

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 angemockert 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 $[c, d]$:= $b - a > d - c$;
- **falsch**, wenn Objekte := Münzen in meiner Geldbörse, Größe \equiv Durchmesser .

Die Syntax der Prädikatenlogik erster Stufe (kurz: PL1)

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

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

- eine abzählbare Menge **VS** von (**Objekt-)** **Variablen** **symbolen**, hier x_1, x_2, \dots – informell u, v, w, x, y, z ; *Mitarbeiter* usw.
- eine abzählbare Menge **KS** von (**Objekt-)** **Konstantensymbolen**, hier a_1, a_2, \dots – informell a, b, c, \dots *Besitzer* usw.

- eine abzählbare Menge **FS** von **Funktionssymbolen**, hier f_1, f_2, \dots – informell f, g, h, \dots , *Chef*, usw. – incl. Abbildung (**F-)****Stelligkeit** : $FS \rightarrow \mathbb{N}_0$
- eine abzählbare Menge **PS** von **Prädikat-(Relations-) symbolen**, hier P_1, P_2, \dots – informell P, Q, R, \dots , *arbeitet_für*, usw. – incl. Abbildung (**P-)****Stelligkeit** : $PS \rightarrow \mathbb{N}_0$
- **Junktoren** $\neg \wedge \vee \rightarrow \leftrightarrow$
- **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, \dots, t_k Terme, so ist $f(t_1, \dots, t_k)$ ein Term.
 Objektkonstanten \equiv „nullstellige“ Funktionen


Die Aussagen der PL1 heißen **Formeln**. PL1-Formeln sind induktiv definiert:

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

Teilformeln, **Teilterme** von Formeln

Induktion/Rekursion über Formelaufbau } analog AL
 Formel- und/oder Termbäume

PL1 in **natürlichsprachlichen** Aussagen, **erste Beispiele**

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

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)

Jetzt also **ausführlichere Beispiele**:

- **DEINE MEINUNG INTERESSIERT MICH NICHT.**
Grundmenge: Personen + persönl. Einstellungen
Konstanten: *ich, du* Personen
Funktionen: *m(x)* Meinung von *x* (Definitions- und Bildbereiche?!)
Relationen: *Int(x,y)* Einstellung *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 ≠ i* bedeutet $\neg =(x,i)$
- **ALLE SCHADENFROHEN LACHEN MICH AUS.**
Prädikate: *Sf(x)* – *x* ist schadenfroh
→ $\forall x (Sf(x) \rightarrow Laus(x,i))$

→ **Grundmuster:**

Für alle mit der Eigenschaft *E* gilt ...

$$\forall x (E(x) \rightarrow \dots)$$

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

→ **Grundmuster:**

Für mindestens eines mit Eigenschaft *E* gilt ...

$$\exists x (E(x) \wedge \dots)$$

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

$$(Q(a) \vee \neg R(f(b), c)) \wedge P(a, b) \rightarrow (Q_a \vee \neg R_f_b_c) \wedge P_a_b$$

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*:

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

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

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

PL1-Varianten

Andere Schreibweisen

– **Quantoren**

$$\forall x F \text{ auch so: } \bigwedge_x F, \quad \forall x: F, (\forall x)F$$

$$\exists x F \text{ auch so: } \bigvee_x F, \quad \exists x: F, (\exists x)F$$

– **Relationen:** auch Infix, Mixfix.

– **Relationen- und Funktionensymbole:** $f_1^k, f_2^k, \dots, P_1^k, P_2^k, \dots$ (pro Stelligkeit *k*)
(auch nullstellige, also konstante Terme/Formeln)

Andere Syntax und Bedeutung (Auswahl)

- PL1 mit **Identität** (PL1=):
Vordefinierte zweistellige (Äquivalenz-)Relation =, ≡ oder ≈.
- PL1 mit **Sorten**:

Objekte sind oft unterschiedlicher Art, z.B. Zahlen, 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

$$\forall x(M(x) \rightarrow \exists y(N(y) \wedge P(x,y)))$$

- PL1 **ohne** besondere **Funktionensymbole**: In der Tat sind Funktionen spezielle Relationen. Nachteil: Man muss Linkstotalität und Rechtseindeutigkeit geeignet explizit „mitnehmen.“

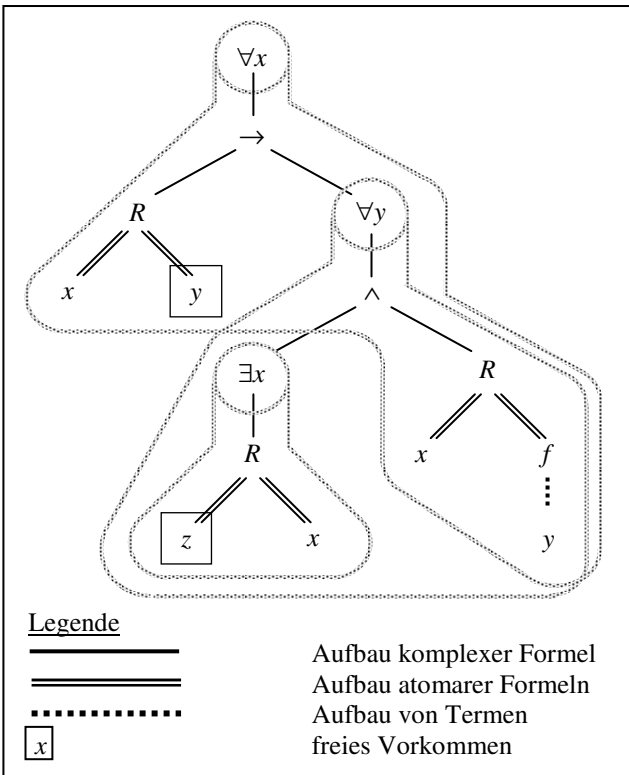
Freie und gebundene Variablen

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

Im Beispiel

$\forall x(R(x, y) \rightarrow \forall y(\exists xR(z, x)$
sind „unterhalb“ $[\forall x]$
alle $[x]$ gebunden, aber
nicht alle durch dieses
 $[\forall x]$: teilweise wird die-
se Bindung von der Bin-
dung durch $[\exists x]$ abge-
löst. 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. | |



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

$I_F : FS_A \rightarrow \bigcup_{k=1}^{\infty} F(U^k, U), FS_A \subseteq FS$,

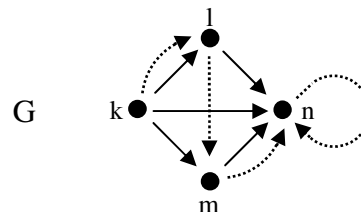
von **Funktionssymbolen** auf ein- oder mehrstellige Funktionen auf U , derart dass k -stellige Symbole auf k -stellige Funktionen abgebildet werden;

$I_P : PS_A \rightarrow \mathbf{P}(\bigcup_{k=0}^{\infty} U^k), PS_A \subseteq PS$,

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 \perp leeren \top Träger, $U = \emptyset$, zulässt. (Achtung – andere Gesetze: Wann gilt $(\forall x F) \rightarrow (\exists x F)$?)

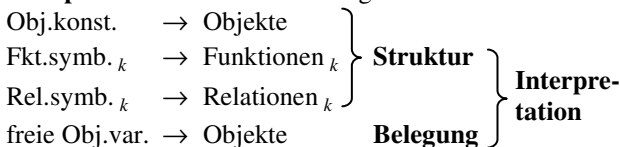
Beispiel: Die Struktur A_{Bsp} sei gegeben durch:



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

PL1-Semantik

Interpretationen sind Zuordnungen ...



Gebundene Variablenvorkommen sind von einer Interpretation „nicht betroffen“, sie werden bei der *Auswertung* (s.u.) verwendet (und dabei systematisch „mit allen möglichen Objekten“ belegt).

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

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, \dots, t_k Terme:

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

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

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

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

Eine **Interpretation** $I = (A, I_V) = (\text{Struktur}, \text{Belegung})$ 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, \dots, t_k Terme:

$$I(f(t_1, \dots, t_k)) := I_F(f)(I(t_1), \dots, I(t_k))$$

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

atomare Formeln:

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

„Junktorenformeln“:

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

„Quantorenformeln“:

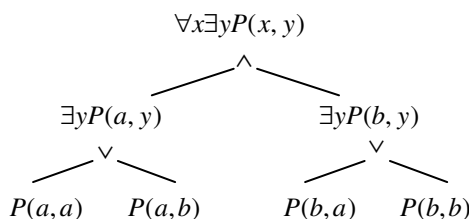
$$I(\forall x\varphi) := \begin{cases} W & \text{wenn für alle } a \in U \text{ gilt : } I_{x:=a}(\varphi) = W \\ F & \text{sonst} \end{cases}$$

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

$I_{x:=a}$: wie I , aber $I_V(x) := a$ – egal ob und wie 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 quantorfreen 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 angetoßen, 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 $I(\varphi) = W$.

Schreibweise: $I \models \varphi$ – Redeweise: φ **gilt unter I** .

Eine Formel φ heißt

- **erfüllbar**, wenn sie mindestens **ein** Modell hat (sonst: **unerfüllbar**);
- **allgemeingültig** (logisch wahr, $\models \varphi$) wenn **jede** passende Interpretation Modell ist (sonst: **widerlegbar**);
- **gültig in A** ($A \models \varphi$), 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. Formelmengemenge M), geschrieben: $\sigma \models \varphi$ ($M \models \varphi$), 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 .

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

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

$$\psi \rightarrow \forall x \varphi \equiv \forall x (\psi \rightarrow \varphi) \qquad \psi \rightarrow \exists x \varphi \equiv \exists x (\psi \rightarrow \varphi)$$

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

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 ?)

Substitution

Satz über AL-Tautologien 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,

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

ersetzt keine Aussagen-, sondern *Objektvariablen*, nämlich paarweise verschiedene Objektvariablen x_1, \dots, x_n durch (nicht unbedingt verschiedene) Terme τ_1, \dots, τ_n .

Termebene:

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

$$f \in FS \quad f(\omega_1, \dots, \omega_k)_{[sub]} := f(\omega_{[sub]_1}, \dots, \omega_{[sub]_k})$$

Übergang zur Aussagenebene:

$$R \in PS \quad R(\omega_1, \dots, \omega_k)_{[sub]} := R(\omega_{[sub]_1}, \dots, \omega_{[sub]_k})$$

Aussagenebene:

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

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

Bisher alles trivial, doch nun ...

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

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

$sub_{\neq x}$ 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:

$$\bullet (\forall x \neg \forall y x = y) \rightarrow (\neg \forall y y = y) \quad ??$$

$$\bullet (\forall x \exists y x < y) \rightarrow (\exists y y < y) \quad ??$$

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

Satz über die Substituierbarkeit

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 „ τ substituierbar für x_i in PL1-Formel φ ?“ bzw. „ $\text{free}(\tau, x_i, \varphi)$?“

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 $\rightarrow \tau$ 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 $\rightarrow \tau$ ist **frei** für x_i in φ & **Stop**.
 - b. Für alle diese $[Qx_k]$:
Kommt x_k in τ vor \rightarrow
 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)
 $\rightarrow \tau$ 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 $\rightarrow \tau$ frei für alle Variablen in allen Formeln.
- keine Variable in τ gebunden in $\varphi \rightarrow \tau$ frei für alle Variablen in φ .
- x_i ist frei für x_i in (jedem) φ .
- φ enthält kein freies $[x_i] \rightarrow$ (jedes) τ ist frei für x_i in φ

Substitutionssatz

$$\text{free}(\tau, x, \varphi) \Rightarrow \forall x \varphi \rightarrow \varphi_{[x/\tau]}$$

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 abzählbaren Menge) als eine einzige Zahl; dadurch sind die TM alle abzählbar T_1, T_2, \dots

(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** $:\Leftrightarrow$

- Falls korrekte Antwort (für alle) JA \Rightarrow Alg endet irgendwann mit JA;
- falls nicht \Rightarrow 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 $MP = \frac{\varphi, \varphi \rightarrow \psi}{\psi}$

Axiome(nschemata)

- ein AL-Axiomensystem **Aussagenlogik AL**

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

- $\varphi \rightarrow \varphi_{[A_1, \dots, A_k / \varphi_1, \dots, \varphi_k]}$ **AL-Gültigkeit AL-G**
wobei φ AL-Formel, φ_i PL1-Formeln
- $(\forall x\varphi) \rightarrow \varphi_{[x/\tau]}$ **Substit.-satz, Spezialisierung SP**
sofern τ f. x in φ erlaubt ist, d.h. $free(\tau, x, \varphi)$
- $\varphi \rightarrow (\forall x\varphi)$ **Generalisierung GN**
sofern x nicht frei in φ vorkommt
- $\forall x(\varphi \rightarrow \psi) \rightarrow (\forall x\varphi \rightarrow \forall x\psi)$ **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

$M \cup \{\varphi\} \vdash \psi \Rightarrow M \vdash (\varphi \rightarrow \psi)$

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

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

$$Sp_+ = \frac{\forall x \varphi}{\varphi_{[x/\tau]}} \quad \text{und} \quad Sp_- = \frac{\neg \exists x \varphi}{\neg \varphi_{[x/\tau]}}$$

Falls c Konstantensymbol, das im Zweig neu ist:

Witness (Zeuge)

$$W_+ = \frac{\exists x \varphi}{\varphi_{[x/c]}} \quad \text{und} \quad W_- = \frac{\neg \forall x \varphi}{\neg \varphi_{[x/c]}}$$

Anwendungsbeispiel:

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

Wir widerlegen die Gegenbehauptung:

| | | | | |
|----|--|--------|-----------|---|
| 1 | $\neg[\forall x(P(x) \vee Q(x)) \rightarrow (\exists x P(x) \vee \forall x Q(x))]$ | (Regel | Prämisse) | |
| 2 | $\forall x(P(x) \vee Q(x))$ | (TR1) | | 1 |
| 3 | $\neg(\exists x P(x) \vee \forall x Q(x))$ | (TR1) | | 1 |
| 4 | $\neg \exists x P(x)$ | (TR3) | | 3 |
| 5 | $\neg \forall x Q(x)$ | (TR3) | | 3 |
| 6 | $\neg Q(c)$ ζ | 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 Widerspruch 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-Werkzeugkasten

für geschl. Formeln

Zusätzliche Ableitungsregeln

| | |
|----------------------------------|--|
| All-Beseitigung, Spezialisierung | $Sp \frac{\forall x \varphi}{\varphi_{[x/\tau]}}$ (τ variabelnfrei) |
| Existenz-Einführung | $EE \frac{\varphi_{[x/\tau]}}{\exists x \varphi}$ (τ variabelnfrei) |
| Existenz-Beseitigung | $EB \frac{\exists x \varphi}{\varphi_{[x/c]}}$ (c neue Konst.) ¹ |

| | |
|----------------------------|---|
| Nicht-Existenz-Beseitigung | $NEB \frac{\neg \exists x \varphi}{\forall x \neg \varphi}$ |
| Nicht-All-Beseitigung | $NAB \frac{\neg \forall x \varphi}{\exists x \neg \varphi}$ |

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)
 | | :
 | $\varphi_{[x/c]}$
 $\forall x \varphi$ (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)) | \perp |
| $\exists x \varphi$ | $\exists x \varphi$ (IB) |

Beweis-Bsp. $\{\forall x(P(x) \rightarrow Q(x)), \neg Q(a)\} \models \neg \forall y P(y)$:

| | |
|--|----------|
| (1) $\forall x(P(x) \rightarrow Q(x))$ | Geg. |
| (2) $\neg Q(a)$ | Geg. |
| Zeige $\neg \forall y P(y)$ | |
| (3) $\forall y P(y)$ | Ann. |
| (4) $P(a)$ | 3, Sp |
| (5) $P(a) \rightarrow Q(a)$ | 1, Sp |
| (6) $Q(a)$ | 4, 5, MP |
| (7) \perp | 2, 6, WE |
| (8) $\neg \forall y P(y)$ | 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. $P(x) \vee (\forall x (\exists x Q(x,x) \wedge R(x)) \vee S(x))$

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

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

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

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

NEIN \rightarrow

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

3. Gehe nach 1.

$$\dots P(x) \vee (\forall x (\exists x Q(x,x) \wedge R(x)) \vee S(x)) \text{ usw.}$$

Algorithmus BPNF: 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:
 - $\neg \forall x \varphi \rightarrow \exists x \neg \varphi$
 - $\neg \exists x \varphi \rightarrow \forall x \neg \varphi$
 - $\neg(\varphi \wedge \vee \psi) \rightarrow (\neg \varphi \vee \wedge \neg \psi)$
[von \wedge nach \vee oder umgekehrt]
 - $\neg \neg \varphi \rightarrow \varphi$
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 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.

$\rightarrow \forall x P(x, f(x))$ – Weg ist der Existenzquantor! Doch was haben wir eigentlich erhalten?

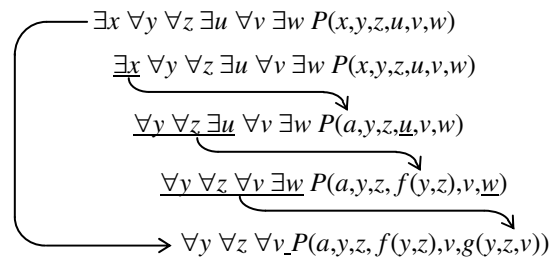
Satz: Skolem-Form

- (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“

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

Skolemisierungs-Beispiel



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 $y_1, \dots, y_n \rightarrow$

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

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 $y_i \rightarrow$

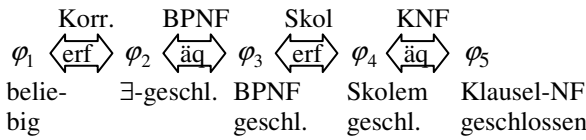
Korollar: Jede PL1-Formel φ mit den freien Variablen y_1, \dots, y_n ist allgemeingültigkeitsäquivalent zu der geschlossenen Formel $\forall y_1 \dots \forall y_n \varphi$.

Klauselnormalform

Eine PL1-Formel in Skolem-Form ist in **Klauselnormalform**, 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.



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

Beispiel

Formel $\forall y \exists z P(x, y, z) \rightarrow \neg \exists y \forall v [Q(x, v) \rightarrow R(x, y)]$
 abg. $\exists x \{ \forall y \exists z P(x, y, z) \rightarrow \neg \exists y \forall v [Q(x, v) \rightarrow R(x, y)] \}$
 berein. $\exists x \{ \forall y \exists z P(x, y, z) \rightarrow \neg \exists u \forall v [Q(x, v) \rightarrow R(x, u)] \}$
 \rightarrow 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 $\{ \neg P(a, b, z), Q(a, f(z, u)) \}, \{ \neg P(a, b, z), \neg R(a, u) \}$

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 $\tau_1, \dots, \tau_n \in D(\varphi) \Rightarrow f(\tau_1, \dots, \tau_n) \in D(\varphi)$.

Beispiel:

$D(\forall x P(g(x), a) \rightarrow Q(f(x, b))) = \{ a, b, g(a), g(b), f(a, a), \dots, g(f(a, g(b))), \dots \}$

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

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

Beispiel:

$E(\forall x P(g(x), a) \rightarrow Q(f(x, b))) = \{ P(g(a), a) \rightarrow Q(f(a, b)), P(g(b), a) \rightarrow Q(f(b, b)), \text{ usw. } \}$

Algorithmus GRes (Grundresolution)

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

- $M := \emptyset$.
- Für $i := 1, 2, \dots$,so lange bis $\emptyset \in M$:
 $M := \text{Resolventenmenge}(M \cup \rho_i)$.

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. Es wird durch die Technik der **Unifikation** beschleunigt werden.

Beispiel

Kl-NF-Formel – Beispiel:
 $\varphi = \forall x [P(x) \wedge P(g(a, b, c)) \wedge \neg P(g(c, f(c), f(f(c))))]$

Die Formel ist unerfüllbar, denn für $\text{sub} = [x / g(c, f(c), f(f(c)))]$ 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 · 10¹³
- 4 f(f(f(f(a)))) ... f(g(cf(c)f(f(c)))) wievielter Term? @

Methodenskizze des Beispiels:

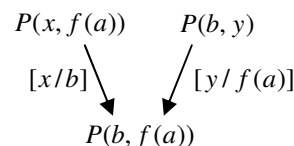
0 Funktionsanwendungen ergeben „nullte“ Teilliste – die Konstanten. Für $i = 1, 2, \dots$:

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

- $[x, y, z / b, f(a), c]$ ein Unifikator mit „Unifikat“ $P(b, f(a), c)$, aber nur
- $[x, y / b, f(a)]$ der allgemeinste Unifikator, mit Unifikat $P(b, f(a), z)$.

Satz:

Der **Unifikationsalgorithmus** (nächste Seite) terminiert, entscheidet, ob die Literalemengung unifizierbar ist und gibt, falls ja, den allgemeinsten Unifikator aus.

Unifikationsalgorithmus

Geg.: $L = \{L_1, \dots, L_n\}$, $n \geq 1$, 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 Zeichenvorkommen nicht Zeichen, \circ ist Hintereinanderausführung von Substitutionen.

Unifikation – Beispiel

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

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:

$$\begin{aligned} & \forall x \forall z [(P(f(x)) \vee \neg Q(z) \vee P(z)) \wedge (\neg P(x) \vee R(g(x), a))] \\ & \equiv \forall x \forall z (P(f(x)) \vee \neg Q(z) \vee P(z)) \wedge \forall x \forall z (\neg P(x) \vee R(g(x), a)) \\ & \equiv \forall x \forall z (P(f(x)) \vee \neg Q(z) \vee P(z)) \wedge \forall y \forall z (\neg P(y) \vee R(g(y), a)) \\ & \equiv \forall x \forall y \forall z (P(f(x)) \vee \neg Q(z) \vee P(z)) \wedge \\ & \quad \forall x \forall y \forall z (\neg P(y) \vee R(g(y), a)) \\ & \equiv \forall x \forall y \forall z [(P(f(x)) \vee \neg Q(z) \vee P(z)) \wedge (\neg P(y) \vee R(g(y), a))] \end{aligned}$$

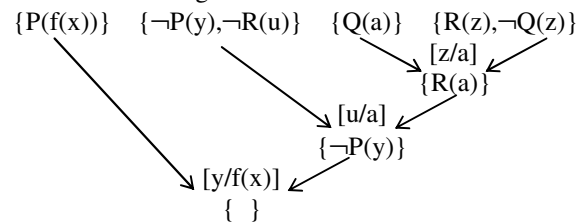
Dann müssen wir für jeden Resolutionsschritt nur noch eine passende Unifikation suchen, **ohne umzubenennen**.

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) KI-NF: $\{ \{P(f(x))\}, \{\neg P(y), \neg R(z)\}, \{Q(a)\}, \{R(u), \neg Q(u)\} \}$

... 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 $K_{1\ sub_1}$ und $K_{2\ sub_2}$ variablenfremd.

Es gebe **Literalmengen** $\{L_1, \dots, L_m\} \subseteq K_{1\ sub_1}$ und $\{L'_1, \dots, L'_n\} \subseteq K_{2\ sub_2}$, so dass $\{L_1, \dots, L_m, L'_1, \dots, L'_n\}$ widersprüchlich unifizierbar mit allgemeinstem Unifikator sub .

Resolvente

$$[(K_{1\ sub_1} \setminus \{L_1, \dots, L_m\}) \cup (K_{2\ sub_2} \setminus \{L'_1, \dots, L'_n\})]_{sub}$$

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

