Von der Aussagenlogik zur Prädikatenlogik

In der Aussagenlogik behandeln wir globale **Aussagen über ganze Systeme**. AL-Aussagen sind je nach betrachtetem System (konkretes System bzw. abstraktes System, d.h. Belegung der Aussagevariablen) wahr oder falsch. Interne Details werden nicht behandelt.

Was ist aber mit (logischen?) Argumentationen wie ...

- „*Wer jemanden* anmeckert, fliegt raus!“ –
  „Aber *Anna* hat *mich* doch gerade angemeckert und ist nicht rausgeflogen!“
- „*Gibt es* Adverbien *die* mit S anfangen?“ –
  „Sicher!“

In der Prädikatenlogik schauen wir in die Systeme hinein und befassen uns mit

- **Objekten** innerhalb eines Systems sowie ihren möglichen
- **Eigenschaften** und
- **Beziehungen** zueinander.

Aussagen sprechen vom Zutreffen von Eigenschaften oder Beziehungen auf einzelne oder alle Objekte innerhalb des Systems. Aussagen (außer den AL-Konstanten, -Tautologien und -Widersprüchen) sind je nach betrachtetem System *und* den bezeichneten Objekten und Eigenschaften oder Beziehungen wahr oder falsch.

Die **PL-Syntax** umfasst die von AL sowie zusätzliche Ausdrucksmittel zur Benennung von ...

- **bestimmten oder beliebigen Objekten:**
  Eigennamen (Konstanten, Variablen),
  Terme (auch mit Funktionen)
- **Eigenschaften** von Objekten: **Prädikate**

oder

Beziehungen zwischen mehreren Objekten: **Relationen**

Einfaches Syntaxbeispiel:

- sprachlich: Zu jedem Objekt existiert ein echt größeres.
- symbolisch: $\forall x \exists y G(y,x)$

Semantikbeispiele: (**Interpretation** der Objekt- und der Eigenschaften-Namen in einem konkreten System)

Die obige Aussage ist

- **wahr**, wenn Objekte := Intervalle $[r,s]$ in $\mathbb{R}$,
  $[a,b]$ größer als $[c,d]$ $:\Leftrightarrow$ $b-a > d-c$;
- **falsch**, wenn
  Objekte := Münzen in meiner Geldbörse,
  Größe $\cong$ Durchmesser .

Die Syntax der Prädikatenlogik erster Stufe (kurz: PL1)

Wir definieren zuerst das **Alphabet**, dann die **Terme** (für Objekte) und schließlich **PL-Formeln**

Das **Alphabet** der Prädikatenlogik erster Stufe PL1 umfasst

- eine abzählbare Menge *VS* von **(Objekt-) Variablensymbolen**, hier $x_1, x_2, ...$
  – informell *u*, *v*, *w*, *x*, *y*, *z*, *Mitarbeiter* usw.
- eine abzählbare Menge *KS* von **(Objekt-) Konstantensymbolen**, hier $a_1, a_2, ...$
  – informell *a*, *b*, *c*, ... , *Besitzer* usw.
- eine abzählbare Menge *FS* von **Funktionssymbolen**, hier $f_1, f_2, ...$
  – informell *f, g, h, ... , Chef,* usw.
  – incl. Abbildung (F-)**Stelligkeit** : $FS \to \mathbb{N}_0$
- eine abzählbare Menge *PS* von **Prädikat-(Relations-) symbolen**, hier $P_1, P_2, ...$
  – informell *P*, *Q*, *R*, ... , *arbeitet_für*, usw.
  – incl. Abbildung (P-)**Stelligkeit** : $PS \to \mathbb{N}_0$
- **Junktoren** $\neg \ \wedge \ \vee \ \to \ \leftrightarrow$
- **Quantoren**
  $\forall$ *für alle ...gilt ...* **Allquantor, Generalisierung**
  $\exists$ *es existiert (mindestens) ein ... so dass ...*
  Existenzquantor, Partikularisierung
- syntaktische **Hilfssymbole**
  Klammern ( ) – auch mal als **{ }** oder **[ ]** geschrieben – und Komma **,** .

Mit **Termen** benennen wir **Objekte**. Die PL1-Terme sind induktiv definiert:

- Alle (Objekt-) Konstanten und Variablen sind Terme;
- Ist *f* ein *k*-stelliges Funktionssymbol ($k>1$) und sind $t_1, ..., t_k$ Terme, so ist $f(t_1, ..., t_k)$ ein Term.
  Alternativer PL1-Dialekt:
  Objektkonstanten $\cong$ „nullstellige“ Funktionen

Die Aussagen der PL1 heißen (PL1-)**Formeln**. Sie sind induktiv so definiert:

- Ist *P* ein *k*-stelliges Relationssymbol ($k>0$) und sind $t_1, ..., t_k$ Terme, so ist $P(t_1, ..., t_k)$ eine (**atomare**) Formel;
  ggf.: nullst. Rel.'n $\cong$ „Aussagenvariablen“;
- Sind *F* und *G* Formeln, dann auch $\neg F$, $(F \wedge G)$, $(F \vee G)$, $(F \to G)$, und $(F \leftrightarrow G)$.
- Ist *F* eine Formel und *x* eine Variable, so sind $\forall x F$ und $\exists x F$ Formeln.
- Man setzt zur besseren Lesbarkeit auch mal andere oder sogar mehr Klammern, z.B. in $\forall x[P(x)] \to \exists y[Q(x,y)]$.

komplexe Formeln: nicht atomar,

Teilformeln, Teilterme von Formeln
Induktion/Rekursion über Formelaufbau
Formel- und/oder Term**bäume**
} analog AL

PL1 in **natürlichsprachlich**en Aussagen**, erste Beispiele**

- DARMSTADT IST EINE GROßSTADT SÜDLICH VON FRANKFURT.
  da := Darmstadt, *f* := Frankfurt,
  $Gr(x) :\Leftrightarrow x$ ist Großstadt,
  $S(x,y) :\Leftrightarrow x$ liegt südlich von y
  → $Gr(da) \wedge S(da, f)$
- KEINE DEUTSCHE GROßSTADT LIEGT NÖRDLICH VON FLENSBURG.
  fl := Flensburg, $Gr(x) :\Leftrightarrow x$ ist Großstadt,
  $N(x,y) :\Leftrightarrow x$ liegt nördlich von y
  → $\neg\exists x(Gr(x) \wedge N(x, fl))$
- ALLE MENSCHEN WERDEN BRÜDER.
  $Mr(x) :\Leftrightarrow x$ ist Mensch,
  $Br(x,y) :\Leftrightarrow x$ wird Bruder von y
  → $\forall x \forall y (((M(x) \wedge M(y)) \wedge x \neq y) \to Br(x,y))$

Dabei haben wir noch die **Grundmenge** der Objekte offen gelassen, hier (denkbar) …

- (deutsche) (Groß-)Städte, Örtlichkeiten, … mit Konstanten Darmstadt, Frankfurt, Flensburg
- Menschen, Lebewesen, Gegenstände und Lebewesen, … (ohne Konstanten)

Ausführlichere Beispiele:

- DEINE MEINUNG INTERESSIERT MICH NICHT.
  Grundmenge: Personen + Meinungen
  Konstanten: *ich*, *du* Personen
  Funktionen: $m(x)$ Meinung von x (wobei für Meinungen w z.B. $m(w) := w$ oder $\varnothing$)
  Relationen: $Int(x,y)$ Meinung x interessiert Person y.
  → $\neg Int(m(du),ich)$
- BRUNOS VATER IST EIN VORGESETZTER VON STEFANIES MUTTER.
  Grundmenge: Personen
  Konstanten: *br*, *st* Personen
  Funktionen: $mu(x)$ Mutter von x
  $va(x)$ Vater von x
  Relationen: $Vor(x,y)$ x ist ein Vorgesetzter von y.
  → $Vor(va(br), mu(st))$

In welchen **Zusammenhängen** tauchen typischerweise **Quantoren** auf ? – einige **Grundmuster:**

- ALLE LACHEN.
  Grundmenge: eine Personengruppe
  Prädikate: $L(x)$ – x lacht
  → $\forall x\, L(x)$

… und mit der-/demselben Grundmenge/Prädikat:

- (MINDESTENS) EINER LACHT.
  → $\exists x\, L(x)$
- KEINER LACHT.
  → $\neg\exists x\, L(x)$
- ALLE LACHEN MICH AUS (SOGAR ICH MICH SELBST).
  Konstante: i – ich
  Relation: $Laus(x,y)$ – x lacht y aus
  → $\forall x\, Laus(x,i)$
- ALLE (ANDEREN) LACHEN MICH AUS.
  → $\forall x\, (x \neq i \rightarrow Laus(x,i))$
  verwendet „festes Prädikat“ = (i.a. infix, PL mit Identität). $x \neq i$ bedeutet $\neg =(x,i)$
- ALLE SCHADENFROHEN LACHEN MICH AUS.
  Prädikate: $Sf(x)$ – x ist schadenfroh
  → $\forall x\, (Sf(x) \rightarrow Laus(x,i))$

→ **1. Grundmuster eingeschränkter Allquantor**: **Für alle mit der Eigenschaft *E*** gilt …
$\forall x\, (E(x) \rightarrow \ldots)$

- SOME LIKE IT HOT. MANCHE MÖGEN'S HEIß.
  Grundmenge: Personen, Prädikat: $Mh(x)$
  → $\exists x\, Mh(x)$
- MANCHE STUDIERENDEN MÖGEN'S HEIß.
  weiteres Prädikat: $St(x)$ – x ist Studierende(r)
  → $\exists x\, (St(x) \wedge Mh(x))$
  (oder >1 gemeint? Dann PL1=!)
- MANCHE STUDIERENDEN LIEBEN LOGIK
  Grundmenge: Personen und Fächer
  Relation: $Li(x,y)$ – Person x liebt y
  Konstante: lo – Logik
  → $\exists x\, (St(x) \wedge Li(x,lo))$
  (oder >1 gemeint? Dann PL1=!)

→ **2. Grundmuster**: **eingeschränkter Ex.-quantor**
Für mindestens eines mit Eigenschaft *E* gilt …
$\exists x\, (E(x) \wedge \ldots)$

PL1 verallgemeinert AL:
Wenn wir

- uns auf nullstellige Prädikate(nsymbole) beschränken und keine Quantoren, Objekte (Variablen, Konstanten), oder Funktionen verwenden,

dann liefert die PL1-Theorie die AL-Theorie. Es reicht sogar, wenn nur

- keine Quantoren und Objektvariablen verwendet werden, wie im Beispiel
  $(Q(a) \vee \neg R(f(b),c)) \wedge P(a,b)$ →
  $(Q\_a \vee \neg R\_f\_b\_c) \wedge P\_a\_b$

In **PL2**, der Prädikatenlogik zweiter Stufe, wird auch über Prädikate und Funktionen quantifiziert, und damit **PL1 verallgemeinert**. Solche Ausdrucksmittel verwendet z.B. das **Induktionsaxiom** der natürlichen Zahlen:

Sei P eine mögliche Eigenschaft natürlicher Zahlen, und

- 1 habe die Eigenschaft P, und
- mit jeder Zahl n habe auch $n+1$ die Eigenschaft P.

Dann haben alle natürlichen Zahlen die Eigenschaft P:

$$\forall P((P(1) \wedge \forall n(P(n) \rightarrow P(plus(n,1)))) \rightarrow \forall n P(n))$$

In einer axiomatischen Mengenlehre lässt sich dieser Satz allerdings auch sinngemäß ohne quantifiziertes Prädikat schreiben und beweisen:

$$\forall M\,((eins \in M \wedge \forall n\,(n \in Nat \cap M \rightarrow plus(n,eins) \in M)) \rightarrow Nat \subseteq M)$$

PL1-Varianten
*Andere **Schreibweisen***

- **Quantoren**
  $\forall x\, F$ auch so: $\bigwedge_x F$, $\forall x\colon F$, $(\forall x)F$
  $\exists x\, F$ auch so: $\bigvee_x F$, $\exists x\colon F$, $(\exists x)F$
- **Relationen**: auch Infix, Mixfix.
- **Relationen- und Funktionensymbole**: $f_1^k, f_2^k, \ldots$, $P_1^k, P_2^k, \ldots$ (pro Stelligkeit k)
  (auch nullstellige, also konstante Terme/Formeln)

*Andere **Syntax und Bedeutung** (Auswahl)*

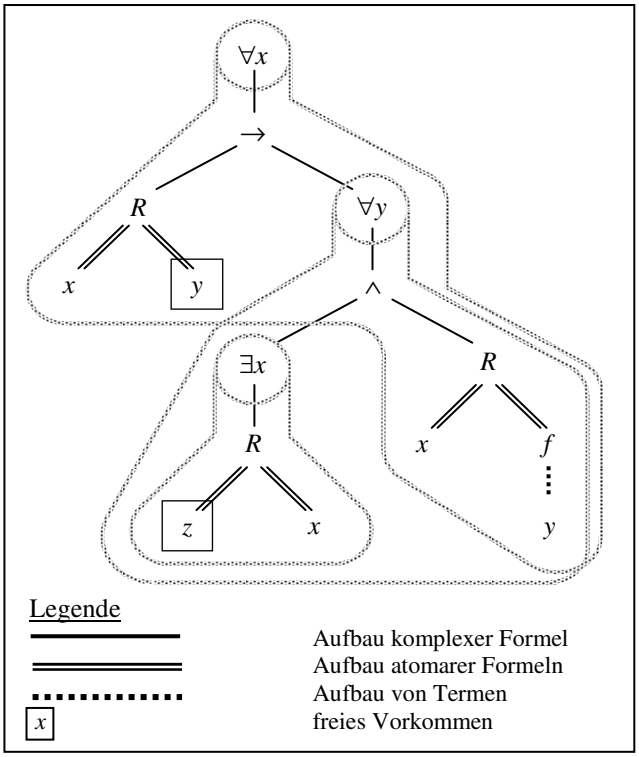
- PL1 mit **Identität** (PL1=):
  Vordefinierte zweistellige (Aquivalenz-)Relation =, ≡ oder ≈.
- PL1 mit **Sorten**:
  Objekte sind oft unterschiedlicher Art, z.B. Zah len, Vektoren, Matrizen. Relationen und Funktionen sind sortenspezifisch, z.B. lineare Abbildung ist „Matrix mal Vektor“, Kreuzprodukt ist „Vektor mal Vektor“.
  Technik: getrennte Variablen- und Konstanten-**Namen-Mengen**, oder Quantoren an Sorten **binden**: $\forall x \in M\ \exists y \in N : P(x,y)$
  Sorten-Ersatz in „unsortierter“ PL1: Prädikate $\forall x(M(x) \rightarrow \exists y(N(y) \wedge P(x,y)))$.
  (vgl. Grundmuster für eingeschränkte Quantoren)
- PL1 **ohne** besondere **Funktionensymbole**: In der Tat sind Funktionen spezielle Relationen.
  Nachteil: Man muss Linkstotalität und Rechtseindeutigkeit geeignet explizit „mitnehmen.“

Freie und gebundene Variablen

Die Tabelle rechts erklärt abgeleitete **syntaktische Definitionen** für PL1.

Im Beispiel $\forall x(R(x,y) \to \forall y(\exists x R(z,x)$ sind „unterhalb“ $[\forall x]$ alle $[x]$ gebunden, aber nicht alle durch dieses $[\forall x]$: teilweise wird diese Bindung von der Bindung durch $[\exists x]$ abgelöst. Ein $[y]$ ist frei, ein $[y]$ ist gebunden (das in $[\forall y]$), aber der Bereich des $[\forall y]$ enthält kein $[y]$. Alle $[z]$ sind frei. Die Begriffe sind im folgenden Baum illustriert.

Begriff	Definition	Im Formelbaum
[*Text*]	Ein **Vorkommen** (**Exemplar**) von *Text* in einer Formel. (Kann ja mehrfach!)	
Scope, Bereich	In $\forall x\, F$ und $\exists x\, F$ ist F der Scope vom äußersten $[\forall x]/[\exists x]$.	der ganze Ast unter dem Quantorknoten
Quantorenvariable	... ist das x in $\forall x\, F$ und $\exists x\, F$	
frei	$[x]$ ist frei in F, wenn es Teilterm von F ist (also keine Qu.-var.) und nicht im Scope eines $[\forall x]/[\exists x]$ steht.	
gebunden durch	$[x]$ ist gebunden durch $[\forall x]/[\exists x]$, wenn es im Scope von $[\forall x]/[\exists x]$ frei vorkommt.	Das unterste $[\forall x]/[\exists x]$ oberhalb $[x]$ bindet x
geschlossene Formel	... hat keine freien Variablen(vorkommen), auch **Satz** oder **Aussage** genannt.	



PL1-Semantik

Interpretationen sind Zuordnungen ...

Obj.konst. → Objekte
Fkt.symb.$_k$ → Funktionen$_k$ } **Struktur**
Rel.symb.$_k$ → Relationen$_k$ } **Struktur**
freie Obj.var. → Objekte **Belegung**
} **Interpretation**

Gebundene Variablenvorkommen sind von einer Interpretation „nicht betroffen“, sie werden bei der *Auswertung* (s.u.) verwendet (und dabei systematisch „mit allen möglichen Objekten“ belegt). Ihre eventuelle globale Interpretation wird ignoriert – ähnlich wie bei lokalen Variablen in Unterprogrammen.

Eine **Struktur** $A=(U, I_A)$ besteht aus

- einer nicht leeren Menge U von Objekten (**Grundmenge**, **Trägermenge**, **Universum**)
- einem Tripel = (I_K, I_F, I_P) (**Interpretation** der Strukturparameter) von Abbildungen

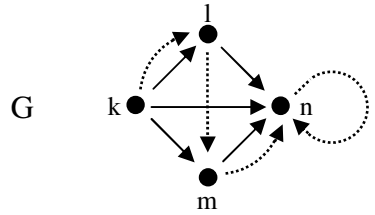
  $I_K : KS_A \to U, \quad KS_A \subseteq KS$, von (einigen) **Konstantensymbolen** auf Objekte in U;

  $I_F : FS_A \to \bigcup_{k=1}^{\infty} F(U^k, U), \; FS_A \subseteq FS$,
  von **Funktionssymbolen** auf ein- oder mehrstellige Funktionen auf U, derart dass k-stellige Symbole auf k-stellige Funktionen abgebildet werden;

  $I_P : PS_A \to \mathbf{P}(\bigcup_{k=0}^{\infty} U^k), \; PS_A \subseteq PS$,
  von **Prädikat-** und **Relationssymbolen** auf ein- oder mehrstellige Relationen über U, derart dass k-stellige Symbole auf k-stellige Relationen abgebildet werden.

[Seltener wird als Variante die **LT-Semantik** verwendet, die einen [l]eeren [T]räger, $U=\varnothing$, zulässt. (Achtung – andere Gesetze: Wann gilt $(\forall x\, F) \to (\exists x\, F)$?)]

Beispiel: Die Struktur A_{Bsp} sei gegeben durch:



- Universum U = Knotenmenge $\{k,l,m,n\}$ des Multigraphen G
- Konstantensymbole: $\{c\}$, Interpretation
  $I_K(c) :=$ Knoten l
- sonstige Funktionssymbole: $\{f\}$, Interpretation
  $I_F(f) :=$ gegeben durch $x \dashrightarrow f(x)$
- Prädikatsymbole $\{P\}$, Interpretation
  $I_P(P)(x,y) :=$ x=y, oder es verläuft ein Pfeil ⟶ von x nach y.

Term-Auswertung

Eine Struktur $A=(U,(I_K, I_F, I_P))$ heißt **passend** oder **ausreichend** für PL1-Term τ, wenn sie alle Konstanten-, Funktions-, und Prädikatsymbole in τ interpretiert.

In diesem Falle könnten wir bereits Variablen- (und Quantoren-) freie Terme und Formeln rekursiv auswerten:

c Konstante: $I(c) := I_K(c)$

f Funktionssymbol und $t_1, t_2, ..., t_k$ Terme:

$$I(f(t_1, ..., t_k))) := I_F(f)(I_A(t_1), ..., I_A(t_k))$$

Eine **Variablen-Interpretation** (oder **Belegung**) in einer Struktur $A = (U, I_A)$ ist eine Abbildung

$$I_V : VS_{I_V} \to U, \quad VS_{I_V} \subseteq VS,$$

von (einigen) Variablen(symbolen) auf Objekte in U.

Eine **Interpretation** $I = (A, I_V)$ = (Struktur, Belegung) bzw. $I = (I_K, I_F, I_P, I_V)$ heißt **passend** oder **ausreichend** für eine PL1-Formel φ, wenn sie alle Konstanten-, Funktions-, und Prädikatsymbole, sowie **alle *freien* Variablen** in φ interpretiert. In diesem Falle kann die **Formel** φ *theoretisch* wie folgt **ausgewertet** werden:

Rekursive **Auswertung von** (Teil-) **Termen** (von φ):

- x Variable: $I(x) := I_V(x)$
- c Konstante: $I(c) := I_K(c)$
- f Funktionssymbol und $t_1, t_2, ..., t_k$ Terme:
  $I(f(t_1, ..., t_k)) := I_F(f)(I(t_1), ..., I(t_k))$

Rekursive Auswertung von (Teil-) Formeln (von φ):

atomare Formeln:

$$I(P(t_1, ..., t_k)) := \begin{cases} W & \text{wenn} \quad (I(t_1), ..., I(t_k)) \in I_P(P) \\ F & \text{sonst} \end{cases}$$

„Junktorenformeln“:

$I(\neg\varphi), I(\varphi \wedge \psi), I(\varphi \vee \psi), I(\varphi \to \psi), I(\varphi \leftrightarrow \psi)$ ergeben sich aus $I(\varphi), I(\psi)$ wie in der Aussagenlogik.

„Quantorenformeln“:

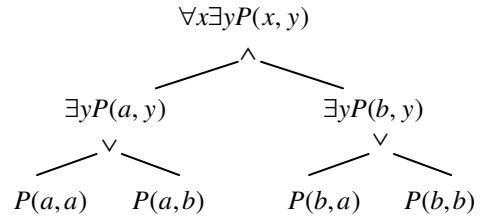
$$I(\forall x\varphi) := \begin{cases} W & \text{wenn für alle } a \in U \text{ gilt : } I_{x:=a}(\varphi) = W \\ F & \text{sonst} \end{cases}$$

$$I(\exists x\varphi) := \begin{cases} W & \text{wenn es } a \in U \text{ gibt mit } I_{x:=a}(\varphi) = W \\ F & \text{sonst} \end{cases}$$

$I_{x:=a}$: wie I, aber $I_V(x) := a$ – egal ob und wie (das in φ freie) x belegt war. Insgesamt ist für x ggf. ganz U durchzuprüfen auch mit **namenlosen** Elementen „a“ von U“!

Die Interpretation und die Auswertung verlaufen also in jeder quantorfreien Formel von innen nach außen (im Formelbaum: von den Blättern zur Wurzel). Für jede quantisierte Teilformel werden aber quasi von außen Auswertungen für im Prinzip alle Belegungen der durch den Quantor gebundenen Variablen mit Werten aus ganz U angestoßen, was zu einer Vielzahl (meist Unzahl) von Auswertungen quantorfreier Formeln führt.

Beispiel mit zweielementigem Universum $U=\{a,b\}$:



$$\forall x\exists y P(x,y) \Leftrightarrow (P(a,a) \vee P(a,b)) \wedge (P(b,a) \vee P(b,b))$$

Wenn U unendlich ist, ist so keine systematische mechanische Wahrheitsprüfung möglich.

Andererseits überblickt man bei einfachen Beispielen intuitiv, ob eine Interpretation (bzw. bereits Struktur, wenn die Formel keine freien Variablen enthält) eine Formel wahr macht – selbst bei unendlichem U.

Semantische Begriffe für PL1

Eine zu einer Formel φ passende Interpretation $I = (A, I_V)$ heißt **Modell von** φ, wenn $I(\varphi) = \mathrm{W}$.

Schreibweise: $\boxed{I \vDash \varphi}$ – Redeweise: φ **gilt unter** I, I **erfüllt** φ.

Eine Formel φ heißt

- **erfüllbar**, wenn sie mindestens **ein** Modell hat (sonst: **unerfüllbar**);
- **allgemeingültig** (logisch wahr, $\boxed{\vDash \varphi}$) wenn **jede** passende Interpretation Modell ist (sonst: **widerlegbar**);
- **gültig in** der Struktur A ($\boxed{A \vDash \varphi}$)**,** wenn für jede passende Belegung I_V die Interpretation $I = (A, I_V)$ Modell ist. Beispiel: $a + b = b + a$ gilt in $\mathbb{N}_0$.

Zwei Formeln φ und ψ heißen (logisch, semantisch) **äquivalent**, $\varphi \equiv \psi$, wenn unter jeder passenden Interpretation I gilt: $I(\varphi) = I(\psi)$.

Eine Formel φ **folgt** (logisch, semantisch) aus einer Formel σ (bzw. Formelmenge M), geschrieben: $\boxed{\sigma \vDash \varphi}$ ($\boxed{M \vDash \varphi}$), wenn jedes Modell von σ (bzw. M) auch Modell von φ ist.

Einige wichtige Äquivalenzen sind:

Dualität von $\forall$ und $\exists$

$\neg\forall x\varphi \equiv \exists x\neg\varphi$ $\qquad$ $\neg\exists x\varphi \equiv \forall x\neg\varphi$

Vertauschung der Quantoren-Reihenfolge

$\forall x\forall y\varphi \equiv \forall y\forall x\varphi$ $\qquad$ $\exists x\exists y\varphi \equiv \exists y\exists x\varphi$

(nur $\forall/\forall$ oder $\exists/\exists$ – nicht gemischt !)

Quantoreinführung und **-beseitigung** – Voraussetzung: x **nicht frei** in φ

$\varphi \equiv \forall x\varphi$ $\qquad$ $\varphi \equiv \exists x\varphi$

(Leere) **Scope-Erweiterung** (auf ψ) – Voraussetzung: x **nicht frei** in ψ.

$\forall x\varphi \wedge \psi \equiv \forall x(\varphi \wedge \psi)$ $\quad$ $\exists x\varphi \vee \psi \equiv \exists x(\varphi \vee \psi)$

$\forall x\varphi \vee \psi \equiv \forall x(\varphi \vee \psi)$ $\quad$ $\exists x\varphi \wedge \psi \equiv \exists x(\varphi \wedge \psi)$

$\psi \to \forall x\varphi \equiv \forall x(\psi \to \varphi)$ $\quad$ $\psi \to \exists x\varphi \equiv \exists x(\psi \to \varphi)$

aber: $\forall x\varphi \to \psi \not\equiv \forall x(\varphi \to \psi)$!

Gegenbeispiel: $\varphi = x > 0$, $\psi = 0 > 0$ in $\mathbb{N}_0$

Distributivität

$\forall x\varphi \wedge \forall x\psi \equiv \forall x(\varphi \wedge \psi)$

$\exists x\varphi \vee \exists x\psi \equiv \exists x(\varphi \vee \psi)$

(Nicht „über Kreuz“, d.h. $\forall + \vee$ oder $\exists + \wedge$! Gegenbeispiele in $\mathbb{N}_0$?)

Ersetzungssatz (Satz über die äquivalente Ersetzung):

Werden in einer Formel φ ein oder mehrere Vorkommen einer Teilformel ψ durch eine zu ψ äquivalente Formel ρ ersetzt, so ist die entstehende Formel zu φ äquivalent.

Substitution

AL-Substitution in PL1:
Werden in einer AL-Tautologie (bzw. in einer unerfüllbaren AL-Formel) alle Vorkommen jeder Aussagevariablen jeweils durch die gleiche PL1-Formel ersetzt, so ist die entstehende PL1-Formel ebenfalls eine Tautologie (bzw. unerfüllbar).

Dies ist eine Abart der AL-Substitution, formale Definition wie dort.

Die eigentliche **Substitution** in PL1,

$$\varphi \mapsto \varphi_{[x_1,...,x_n/\tau_1,...,\tau_n]}$$

ersetzt keine Aussagen-, sondern ***Objektvariablen***, nämlich paarweise verschiedene Objektvariablen $x_1, ..., x_n$ durch (nicht unbedingt verschiedene) Terme $\tau_1, ..., \tau_n$.

Termebene:

$$x \in VS \cup KS: \quad \overbrace{x_{[x_1,...,x_n/\tau_1,...,\tau_n]}}^{sub} := \begin{cases} \tau_i & \text{für } x = x_i \\ x & \text{sonst} \end{cases},$$

$$f \in FS \quad f(\omega_1,...,\omega_k)_{[sub]} := f(\omega_{1[sub]},...,\omega_{k[sub]})$$

Übergang zur Aussagenebene:

$$R \in PS \quad R(\omega_1,...,\omega_k)_{[sub]} := R(\omega_{1[sub]},...,\omega_{k[sub]})$$

Aussagenebene:

Negation $(\neg\varphi)_{[sub]} := \neg(\varphi_{[sub]})$

$\otimes \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$: $(\varphi \otimes \rho)_{[sub]} := \varphi_{[sub]} \otimes \rho_{[sub]}$

Bisher alles trivial, doch nun ...

Quantoren $(\forall x\varphi)_{[sub]} := \forall x(\varphi_{[sub \neq x]})$

$(\exists x\varphi)_{[sub]} := \exists x(\varphi_{[sub \neq x]})$

$sub_{\neq x}$ ist „wie *sub*, aber *gebundene* [*x*] werden *nicht* ersetzt". (Warum?!)

Satz über die gebundene Umbenennung
Sei φ eine Formel, die die Variable *y* nicht frei enthält.
Dann gilt $\exists x\varphi \equiv \exists y\varphi_{[x/y]}$ und $\forall x\varphi \equiv \forall y\varphi_{[x/y]}$.

Ein Haken bei der Substitution in PL1

Da das, was im Allgemeinen gilt, auch im Besonderen gilt, sollte man meinen, dass für jede Substitution gilt:

$(\forall x\varphi) \rightarrow \varphi_{[x/\tau]}$!?

In der Tat ist das **nicht** so:

- $(\forall x \neg\forall y\ x = y) \rightarrow (\neg\forall y\ y = y)$??
- $(\forall x \exists y\ x < y) \rightarrow (\exists y\ y < y)$??

Analyse: Anscheinend kann Unsinn herauskommen, wenn beim Substituieren *ein in einer Teilformel freies Variablenvorkommen gebunden wird*.

Satz über die Substitutierbarkeit
Für jeden PL1-Term τ, jede Variable *x* und jede PL1-Formel φ sind äquivalent:

1. Beim Übergang von φ nach $\varphi_{[x/\tau]}$ wird kein freies Variablenvorkommen gebunden. (Formalisierung?)
2. Kein freies [*x*] in φ liegt im Scope eines Quantorvorkommens mit einer Quantorvariable *y*, die in τ vorkommt.

Man sagt dann: τ ist **erlaubt** (man sagt auch **frei**, **substituierbar**) **für** *x* **in** PL1-Formel φ, bzw. **free**(τ ,*x*, φ).

Hilfsalgorithmus: Substituierbar oder nicht?

... beantwortet die Frage „Ist τ substituierbar für x_i in PL1-Formel φ?" bzw. „free(τ, x_i, φ)?"

Geg.: τ, x_i, φ (*i* liegt fest!)

1. Bestimme alle freien [x_i] (freie Vorkommen des gegebenen x_i) in φ.
   Gibt es keine ➔ τ ist frei für x_i in φ & Stop.
2. Für alle freien [x_i] in φ: Bestimme in φ alle Quantisierungen [Qx_k], automatisch mit $i \neq k$, in deren Scope dieses [x_i] liegt.
   a. Gibt es keine ➔ τ ist frei für x_i in φ & Stop.
   b. Für alle diese [Qx_k]:
      Kommt x_k in τ vor ➔
      τ ist nicht frei für x_i in φ & Stop.
3. (Sind nun also alle freien Vorkommen [x_i] in φ durchprobiert, ohne auf Stop zu laufen)
   ➔ τ ist frei für x_i in φ & Stop.

Manche sagen auch „*x* frei für τ" anstatt „τ frei für *x*".

Einfache Folgerungen:

- τ enthält keine Variable ➔ τ frei für alle Variablen in allen Formeln.
- keine Variable in τ gebunden in φ ➔ τ frei für alle Variablen in φ.
- x_i ist frei für x_i in (jedem) φ.
- φ enthält kein freies [x_i] ➔ (jedes) τ ist frei für x_i in φ

Substitutionssatz
free(τ ,*x*, φ) $\Rightarrow$ $\forall x\varphi \rightarrow \varphi_{[x/\tau]}$

Was geht bei PL1 nicht?

PL1-Vorteil: Fast die ganze Mathematik lässt sich damit ausdrücken. Daher würde man gerne entscheiden, welche Formeln wahr und welche falsch sind!

In der Aussagenlogik sind semantische Eigenschaften prinzipiell entscheidbar: die Wahrheitstafel erlaubt die *vollständige Prüfung anhand aller* (*endlich vielen*!) *Interpretationen* der Auss.-Variablen (= AL-Belegungen). Prädikatenlogik hat diesbezüglich einen doppelten Nachteil: bei ihr wären

- **unendlich viele Strukturen** (Grundmengen, auch beliebige unendliche, Interpretationen von Konstanten, Funktionen und Prädikaten), und
- ggf. **unendlich viele** Variablen-**Interpretationen**

zu untersuchen.

Satz über die **Unentscheidbarkeit**
Es gibt keinen Algorithmus, der Allgemeingültigkeit (oder Erfüllbarkeit oder Folgerung) entscheidet.

[Mögliche Beweise einer PL1-Formel (falls hinreichend formal und d.h. auch: sehr ausführlich, z.B. auf Werkzeugkastenniveau) können jedoch algorithmisch *überprüft* werden.]

Frage: *Noch* keinen – weil einfach noch keiner schlau genug war einen zu (er)finden? Oder gar endgültig?

Antwort: Nein, *vermutlich* endgültig, weil es „*fast beweisbar*" keinen geben kann! (Church, Turing 1936)

Rückfrage: *Fast beweisbar*, was soll denn das bedeuten?

Es gibt zahlreiche mögliche mathematische Definitionen des Begriffs des (beliebigen!) Algorithmus, z.B.

- Turing-Maschine (TM) (besonders elementare Programmschritte und Speicher)
- Rekursive Funktionen,
- Markov-Systeme,
- Post-Systeme, usw.

Satz über **bisherige Algorithmus-Definitionen** (und **Churchsche These** über **zukünftige**):
Alle „überzeugenden" Algorithmusdefinitionen sind äquivalent. Die Klasse der intuitiv berechenbaren Funktionen ist genau die Klasse der Turing-berechenbaren Funktionen.

Beweisstrategie der Unentscheidbarkeit:
(1) Einführung der **Gödel-Nummern**: eindeutige Codierung einer TM (allgemeiner: Elementen einer aufzählbaren Menge) als eine einzige Zahl; dadurch sind die TM alle aufzählbar T_1, T_2, ...
(2) Das **Halteproblem** (Es gibt keine TM, die für jede TM und beliebige Eingaben entscheidet, ob diese TM auf dieser Eingabe terminiert) per **Diagonalargument** beweisen. *Annahme eine TM T_{super} könne das. Dann berechne TM $T_{seltsam}$* (unter Verwendung von T_{super})*:*

$$T_{seltsam}(T_i) := \begin{cases} T_i(T_i)+1 & \text{wenn } T_i(T_i) \text{ existiert} \\ 0 & \text{wenn } T_i(T_i) \text{ nicht stoppt} \end{cases}$$

Dann wäre aber $T_{seltsam}$ nicht in der Aufzählung! Also ist die Annahme falsch.

(3) PL1-Erfüllbarkeitsentscheidung auf T_{super} zurückführen. (technische Fleißaufgabe).

Was geht überhaupt bei PL1?

Eine Frage(nmenge) ist durch Algorithmus *Alg* **halbentscheidbar** :⇔

- Falls korrekte Antwort (für alle) JA ⇒ Alg endet irgendwann mit JA;
- falls nicht ⇒ Alg endet nicht.

Für die Allgemeingültigkeit (also auch Unerfüllbarkeit und Folgerung) lernen wir noch Beweisregeln und Halbentscheidungs-Algorithmen kennen:

- Beweiskalküle
- Beweisalgorithmus
- PL1-Tableaux
- PL1-Resolution
- Natürliches Schließen (Werkzeugkasten)

Also kann es keinen Halbentscheidungs-Algorithmus für die Erfüllbarkeit oder Widerlegbarkeit geben! Sonst könnte man z.B. „verzahnt" φ auf Allgemeingültigkeit und $\neg\varphi$ auf Widerlegbarkeit prüfen („halb-entscheiden"), wovon eines irgendwann JA ergibt, zusammen also doch über die Allgemeingültigkeit entscheiden.

Eine Gruppe von Prädikatenkalkülen

Sprache

- **PL1-Formeln** (mit Aussagevariablen)

Regeln

- **Modus Ponens** $\text{MP} = \dfrac{\varphi, \varphi \to \psi}{\psi}$

Axiome(nschemata)

- ein AL-Axiomensystem **Aussagenlogik AL**

und beliebige Verallgemeinerungen $\forall x_1 \ldots \forall x_1$ von Formeln der Arten ...

- $\varphi \to \varphi_{[A_1,\ldots,A_k / \varphi_1,\ldots,\varphi_k]}$ **AL-Gültigkeit AL-G**
  wobei φ AL-Formel, φ_i PL1-Formeln
- $(\forall x\varphi) \to \varphi_{[x/\tau]}$ **Substit.-satz, Spezialisierung SP**
  sofern τ f. x in φ erlaubt ist, d.h. free(τ ,x, φ)
- $\varphi \to (\forall x\varphi)$ **Generalisierung GN**
  sofern x nicht frei in φ vorkommt
- $\forall x(\varphi \to \psi) \to (\forall x\varphi \to \forall x\psi)$
  Generalisierungsdistribution GD
- $\exists x\varphi \leftrightarrow \neg\forall x\neg\varphi$ **Existenz**„definition" **ED**

Satz: **Korrektheit dieser PL1-Kalküle**

- Alle ableitbaren Formeln sind allgemeingültig.
- Alle aus einer Formelmenge M ableitbaren Formeln sind Folgerungen von M.

Gödelscher Vollständigkeitssatz
... und umgekehrt! –

- Alle allgemeingültigen Formeln und Folgerungen sind damit ableitbar.

Generalisierungssatz **GS**
$M \vdash \varphi$ und x in keinem $\psi \in M$ frei $\Rightarrow$ $M \vdash \forall x\varphi$.

Deduktionssatz **DS**
$M \cup \{\varphi\} \vdash \psi \;\Rightarrow\; M \vdash (\varphi \to \psi)$

Beispiel für eine Meta-Ableitung (d.h. unter Verwendung semantischer Sätze)

1. $(A_1 \to \neg A_2) \to (A_2 \to \neg A_1)$ aus AL, MP (mehrfach)
2. $(\forall y\neg P(y) \to \neg P(x)) \to (P(x) \to \neg\forall y\neg P(y))$ 1, AL-G
3. $\forall x[\,(\forall y\neg P(y) \to \neg P(x)) \to (P(x) \to \neg\forall y\neg P(y))\,]$ 2, GS (M=Ø)
4. $\forall x\,(\forall y\neg P(y) \to \neg P(x)) \to \forall x(P(x) \to \neg\forall y\neg P(y))$ 3, GD
5. $\forall y\neg P(y) \to \neg P(x)$ SP (mit $\forall$x)
6. $\forall x(\,\forall y\neg P(y) \to \neg P(x)\,)$ 5, GS
7. $\forall x(P(x) \to \neg\forall y\neg P(y))$ 4,6, MP

Jetzt verfügen wir über Halbentscheidungsverfahren der Allgemeingültigkeit von φ:

Brute Force (Rohe Gewalt) **(ineffizient!)**
Dazu produzieren wir mittels eines PL1-Kalküls systematisch der Reihe nach alle (!) möglichen Ableitungen (*), und nach jeder solchen Produktion:
Erzeugte Ableitung endet mit φ?

- ja: JA, Stop.
- nein: nächste Produktion.

(*) Nicht ganz trivial, denn SP hat ja ∞ viele Varianten! Also nicht so: „Stufe n: n Regel-Anwendungen auf alle Axiome. n=0, 1, ..."
Man kann aber auch gezielter vorgehen, z.B. so ...

PL1-Tableaux

Anwendung für geschlossene Formeln (ohne freie Var.)

Wie AL-Tableaux + *zusätzliche* Regeln

Falls free(τ,x,φ): **Spezialisierung**

$\mathrm{Sp}_+ = \dfrac{\forall x\varphi}{\varphi_{[x/\tau]}}$ und $Sp_- = \dfrac{\neg\exists x\varphi}{\neg\varphi_{[x/\tau]}}$

Falls c Konstantensymbol, das im Zweig neu ist:

Witness (**Zeuge**)

$\mathrm{W}_+ = \dfrac{\exists x\varphi}{\varphi_{[x/c]}}$ und $\mathrm{W}_- = \dfrac{\neg\forall x\varphi}{\neg\varphi_{[x/c]}}$

Anwendungsbeispiel:
Zu beweisen: $\forall x(P(x)\vee Q(x)) \rightarrow (\exists xP(x)\vee\forall xQ(x))$

Wir widerlegen die Gegenbehauptung:

		(Regel	Prämisse)
1	$\neg[\forall x(P(x)\vee Q(x)) \rightarrow (\exists xP(x)\vee\forall xQ(x))]$		
2.	$\forall x(P(x)\vee Q(x))$	(TR1)	1
3.	$\neg(\exists xP(x)\vee\forall xQ(x))$	(TR1)	1
4.	$\neg\exists xP(x)$	(TR3)	3
5.	$\neg\forall xQ(x)$	(TR3)	3
6.	$\neg Q(c)$ ····· ↯	W_	5
↯ ····· 7.	$\neg P(c)$	Sp_	4
8.	$P(c)\vee Q(c)$	Sp+	2
9. $P(c)$	10. $Q(c)$	(TR8)	8

AL-Tableau-Regeln s.S.15

Satz: **PL1-Tableaux-Eigenschaften**
Als *Kalkül* ist PL1-Tableaux **korrekt** und **vollständig**:

- Enthalten alle Zweige einen Widerpruch zwischen zwei Literalen, ist die Wurzel unerfüllbar bzw. ihr Gegenteil allgemeingültig
- Ist die Wurzel unerfüllbar bzw. ihr Gegenteil allgemeingültig, so existiert ein daraus abgeleitetes Tableau mit einem Widerspruch zwischen 2 Literalen.

Als Algorithmus sind PL1-Tableaux leider *nicht* unbedingt *terminierend*.
Bei geeigneter Variation und Steuerung, z.B. im Rahmen eines Brute-Force-Verfahrens oder mit effizienteren Techniken taugen sie als Halbentscheidungsverfahren:

Die Terminierung bei *unerfüllbarer* Wurzel kann garantiert werden.

PL1-Werkzeug-Ergänzungs-Kasten

für geschl. Formeln

Zusätzliche Ableitungsregeln

All-Beseitigung, Spezialisierung	Sp $\dfrac{\forall x\varphi}{\varphi_{[x/\tau]}}$ (τ variablenfrei)
Existenz-Einführung	EE $\dfrac{\varphi_{[x/\tau]}}{\exists x\varphi}$ (τ variablenfrei)
Existenz-Benutzung	EB $\dfrac{\exists x\varphi}{\varphi_{[x/c]}}$ (c neue Konst.)[1]
Nicht-Existenz-Benutzung	NEB $\dfrac{\neg\exists x\varphi}{\forall x\neg\varphi}$
Nicht-All-Benutzung	NAB $\dfrac{\neg\forall x\varphi}{\exists x\neg\varphi}$

[1]) Kein Beweis der Formel mit c, nur eine temporäre Namensgebung, informell: „ ..., nennen wir es c“.

PL1-spezifisches Beweis- (und Block-) Schema

All-Beweis-Schema: `Zeige` $\forall x\varphi$, unmittelbar gefolgt von weiterem Blockbeginn `Zeige` $\varphi_{[x/c]}$ (c im Beweis neuer Konstantenname). Sobald im gleichen Block $\varphi_{[x/c]}$ erreicht: Blockende. Dann weiteres Blockende mit $\forall x\varphi$ (AB).

All-Beweis:
Zeige $\forall x\varphi$
| Zeige $\varphi_{[x/c]}$ (c neue Konstante)
| | ⋮
| $\varphi_{[x/c]}$ (…B)
$\forall x\varphi$ (AB)

Satz:
Der AL&PL1-Werkzeugkasten ist vollständig und korrekt für **geschlossene** Formeln (d.h. ohne freie Variablen).

2 Beweisstrategien für Existenzaussagen:

Direkter Beweis:	Indirekter Beweis:
Zeige $\exists x\varphi$	Zeige $\exists x\varphi$
. | ⋮	| $\neg\exists x\varphi$ (Ann)
. | $\varphi_{[x/\tau]}$	| ⋮
| $\exists x\varphi$ (EE)	| $\bot$ (WE)
$\exists x\varphi$ (DB)	$\exists x\varphi$ (IB)

Beispiel 1: $\forall xP(x) \rightarrow \exists yP(y)$?

(1)	Zeige $\forall xP(x) \rightarrow \exists yP(y)$	
(2)	| $\forall xP(x)$	Ann.
(3)	| Zeige $\exists yP(y)$	
(4)	| | $P(a)$	1, Sp
(5)	| | $\exists yP(y)$	2, EE
(6)	| $\exists yP(y)$	DB
(7)	$\forall xP(x) \rightarrow \exists yP(y)$	BB

Beispiel 2: $\{\forall x(P(x) \rightarrow Q(x)), \neg Q(a)\} \models \neg\forall yP(y)$?

(1)	$\forall x(P(x) \rightarrow Q(x))$	Geg.
(2)	$\neg Q(a)$	Geg.
	Zeige $\neg\forall yP(y)$	
(3)	| $\forall yP(y)$	Ann.
(4)	| $P(a)$	3, Sp
(5)	| $P(a) \rightarrow Q(a)$	1, Sp
(6)	| $Q(a)$	4, 5, MP
(7)	| $\bot$	2, 6, WE
(8)	$\neg\forall yP(y)$	IB

PL1-Normalformen

Pränexe Normalform

Eine PL1-Formel ist in **PNF** (Pränexer Normalform), wenn „alle Quantoren vorne stehen.“ (Formal?!).

Normalformensatz (sogar „bereinigte PNF“)

Zu jeder PL1-Formel gibt es mindestens eine äquivalente **bereinigte PNF-Formel**, d.h. in der zusätzlich

- keine Variable sowohl gebunden als auch frei vorkommt und
- alle Quantorvariablen unterschiedlich sind.

Algorithmus Ber: Bereinigung

+ Schritt-Beispiele anhand der Ausgangsformel

$P(x) \vee (\forall x\, (\exists x Q(x,x) \wedge R(x)\,) \vee\ S(x)\,)$

1. Suche von links nach rechts eine noch nicht „erledigte“ Quantorvariable. Gibt es keine, ist die Bereinigung fertig. $P(x) \vee (\forall \mathbf{x}\, (\exists x Q(x,x) \wedge R(x)\,) \vee S(x)\,)$

2. Gibt es ein kollidierendes (erledigtes oder freies) Vorkommen derselben Variable?

   JA → $P(\mathbf{x}) \vee (\forall \mathbf{x}\, (\exists x Q(x,x) \wedge R(x)\,) \vee\ S(x)\,)$

   - Benenne eine Quantorvariable der beiden sowie alle durch sie gebundenen Variablenvorkommen um zu einem neuen Variablennamen.
   - Markiere die unerledigte beider Vorkommenmengen als erledigt.

     $P(x) \vee (\forall \mathbf{y}\, (\exists x Q(x,x) \wedge R(\mathbf{y})\,) \vee\ S(x)\,)$

   NEIN →

   - Markiere die gefundene Quantorvariable sowie alle durch sie gebundenen Variablenvorkommen als erledigt.

3. Gehe nach 1.

   … $P(x) \vee (\forall \mathbf{y}(\exists \mathbf{x} Q(x,x) \wedge R(\mathbf{y})\,) \vee\ S(x)\,)$ usw.

Algorithmus BPNF: Ber. PNF Umformung

1. Bereinigung durch Ber (s.o.)
2. $\rightarrow$ und $\leftrightarrow$ durch $\neg \wedge \vee$ ersetzen (vgl. AL)
3. Alle Negationen hinter die Quantoren ziehen:
   a) $\neg\forall x\varphi \rightarrow \exists x\neg\varphi$
   b) $\neg\exists x\varphi \rightarrow \forall x\neg\varphi$
   c) $\neg(\varphi \wedge/\vee \psi) \rightarrow (\neg\varphi \vee/\wedge \neg\psi)$
   [falls Quantor(en) in φ oder ψ]
   d) $\neg\neg\varphi \rightarrow \varphi$
   [falls Quantor(en) in φ – verzichtbar, beschleunigt)
4. Alle Quantifikationen in der vorliegenden Reihenfolge vor die ganze Formel ziehen (leere Scope-Erweiterungen).

Gemischte Schrittfolgen sind möglich, solange nur legitime Scope-Erweiterungen durchgeführt werden.

Satz: **PNF Umformung**
BPNF terminiert und ist korrekt.

Skolem-Form

Eine PL1-Formel ist in Skolem-Form, wenn sie in BPNF ist und kein $\exists$ enthält. Oft gibt es zu einer PL1-Formel φ *keine* äquivalente Formel in Skolem-Form – z.B. gibt es keine zu $\forall x\exists y P(x,y)$. Aber, ...

z.B. im Falle $\forall x\exists y P(x,y)$ können wir zu jedem x aus allen zu x „passenden“ y eines willkürlich herausgreifen und als $f(x)$ dem x zuordnen.
→ $\forall x P(x, f(x))$ und weg ist der Existenzquantor! Doch was haben wir eigentlich erhalten?

Satz: **Skolem-Form**

(1) Zu jeder PL1-Formel φ existiert eine Formel ψ in Skolem-Form, die genau dann **erfüllbar** ist, wenn φ erfüllbar ist.

(2) Solch eine **erfüllbarkeitsäquivalente** Formel ψ erhält man in endlich vielen Schritten durch den Algorithmus Skol.

Algorithmus **Skol** – „**Skolemisierung**“

- Wende BPNF an.
- Wenn nun kein Existenzquantor vorkommt: STOP.
- Wiederhole, solange die Formel einen Existenzquantor enthält:
  Sie hat die Form $\exists y\, \rho$ oder $\forall y_1 \cdots \forall y_n \exists z\, \rho$.
  - Im ersten Fall ersetze $\exists y\, \rho$ durch $\rho_{[y/c]}$ mit einer Konstante c, die in ρ nicht vorkommt.
  - Im zweiten Fall ersetze $\forall y_1 \cdots \forall y_n \exists z\, \rho$ durch $\forall y_1 \cdots \forall y_n\, \rho_{[z/f(y_1,\ldots,y_n)]}$ mit einem n-stelligen Funktionssymbol f, das in ρ nicht vorkommt.

  (c, f : **Skolem-Funktion**)

Skolemisierungs-Beispiel

$\exists x\, \forall y\, \forall z\, \exists u\, \forall v\, \exists w\, P(x,y,z,u,v,w)$

$\exists x\, \forall y\, \forall z\, \exists u\, \forall v\, \exists w\, P(x,y,z,u,v,w)$

$\forall y\, \forall z\, \exists u\, \forall v\, \exists w\, P(a,y,z,u,v,w)$

$\forall y\, \forall z\, \forall v\, \exists w\, P(a,y,z,f(y,z),v,w)$

$\forall y\, \forall z\, \forall v\, P(a,y,z,f(y,z),v,g(y,z,v))$

Einschub:

Äquivalenz bzgl. Erfüllbarkeit / Allgem.-gültigkeit

Lemma: Jede PL1-Formel φ mit (mindestens) einer freien Variablen y_i ist erfüllbarkeitsäquivalent zu der Formel $\exists y_i\, \varphi$.

Grund: Die Erfüllbarkeit sowohl von φ mit der freien Variablen y_i als auch von $\exists y_i\, \varphi$ bedeutet die Existenz einer passenden Struktur und die Existenz eines passenden Wertes für y_i, die φ wahr machen.

Mehrfach angewendet auf die freien Variablen $y_1, \cdots, y_n$ →

Korollar Ex.-Abschluss: Jede PL1-Formel φ mit den freien Variablen $y_1, \cdots, y_n$ ist erfüllbarkeitsäquivalent zu der geschlossenen Formel $\exists y_1 \cdots \exists y_n\, \varphi$.

Analog/dual: **Allgemeingültigkeits-Äquivalenz!**

Lemma: Jede PL1-Formel φ mit (mindestens) einer freien Variablen y_i ist genau dann allgemeingültig, wenn die Formel $\forall y_i\, \varphi$ allgemeingültig ist.

Grund: Die Erfüllbarkeit sowohl von φ mit der freien Variablen y_i als auch von $\forall y_i\, \varphi$ bedeutet dass in allen passenden Strukturen alle Werte für y_i die Formel φ wahr machen.

Mehrfach angewendet auf alle y_i →

Korollar: Jede PL1-Formel φ mit den freien Variablen $y_1, \cdots, y_n$ ist allgemeingültigkeitsäquivalent zu der geschlossenen Formel $\forall y_1 \cdots \forall y_n\, \varphi$.

Klauselnormalform

Eine PL1-Formel in Skolem-Form ist in **Klausel-Normalform**, wenn ihre Matrix (:= der Rest hinter den Quantoren) in KNF ist (mit atomaren Formeln anstelle der Aussagevariablen).

Satz: **Erfüllbarkeitsäquivalente Klauselnormalform**
Zu jeder PL1-Formel existiert eine erfüllbarkeitsäquivalente Formel in Klauselnormalform.

	∃-Abschl.		BPNF		Skol		KNF	
φ_1	erf	φ_2	äq	φ_3	erf	φ_4	äq	φ_5
belie-big		geschl.		BPNF geschl.		Skolem geschl.		Klausel-NF geschlossen

Man lässt meist die (nun redundanten) Allquantoren weg und schreibt φ_5 in $\{\ \}_{KNF}$-Form.

Beispiel für KlNF-Berechnung

Formel $\forall y\exists z P(x,y,z) \to \neg\exists y\forall v[Q(x,v) \to R(x,y)]$

abg. $\exists x\,\{\forall y\exists z P(x,y,z) \to \neg\exists y\forall v[Q(x,v) \to R(x,y)]\}$

berein. $\exists x\,\{\forall y\exists z P(x,y,z) \to \neg\exists u\forall v[Q(x,v) \to R(x,u)]\}$

$\to$ ers. $\exists x\,\{\neg\forall y\exists z P(x,y,z) \vee \neg\exists u\forall v[\neg Q(x,v) \vee R(x,u)]\}$

$\neg$ rein $\exists x\,\{\exists y\forall z\neg P(x,y,z) \vee \forall u\exists v[Q(x,v) \wedge \neg R(x,u)]\}$

pränex $\exists x\,\exists y\forall z\forall u\exists v\,\{\neg P(x,y,z) \vee [Q(x,v) \wedge \neg R(x,u)]\}$

skolemis. $\forall z\forall u\{\neg P(a,b,z) \vee [Q(a,f(z,u)) \wedge \neg R(a,u)]\}$

Klausel-NF

$\forall z\forall u\,\{[\neg P(a,b,z) \vee Q(a,f(z,u))] \wedge [\neg P(a,b,z) \vee \neg R(a,u)]\}$

als Mengen

$\big\{\{\neg P(a,b,z), Q(a,f(z,u))\}, \{\neg P(a,b,z), \neg R(a,u)\}\big\}$

Zwei Herbrand-Konzepte

Für eine geschlossene Skolem-Formel φ ist die Menge der Objekte, „*über die man konkret reden kann*“, das **Herbrand-Universum** $D(\varphi)$, d.h. die wie folgt induktiv definierte Termemenge:

- Konstanten in φ gehören zu $D(\varphi)$; falls es keine gibt, dann „c“ (a_1).
- f n-stelliges Funktionssymbol in φ und $\tau_1,...,\tau_n \in D(\varphi) \Rightarrow f(\tau_1,...,\tau_n) \in D(\varphi)$.

Beispiel:

$D(\forall x\, P(g(x),a) \to Q(f(x,b))) =$

$\{a, b, g(a), g(b), f(a,a), ..., g(f(a,g(b))), ...\}$

Für eine geschl. Skolem-Formel φ ist die **Herbrand-Expansion** von φ die Formelmenge

$E(\varphi) := \{Matrix(\varphi)_{[y_1,...,y_n\,/\,\tau_1,...,\tau_n]} \mid \tau_1,...,\tau_n \in D(\varphi)\}$

(Matrix: alle Quantifizierungen streichen)

Beispiel:

$E(\forall x\, P(g(x),a) \to Q(f(x,b))) =$

$\{P(g(a),a) \to Q(f(a,b)), P(g(b),a) \to Q(f(b,b)), \text{usw.}\}$

Algorithmus GRes (Grundresolution)

Sei φ eine geschlossene Formel in Klauselnormalform.
Sei $E(\varphi) = \{\rho_1, \rho_2 ...\}$ die Herbrand-Expansion (ab-/aufzählbar!).

- $M := \varnothing$.
- Für $i := 1,2,...$,so lange bis $\varnothing \in M$:
  $M :=$ Resolventenmenge$(M \cup \rho_i)$.

… entdeckt jeden Widerspruch zwischen endlich vielen Elementen von $E(\varphi)$.

… und das reicht bei unerfüllbaren Formeln! …

Satz von Herbrand:
Grundresolution ist Halbentscheidungsverfahren

Der Algorithmus terminiert für die Formel φ genau dann, wenn φ unerfüllbar ist.

Das Verfahren ist sehr langwierig und bei Erfüllbarkeit ohnehin fruchtlos. Wir werden es durch die Technik der **Unifikation** beschleunigen.

Beispiel

Kl-NF-Formel –

$\varphi = \forall x[P(x) \wedge P(g(a,b,c)) \wedge \neg P(g(c, f(c), f(f(c))))]$

Die Formel ist unerfüllbar, denn für

sub = [x / $g(c, f(c), f(f(c)))$]

ergibt sich sowohl $P(g(c, f(c), f(f(c))))$ als auch $\neg P(g(c, f(c), f(f(c))))$.

Wie löst das eine einfach programmierte GRes? Wie probiert sie systematisch für x das Herbrand-Universum und so die Herbrand-Expansion von φ durch?

0 a b c #=3

1 f(a) f(b) f(c) g(aaa) g(aab) … g(ccc) #=30

2 f(f(a)) … f(g(ccc)) g(aaf(a)) … g(g(ccc) g(ccc) g(ccc)) #=35910

3 f(f(f(a))) … g(g(g(ccc),…),…) #=ca $3 \cdot 10^{13}$

4 f(f(f(f(a)))) … f(g(cf(c)f(f(c)))) wievielter Term? ☹

Methodenskizze des Beispiels:

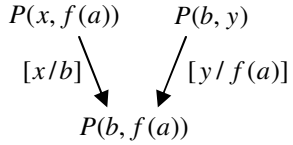
0 Funktionsanwendungen ergeben „nullte“ Teilliste – die Konstanten. Für i =1,2,…:

i -te Teilliste dazu, für maximal i Funktionsanwendungen übereinander, d.h. für jede vorkommende Funktion systematisch:

Anwenden auf alle möglichen Argumentkombinationen, gebildet aus der bisherigen Liste, darunter mindestens ein Argument aus (i–1)-ter Teilliste. Alle diese Terme zur i-ten Teilliste hinzuschreiben.

Unifikation

Ziel: Die Variablen in den Termen in mehreren atomaren PL1-Formeln so substituieren, dass die Formeln identisch werden.



Nicht unifizierbar sind...

- $P(x, f(a))$ und $Q(b, y)$
- $P(x, f(a))$ und $P(b, f(x))$
- $P(x, x)$ und $P(y, f(y))$

Ein **allgemeinster Unifikator** macht *keine überflüssigen Festlegungen* durch Konstanten; jeden Unifikator erhält man aus ihm durch weitere Substitution.

Beispiel: Für $P(x, f(a), z)$ und $P(b, y, z)$ ist

- $[x,y,z\,/\,b,f(a),c]$ zwar ein Unifikator mit „Unifikat“ $P(b, f(a), c)$, aber nur
- $[x,y\,/\,b,f(a)]$ ist der allgemeinste Unifikator, mit Unifikat $P(b, f(a), z)$. Er liefert mit $[z/c]$ den vorigen.

Satz:
Der **Unifikationsalgorithmus** unten terminiert, entscheidet, ob die Literalemenge unifizierbar ist und gibt, falls ja, den allgemeinsten Unifikator aus.

Unifikationsalgorithmus

Geg.: $L = \{L_1,...,L_n\}, n \geq 1$, alle L_i PL1-Literale (PL1-Term oder Negation davon)

1. Literalarbeitsmenge $M := L$, Substitution $s := [\ /\ ]$ (leer)
2. IF M einelementig THEN {Ausgabe := s; STOP} ELSE continue
3. Bestimme $L_1, L_2 \in$ M mit $L_1 \neq L_2$; z_1, z_2 seien die ersten in L_1, L_2 voneinander verschiedenen Zeichen.
4. IF weder z_1 noch z_2 Variablenname THEN {Ausgabe := "geht nicht"; STOP}
5. O.B.d.A. sei z_1 Variablenname; und t der Teilterm von L_2, der mit z_2 beginnt. IF z_1 kommt in t vor THEN {Ausgabe := "geht nicht"; STOP}
6. $s := s \circ [z_1 / t]$; $M := \{q_{[z_1/t]} \mid q \in M\}$; GOTO 2.

Achtung: eigentlich Zeichen*vorkommen*, nicht Zeichen,
$\circ$ ist Hintereinanderausführung von Substitutionen.

Unifikation – Beispiel

L = { P(f(x),w,f(a)), P(f(w),w,f(z)), P(f(g(z)),x,y) }
s=[/]
M = { P(f(x),w,f(a)), P(f(w),w,f(z)), P(f(g(z)),x,y) }
P(f(**x**),w,f(a)) ← L_1
P(f(**w**),w,f(z)) ← L_2
s=[x/w]
M = { P(f(w),w,f(a)), P(f(w),w,f(z)), P(f(g(z)),w,y) }
P(f(w),w,f(**a**))
P(f(w),w,f(**z**))
s=[x/w] [z/a], M = { P(f(w),w,f(a)), P(f(g(a)),w,y) }
P(f(**w**),w,f(a))
P(f(**g**(a)),w,y)
s=[x/w] [z/a] [w/g(a)]
M = { P(f(g(z)),g(z),f(a)), P(f(g(z)),g(z),y) }
P(f(g(a)),g(a),**f**(a))
P(f(g(a)),g(a),**y**)
s=[x/w] [z/a] [w/g(a)] [y/f(a)], M={ P(f(g(a)),g(a),f(a)) }

Ohne die Variablentrennung der Klauseln würden uns einige Resolutionsmöglichkeiten verloren gehen – z.B. gleich im vorigen Beispiel! Damit wir nicht bei jeder Suche nach Unifikationsmöglichkeiten die Klauseln erneut **variablenfremd** machen müssen, erledigen wir dies am besten einmal **vorab**, indem wir zur KNF-Formel eine (erfüllbarkeits-) äquivalente **auseinanderbenannte KNF-Formel** (**ABKNF**) erzeugen.

Im vorigen Beispiel:

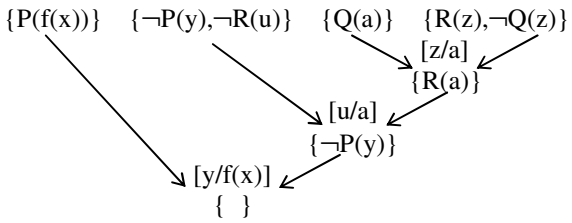
$$\forall x \forall z\, [(P(f(x)) \vee \neg Q(z) \vee P(z)) \wedge (\neg P(x) \vee R(g(x),a))]$$
$$\equiv \forall x \forall z\, (P(f(x)) \vee \neg Q(z) \vee P(z)) \wedge \forall x \forall z\, (\neg P(x) \vee R(g(x),a))$$
$$\equiv \forall x \forall z\, (P(f(x)) \vee \neg Q(z) \vee P(z)) \wedge \forall y \forall z\, (\neg P(y) \vee R(g(y),a))$$
$$\equiv \forall x \forall y \forall z\, (P(f(x)) \vee \neg Q(z) \vee P(z)) \wedge \forall x \forall y \forall z\, (\neg P(y) \vee R(g(y),a))$$
$$\equiv \forall x \forall y \forall z\, [(P(f(x)) \vee \neg Q(z) \vee P(z)) \wedge (\neg P(y) \vee R(g(y),a))]$$

Dann müssen wir für jeden Resolutionsschritt nur noch eine passende Unifikation suchen, **ohne umzubenennen**.

Resolutionssatz: Eine PL1-Formel ist genau dann **unerfüllbar**, wenn aus ihrer Klauselnormalform per PL1-**Resolution** die **leere Klausel** erzeugt werden kann.

Beispiel

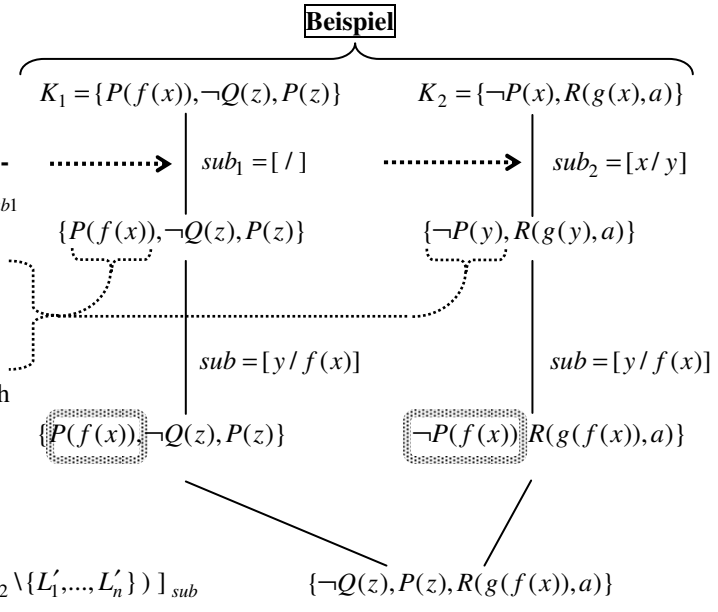
Gegeben sei – bereits in (auseinanderbenannter) Kl-NF:
{ {P(f(x))}, {¬P(y),¬R(z)}, {Q(a)}, {R(u),¬Q(u)} }
... unerfüllbar wegen ...



PL1-Resolutionsschritt, informell:

K_1 und K_2 seien Klauseln aus PL1-Literalen (d.h. atomare Formel oder Negation davon) in Mengenform.
Es gebe Variablen-**Umbenennungen** sub_1 und sub_2, so dass $K_{1\,sub1}$ und $K_{2\,sub2}$ variablenfremd.
Es gebe **Literalmengen** $\{L_1,...,L_m\} \subseteq K_{1\,sub1}$ und $\{L_1',...,L_n'\} \subseteq K_{2\,sub2}$, so dass $\{L_1,...,L_m,\ L_1',...,L_n'\}$ widerspruchlich unifizierbar mit allgemeinstem Unifikator *sub*.

Beispiel



Resolvente

$[\ (K_{1\,sub1} \setminus \{L_1,...,L_m\}) \cup (K_{2\,sub2} \setminus \{L_1',...,L_n'\})\ ]_{sub}$

Es geht übrigens auch anders – mit denselben K_1 und K_2!