow t \in AL(\mathcal{D})$.

(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
$\underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \alpha \in AL(\mathcal{D})$.

Beispiel (Syntax von $AL(\mathcal{D})$)

Wir werden vorallem $AL(\mathbb{Z}) = ALI$ (Assignment Language over Integers) analysieren.

Ein ALI-Programm zur Multiplikation zweier positiver Zahlen:

$$
\begin{array}{l}
\underline{\texttt{begin}} \\
\underline{\mathrm{z}} \underline{\leftarrow} \underline{0} \underline{;} \\
\underline{\texttt{while}} \ \underline{(} \underline{0} \underline{<} \underline{\mathrm{y}} \underline{)} \ \underline{\texttt{do}} \ \underline{\texttt{begin}} \\
\quad \underline{\mathrm{z}} \underline{\leftarrow} \underline{(} \underline{\mathrm{z}} \underline{+} \underline{\mathrm{x}} \underline{)} \underline{;} \\
\quad \underline{\mathrm{y}} \underline{\leftarrow} \underline{(} \underline{\mathrm{y}} \underline{-} \underline{1} \underline{)} \\
\quad \underline{\texttt{end}} \\
\underline{\texttt{end}}
\end{array}
$$

Beachte: Die intendierte Semantik muss erst gerechtfertigt werden!

Syntaktische Analyse

Bottom-Up-Analyse:

von atomaren Teilprogrammen (Zuweisungen) zu komplexen Statements

- $\underline{\mathrm{z}} \in \mathit{IVS},\ \underline{0} \in \mathcal{T}(\mathbb{Z}) \overset{AL1}{\Longrightarrow} \alpha_1 = \underline{\mathrm{z}}\ \underline{\leftarrow}\ \underline{0} \in \mathit{ALI}$
- $\underline{\mathrm{z}} \in \mathit{IVS},\ \underline{(}\ \underline{\mathrm{z}}\ \underline{+}\ \underline{\mathrm{x}}\ \underline{)} \in \mathcal{T}(\mathbb{Z}) \overset{AL1}{\Longrightarrow} \alpha_2 = \underline{\mathrm{z}}\ \underline{\leftarrow}\ \underline{(}\ \underline{\mathrm{z}}\ \underline{+}\ \underline{\mathrm{x}}\ \underline{)} \in \mathit{ALI}$
- $\underline{\mathrm{y}} \in \mathit{IVS},\ \underline{(}\ \underline{\mathrm{y}}\ \underline{-}\ \underline{1}\ \underline{)} \in \mathcal{T}(\mathbb{Z}) \overset{AL1}{\Longrightarrow} \alpha_3 = \underline{\mathrm{y}}\ \underline{\leftarrow}\ \underline{(}\ \underline{\mathrm{y}}\ \underline{-}\ \underline{1}\ \underline{)} \in \mathit{ALI}$
- $\alpha_2, \alpha_3 \in \mathit{AL} \overset{AL2}{\Longrightarrow} \beta = \underline{\texttt{begin}}\ \alpha_2\underline{;}\ \alpha_3\ \underline{\texttt{end}} \in \mathit{ALI}$
- $\underline{(}\ \underline{0} \underline{\leq} \underline{\mathrm{y}}\ \underline{)} \in \mathcal{BA}(\mathbb{Z}),\ \beta \in \mathit{ALI} \overset{AL4}{\Longrightarrow} \gamma = \underline{\texttt{while}}\ \underline{(}\ \underline{0} \underline{\leq} \underline{\mathrm{y}}\ \underline{)}\ \underline{\texttt{do}}\ \beta \in \mathit{ALI}$
- $\alpha_1, \gamma \in \mathit{ALI} \overset{AL2}{\Longrightarrow} \underline{\texttt{begin}}\ \alpha_1\underline{;}\ \gamma\ \underline{\texttt{end}} \in \mathit{ALI}$

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:

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

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

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

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

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

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

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

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

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

Was Sie nun erwartet

4. Die Semantik der Prädikaten- (und der Aussagen-)logik
   - 4.1 Semantik der AL (*WH aus FMod*)
   - 4.2 PL-Interpretationen
   - 4.3 Semantische Grundbegriffe
5. Formalisierung natürlichsprachlicher Information mit PL (*großteils WH aus FMod*)
6. Ausdrücken formaler Information in PL
7. Das Konzept der logischen Konsequenz

Ein (einfaches?) Beispiel [Wason, 1966]

Vor Ihnen liegen 4 Karten, die jeweils auf einer Seite mit einem Buchstaben, auf der anderen Seite mit einer Zahl beschriftet sind.

Beispiel:

A	C	6	7

Regel: (Denken Sie an "Zusicherungen über Daten"!)

> *Wenn auf der einen Seite ein Vokal steht,*
> *dann steht auf der anderen Seite eine gerade Zahl.*

Aufgabe:
Welche Karten müssen umgedreht werden um zu bestimmen, ob die Regel verletzt bzw. eingehalten wurde?

Erinnerung: PL- (und AL-)Syntax

(allgemeine) 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/Atome)

(P2) $p(t_1, \ldots, t_n) \in \mathcal{PF}_\Sigma$ (allgemeine) atomare Formel/Atom
wenn $p \in PS_n$ und $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$ (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

Wir haben die Bedeutung von Formeln noch nicht definiert!

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 $\mathrm{val} : \mathcal{I} \times \mathcal{AF} \mapsto \mathcal{B}$:

(MA1) $\mathrm{val}_I(A) = I(A)$ für $A \in AV$;

(MA2) $\mathrm{val}_I(\top) = 1/\mathbf{t}$ und $\mathrm{val}_I(\bot) = 0/\mathbf{f}$;

(MA3) $\mathrm{val}_I(\neg F) = \mathit{not}\, \mathrm{val}_I(F)$;

(MA4) $\mathrm{val}_I(\,(F * G)\,) = \mathrm{val}_I(F) \circledast \mathrm{val}_I(G)$,
wobei $\circledast$ die logische Funktion zum Operator $*$ ist.

Aussagenlogische Funktionen

Achtung: Wir schreiben oft 0 statt **f** und 1 statt **t**

true	*false*
1	0
$\top$	$\bot$

x	*not*
1	0
0	1
	$\neg$

x	y	*and*	*nand*	*or*	*nor*	*iff*	*xor*	*implies*		*if*	
1	1	1	0	1	0	1	0	1	0	1	0
1	0	0	1	1	0	0	1	0	1	1	0
0	1	0	1	1	0	0	1	1	0	0	1
0	0	0	1	0	1	1	0	1	0	1	0
		$\wedge$	$\uparrow$	$\vee$	$\downarrow$	$\leftrightarrow$	$\not\leftrightarrow$	$\supset$	$\not\supset$	$\subset$	$\not\subset$

Beispiel einer Formelauswertung

(MA1) $\mathrm{val}_I(A) = I(A)$ für $A \in AV$;

von (MA2) $\mathrm{val}_I(\bot) = 0$;

(MA3) $\mathrm{val}_I(\neg F) = \textit{not}\, \mathrm{val}_I(F)$;

von (MA4) $\mathrm{val}_I(\,(F \wedge G)\,) = \mathrm{val}_I(F)\ \textit{and}\ \mathrm{val}_I(G) = 1$, d.h.:
$\mathrm{val}_I(\,(F \wedge G)\,) = 1$ gdw. $\mathrm{val}_I(F) = \mathrm{val}_I(G) = 1$;

von (MA4) $\mathrm{val}_I(\,(F \supset G)\,) = \mathrm{val}_I(F)\ \textit{implies}\ \mathrm{val}_I(G) = 1$, d.h.:
$\mathrm{val}_I(\,(F \supset G)\,) = 1$ gdw. $\mathrm{val}_I(F) = 0$ oder $\mathrm{val}_I(G) = 1$

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

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

Wahrheitstafeln

- Kompakte Berechnung der Formelwerte für alle Interpretationen
- Unter jedem Operator steht der Wert der entsprechenden Teilformel.

A	B	$((A \wedge \neg B) \supset \bot)$
1	1	1 0 0 1 1 0
1	0	1 1 1 0 0 0
0	1	0 0 0 1 1 0
0	0	0 0 1 0 1 0

bedeutet:

$I(A) = \mathbf{t}$, $I(B) = \mathbf{t}$: $\mathrm{val}_I(\cdots) = \cdots = \mathbf{t}$

$I(A) = \mathbf{t}$, $I(B) = \mathbf{f}$: $\mathrm{val}_I(\cdots) = \cdots = \mathbf{f}$

$I(A) = \mathbf{f}$, $I(B) = \mathbf{t}$: $\mathrm{val}_I(\cdots) = \cdots = \mathbf{t}$

$I(A) = \mathbf{f}$, $I(B) = \mathbf{f}$: $\mathrm{val}_I(\cdots) = \cdots = \mathbf{t}$

false
0
$\bot$

x	*not*
1	0
0	1
	$\neg$

x	y	*and*	*implies*
1	1	1	1
1	0	0	0
0	1	0	1
0	0	0	1
		$\wedge$	$\supset$

Semantik der PL – Hinweise zur Ausdrucksstärke

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 von Modellstrukturen (Axiomatisierung)

Modellstruktur $\mathbb{Z}$:

- $\forall x \forall y\, x + y = y + x$ (Addition ist kommutativ)
- $\forall x \exists y\, x + y = 0$ (Invertierbarkeit der Addition)
- $\forall x \forall y (x < y \supset 0 - y < 0 - x)$ ("Minus" ist anti-monoton)

Ausdrücken von komplexen Prädikaten

Über $\mathbb{N}$ lässt sich "x ist eine Primzahl" wie folgt ausdrücken:

- $\forall y \forall z [x = y * z \supset (y = 1 \vee z = 1)] \wedge 1 < x$

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 $\Longrightarrow \Phi(P)$: $D^n \to \{\mathbf{f}, \mathbf{t}\}$
  - $c \in KS_\Sigma \Longrightarrow \Phi(c) \in D$
  - $f \in FS_\Sigma$ n-stellig $\Longrightarrow \Phi(f)$: $D^n \to D$
- $\xi : IVS \to D$ 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 $\mathcal{I}' \overset{v}{\sim} \mathcal{I}$

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

Erinnerung: $\xi \overset{v}{\sim} \xi' \ldots \xi(w) = \xi'(w)$ 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 $FV(F) = \{\}$!

$$
\begin{aligned}
\mathrm{val}_{\mathcal{I}}(F) = \mathbf{t} &\overset{MP6}{\Longleftrightarrow} \mathrm{val}_{\mathcal{I}'}(\exists y \neg P(x, f(y))) = \mathbf{t} \text{ für alle } \mathcal{I}' \overset{x}{\sim} \mathcal{I} \\
&\overset{MP7}{\Longleftrightarrow} \mathrm{val}_{\mathcal{I}''}(\neg P(x, f(y))) = \mathbf{t} \text{ für ein } \mathcal{I}'' \overset{y}{\sim} \mathcal{I}' \\
&\overset{MP2}{\Longleftrightarrow} \mathrm{val}_{\mathcal{I}''}(P(x, f(y))) = \mathbf{f} \\
&\overset{MP1}{\Longleftrightarrow} \mathcal{M}_{\mathcal{T}}(\xi'', x) \geq \mathcal{M}_{\mathcal{T}}(\xi'', f(y)) \text{ für } \mathcal{I}'' = \langle D, \Phi, \xi'' \rangle \\
&\overset{MT1,MT3}{\Longleftrightarrow} \xi''(x) \geq id(\mathcal{M}_{\mathcal{T}}(\xi'', y)) \iff \xi''(x) \geq \xi''(y)
\end{aligned}
$$

$\mathrm{val}_{\mathcal{I}}(F) = \mathbf{t}$, 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).

Beachte: Keine Änderung gegenüber der Aussagenlogik,
außer dass Interpretationen nun komplexere Gebilde sind.

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 Programm für das ganze Semester

- Was genau sind "Formeln der Prädikatenlogik erster Stufe"?
- Was ist eine "axiomatische Theorie" in dieser Sprache (PL)?
- Wie kann man die Bedeutung unserer Sprache präzise festlegen?
  Erinnerung: Frege und Russell sind letztlich auch deshalb gescheitert, weil sie keine klare Syntax/Semantik-Trennung kannten!
- Wie drückt man mathematische – und andere – Konzepte in PL aus?
- Was genau heißt "ein Satz folgt (logisch) aus Annahmen/Axiomen"?
- Wie, also mit welchen Regeln, kann bzw. soll man "herleiten"?
- Kann man wirklich alles herleiten, was logisch folgt?
- Kann man Leibniz' bzw. Hilberts Traum verwirklichen?
- Was hat das alles mit beweisbar korrekten Programmen zu tun?

Formalisieren natürlichsprachlicher Information

[WH aus bzw. Vertiefung zu FMod]

Wahl der Stelligkeit von Prädikatensymbolen:

"Max liest Zeitung."

Folgende Möglichkeiten bestehen:

- Nullstelliges Prädikat (Aussagenvariable):
  $Max_liest_Zeitung$
- Einstelliges Prädikat:
  $Liest_Zeitung(max)$
- Zweistelliges Prädikat:
  $Liest(max, zeitung)$

Generell gilt:

Die beste Wahl hängt vom Kontext ab.

Maxime:

Wähle ein eigenes Symbol für Satzteile, die öfter auftreten (können).

Zeit-, Eigenschafts- und Hauptwörter

Sokrates ist sterblich.

Möglichkeit 1: Hauptwort wird zum Prädikat.

Sehr problematisch:

$Sokrates_ist(sterblich)$

Möglichkeit 2: Zeit-/Eigenschaftswort wird zum Prädikat.

$Ist_sterblich(sokrates)$

Die zweite Möglichkeit ist fast immer die richtige.

Maxime:

Mache Zeitwörter bzw. Eigenschaften zu Prädikatensymbole und Hauptwörter zu ihren Argumenten.

Fürwörter (Pronomen)

Persönliche Fürwörter (ich, du, er, sie, es, ...) oder bezügliche Fürwörter (der, welcher, ...) vermeiden Wiederholungen und stellen Bezüge her.

Mia liebt Max, der wiederum sie liebt.

Umformung in

Mia liebt Max und Max liebt Mia.

ergibt die Formel

$$Liebt(mia, max) \wedge Liebt(max, mia)$$

Maxime:

Ersetze Fürwörter durch Namen. Wiederhole Satzteile, die durch Fürwörter ersetzt wurden. Mache dabei implizite Konjunktionen explizit.

Unbestimmte Fürwörter wie „nichts“ ziehen Subjekt/Objekt und Negation zusammen.

Nichts währt ewig.

Trenne Verneinung und Subjekt:

Es existiert nicht etwas, das ewig währt.

Ergibt die Formel

$\neg\exists x\, \mathit{Währt_ewig}(x)$

Ich kann nichts sehen. $\neg\exists x\, \mathit{Kann_sehen}(\mathit{chris}, x)$

Maximen:

Führe Variablen für unbestimmte Fürwörter ein.
Mache Negationen explizit.

Fürwörter und Bindewörter

Vor mir ist etwas Großes, und es ist hungrig.

Die zwei Teilsätze können nicht getrennt behandet werden: „es“ bezieht sich auf „etwas Großes“. Die entsprechende Variable muss überall dieselbe sein.

Es gibt etwas, das vor mir ist, das groß ist, und das hungrig ist.
$\exists x\,(\mathit{Befindet_sich_vor(x,chris)} \wedge \mathit{Groß}(x) \wedge \mathit{Hungrig}(x))$

Maxime:

Erstrecken sich die Bezüge von Fürwörtern über Bindewörter hinweg, behandle die Fürwörter vor den Bindewörtern.

Quantoren, Typen und Beziehungen (1)

Jeder Student ist jünger als irgendein Professor.

Maxime:

Identifiziere Individuenarten und drücke durch einstellige Typprädikate aus.

„ist Student“ $\Rightarrow Stud(\ldots)$
„ist Professor“ $\Rightarrow Prof(\ldots)$

Max ist Student. $\Rightarrow Stud(max)$
Gernot ist Professor. $\Rightarrow Prof(gernot)$

Maxime:

Identifiziere Beziehungen (Relationen) und formalisiere sie durch Prädikatensymbole mit entsprechender Stelligkeit.

„ist jünger als“ $\Rightarrow Jünger(\ldots,\ldots)$
Max ist jünger als Chris. $\Rightarrow Jünger(max, chris)$

Quantoren, Typen und Beziehungen (2)

Jeder Student ist jünger als irgendein Professor.

Maxime:

Verwende $\forall$-Quantoren für „alle“/„jede“ und $\exists$-Quantoren für „es gibt“/„jemand“/„irgendein“.

$\forall x\,(Stud(x) \supset \exists y\,(Prof(y) \wedge Jünger(x, y)))$

Für alle Individuen x gilt:
Falls x ein Student ist, gibt es mindestens ein Individuum y,
sodass y Professor ist und x jünger als y ist.

Quantoren und Eigenschaften

Alle vernünftigen Leute verabscheuen Gewalt.

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

$\forall x\,(\mathit{Vernünftig}(x) \supset \mathit{Verabscheut}(x, \mathit{gewalt}))$

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!)

$\exists x\,(\mathit{Vernünftig}(x) \wedge \mathit{Verabscheut}(x, \mathit{gewalt}))$

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

Alternierende Quantoren

Vergleichen Sie:

$\forall x\, \exists y\, \mathit{Mutter}(y, x)$	$\exists y\, \forall x\, \mathit{Mutter}(y, x)$
„Jeder hat eine Mutter.“	„Jemand ist die Mutter von allen.“
y hängt vom gewählten x ab.	y ist unabhängig vom gewählten x.

Vertauschung unterschiedlicher Quantoren ändert die Bedeutung!

Funktionssymbole

Jedes Kind ist jünger als seine Mutter.

„x ist ein Kind“ $\Rightarrow Kind(x)$
„y ist Mutter von x“ $\Rightarrow Mutter(y, x)$

$\forall x \, \forall y [(Kind(x) \wedge Mutter(y, x)) \supset Jünger(x, y)]$

Diese Formalisierung berücksichtigt nicht, dass jedes Kind genau eine genetische Mutter hat.

Besser: Fasse „Mutter“ als Funktion auf, die jedem Kind seine eindeutig bestimmte Mutter zuordnet.

„Mutter von x“ $\Rightarrow mutter(x)$

$\forall x \, (Kind(x) \supset Jünger(x, mutter(x))$

Maxime:

Wähle Funktionssymbole für strikt nacheindeutige Beziehungen.

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) – Pferd(x) 2-stellig: – *Besitzt*(x,y)
– *Pferd*(x) – *Klein*(x) – *Füttert*(x,y)
– *Futter_reicht_für*(x) – *Groß*(x)

Alternativ (z.B.): *reicht_für*(*futter*,x) statt *Futter_reicht_für*(x)

- 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:
  $\mathrm{DFrnfaP} = \neg\forall y(Pferd(y) \supset Futter_reicht_für(y))$

Alternativ: Wenn man "alle Pferde" als "alle seine Pferde" interpretiert:

$\mathrm{DFrnfaP}(x) = \neg\forall y((Pferd(y) \wedge Besitzt(x,y)) \supset Futter_reicht_für(y))$

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

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 noch 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]:
  $\forall y \forall z [x = y * z \supset (y = 1 \vee z = 1)] \wedge 1 < x$
- Es gibt nicht immer ein inverses Element bzgl. Addition [Satz]:
  $\neg \forall x \exists y \; x + y = 0$
- x ist mindesten dreimal so groß wie y [binäres Prädikat]:
  $(y + y) + y = x \vee (y + y) + y < x$
  Achtung: Weder für $\leq$ noch für 2 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 ('$x - y$' nicht '$x \dot{-} y$'!) ist als 3-stellige Relation '$x - y = z$' 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)

<u>Jede</u> 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'):
  $\forall x \forall y \forall z [(R(x,y) \wedge R(x,z)) \supset \exists u (R(y,u) \wedge R(z,u))]$

Ausdrücken mit PL-Formeln (Forts.)

Im Kontext relationaler Datenbanken ist funktionale Abhängigkeit eine oft gebrauchte Eigenschaft.

Beachte: Jede Relationale Datenbank lässt sich als eine Modellstruktur ohne Funktionen verstehen, bei der alle Elemente des Gegenstandsbereichs auch Konstante sind. Die Argumentstellen der Relationen (Prädikate) nennt man in diesem Kontext Attribute.

Zitat aus `http://en.wikipedia.org/wiki/Functional_dependency`:
Given a relation R, a set of attributes X in R is said to functionally determine another set of attributes Y, also in R, (written $X \to Y$) if, and only if, each X value is associated with precisely one Y value; R is then said to satisfy the functional dependency $X \to Y$.
Beachte: Die Definition ist nicht ganz frei von Zweideutigkeit!

Folgende PL-Formel drückt $\{x_1, x_2\} \to \{x_3, x_4\}$ in $R(x_1, x_2, x_3.x_4)$ aus und beseitigt eventuelle Unklarheiten (z.B. bzgl. "one Y value"):

$$\forall x_1 \forall x_2 \forall y \forall z \forall u \forall v [(R(x_1, x_2, y, z) \land R(x_1, x_2, u, v)) \supset (y = u \land z = v)]$$

Ein Programm für das ganze Semester

- Was genau sind "Formeln der Prädikatenlogik erster Stufe"?
- Was ist eine "axiomatische Theorie" in dieser Sprache (PL)?
- Wie kann man die Bedeutung unserer Sprache präzise festlegen?
  Erinnerung: Frege und Russell sind letztlich auch deshalb gescheitert, weil sie keine klare Syntax/Semantik-Trennung kannten!
- Wie drückt man mathematische – und andere – Konzepte in PL aus?
- Was genau heißt "ein Satz folgt (logisch) aus Annahmen/Axiomen"?
- Wie, also mit welchen Regeln, kann bzw. soll man "herleiten"?
- Kann man wirklich alles herleiten, was logisch folgt?
- Kann man Leibniz' bzw. Hilberts Traum verwirklichen?
- Was hat das alles mit beweisbar korrekten Programmen zu tun?

Was bisher geschah & was Sie heute erwartet

1. Logik: Woher? Warum? Wie?
2. Die Sprache der Prädikaten (und Aussagen-)Logik (Syntax)
3. Von Syntax zu Semantik
4. Die Semantik der Prädikaten- (und der Aussagen-)logik
5. Formalisierung natürsprachlicher Information mit PL
6. Ausdrücken formaler Information in PL
7. Das Konzept der logischen Konsequenz

Gültige und ungültige Schlüsse

Alle ganzen Zahlen sind rational.
5 ist eine ganze Zahl.
} Prämissen, Annahmen

---

5 ist rational. — Konklusion, Folgerung

Alle rationalen Zahlen sind ganz.
Die Zahl π ist rational.

Die Zahl π ist ganz.

Alle Psycholytiker sind schmöll.
Adi ist ein Psycholytiker.

Adi ist schmöll.

Ungültige Schlüsse (Inferenzen)

Alle ganzen Zahlen sind rational.
5 ist rational.

---

5 ist eine ganze Zahl.

5 ist rational.
5 ist eine ganze Zahl.

Alle ganzen Zahlen sind rational.

Kriterium für die Gültigkeit von Schlüssen

Immer wenn alle Prämissen wahr sind, ist auch die Konklusion wahr.

Logisch korrektes Schließen

Kriterium für die Gültigkeit von Schlüssen

Immer wenn alle Prämissen wahr sind, ist auch die Konklusion wahr.

Was bedeutet hier "immer"?

Achtung: In der Logik gibt es Wahrheit nur relativ zu Interpretationen!

Alle Psycholytiker sind schmöll.	Alle P sind Q.	$\forall x(P(x) \supset Q(x))$
Adi ist ein Psycholytiker.	A ist P.	$P(a)$
Adi ist schmöll.	A ist Q.	$Q(a)$

Kriterium für die logische Gültigkeit von Schlüssen (genauer)

Für alle Interpretationen $\mathcal{I}$ gilt Folgendes: Wenn alle Prämissen in $\mathcal{I}$ wahr sind, dann ist auch die Konklusion in $\mathcal{I}$ wahr.

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 $\mathrm{val}_{\mathcal{I}}(F_n) = \mathbf{t}$ 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 $F_1, ..., F_n$ (der Formelmenge $\mathcal{F}$).“

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

Beachte:
Ein Schluss von G aus den Prämissen $F_1, \ldots, F_n$ ist genau dann gültig wenn $F_1, \ldots, F_n \models G$ (also genau dann wenn die Konklusion aus den Prämissen tatsächlich logisch folgt).

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. $\models F \leftrightarrow G$.

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:

- $\neg\forall x F \equiv \exists x \neg F$, $\neg\exists x F \equiv \forall x \neg F$, $\neg\forall x \neg F \equiv \exists x F$, $\neg\exists x \neg F \equiv \forall x F$
- $\forall x \forall y F \equiv \forall y \forall x F$, $\exists x \exists y F \equiv \exists y \exists x F$
- $\forall x (F \wedge G) \equiv \forall x F \wedge \forall x G \equiv \forall x F \wedge \forall y F(x/y)$
- $\exists x (F \vee G) \equiv \exists x F \vee \exists x G \equiv \exists x F \vee \exists y G(x/y)$
- $\forall x (F^{-x} \supset G) \equiv F \supset \forall x G$, $\exists x (F^{-x} \supset G) \equiv F \supset \exists x G$
  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!

Aussagenlogische Schlüsse

Manche logische Beziehungen zwischen Sätzen nehmen nur auf die aussagenlogische Form Bezug.

Beispiel *Modus Ponens*:

"Wenn Fritz eine Pistole hat, ist er der Mörder. Fritz hat eine Pistole. Daher ist Fritz der Mörder." ist aussagenlogisch gültig, da die relevante Beziehung zwischen den drei Sätzen durch $F \supset P,\ F \models P$ repräsentiert werden kann: P ist immer wahr ist, wenn F und $F \supset P$ wahr sind.

Beachte: die prädikatenlogische Form von F und P ist hier irrelevant.

Beispiel *Schluss aus Disjunktion*:

"Es gibt Eis oder Torte. Es gibt kein Eis mehr. Daher gibt es Torte." ist logisch gültig – unabhängig davon ob "oder" inklusiv oder exklusiv interpretiert wird.
Folgender Schluss ist nur für exklusive Disjunktion logisch korrekt:
"Die Ampel zeigt grün oder rot. Sie zeigt rot. Daher zeigt sie nicht grün."

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?

Ein Programm für das ganze Semester

- Was genau sind "Formeln der Prädikatenlogik erster Stufe"?
- Was ist eine "axiomatische Theorie" in dieser Sprache (PL)?
- Wie kann man die Bedeutung unserer Sprache präzise festlegen?
  Erinnerung: Frege und Russell sind letztlich auch deshalb gescheitert, weil sie keine klare Syntax/Semantik-Trennung kannten!
- Wie drückt man mathematische – und andere – Konzepte in PL aus?
- Was genau heißt "ein Satz folgt (logisch) aus Annahmen/Axiomen"?
- Wie, also mit welchen Regeln, kann bzw. soll man "herleiten"?
- Kann man wirklich alles herleiten, was logisch folgt?
- Kann man Leibniz' bzw. Hilberts Traum verwirklichen?
- Was hat das alles mit beweisbar korrekten Programmen zu tun?

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]

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

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

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

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 $\mathcal{F} \models G$ <u>nicht</u> 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 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!)

Was bisher geschah & was Sie nun erwartet

1. Logik: Woher? Warum? Wie?
2. Die Sprache der Prädikaten (und Aussagen-)Logik (Syntax)
3. Von Syntax zu Semantik
4. Die Semantik der Prädikaten- (und der Aussagen-)logik
5. Formalisierung natürsprachlicher Information mit PL
6. Ausdrücken formaler Information in PL
7. Das Konzept der logischen Konsequenz
8. Logische Kalküle – der Tableau-Kalkül
   - 8.1 Kalkül-Typen: Frege-Hilbert, Sequentialkalkül, Natürliches Schließen
   - 8.2 Tableau-Kalkül — Teil 1: AL-Tableaux
   - 8.3 Tableau-Kalkül — Teil 2: PL-Tableaux ohne Gleichheit
   - 8.4 Tableau-Kalkül — Teil 3: Gleichheitsregeln für PL-Tableaux
   - * Jeweils mit vielen Beispielen!

Logische Kalküle

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

[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:

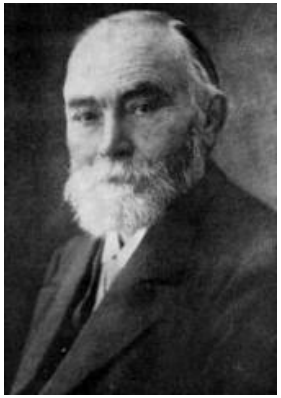
Korrektheit von $\mathcal{K}$:

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

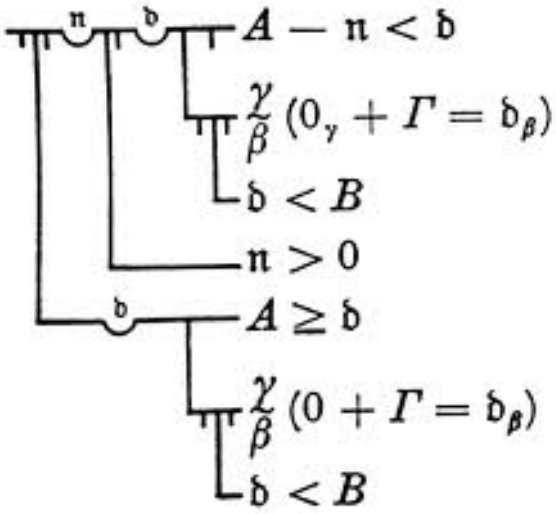
Erinnerung an einen bekannten Namen



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*:



Zwei weitere alte Bekannte

David Hilbert (1862 – 1943). Einer der gößten und einflussreichsten Mathematiker aller Zeiten. Vielleicht der letzte große mathematische Universalist: entscheidende Beiträge zu Geometrie, Topologie, Algebra, Zahlentheorie, Funktionalanalysis, Logik, aber auch zur theoretischen Physik.

Kurt Gödel (1906 - 1978). Österreichisch-amerikan. Logiker, Mathematiker und Philosoph. Berühmt vor allem für seine *Unvollständigkeitssätze* der Arithmetik. Seine Erkenntnisse sind – ähnlich wie die von Alan Turing – von grundlegender Bedeutung für die Informatik.

Ein (Frege-)Hilbert-Typ-Kalkül für die klassische AL

Axiome:
1. $(A \supset (B \supset A))$
2. $((A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C)))$
3. ...

Regel: $\dfrac{A \quad A \supset B}{B}$ (Modus ponens)

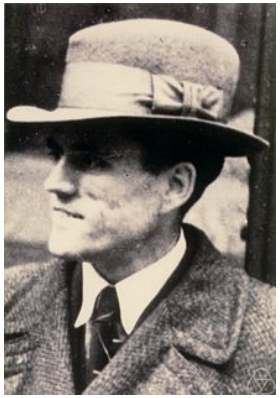
$$\dfrac{\dfrac{\text{Axiom 1: } B \mapsto A}{A \supset (A \supset A)} \qquad \dfrac{\dfrac{\text{Axiom 1: } B \mapsto (A \supset A)}{A \supset ((A \supset A) \supset A)} \quad \dfrac{\text{Axiom 2: } B \mapsto (A \supset A),\ C \mapsto A}{(A \supset ((A \supset A) \supset A)) \supset ((A \supset (A \supset A)) \supset (A \supset A))}}{(A \supset (A \supset A)) \supset (A \supset A)}}{A \supset A}$$

Nachteile:

- Vollständigkeit ist nicht einfach nachzuweisen. (Gödels Dissertation!) [Korrektheit hingegen ist einfach – überlegen Sie warum! Induktion!]
- Undurchsichtige Beweise auch von einfachen Formeln (siehe $A \supset A$).
- Ableitungen entsprechen nicht typischen natürlichsprachlichen bzw. mathematischen Argumenten und Argumentationsketten.
- Keine direkte Beziehung zur Semantik.
- Beweissuche ist schwer zu automatisieren.

Zwei weitere Kalkültypen

Gerhard Gentzen: *Untersuchungen über das logische Schließen.* (1934/35)



Gerhard Gentzen (1909 - 1945). Deutscher Logiker, Mitarbeiter von David Hilbert. `http://de.wikipedia.org/`: "Die nachhaltige Bedeutung der von ihm entwickelten Methoden, Regeln und Strukturen zeigt sich heute vor allem in wichtigen Teilgebieten der Informatik, der Verifikation von Programmen. Dabei werden formale Beweise selbst als Programme gedeutet."

Natürliches Schließen

Grundidee: Logisches Schließen benützt nicht Axiome (vgl. Hilbert-Typ), sondern erfolgt nach Regeln, die dem natürlich-sprachlichen Gebrauch der logischen Konnektive – in Prämissen bzw. Konklusionen – entsprechen.
Beziehungen zu funktionalen Programmen (Curry-Howard-Isomorphismus)
Nachteil: Ebenfalls nicht zur Beweissuche geeignet.

Sequentialkalül

Grundidee: Statt einzelner Formeln analysiert man ganze "Sequente" $F_1, \ldots, F_n \longrightarrow G$. Nur die Schnittregel verhindert effiziente Beweissuche. Aber nach Gentzens "Hauptsatz" ist diese eliminierbar.

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 $(A \supset B) \wedge A \models B$

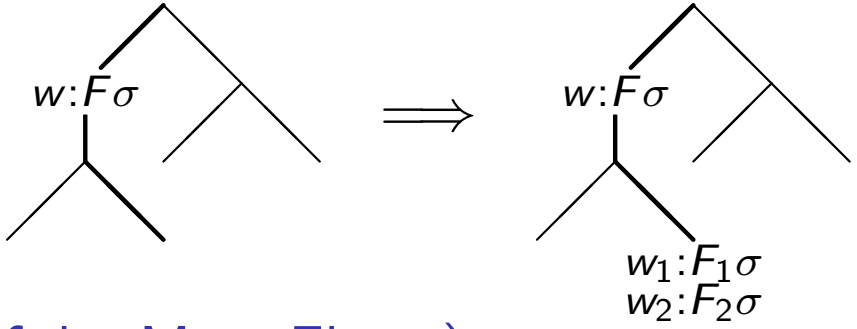
	(1) $\mathbf{t} : (A \supset B) \wedge A$	Annahme
	(2) $\mathbf{f} : B$	Annahme
	(3) $\mathbf{t} : A \supset B$	Zerlegung von (1)
	(4) $\mathbf{t} : A$	Zerlegung von (1)
(5) $\mathbf{f} : A$	(6) $\mathbf{t} : B$	Zerlegung von (3)
$\times$	$\times$	Widerspruch zu (4) bzw. (2)

Tableau-Expansion

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

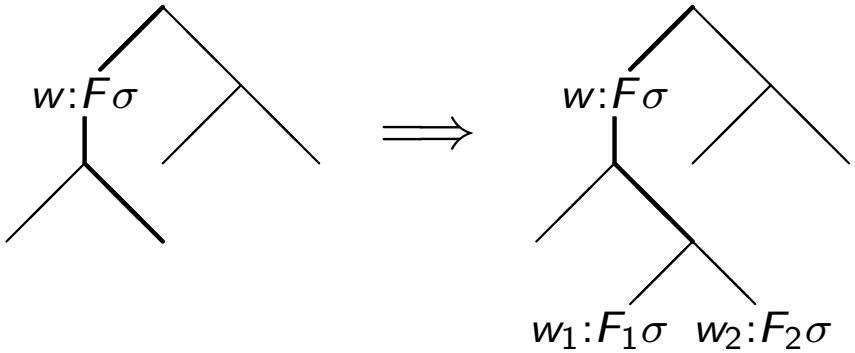
α-Regeln (Konjunktion auf der Meta-Ebene):

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



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 $\mathbf{t}{:}F_1$,..., $\mathbf{t}{:}F_n, \mathbf{f}{:}G$ offen ist, so gibt es ein Gegenbeispiel zu $F_1, ..., F_n \models G$.

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 $F_1, ..., F_n \models G$, 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
(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 $\Uparrow$	
(11)	$\mathbf{t} : C$ von 7 $\times$ Wid. (9/11)	

dieser Ast muss nicht mehr weiter expandiert werden

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

$(^2)$: x darf in B nicht frei vorkommen ($x \notin FV(A)$)

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!

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 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)

Hinweis: 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 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 zweimal auf die Formel $\mathbf{t} : \forall x P(x)$ – und zwar mit jeweils verschiedenen Instanzen – angewendet werden.

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 könnte.

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

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

PL-Tableaux – Beispiele (8) [Forts.]

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

(1)	$\mathbf{t} : \forall x \forall y \forall z[(R(x,y) \wedge R(y,z)) \supset R(x,z)]$	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)	$\mathbf{t} : \forall y \forall z[(R(c,y) \wedge R(y,z)) \supset R(c,z)]$	von 1 (γ)
(11)	$\mathbf{t} : \forall z[(R(c,d) \wedge R(d,z)) \supset R(c,z)]$	von 10(γ)
(12)	$\mathbf{t} : (R(c,d) \wedge R(d,c)) \supset R(c,c)$	von 11
(13)	$\mathbf{f}{:}R(c,d)$ v. 9 $\times$ (7/13) | (14) $\mathbf{t} : R(d,c)$	von 9
	(15) $\mathbf{f} : R(c,d) \wedge R(d,c)$ 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 Gleichungsatom 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→3
	$\times$	$AB^{=}$

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

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

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

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

PL-Tableaux mit Gleichheit – Beispiele (5)

$Th(\mathbb{S}) \models \forall x \forall y[x = \underline{\text{push1}}(y) \supset \neg\underline{\text{ist0?}}(x)]$

(1)	$\mathbf{t} : \forall x \exists y[\underline{\text{ist0?}}(x) \supset x = \underline{\text{push0}}(y)]$	Annahme (γ)
(2)	$\mathbf{t} : \forall x \forall y\, \underline{\text{push0}}(x) \neq \underline{\text{push1}}(y)$	Annahme (γ)
(3)	$\mathbf{f} : \forall x \forall y[x = \underline{\text{push1}}(y) \supset \neg\underline{\text{ist0?}}(x)]$	Annahme (δ)
(4)	$\mathbf{f} : \forall y[a = \underline{\text{push1}}(y) \supset \neg\underline{\text{ist0?}}(a)]$	von 3 (δ)
(6)	$\mathbf{f} : a = \underline{\text{push1}}(b) \supset \neg\underline{\text{ist0?}}(a)$	von 4
(7)	$\mathbf{t} : a = \underline{\text{push1}}(b)$	von 6
(8)	$\mathbf{f} : \neg\underline{\text{ist0?}}(a)$	von 6
(9)	$\mathbf{t} : \underline{\text{ist0?}}(a)$	von 8
(10)	$\mathbf{t} : \exists y[\underline{\text{ist0?}}(a) \supset a = \underline{\text{push0}}(y)]$	von1 (δ)
(11)	$\mathbf{t} : \underline{\text{ist0?}}(a) \supset a = \underline{\text{push0}}(c)$	von 10
(12)	$\mathbf{f} : \underline{\text{ist0?}}(a)$	
	$\times$ Wid.: 9/12	
(13)	$\mathbf{t} : a = \underline{\text{push0}}(c)$	von 11
(14)	$\mathbf{t} : \forall y\, \underline{\text{push0}}(c) \neq \underline{\text{push1}}(y)$	von 2 (γ)
(15)	$\mathbf{t} : \underline{\text{push0}}(c) \neq \underline{\text{push1}}(b)$	von 14
(16)	$\mathbf{f} : \underline{\text{push0}}(c) = \underline{\text{push1}}(b)$	von 15
(17)	$\mathbf{f} : a = \underline{\text{push1}}(b)$	$S^{=}$:13$\rightarrow$16
	$\times$ Wid.: 7/17	

Überlege: Wie finden wir wirklich benötigte Axiome (hier 1 und 2)?

Beispiele (6): Nochmals Alfred und Hans ... [WH]

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?

(1)	$\mathbf{t} : \exists x(F(x) \wedge S(x,a))$	Ann.(δ)
(2)	$\mathbf{t} : \neg F(h)$	Ann.
(3)	$\mathbf{t} : \forall x \forall y(S(x,y) \supset M(x,y))$	Ann.(γ)
(4)	$\mathbf{t} : \exists x[M(x,a) \wedge \forall y(M(y,a) \supset y = x)]$	Ann.(δ)
(5)	$\mathbf{f} : \neg M(h,a)$	Ann.
(6)	$\mathbf{t} : F(b) \wedge S(b,a)$	von 1
(7)	$\mathbf{t} : M(c,a) \wedge \forall y(M(y,a) \supset y = c)$	von 4
(8)	$\mathbf{t} : M(c,a)$	von 7
(9)	$\mathbf{t} : \forall y(M(y,a) \supset y = c)$	von 7 (γ)
(10)	$\mathbf{t} : F(b)$	von 6
(11)	$\mathbf{t} : S(b,a)$	von 6
(12)	$\mathbf{t} : \forall y(S(b,y) \supset M(b,y))$	von 3 (γ)
(13)	$\mathbf{t} : S(b,a) \supset M(b,a)$	von 12
(14)	$\mathbf{f} : S(b,a)$	von 13
	$\times$ (11/14)	
(15)	$\mathbf{t} : M(b,a)$	von 13
(16)	$\mathbf{t} : M(b,a) \supset b = c$	von 9
(17)	$\mathbf{f} : M(b,a)$	von 16
	$\times$ (15/17)	
(18)	$\mathbf{t} : b = c$	von 16
(19)	$\mathbf{t} : M(h,a)$	von 5
(20)	$\mathbf{t} : M(h,a) \supset h = c$	von 9
(21)	$\mathbf{f} : M(h,a)$	von 20
	$\times$ (19/21)	
(22)	$\mathbf{t} : h = c$	von 20
(23)	$\mathbf{f} : F(h)$	von 2
(24)	$\mathbf{f} : F(c)$	$S^{=}$:22$\rightarrow$23
(25)	$\mathbf{t} : F(c)$	$S^{=}$:18$\rightarrow$10
	$\times$ (24/25)	

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.

Ein Programm für das ganze Semester

- Was genau sind "Formeln der Prädikatenlogik erster Stufe"?
- Was ist eine "axiomatische Theorie" in dieser Sprache (PL)?
- Wie kann man die Bedeutung unserer Sprache präzise festlegen?
  Erinnerung: Frege und Russell sind letztlich auch deshalb gescheitert, weil sie keine klare Syntax/Semantik-Trennung kannten!
- Wie drückt man mathematische – und andere – Konzepte in PL aus?
- Was genau heißt "ein Satz folgt (logisch) aus Annahmen/Axiomen"?
- Wie, also mit welchen Regeln, kann bzw. soll man "herleiten"?
- Kann man wirklich alles herleiten, was logisch folgt?
- Kann man Leibniz' bzw. Hilberts Traum verwirklichen?
- Was hat das alles mit beweisbar korrekten Programmen zu tun?

Was bisher geschah & was Sie nun erwartet

1. Logik: Woher? Warum? Wie?
2. Die Sprache der Prädikaten (und Aussagen-)Logik (PL/AL-Syntax)
3. Von Syntax zu Semantik
4. Die Semantik der Prädikaten- (und der Aussagen-)logik
5. Formalisierung natürlichsprachlicher Information mit PL
6. Ausdrücken formaler Information in PL
7. Das Konzept der logischen Konsequenz
8. Logische Kalküle – der Tableau-Kalkül
9. Wichtige Eigenschaften der PL
   - Die Unentscheidbarkeit der PL

Wichtige Eigenschaften der Prädikatenlogik

- (Abstrakte) Vollständigkeit:
  Es gibt (nicht terminierende) Algorithmen, die genau alle gültigen Formeln aufzählen. Bzw.: Es gibt vollständige Kalküle für die PL. (Gödel, 1929)
- Unentscheidbarkeit:
  Es gibt keinen Algorithmus mit dem man die Gültigkeit (bzw. die Erfüllbarkeit) für beliebige PL-Formeln entscheiden kann. (Turing, Church, 1936)
- Satz von Löwenheim-Skolem:
  Jede erfüllbare Formel hat ein Modell dessen Gegenstandsbereich aus natürlichen Zahlen besteht. (Löwenheim, 1915; Skolem, 1919)
- 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)

Noble Träume [Erinnerung]

Gottfried Wilhelm Leibniz (1646 – 1716). Deutscher Philosoph, Wissenschaftler, Mathematiker, Diplomat, Physiker, Theologe, Historiker, Politiker, Bibliothekar.
Informatisch bedeutsam: Binärsystem, Rechenmaschinen, Ideen zur Logik und zur Automatischen Deduktion.

Leibniz' Traum (bezogen auf seine "Characteristica Universalis"):

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

Hilberts Traum (etwas bescheidener, weil nur auf Mathematik bezogen):

> *Da ist das Problem, suche die Lösung. Du kannst sie durch reines Denken finden; denn in der Mathematik gibt es kein Ignorabimus.*

Überlegen bzw. beurteilen Sie selbst:

Welche (positiven und negativen!) Konsequenzen haben die Entdeckungen der Logik/Informatik im 20. Jahrhundert für Leibniz' und Hilberts Träume?

Die Unentscheidbarkeit der Prädikatenlogik

Zur Erinnerung:

Entscheidbare (= rekursive) Sprachen

Eine formale Sprache heißt **rekursiv** oder **entscheidbar** wenn sie von einer (deterministischen) Turingmaschine akzeptiert wird, die **immer hält**.
Man sagt dann auch: die TM **entscheidet** die Sprache.

Satz (Unentscheidbarkeit prädikatenlogischer Gültigkeit)

Die Menge der gültigen PL-Formeln ist keine entscheidbare Sprache.

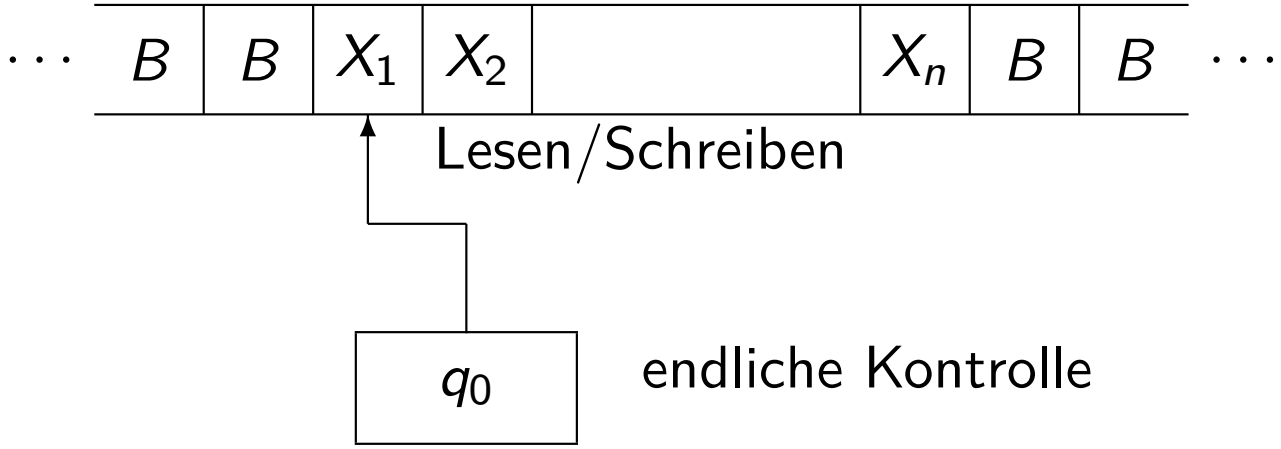
Wegen des Deduktionstheorems und der Church-Turing-These gilt auch:

Satz (Unentscheidbarkeit prädikatenlogischer Konsequenz)

Es gibt kein Verfahren, das für beliebige PL-Formeln $F_1, \ldots F_n, G$ feststellen kann, ob $F_1, \ldots F_n \models G$ oder nicht.

Überlege: Warum und wie ergibt sich der zweite Satz aus dem ersten?

Turingmaschinen [WH]



- Das Band enthält die Eingabe. Alle weiteren Zellen enthalten ein besonderes Blank-Symbol.
- Lese-/Schreib-Kopf ist immer über einer Bandzelle positioniert.
- Am Anfang befindet sich der Lese-/Schreib-Kopf über der am weitesten links stehenden Eingabezelle.

Abhängig vom Zustand und dem gelesenen Bandsymbol:

- TM wechselt den Zustand
- TM schreibt ein Bandsymbol in die gelesene Zelle
- Lese-/Schreib-Kopf geht nach links oder rechts, bzw. bleibt stehen

Formale Beschreibung einer Turingmaschine [WH]

Eine **Turingmaschine** ist ein Siebentupel

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F),$$

- Q endliche Menge von Zuständen,
- Σ endliche Menge der Eingabesymbole,
- Γ endliche Menge der Bandsymbole, $\Sigma \subset \Gamma$,
- $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R, S\}$ Übergangsfunktion,
- $q_0 \in Q$ Startzustand,
- $B \in \Gamma - \Sigma$ Blank-Symbol,
- $F \subseteq Q$ Menge von Endzuständen.

Übergänge [WH]

$$(q, X; p, Y, D) \in \delta$$

M liest im Zustand q auf dem Band Symbol X

- wechselt nun davon abhängig in den Zustand p,
- Überschreibt Symbol X durch Symbol Y,
- bewegt den Lese-/Schreibkopf in die durch D beschriebene Richtung ($D \in \{L, R, S\}$).

Deterministische Turingmaschine (DTM)

Eine Turingmaschine M heißt **deterministisch**, wenn für alle $(q, X) \in Q \times \Sigma \times \Gamma$ höchstens ein Element $(q, X; p, Y, D) \in \delta$ existiert, $D \in \{L, R, S\}$; wir schreiben dann auch $\delta(q, X) = (p, Y, D)$.

Konfiguration [WH] mit einer kleinen Änderung

Am Anfang befindet sich nur das Eingabewort auf dem Band, welches nach links und rechts mit unendlich vielen Blank-Symbolen umgeben ist. Die TM ist im Startzustand und der Lese-/Schreibkopf befindet sich über dem am weitesten links stehenden Eingabesymbol.

Mit jeder Bewegung ändert die TM ihre Konfiguration.

Darstellung einer Konfiguration:

$$X_1 X_2 \ldots X_{i-1} q X_i X_{i+1} \ldots X_n$$

wobei:
– q bezeichnet den aktuellen Zustand der TM,
– der Lese-/Schreibkopf steht über dem i-ten Symbol von links,
– alle Symbole in $X_1 X_2 \ldots X_n$ (auch X_1 und X_n!) können auch 'B' sein

Die kleine Änderung bedeutet:
Die Darstellung einer gegeben Konfiguration ist nicht mehr eindeutig:
$Xq = BXq = XqB = BXqBB = BBBBBBXqBBBBBBBB = \ldots$
Dafür wird unser Unentscheidbarkeitsbeweis einfacher zu verstehen!

Konfiguration [WH]

$\Rightarrow$ ... Übergang zwischen Konfigurationen in einem Schritt[1]
$\Rightarrow^*$ ... Übergang zwischen Konfigurationen in beliebig vielen Schritten

Erfolgt die nächste Bewegung nach

- links, d.h. $\delta(q, X_i) = (p, Y, L)$:

$$X_1 X_2 ... X_{i-1} \mathbf{q} \mathbf{X_i} X_{i+1} ... X_n \Rightarrow X_1 X_2 ... X_{i-2} \mathbf{p} X_{i-1} \mathbf{Y} X_{i+1} ... X_n$$

- rechts, d.h. $\delta(q, X_i) = (p, Y, R)$:

$$X_1 X_2 ... X_{i-1} \mathbf{q} \mathbf{X_i} X_{i+1} ... X_n \Rightarrow X_1 X_2 ... X_{i-1} \mathbf{Y} \mathbf{p} X_{i+1} ... X_n$$

Bleibt der Kopf stehen:

- d.h., $\delta(q, X_i) = (p, Y, S)$:

$$X_1 X_2 ... X_{i-1} \mathbf{q} \mathbf{X_i} X_{i+1} ... X_n \Rightarrow X_1 X_2 ... X_{i-1} \mathbf{p} \mathbf{Y} X_{i+1} ... X_n$$

[1] in [HMU] wird $\Rightarrow$ mit $\vdash$ bezeichnet.

(Deterministische) Turingmaschinen: akzeptierte Sprache [WH]

Akzeptierte Sprache

Die von der (deterministischen) Turingmaschine M **akzeptierte Sprache** $L(M)$ besteht aus genau all jenen Wörtern, bei deren Analyse M einen Endzustand erreicht:

$$L(M) = \{w \in \Sigma^* \mid q_0 w \Rightarrow^* \alpha p \beta, \quad p \in F, \quad \alpha, \beta \in \Gamma^*\}$$

Rekursiv aufzählbare Sprachen

Eine formale Sprache heißt **rekursiv aufzählbar**[2] wenn sie von einer (deterministischen) Turingmaschine akzeptiert wird.

[2] engl: recursively enumerable, auch bezeichnet als: Turing-erkennbar (Turing-recognizable), semi-entscheidbar (semi-decidable)

Entscheidbare (Rekursive) Sprachen [WH]

Wird eine Turingmaschine M auf einem Input w gestartet, so gibt es folgende Möglichkeiten:

- M hält in einem Zustand $q \in F$ und akzeptiert somit die Eingabe
- M hält in einem Zustand $q \notin F$ und verwirft somit die Eingabe
- M gerät in eine Endlosschleife und hält nicht

Rekursive (Entscheidbare) Sprachen

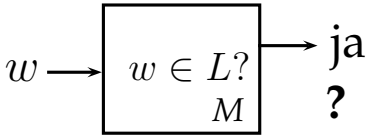
Eine formale Sprache heißt **rekursiv** oder **entscheidbar**[3] wenn sie von einer (deterministischen) Turingmaschine akzeptiert wird, die **immer hält**. (Man sagt dann auch: die TM **entscheidet** die Sprache)

[3] engl: recursive, decidable, auch bezeichnet als: Turing-entscheidbar (Turing-decidable)

Problemlandschaft [WH]

Rekursiv aufzählbare Sprachen L:

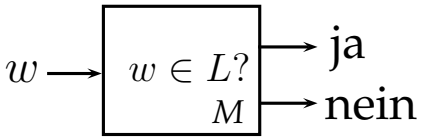
Es gibt eine TM M die akzeptiert und hält wenn $w \in L$, sonst aber u.U. unendlich läuft.

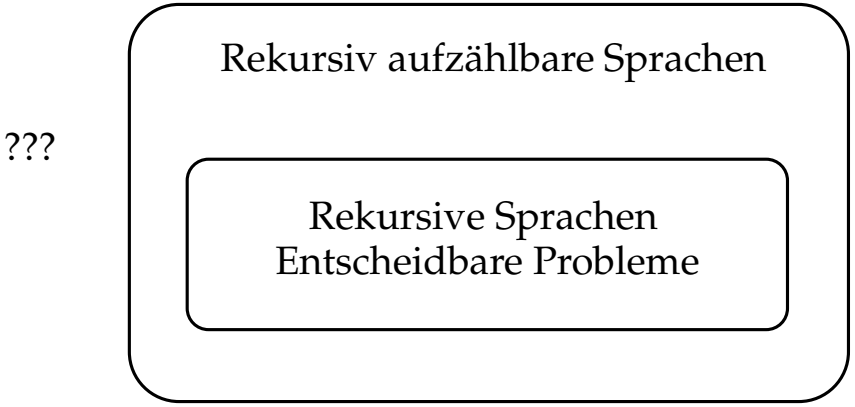


Rekursive Sprachen (**Entscheidbare Probleme**) L:

Es gibt eine TM M mit $L(M) = L$ die immer hält.

(d.h., es gibt einen **Algorithmus**, der bei Eingabe von w "ja" antwortet, wenn $w \in L$ und "nein" antwortet, wenn $w \notin L$)



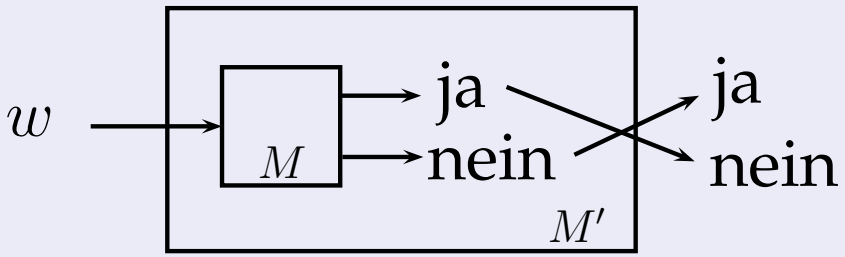


Rekursive (entscheidbare) Sprachen und ihr Komplement [WH]

Das Komplement einer entscheidbaren Sprache ist entscheidbar.

Beweis

Sei L rekursiv (= entscheidbar) und M eine TM mit $L(M) = L$, die immer hält. Wir konstruieren M' aus M so, dass M' stoppt ohne zu akzeptieren, wenn M auf einer Eingabe w in einen Endzustand übergeht. Hält M ohne zu akzeptieren, so geht M' in einen Endzustand über. Da eines dieser beiden Ereignisse eintritt, ist $\overline{L} = L(M')$ (das Komplement von L) eine rekursive Sprache.



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.

Reduktion des Halteproblems auf PL-Konsequenz

Beweisidee: Wir zeigen, wie man zu jeder gegebener TM M und jedem Wort $w \in \{0,1\}^*$ PL-Formeln $F_1, \ldots, F_n$, G konstruieren kann, sodass gilt:

$$M \text{ akzeptiert } w \text{ genau dann, wenn } F_1, \ldots, F_n \models G$$

Nützliche Terminologie:
Die durch M bestimmte Folge von Konfigurationen

$$q_0 b_1 \ldots b_n \Rightarrow \ldots \Rightarrow X_1 X_2 \ldots X_{i-1} q X_i \ldots . X_{n-1} X_n \; [\Rightarrow \ldots]$$

heißt Berechnung von $M = (Q, \{0,1\}, \Gamma, \delta_M, q_0, B, F)$
bei Eingabe $w = b_1 \ldots b_n$ ($b_i \in \{0,1\}$).

Beachte: M akzeptiert w g.d.w. ein Endzustand erreicht wird ($q \in F$).

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. $\delta(q, X_i) = (p, Y, S)$

  $$X_1 X_2 \ldots X_{i-1} \mathbf{q} \mathbf{X_i} X_{i+1} \ldots X_n \;\Rightarrow\; X_1 X_2 \ldots X_{i-1} \mathbf{p} \mathbf{Y} X_{i+1} \ldots X_n$$

  Beachte: $X_i X_{i+1} \ldots X_n$ 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. $\delta(q, X_i) = (p, Y, R)$

  $$X_1 X_2 \ldots X_{i-1} \mathbf{q} \mathbf{X_i} X_{i+1} \ldots X_n \;\Rightarrow\; X_1 X_2 \ldots X_{i-1} \mathbf{Y} \mathbf{p} X_{i+1} \ldots X_n$$

  $\xi_1(\beta) = \forall u \forall v [P_{\mathbf{q}}(u, f_{\mathbf{X_i}}(v)) \supset P_{\mathbf{p}}(f_{\mathbf{Y}}(u), v)]$
  $\xi_2(\beta) = \forall u [P_{\mathbf{q}}(u, b) \supset P_{\mathbf{p}}(f_{\mathbf{Y}}(u), b)]$
- 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: $K_0 = q_0 0011 \Rightarrow_\beta X q_1 011$

$\xi(K_0) = P_{q_0}(b, f_0(f_0(f_1(f_1(b)))))$

$\xi(K_1) = P_{q_1}(f_X(b), f_0(f_1(f_1(b))))$

$\xi_1(\beta) = \forall u \forall v [P_{q_0}(u, f_0(v)) \supset P_{q_1}(f_X(u), v)]$

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:

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

dabei ist $\{p_1, \ldots, p_n\}$ 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 ($\delta = \{\beta_1, \ldots, \beta_n\}$) gilt:
Die Turingmaschine M akzeptiert das Wort w genau dann, wenn
$\xi(K_0(M, w)), \xi_1(\beta_1), \xi_2(\beta_1), \ldots, \xi_1(\beta_n), \xi_2(\beta_n) \models Akzept(M)$.

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).

Ein Programm für das ganze Semester

- Was genau sind "Formeln der Prädikatenlogik erster Stufe"?
- Was ist eine "axiomatische Theorie" in dieser Sprache (PL)?
- Wie kann man die Bedeutung unserer Sprache präzise festlegen?
  Erinnerung: Frege und Russell sind letztlich auch deshalb gescheitert, weil sie keine klare Syntax/Semantik-Trennung kannten!
- Wie drückt man mathematische – und andere – Konzepte in PL aus?
- Was genau heißt "ein Satz folgt (logisch) aus Annahmen/Axiomen"?
- Wie, also mit welchen Regeln, kann bzw. soll man "herleiten"?
- Kann man wirklich alles herleiten, was logisch folgt?
- Kann man Leibniz' bzw. Hilberts Traum verwirklichen?
- Was hat das alles mit beweisbar korrekten Programmen zu tun?

Was bisher geschah & was Sie nun erwartet

1. Logik: Woher? Warum? Wie?
2. Die Sprache der Prädikaten (und Aussagen-)Logik (PL/AL-Syntax)
3. Von Syntax zu Semantik
4. Die Semantik der Prädikaten- (und der Aussagen-)logik
5. Formalisierung natürsprachlicher Information mit PL
6. Ausdrücken formaler Information in PL
7. Das Konzept der logischen Konsequenz
8. Logische Kalküle – der Tableau-Kalkül
9. Wichtige Eigenschaften der PL
10. Korrektheitsbeweise – der Hoare-Kalkül

(Partielle) Korrektheitsaussagen

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 bedeudet: 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 Korrektheitsaussagen ausdrücken, dass ein Programm nicht terminiert:

$$\mathcal{D} \models \top\ \{\alpha\}\ \bot$$

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

Beispiel

Bestimmen Sie jene a, b in Abhängigkeit von $I(x)$ und $I(y)$ für die in $\mathbb{N}$ gilt:

$$\mathrm{x} < \mathrm{y}\ \{\underline{\texttt{while}}\ \mathrm{x} < \mathrm{y}\ \underline{\texttt{do}}\ \underline{\texttt{begin}}\ \mathrm{x} \leftarrow \mathrm{x} + 2;\ \mathrm{y} \leftarrow \mathrm{y} + 1\ \underline{\texttt{end}}\}\ (\mathrm{x} = a \wedge \mathrm{y} = b)$$

Lösung:

Fall 1 $I(\mathrm{x}) \geq I(\mathrm{y})$: Die Precondition ist falsch und daher ist die Korrektheitsaussage wahr für alle $a, b \in \omega$.

Fall 2 $I(\mathrm{x}) < I(\mathrm{y})$: Precondition und Schleifenbedingung treffen zu.
Im Folgenden bezeichnet I_k die Umgebung nach k-maliger Ausführung des Schleifenkörpers. D.h., $I_0 = I$ und $I_{k+1} = \mathcal{M}_{AL}(I_k, \underline{\texttt{begin}} \ldots \underline{\texttt{end}})$.
Wir erhalten die Rekursionen

$$\begin{aligned} I_0(\mathrm{x}) &= I(\mathrm{x}), \quad I_{k+1}(\mathrm{x}) = I_k(\mathrm{x}) + 2 \\ I_0(\mathrm{y}) &= I(\mathrm{y}), \quad I_{k+1}(\mathrm{y}) = I_k(\mathrm{y}) + 1 \end{aligned}$$

Auflösen der Rekursionen liefert

$$\begin{aligned} I_k(\mathrm{x}) &= I(\mathrm{x}) + 2k \\ I_k(\mathrm{y}) &= I(\mathrm{y}) + k \end{aligned}$$

Beispiel (Fortsetzung)

Die Zahl der Schleifendurchläufe ist das kleinste k, sodass $I_k(\mathrm{x}) \geq I_k(\mathrm{y})$ also $I(\mathrm{x}) + 2k \geq I(\mathrm{y}) + k$ gilt.
Daher ergibt sich $k = I(\mathrm{y}) - I(\mathrm{x})$.

Wir erhalten daher $\mathcal{M}_{AL}(I, \underline{\texttt{while}} \ldots) = I_k$ mit
$I_k(\mathrm{x}) = I(\mathrm{x}) + 2(I(\mathrm{y}) - I(\mathrm{x})) = 2I(\mathrm{y}) - I(\mathrm{x})$ und
$I_k(\mathrm{y}) = I(\mathrm{y}) + (I(\mathrm{y}) - I(\mathrm{x})) = 2I(\mathrm{y}) - I(\mathrm{x})$.

Die Korrektheitsaussage ist also in diesem Fall wahr für
$a = b = 2I(\mathrm{y}) - I(\mathrm{x})$.

Wer oder was ist “Hoare”?

Sir Charles Antony Richard Hoare besser bekannt als **Tony Hoare** (geb. 1934). Britischer Informatiker, Turing-Award-Winner. Bekannt vor allem für die Erfindung von *Quicksort*, die Entwicklung des *Hoare-Kalküls* zur Programmverfikation und der Spezifikationssprache *Communicating Sequential Processes* (CSP).

Die Menge $\mathcal{H}$ der wahren (partiellen) Korrektheitsaussagen

Notation:

$P \supset Q\left[\begin{smallmatrix} t \\ v \end{smallmatrix}\right]$ ... in Q wurden alle Vorkommen von v durch t ersetzt und außerdem kommt v weder in P noch in Q gebunden vor

$$\frac{\text{Wenn} \quad \mathcal{D} \models P \supset Q\left[\begin{smallmatrix} t \\ v \end{smallmatrix}\right]}{\text{dann} \quad P\ \{v \underline{\leftarrow} t\}\ Q \in \mathcal{H}}$$

$$\frac{\text{Wenn} \quad P\ \{\alpha\}\ Q \in \mathcal{H} \quad \text{und} \quad Q\ \{\beta\}\ R \in \mathcal{H}}{\text{dann} \quad P\ \{\underline{\texttt{begin}}\ \alpha\underline{;}\ \beta\ \underline{\texttt{end}}\}\ R \in \mathcal{H}}$$

$$\frac{\text{Wenn} \quad (P \wedge B)\ \{\alpha\}\ Q \in \mathcal{H} \quad \text{und} \quad (P \wedge \neg B)\ \{\beta\}\ Q \in \mathcal{H}}{\text{dann} \quad P\ \{\underline{\texttt{if}}\ B\ \underline{\texttt{then}}\ \alpha\ \underline{\texttt{else}}\ \beta\}\ Q \in \mathcal{H}}$$

$$\frac{\begin{array}{c}\text{Wenn} \quad \mathcal{D} \models P \supset INV \quad \text{und} \quad \mathcal{D} \models (INV \wedge \neg B) \supset Q \\ \text{und} \quad (INV \wedge B)\ \{\alpha\}\ INV \in \mathcal{H}\end{array}}{\text{dann} \quad P\ \{\underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \alpha\}\ Q \in \mathcal{H}}$$

Der Hoare-Kalkül

$$\text{(H1)}\ \frac{P \supset Q\left[\begin{smallmatrix} t \\ v \end{smallmatrix}\right]}{P\ \{v \underline{\leftarrow} t\}\ Q}$$

Beachte:
v kommt in P, Q nur frei vor

$$\text{(H2)}\ \frac{P\ \{\alpha\}\ Q \qquad Q\ \{\beta\}\ R}{P\ \{\underline{\texttt{begin}}\ \alpha\underline{;}\ \beta\ \underline{\texttt{end}}\}\ R}$$

$$\text{(H3)}\ \frac{(P \wedge B)\ \{\alpha\}\ Q \quad (P \wedge \neg B)\ \{\beta\}\ Q}{P\ \{\underline{\texttt{if}}\ B\ \underline{\texttt{then}}\ \alpha\ \underline{\texttt{else}}\ \beta\}\ Q}$$

$$\text{(H4)}\ \frac{P \supset INV \quad (INV \wedge B)\ \{\alpha\}\ INV \quad (INV \wedge \neg B) \supset Q}{P\ \{\underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \alpha\}\ Q}$$

Korrektheit der Assignment-Regel (H1)

Satz

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

(1) $\mathcal{M}_{\mathcal{T}}(I', s) = \mathcal{M}_{\mathcal{T}}(I, s[^t_v])$ *für alle* $s \in \mathcal{T}(\mathcal{D})$.

(2) $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I', A) = \mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, A[^t_v])$ *für alle atomaren Formeln* $A \in \mathcal{PF}_{\Sigma(\mathcal{D})}$.

(3) $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I', F) = \mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, F[^t_v])$ *für alle Formeln* $F \in \mathcal{PF}_{\Sigma(\mathcal{D})}$.

(4) *Aus* $\mathcal{D} \models P \supset Q[^t_v]$ *folgt* $P \ \{v \leftarrow t\} \ Q \in \mathcal{H}$ *für alle* $P, Q \in \mathcal{PF}_{\Sigma(\mathcal{D})}$.

Korrektheit der Hintereinanderausführungsregel (H2)

Satz

Für alle $P, Q, R \in \mathcal{PF}_{\Sigma(\mathcal{D})}$ *und* $\alpha, \beta \in AL(\mathcal{D})$*:*
Wenn $\mathcal{D} \models P\ \{\alpha\}\ Q$ *und* $\mathcal{D} \models Q\ \{\beta\}\ R$
dann $\mathcal{D} \models P\ \{\underline{\texttt{begin}}\ \alpha\underline{;}\ \beta\ \underline{\texttt{end}}\}\ R$.

Beweis:

Zu zeigen ist Folgendes:
Aus $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, P) = \mathbf{t}$ und $I_2 = \mathcal{M}_{AL}(I, \underline{\texttt{begin}}\ \alpha\underline{;}\ \beta\ \underline{\texttt{end}})$ folgt $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I_2, R) = \mathbf{t}$.

Aus der Definition von $\mathcal{M}_{AL}$ folgt:
$I_2 = \mathcal{M}_{AL}(\mathcal{M}_{AL}(I, \alpha), \beta) = \mathcal{M}_{AL}(I_1, \beta)$ wobei $I_1 = \mathcal{M}_{AL}(I, \alpha)$.
Aus $\mathcal{D} \models P\ \{\alpha\}\ Q$ und $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, P) = \mathbf{t}$ ergibt sich $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I_1, Q) = \mathbf{t}$.

Aus $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I_1, Q) = \mathbf{t}$ und $\mathcal{D} \models Q\ \{\beta\}\ R$ folgt $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I_2, R) = \mathbf{t}$, QED.

Korrektheit der Konditionalregel (H3) und der While-Regel (H4)

Satz

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

Satz

Wenn $\mathcal{D} \models P \supset INV$ *und* $\mathcal{D} \models (INV \wedge \neg B) \supset Q$
und $\mathcal{D} \models (INV \wedge B)\ \{\alpha\}\ INV$
dann $P\ \{\underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \alpha\}\ Q$.

Lemma

Aus $\mathcal{D} \models (INV \wedge B)\ \{\alpha\}\ INV$ *folgt*
$\mathcal{D} \models INV\ \{\underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \alpha\}\ (INV \wedge \neg B)$.

Nützliche Hilfsregeln

$$\text{(T1)}\ \frac{(P \wedge v = t)\ \{\beta\}\ R}{P\ \{\underline{\texttt{begin}}\ v \underline{\leftarrow} t\underline{;}\ \beta\ \underline{\texttt{end}}\}\ R}$$

v kommt weder in P noch in t vor

$$\text{(T2)}\ \frac{P\ \{\alpha\}\ R\left[{}^{t}_{v}\right]}{P\ \{\underline{\texttt{begin}}\ \alpha\underline{;}\ v \underline{\leftarrow} t\ \underline{\texttt{end}}\}\ R}$$

$$\text{(T3)}\ \frac{P\ \{\alpha\}\ INV \quad (INV \wedge B)\ \{\beta\}\ INV \quad (INV \wedge \neg B) \supset R}{P\ \{\underline{\texttt{begin}}\ \alpha\underline{;}\ \underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \beta\ \underline{\texttt{end}}\}\ R}$$

Erläuterung der Hilfsregel (T1)

(T1) ist für Programme der Form $\underline{\texttt{begin}}\ v \underline{\leftarrow} t\underline{;}\ \beta\ \underline{\texttt{end}}$.

Wenn v weder in P noch in t vorkommt, wählen wir $P \wedge v = t$ als Zwischenbedingung:

$$\text{(H2)}\ \frac{\text{(H1)}\ \dfrac{P \supset (P \wedge v = t)\left[{t \atop v}\right]}{P\ \{v \underline{\leftarrow} t\}\ (P \wedge v = t)} \qquad (P \wedge v = t)\ \{\beta\}\ R}{P\ \{\underline{\texttt{begin}}\ v \underline{\leftarrow} t\underline{;}\ \beta\ \underline{\texttt{end}}\}\ R}$$

Weil v nicht in P und t vorkommt, gilt
$(P \wedge v = t)\left[{t \atop v}\right] = (P \wedge (t{=}t)) = P$.
Daher $\mathcal{D} \models \left(P \supset (P \wedge v = t)\left[{t \atop v}\right]\right)$.

Es bleibt also

$$\text{(T1)}\ \frac{(P \wedge v = t)\ \{\beta\}\ R}{P\ \{\underline{\texttt{begin}}\ v \underline{\leftarrow} t\underline{;}\ \beta\ \underline{\texttt{end}}\}\ R} \quad \begin{array}{l} v \text{ kommt weder} \\ \text{in } P \text{ noch in } t \text{ vor} \end{array}$$

Erläuterung der Hilfsregel (T2)

(T2) ist für Programme der Form $\underline{\texttt{begin}}\ \alpha\underline{;}\ v \underline{\leftarrow} t\ \underline{\texttt{end}}$.

$$\text{(H2)}\ \frac{P\ \{\alpha\}\ R\left[\begin{smallmatrix} t \\ v \end{smallmatrix}\right] \qquad \text{(H1)}\ \dfrac{R\left[\begin{smallmatrix} t \\ v \end{smallmatrix}\right] \supset R\left[\begin{smallmatrix} t \\ v \end{smallmatrix}\right]}{R\left[\begin{smallmatrix} t \\ v \end{smallmatrix}\right]\ \{v \underline{\leftarrow} t\}\ R}}{P\ \{\underline{\texttt{begin}}\ \alpha\underline{;}\ v \underline{\leftarrow} t\ \underline{\texttt{end}}\}\ R}$$

Es gilt $\mathcal{D} \models \left(R\left[\begin{smallmatrix} t \\ v \end{smallmatrix}\right] \supset R\left[\begin{smallmatrix} t \\ v \end{smallmatrix}\right]\right)$.

Es bleibt also

$$\text{(T2)}\ \frac{P\ \{\alpha\}\ R\left[\begin{smallmatrix} t \\ v \end{smallmatrix}\right]}{P\ \{\underline{\texttt{begin}}\ \alpha\underline{;}\ v \underline{\leftarrow} t\ \underline{\texttt{end}}\}\ R}$$

Erläuterung der Hilfsregel (T3)

(T3) ist für Programme der Form $\underline{\texttt{begin}}\ \alpha\underline{;}\ \underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \beta\ \underline{\texttt{end}}$.

Wir wählen die Schleifeninvariante *INV* als Zwischenbedingung:

$$\text{(H2)}\ \frac{\overset{(1)}{P\ \{\alpha\}\ INV} \quad \overset{(2)}{INV\ \{\underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \beta\}\ R}}{P\ \{\underline{\texttt{begin}}\ \alpha\underline{;}\ \underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \beta\ \underline{\texttt{end}}\}\ R}$$

Setzt man nun (H4) auf die Korrektheitsaussage (2) an ergibt sich:

$$\text{(H4)}\ \frac{\overset{(3)}{INV \supset INV} \quad \overset{(4)}{(INV \wedge B)\ \{\alpha\}\ INV} \quad \overset{(5)}{(INV \wedge \neg B) \supset R}}{INV\ \{\underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \alpha\}\ R}$$

Wegen $\mathcal{D} \models (INV \supset INV)$ bleiben die Prämissen (1), (4) und (5).
Zusammenfassend ergibt sich:

$$\text{(T3)}\ \frac{P\ \{\alpha\}\ INV \quad (INV \wedge B)\ \{\beta\}\ INV \quad (INV \wedge \neg B) \supset R}{P\ \{\underline{\texttt{begin}}\ \alpha\underline{;}\ \underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \beta\ \underline{\texttt{end}}\}\ R}$$

Beispiel

Wir zeigen die Gültigkeit der partiellen Korrektheitsaussage

$$\mathrm{y} > 1\ \{\underline{\texttt{begin}}\ \mathrm{x} \underline{\leftarrow} 3;\ \underline{\texttt{if}}\ \mathrm{x}+\mathrm{y} < 19\ \underline{\texttt{then}}\ \mathrm{x} \underline{\leftarrow} \mathrm{x}+19\ \underline{\texttt{else}}\ \mathrm{x} \underline{\leftarrow} \mathrm{y}+\mathrm{y}\ \underline{\texttt{end}}\}\ \mathrm{x} > \mathrm{y}$$

über dem Datentyp der natürlichen Zahlen.

Mit den Abkürzungen P: $\mathrm{y} > 1$, Q: $\mathrm{x} > \mathrm{y}$ und B: $\mathrm{x}+\mathrm{y} < 19$ ergibt sich folgende Ableitung:

$$\text{(T1)}\ \dfrac{\text{(H3)}\ \dfrac{\text{(H1)}\ \dfrac{(1)}{((P \wedge \mathrm{x}=3) \wedge B) \supset Q\left[{}^{\mathrm{x}+19}_{\mathrm{x}}\right]} \qquad \text{(H1)}\ \dfrac{(2)}{((P \wedge \mathrm{x}=3) \wedge \neg B) \supset Q\left[{}^{\mathrm{y}+\mathrm{y}}_{\mathrm{x}}\right]}}{(P \wedge \mathrm{x}=3)\ \{\underline{\texttt{if}}\ B\ \underline{\texttt{then}}\ \mathrm{x} \underline{\leftarrow} \mathrm{x}+19\ \underline{\texttt{else}}\ \mathrm{x} \underline{\leftarrow} \mathrm{y}+\mathrm{y}\}\ Q}}{P\ \{\underline{\texttt{begin}}\ \mathrm{x} \underline{\leftarrow} 3;\ \underline{\texttt{if}}\ B\ \underline{\texttt{then}}\ \mathrm{x} \underline{\leftarrow} \mathrm{x}+19\ \underline{\texttt{else}}\ \mathrm{x} \underline{\leftarrow} \mathrm{y}+\mathrm{y}\ \underline{\texttt{end}}\}\ Q}$$

Beispiel (Fortsetzung)

Es bleibt die Gültigkeit der Formeln (1) und (2) zu zeigen.

(1) $(((y > 1 \wedge x = 3) \wedge x + y < 19) \supset y < x + 19)$
$y < x + y$, da $x = 3$ (Prämisse)
$x + y < 19$ (Prämisse)
$19 < x + 19$, da $x = 3$ (Prämisse)
insgesamt gilt also $y < x + 19$;
(1) ist daher gültig in $\mathbb{N}$.

(2) $(((y > 1 \wedge x = 3) \wedge x + y \geq 19) \supset y < y + y)$
Wegen der Prämisse $y > 1$ gilt auch $y < y + y$;
(2) ist daher gültig in $\mathbb{N}$.

Finden der Invariante (spezielle "Kochrezepte")

Satz

P $\{\underline{\texttt{while}}\ P\ \underline{\texttt{do}}\ \cdots\}$ Q ist genau dann ableitbar, wenn die Korrektheitsaussage mit der Invariante $(P \vee Q)$ ableitbar ist.

Nützliche Rezepte für Spezialfälle im Datentyp $\mathbb{Z}$ (bzw. $\mathbb{N}$) :

Rezept 1: Invariante $aw + bv = c$ bei

$$\underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \underline{\texttt{begin}}\ v \underline{\leftarrow} v + a\underline{;}\ w \underline{\leftarrow} w - b\ \underline{\texttt{end}}$$
$$\underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \underline{\texttt{begin}}\ w \underline{\leftarrow} w - b\underline{;}\ v \underline{\leftarrow} v + a\ \underline{\texttt{end}}$$

denn es gilt, z.B., $a(w - b) + b(v + a) = aw - ab + bv + ab = aw + bv$
Wichtig: a, b, c sind innerhalb der Schleife konstant.

Rezept 2: Invariante $aw = bv + c$ bzw. $aw + c = bv$ bei

$$\underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \underline{\texttt{begin}}\ v \underline{\leftarrow} v + a\underline{;}\ w \underline{\leftarrow} w + b\ \underline{\texttt{end}}$$
$$\underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \underline{\texttt{begin}}\ v \underline{\leftarrow} v - a\underline{;}\ w \underline{\leftarrow} w - b\ \underline{\texttt{end}}$$

Beispiel

Wir zeigen die Gültigkeit der partiellen Korrektheitsaussage

$$(\,\mathrm{x}=3 \wedge \mathrm{y}=1\,)\ \{\underline{\texttt{while}}\ \mathrm{x}>0\ \underline{\texttt{do}}\ \underline{\texttt{begin}}\ \mathrm{y} \underline{\leftarrow} \mathrm{y}+3;\ \mathrm{x} \underline{\leftarrow} \mathrm{x}-1\ \underline{\texttt{end}}\}\ \mathrm{y}=10$$

über dem Datentyp $\mathbb{Z}$ der ganzen Zahlen.

Mit den Abkürzungen P: $(\,\mathrm{x}=3 \wedge \mathrm{y}=1\,)$, B: $\mathrm{x}>0$ und Q: $\mathrm{y}=10$ ergibt sich folgende Ableitung:

$$\text{(H4)}\ \dfrac{\overset{(1)}{P \supset I} \qquad \text{(T2)}\ \dfrac{\text{(H1)}\ \dfrac{\overset{(2)}{(I \wedge B) \supset I\left[\begin{smallmatrix}\mathrm{x}-1\\ \mathrm{x}\end{smallmatrix}\right]\left[\begin{smallmatrix}\mathrm{y}+3\\ \mathrm{y}\end{smallmatrix}\right]}}{(I \wedge B)\ \{\mathrm{y} \underline{\leftarrow} \mathrm{y}+3\}\ I\left[\begin{smallmatrix}\mathrm{x}-1\\ \mathrm{x}\end{smallmatrix}\right]}}{(I \wedge B)\ \{\underline{\texttt{begin}}\ \mathrm{y} \underline{\leftarrow} \mathrm{y}+3;\ \mathrm{x} \underline{\leftarrow} \mathrm{x}-1\ \underline{\texttt{end}}\}\ I} \qquad \overset{(3)}{(I \wedge \neg B) \supset Q}}{P\ \{\underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \underline{\texttt{begin}}\ \mathrm{y} \underline{\leftarrow} \mathrm{y}+3;\ \mathrm{x} \underline{\leftarrow} \mathrm{x}-1\ \underline{\texttt{end}}\}\ Q}$$

Wir können Rezept 1 anwenden.

Beispiel (Fortsetzung)

Rezept 1 ergibt die Form $3x + y = c$ für die Invariante I. Wegen der Precondition $(x = 3 \wedge y = 1)$ erhält man $I = 3x + y = 10$.

Das Korrekheitsproblem ist damit auf folgende einfache Gültigkeitsnachweise in $\mathbb{Z}$ reduziert

(1) $(P \supset I)$
$((\mathrm{x} = 3 \wedge \mathrm{y} = 1) \supset 3\mathrm{x} + \mathrm{y} = 10)$ ist gültig in $\mathbb{Z}$, da in den ganzen Zahlen $3 \cdot 3 + 1 = 10$ gilt.

(2) $((I \wedge B) \supset I\left[\begin{smallmatrix}\mathrm{x} - 1\\ \mathrm{x}\end{smallmatrix}\right] I\left[\begin{smallmatrix}\mathrm{y} + 3\\ \mathrm{y}\end{smallmatrix}\right])$
$((3\mathrm{x} + \mathrm{y} = 10 \wedge \mathrm{x} > 0) \supset 3(\mathrm{x} - 1) + (\mathrm{y} + 3) = 10)$
Die rechte Seite vereinfacht sich zu $3\mathrm{x} + \mathrm{y} = 10$.
Wir erhalten die Tautologie $((I \wedge B) \supset I)$;
Formel (2) ist somit gültig in $\mathbb{Z}$.

(3) $((I \wedge \neg B) \supset Q)$
$((3\mathrm{x} + \mathrm{y} = 10 \wedge \mathrm{x} \leq 0) \supset \mathrm{y} \geq 10)$
$\mathrm{x} \leq 0$ impliziert $3\mathrm{x} \leq 0$. Damit folgt $\mathrm{y} \geq 10$ aus $3\mathrm{x} + \mathrm{y} = 10$.

Beispiel – Divisionsalgorithmus

DIV[dividend, divisor, quot, rem] =

```
begin
  begin
    rem ← dividend;
    quot ← 0
  end;
  while rem≥divisor do begin
    rem ← rem ∸ divisor;
    quot ← quot + 1
  end
end
```

Beispiel – Verfikation des Divisionsalgorithmus

Wir benützen folgende Abkürzungen:

$$
\begin{array}{lcl}
P & : & \mathrm{divisor} > 0 \\
B & : & \mathrm{rem} \geq \mathrm{divisor} \\
Q & : & (\,\mathrm{dividend} = \mathrm{rem} + \mathrm{quot} * \mathrm{divisor} \wedge \mathrm{rem} < \mathrm{divisor}\,)
\end{array}
$$

Wir zerlegen die Korrektheitsaussage zunächst gemäß Zusatzregel (T3):

$$
(\mathrm{T3})\ \frac{\overset{(1)}{P\ \{\underline{\texttt{begin}}\ \ldots\ \underline{\texttt{end}}\}\ I} \quad \overset{(2)}{(I \wedge B)\ \{\underline{\texttt{begin}}\ \ldots\ \underline{\texttt{end}}\}\ I} \quad \overset{(3)}{(I \wedge \neg B) \supset Q}}{P\ \{\underline{\texttt{begin}}\ \underline{\texttt{begin}}\ \ldots\ \underline{\texttt{end}};\ \underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \underline{\texttt{begin}}\ \ldots\ \underline{\texttt{end}}\ \underline{\texttt{end}}\}\ Q}
$$

Beispiel – Verfikation des Divisionsalgorithmus (Forts.)

Für die Korrektheitsaussagen (1) und (2) erhalten wir die folgenden Ableitungen:

(1) $\longrightarrow$ (4):

$$\text{(T2)}\ \frac{\text{(H1)}\ \dfrac{P \supset I\left[{}^{0}_{\text{quot}}\right]\left[{}^{\text{dividend}}_{\text{rem}}\right]}{P\ \{\text{rem} \underline{\leftarrow} \text{dividend}\}\ I\left[{}^{0}_{\text{quot}}\right]}}{P\ \{\underline{\texttt{begin}}\ \text{rem} \underline{\leftarrow} \text{dividend}\underline{;}\ \text{quot} \underline{\leftarrow} 0\ \underline{\texttt{end}}\}\ I}$$

(2) $\longrightarrow$ (5):

$$\text{(T2)}\ \frac{\text{(H1)}\ \dfrac{(\,I \wedge B\,) \supset I\left[{}^{\text{quot}+1}_{\text{quot}}\right]\left[{}^{\text{rem} \dot{-} \text{divisor}}_{\text{rem}}\right]}{(\,I \wedge B\,)\ \{\text{rem} \underline{\leftarrow} \text{rem} \dot{-} \text{divisor}\}\ I\left[{}^{\text{quot}+1}_{\text{quot}}\right]}}{(\,I \wedge B\,)\ \{\underline{\texttt{begin}}\ \text{rem} \underline{\leftarrow} \text{rem} \dot{-} \text{divisor}\underline{;}\ \text{quot} \underline{\leftarrow} \text{quot} + 1\ \underline{\texttt{end}}\}\ I}$$

Als Invariante eignet sich I: $\text{dividend} = \text{rem} + \text{quot} * \text{divisor}$.

Beispiel – Verfikation des Divisionsalgorithmus (Forts.)

Mit der Invariante I: $\mathrm{dividend} = \mathrm{rem} + \mathrm{quot} * \mathrm{divisor}$ erhält man:

(3) $(\,(\,I \wedge \neg\, B\,) \supset Q\,)$ ist eine Tautologie, da Q: $(\,I \wedge \neg\, B\,)$ ist.

(4) $(\,P \supset \mathrm{dividend} = \mathrm{dividend} + 0 * \mathrm{divisor}\,)$
Die rechte Seite der Implikation ist in $\mathbb{N}$ immer wahr,
daher ist die gesamte Formel gültig in $\mathbb{N}$.
$\Rightarrow$ P kann beliebig gewählt werden.

(5) $(\,(\,I \wedge B\,) \supset I\left[{\mathrm{quot}+1 \atop \mathrm{quot}}\right]\left[{\mathrm{rem} \dot{-} \mathrm{divisor} \atop \mathrm{rem}}\right]\,)$
$(\,I \wedge \mathrm{rem} \geq \mathrm{divisor}\,)$
$\supset \mathrm{dividend} = (\mathrm{rem} \dot{-} \mathrm{divisor}) + (\mathrm{quot} + 1) * \mathrm{divisor}$

Ist die linke Seite wahr, muss $\mathrm{rem} \geq \mathrm{divisor}$ gelten.
Dann ist aber die rechte Seite in $\mathbb{N}$ äquivalent zu
$\mathrm{dividend} = \mathrm{rem} + \mathrm{quot} * \mathrm{divisor}$.
(5) ist daher in $\mathbb{N}$ gleichwertig zur Tautologie $(\,(\,I \wedge B\,) \supset I\,)$
und damit gültig in $\mathbb{N}$.

Beispiel

Wir zeigen die Korrektheit der partielle Korrektheitsaussage

$$\mathrm{ist0?(x)}\ \{\underline{\texttt{while}}\ \mathrm{ist0?(x)}\ \underline{\texttt{do}}\ \underline{\texttt{begin}}\ \mathrm{y} \underline{\leftarrow} \mathrm{push0(y)}\underline{;}\ \mathrm{x} \underline{\leftarrow} \mathrm{pop(x)}\ \underline{\texttt{end}}\}\ \mathrm{ist0?(y)}$$

über dem Datentyp $\mathbb{S}$ der binären Stacks.

Abkürzungen $P = B$: $\mathrm{ist0?(x)}$ und Q: $\mathrm{ist0?(y)}$:

$$\text{(H4)}\ \frac{\overset{(1)}{P \supset I} \qquad \text{(T2)}\ \dfrac{\text{(H1)}\ \dfrac{\overset{(2)}{(I \wedge B) \supset I\left[\mathrm{pop}_{\mathrm{x}}(\mathrm{x})\right]\left[\mathrm{push0}_{\mathrm{y}}(\mathrm{y})\right]}}{(I \wedge B)\ \{\mathrm{y} \underline{\leftarrow} \mathrm{push0(y)}\}\ I\left[\mathrm{pop}_{\mathrm{x}}(\mathrm{x})\right]}}{(I \wedge B)\ \{\underline{\texttt{begin}}\ \mathrm{y} \underline{\leftarrow} \mathrm{push0(y)}\underline{;}\ \mathrm{x} \underline{\leftarrow} \mathrm{pop(x)}\ \underline{\texttt{end}}\}\ I} \qquad \overset{(3)}{(I \wedge \neg B) \supset Q}}{P\ \{\underline{\texttt{while}}\ B\ \underline{\texttt{do}}\ \underline{\texttt{begin}}\ \mathrm{y} \underline{\leftarrow} \mathrm{push0(y)}\underline{;}\ \mathrm{x} \underline{\leftarrow} \mathrm{pop(x)}\ \underline{\texttt{end}}\}\ Q}$$

Beispiel – Fortsetzung

Unter Verwendung der Invariante I: $(P \vee Q)$ erhält man:

(1) $(P \supset (P \vee Q))$ ist eine Tautologie.

(2) $((I \wedge B) \supset I\big[^{\mathrm{pop}(\mathrm{x})}_{\mathrm{x}}\big]\big[^{\mathrm{push0}(\mathrm{y})}_{\mathrm{y}}\big])$
$((I \wedge B) \supset (\mathrm{ist0?}(\mathrm{pop}(\mathrm{x})) \vee \mathrm{ist0?}(\mathrm{push0}(\mathrm{y}))))$
Da in $\mathbb{S}$ für alle $I \in ENV$ gilt, dass $\mathrm{ist0?}(\mathrm{push0}(\mathrm{y}))$ den Wert **t** liefert, ist die rechte Seite der Implikation und damit die gesamte Implikation wahr.
(2) ist daher gültig in $\mathbb{S}$.

(3) $(((P \vee Q) \wedge \neg B) \supset Q)$ ist äquivalent zu (Distributivgesetz!)
$(((P \wedge \neg B) \vee (Q \wedge \neg B)) \supset Q)$
Da $B = P$ ist, evaluiert $(P \wedge \neg B)$ immer zu **f**, die Formel ist gleichwertig zur Tautologie $((Q \wedge \neg B) \supset Q)$.
Daher ist auch (3) eine Tautologie.

Beispiel

Aufgabe:
Zeige, dass die partielle Korrektheitsaussage

$$\mathrm{y}=1\ \{\underline{\texttt{while}}\ \mathrm{x}>0\ \underline{\texttt{do}}\ \underline{\texttt{begin}}\ \mathrm{x} \leftarrow \mathrm{x} \dot{-} 1;\ \mathrm{y} \leftarrow \mathrm{y}+\mathrm{y}\ \underline{\texttt{end}}\}\ \mathrm{y}=1$$

nicht im Hoare-Kalkül über dem Datentyp $\mathbb{N}$ ableitbar ist.

Lösung:
Wegen der Korrektheit des Hoare-Kalküls genügt es nachzuweisen, dass die Korrektheitsaussage falsch ist!

Laut Definition gilt ist $P\ \{\alpha\}\ Q$) genau dann falsch, wenn ein I mit $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, P) = \mathbf{t}$ existiert, für das $\mathcal{M}_{AL}(I, \alpha)$ definiert ist (also α in I terminiert), aber $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(\mathcal{M}_{AL}(I, \alpha), Q) = \mathbf{f}$.
Wir suchen also ein Environment I, sodass:

- $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, \mathrm{y}=1) =_{\mathrm{PL}} \mathbf{t}$,
- $I' = \mathcal{M}_{AL}(I, \underline{\texttt{while}}\ \mathrm{x}>0\ \underline{\texttt{do}}\ \cdots)$ definiert ist und
- $\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I', \mathrm{y}=1) =_{\mathrm{PL}} \mathbf{f}$.

Beispiel (Fortsetzung)

Wir müssen $I(\mathrm{y}) = 1$ und können $I(\mathrm{x}) = 1$ wählen:

$$
\begin{array}{ll}
\mathcal{M}_{AL}(I, \underline{\texttt{while}}\ \mathrm{x} > 0\ \underline{\texttt{do}}\ \cdots) & \\
\quad [\ \mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I, \mathrm{x} > 0) =_{\mathrm{PL}} \mathbf{t}\ ] & \\
= \quad \mathcal{M}_{AL}(I', \underline{\texttt{while}}\ \mathrm{x} > 0\ \underline{\texttt{do}}\ \cdots) & \\
\quad [\ I'(\mathrm{x}) = 0,\ I'(\mathrm{y}) = 2,\ I'(v) = I(v) \text{ für } v \notin \{\mathrm{x}, \mathrm{y}\}\ ] & \\
\quad [\ \mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I', \mathrm{x} > 0) =_{\mathrm{PL}} \mathbf{f}\ ] & \\
= \quad I' &
\end{array}
$$

Die Auswertung der Postcondition im Environment I' liefert

$$\mathcal{M}^{\mathcal{D}}_{\mathcal{PF}}(I', \mathrm{y} = 1) \quad =_{\mathrm{PL}} \quad (2 = 1) \quad =_{\mathrm{PL}} \quad \mathbf{f}.$$

Daher liegt eine falsche Korrektheitsaussage vor.