Prädikatenlogik

In der Aussagenlogik ist es nicht möglich, Aussagen über ganze Klassen von Objekten zu machen, so daß Schlußfolgerungen für individuelle Objekte möglich sind.

Es sei gegeben:

> Martin ist ein Informatiker. Peter ist ein Informatiker.
> Jeder Informatiker kann programmieren.

Wir wollen folgern:

> Martin kann programmieren. Peter kann programmieren.

Syntax der Prädikatenlogik

Mit der Prädikatenlogik (1. Stufe) wollen wir Sachverhalte beschreiben, die folgendes enthalten können.

- Objekte, z.B. Personen oder Sachen
- Funktionen auf den Objekten, z.B. Größe, Gewicht, Hochzeitstag
- Eigenschaften von Objekten
- Beziehungen zwischen Objekten
- Aussagen über Objekte, auch quantifizierende

Wie in der Aussagenlogik beschreiben wir zunächst die Syntax der Wissensrepräsentationssprache.

Wir haben in der Prädikatenlogik folgende syntaktischen Elemente:

- Eine *Konstante* repräsentiert ein spezifisches Element aus einer Menge von Objekten. Wir stellen Konstanten durch Bezeichner dar, die mit einem Kleinbuchstaben beginnen.
  Beispiel: $peter, martin$
- Eine *Variable* repräsentiert ein unspezifisches Element aus einer Menge von Objekten. Wir stellen Variablen durch Bezeichner dar, die mit einem Großbuchstaben beginnen.
  Beispiel: $X, Person$
- Eine *Funktion* repräsentiert einen funktionalen Zusammenhang zwischen Objekten. Solche Funktionen haben eine feste Stelligkeit. Wir stellen Funktionen durch kleingeschriebene Bezeichner dar mit den Argumenten in Klammern.
  Beispiel: $alter(martin), f(X, Y, peter)$

- *Prädikate* repräsentieren Eigenschaften oder Beziehungen zwischen Objekten. Solche Prädikate haben eine feste Stelligkeit. Wir stellen Prädikate durch großgeschriebene Bezeichner dar mit den Argumenten in Klammern.

  Beispiel: $\mathrm{Informatiker}(\mathrm{martin}), \mathrm{Programmieren}(\mathrm{X}), \mathrm{Vater}(\mathrm{lars}, \mathrm{peter})$

  Bemerkung: Im Gegensatz zu Funktionen kann einem Prädikatenausdruck ein Wahrheitswert zugeordnet werden.
- Die *logischen Junktoren* wie in der Aussagenlogik: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- *Quantoren*, die mit einer Variable verbunden sind, dienen dazu, die Gültigkeit von Prädikaten zu quantifizieren. Hierzu stehen der Allquantor $\forall$ und der Existenzquantor $\exists$ zur Verfügung.

  Beispiel: $\exists \mathrm{X}\ \mathrm{Programmieren}(\mathrm{X})$

Prädikatenlogische Signatur

Definition 3.12. Eine *(PL1-)Signatur* $\Sigma = (\mathtt{Func}, \mathtt{Pred})$ besteht aus

- einer Menge $\mathtt{Func}$ von *Funktionssymbolen* und
- einer Menge $\mathtt{Pred}$ von *Prädikatensymbolen*.

Jedes Symbol $s \in \mathtt{Func} \cup \mathtt{Pred}$ hat eine feste *Stelligkeit* ≥ 0.

Ein Funktionssymbol mit der Stelligkeit 0 heißt *Konstante*.

PL1-Terme

Definition 3.13. Es sei V eine Menge von Variablensymbolen und $\Sigma = (\mathtt{Func}, \mathtt{Pred})$ sei eine PL1-Signatur. Dann ist die Menge $\mathtt{Term}_\Sigma(V)$ der *(PL1-)Terme* wie folgt definiert:

1. Jede Konstante $c \in \mathtt{Func}$ ist ein PL1-Term.
2. Jedes Variablensymbol $X \in V$ ist ein PL1-Term.
3. Ist $f \in \mathtt{Func}$ ein n-stelliges Funktionssymbol ($n \geq 1$) und sind $t_1, \ldots, t_n$ PL1-Terme, so ist auch $f(t_1, \ldots, t_n)$ ein PL1-Term.

Beispiel 3.14. $f(g(X), Y), jahreVerheiratet(klaus, marie)$

PL1-Formeln

Definition 3.14. Es sei V eine Menge von Variablensymbolen und $\Sigma = (\mathtt{Func}, \mathtt{Pred})$ sei eine PL1-Signatur. Dann ist die Menge $\mathtt{Formel}_\Sigma(V)$ der *PL1-Formeln* wie folgt definiert:

1. Ist $p \in \mathtt{Pred}$ ein n-stelliges Prädikatensymbol und sind $t_1, \ldots, t_n$ Terme, so ist

$$p(t_1, \ldots, t_n)$$

   eine *(atomare) PL1-Formel*.
2. Sind F und G PL1-Formeln, dann sind auch

$$\neg F, F \wedge G, F \vee G, F \rightarrow G, F \leftrightarrow G$$

   PL1-Formeln.

3. Ist F eine PL1-Formel und $X \in \mathcal{V}$, dann sind auch

$$\exists X\, \mathsf{F} \text{ und } \forall X\, \mathsf{F}$$

PL1-Formeln.

Beispiel 3.15. Den anfangs dargestellten Sachverhalt könnten wir durch folgende Formeln ausdrücken.

$$\begin{aligned}
&\texttt{Informatiker}(martin)\\
&\texttt{Informatiker}(peter)\\
&\forall X\ \texttt{Informatiker}(X) \rightarrow \texttt{Programmieren}(X)
\end{aligned}$$

Die Frage, ob Martin und Peter programmieren können, würde dann als PL1-Formel lauten:

$$\texttt{Programmieren}(martin) \wedge \texttt{Programmieren}(peter)$$

Bemerkung 3.4.

- In PL1 ist nur eine Quantifizierung über Objekt-Variablen (also 0-stellige Funktionen) zulässig. Somit wäre die Formel

$$\exists \mathrm{X}\, \mathrm{X}(\mathrm{peter})$$

  keine PL1-Formel.
- Wir verlangen zusätzlich, daß alle Variablen quantifiziert werden, d.h. eine Formel der Art

$$\mathrm{p}(\mathrm{X}) \vee \mathrm{q}(\mathrm{Y})$$

  wäre verboten. Formeln, die ausschließlich quantifizierte Variablen enthalten, bezeichnet man als *geschlossene Formeln*.

Prädikatenlogik und PROLOG

In der Programmiersprache PROLOG können (gewisse) PL1-Formeln direkt repräsentiert werden.

Beispiel 3.16. Der bekannte Sachverhalt als PROLOG-Programm:

```
programmieren(X) :- informatiker(X).

informatiker( martin ).
informatiker( peter ).
```

An solch ein PROLOG-Programm kann man eine Anfrage stellen:

```
?- programmieren( peter ),programmieren( martin ).
Yes
```

Prädikatenlogische Interpretation

Bisher haben wir wieder nur Syntax. Wir müssen nun die Funktions- und Prädikatensymbole mit einer Bedeutung belegen.

Definition 3.15. Es sei $\Sigma = (\mathtt{Func}, \mathtt{Pred})$ eine PL1-Signatur. Eine *Σ-Interpretation* $I = (U_I, S_I)$ besteht aus:

- einer nichtleeren Menge U_I, der *Trägermenge (Grundbereich, Universum)*,
- sowie einer Abbildung S_I, die jedem n-stelligen Funktionssymbol $f \in \mathtt{Func}$ eine n-stellige Funktion $S_I(f) : U_I^n \to U_I$ zuordnet und
- jedem n-stelligen Prädikatensymbol $p \in \mathtt{Pred}$ eine n-stellige Relation $S_I(p) \subseteq U_I^n$ zuordnet.

Termauswertung

Es sind noch nicht alle sprachlichen Symbole der PL mit einer Bedeutung belegt, es fehlen noch die Variablen.

Definition 3.16. Es sei $I = (U_I, S_I)$ eine Σ-Interpretation und V eine Menge von Variablensymbolen. Dann ist eine Variablenbelegung α eine Funktion $\alpha : V \to U_I$.

Definition 3.17. Gegeben sein ein Term $t \in \mathtt{Term}_\Sigma(V)$, eine Σ-Interpretation $I = (U_I, S_I)$ und eine Variablenbelegung $\alpha : V \to U_I$.

Die *Termauswertung* von t in I unter α ist die wie folgt definierte Funktion $[\![\ ]\!]_{I,\alpha} : \mathtt{Term}_\Sigma(V) \to U_I$:

$$\begin{aligned} [\![X]\!]_{I,\alpha} &= \alpha(X) \text{ für } X \in V \\ [\![f(t_1, \ldots, t_n)]\!]_{I,\alpha} &= S_I(f)([\![t_1]\!]_{I,\alpha}, \ldots, [\![t_n]\!]_{I,\alpha}) \end{aligned}$$

PL1-Semantik

Definition 3.18. Es sei $I = (U_I, S_I)$ eine Σ-Interpretation, V eine Menge von Variablensymbolen und $\alpha : V \longrightarrow U_I$ eine Variablenbelegung.

$\alpha_{X/a} : V \longrightarrow U_I$ bezeichne die Modifikation von α an der Stelle X zu a, d.h:

$$\alpha_{X/a}(t) = \begin{cases} \alpha(t) & \text{für } t \neq X \\ a & \text{für } t = X \end{cases}$$

Dann ist der Wahrheitswert einer Formel $F \in \texttt{Formel}_\Sigma(V)$ in I unter α (geschrieben $[\![F]\!]_{I,\alpha}$) wie folgt definiert:

- Für eine atomare Formel $p(t_1, \ldots, t_n)$ gilt:

$$[\![p(t_1, \ldots, t_n)]\!]_{I,\alpha} \text{ ist wahr} :\Leftrightarrow ([\![t_1]\!]_{I,\alpha}, \ldots, [\![t_n]\!]_{I,\alpha}) \in S_I(p)$$

- Für $F \wedge G, F \vee G, F \rightarrow G$ und $\neg F$ gelten die selben Regeln wie in der Aussagenlogik.
- Für eine Formel $\forall X\, F$ gilt:

$$[\![\forall X\, F]\!]_{I,\alpha} \text{ ist wahr} :\Leftrightarrow \text{für jedes } a \in U_I \text{ gilt:} [\![F]\!]_{I,\alpha_{X/a}} \text{ ist wahr}$$

- Für eine Formel $\exists X\, F$ gilt:

$$[\![\exists X\, F]\!]_{I,\alpha} \text{ ist wahr} :\Leftrightarrow \text{es gibt ein } a \in U_I \text{ mit:} [\![F]\!]_{I,\alpha_{X/a}} \text{ ist wahr}$$

Bemerkung 3.5.

- Gemäß Bemerkung 3.4 betrachten wir nur geschlossene Formeln. Dies sind Formeln, in denen alle Variablen gebunden sind.
- Der Wahrheitswert einer geschlossenen Formel ist aber unabhängig von einer Variablenbelegung α.
- Für geschlossene Formeln können wir deshalb auch einfach $[\![\mathrm{F}]\!]_{\mathrm{I}}$ schreiben.
- Die Begriffe *Modell* und *semantische Folgerung* werden analog zur Aussagenlogik definiert.

Beispiel 3.17. Durch die Mengen

$$\begin{aligned} \mathrm{Programmieren} &= \{\mathrm{peter}, \mathrm{martin}\} \text{ und} \\ \mathrm{Informatiker} &= \{\mathrm{peter}, \mathrm{martin}\} \end{aligned}$$

ist ein Modell für unser Beispiel gegeben. Es ist sogar das “kleinste Modell” für diese Formelmenge. Solche Modelle nennt man *Herbrand-Modell*.

Die Interpretation

$$\begin{aligned} \mathrm{Programmieren} &= \{\mathrm{peter}, \mathrm{martin}\} \text{ und} \\ \mathrm{Informatiker} &= \{\mathrm{peter}, \mathrm{martin}, \mathrm{klaus}\} \end{aligned}$$

ist dagegen kein Modell für diese Formelmenge.

Äquivalenzen für PL1-Formeln

Definition 3.19. Zwei PL1-Formeln F und G heißen *semantisch äquivalent* gdw. für alle Σ-Interpretationen I gilt:

$$[\![\mathrm{F}]\!]_{\mathrm{I}} = [\![\mathrm{G}]\!]_{\mathrm{I}}$$

Lemma 3.6.

$$\begin{array}{rcl}
\neg\forall \mathrm{X}\mathrm{F} & \equiv & \exists \mathrm{X}\neg\mathrm{F} \\
\neg\exists \mathrm{X}\mathrm{F} & \equiv & \forall \mathrm{X}\neg\mathrm{F} \\
(\forall \mathrm{X}\mathrm{F}) \wedge (\forall \mathrm{X}\mathrm{G}) & \equiv & \forall \mathrm{X}(\mathrm{F} \wedge \mathrm{G}) \\
(\exists \mathrm{X}\mathrm{F}) \vee (\exists \mathrm{X}\mathrm{G}) & \equiv & \exists \mathrm{X}(\mathrm{F} \vee \mathrm{G}) \\
\forall \mathrm{X}\forall \mathrm{Y}\mathrm{F} & \equiv & \forall \mathrm{Y}\forall \mathrm{X}\mathrm{F} \\
\exists \mathrm{X}\exists \mathrm{Y}\mathrm{F} & \equiv & \exists \mathrm{Y}\exists \mathrm{X}\mathrm{F} \\
\forall \mathrm{X}\mathrm{F} & \equiv & \forall \mathrm{Y}\mathrm{F}[\mathrm{X}/\mathrm{Y}] \\
\exists \mathrm{X}\mathrm{F} & \equiv & \exists \mathrm{Y}\mathrm{F}[\mathrm{X}/\mathrm{Y}]
\end{array}$$

Normalformen

Definition 3.20. Eine Formel F, in der alle Quantoren außen stehen, heißt *Pränexform*.

Eine Pränexform, die als Junktor nur noch Konjunktion, Disjunktion und Negation enthält, wobei die Negation nur unmittelbar vor Atomen auftritt, heißt *vereinigungstechnische Normalform (VNF)*.

Mit folgenden Schritten kann jede PL1-Formel in eine äquivalente Formel in VNF überführt werden:

1. Umbennung von Variablen derart, daß keine Variable sowohl frei als auch gebunden auftritt und hinter allen vorkommenden Quantoren stehen verschiedene Variablen.
2. Beseitigung der Junktoren $\rightarrow$ und $\leftrightarrow$.

3. Die Negation wird ganz nach innen gezogen, so daß sie nur noch unmittelbar vor Atomen auftritt.
4. Die Quantoren werden ganz nach außen geschoben.

Skolemisierung

Skolemisierung dient der Elimination von Existenzquantoren. Wir betrachten eine Pränexform:

$$\forall X_1 \ldots \forall X_k \exists Y \, F(\ldots, Y, \ldots)$$

Zum Ersatz von $\exists Y$ wählen wir ein neues Funktionszeichen f der Stelligkeit k und führen folgende Operation aus:

1. Streiche $\exists Y$ aus der Pränexform.
2. Ersetze in der verbleibenden Formel Y an allen Stellen durch $f(X_1, \ldots, X_k)$.

Wir erhalten somit

$$\forall X_1 \ldots \forall X_k \, F(\ldots, f(X_1, \ldots, X_k), \ldots)$$

Die Funktion f heißt *Skolemfunktion* bzw. für $k = 0$ *Skolemkonstante*.

Normalformen (2)

5. Alle Existenzquantoren werden durch Skolemisierung entfernt.
6. Jetzt können alle Allquantoren entfernt werden.
7. Mit Hilfe der de Morganschen Regeln können disjunktive Normalform DNF bzw. konjunktive Normalform KNF analog zur Aussagenlogik erzeugt werden.

Ebenfalls analog zur Aussagenlogik kann eine PL1-Formel in KNF in Klauselform dargestellt werden.

Substitution

Für die maschinelle Inferenz mit PL wird noch ein Mechanismus zur Instanziierung von Variablen benötigt.

Definition 3.21. Es sei $\mathcal{X} \subseteq V$ eine endliche Teilmenge von Variablensymbolen von V. Dann ist eine Abbildung $\sigma : \mathcal{X} \longrightarrow \mathtt{Term}_\Sigma(V)$ eine *Substitution*.

Durch

$$\sigma(X) = \begin{cases} \sigma(X) & \text{für } X \in \mathcal{X} \\ X & \text{für } X \in V \setminus \mathcal{X} \end{cases}$$

wird der Definitionsbereich von σ zunächst auf V ausgedehnt.

Durch $\sigma(c) = c$ für Konstanten und die rekursive Anwendung von σ in der Form

$$\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$$

erhält man eine Abbildung $\sigma : \mathtt{Term}_\Sigma(V) \longrightarrow \mathtt{Term}_\Sigma(V)$.

Eine Substitution σ geben wir in der Form $\{X_1/t_1, \ldots, X_k/t_k\}$ an.

Unifikation

Die Unifikation ist eine Substitution, bei der Terme gleichgemacht werden.

Definition 3.22. Eine Substitution σ heißt *Unifikator* der Terme s und t gdw. $\sigma(s) = \sigma(t)$ gilt. s und t sind dann *unifizierbar*.

Beispiel 3.18. Sind X, Y, Z Variablensymbole und a, b, c, Konstanten, so sind die Terme $f(X, b)$ und $f(a, c)$ nicht unifizierbar.

Die Terme $f(X, b)$ und $f(a, b)$ sind unifizierbar mit $\sigma = \{X/a\}$.

Die Substitutionen $\sigma = \{X/b, Y/a, Z/g(a, a)\}$ und $\mu = \{X/b, Z/g(a, Y)\}$ sind Unifikatoren für die Terme $f(X, g(a, Y))$ und $f(b, Z)$.

Allgemeinster Unifikator

Definition 3.23. Eine Sustitution σ heißt *allgemeinster Unifikator* für die Terme s und t gdw.

- σ ist Unifikator von s und t und
- Für jeden Unifikator λ von s und t existiert eine Substitution τ, so daß $\lambda = \tau \circ \sigma$ gilt.

Bemerkung 3.6. Es gibt einen Algorithmus, der für zwei Terme s und t entscheidet, ob sie unifizierbar sind und gegebenenfalls einen allgemeinsten Unifikator berechnet.

Gegeben seien zwei Terme s und t. Die Berechung des allgemeinsten Unifikators geschieht wie folgt: Man startet mit $\sigma = \{\}$ als allgemeinsten Unifikator und wendet sukzessive die folgenden Regeln an:

(a) Sind s und t Konstanten, so sind sie unifizierbar gdw. s gleich t ist.

(b) Ist s eine Variable und t eine Konstante, so sind s und t unifizierbar. Man erweitere hierzu σ um $\{s/t\}$.

(c) Ist s eine Variable und t ein Term der Form $f(t_1, \ldots, t_n)$, so sind sie unifizierbar gdw. s nicht in t vorkommt. Man erweitere hierzu σ um $\{s/t\}$.

(d) Sind s und t zusammengesetzte Terme, so sind sie unifizierbar gdw. $s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$ und wenn jeweils s_i und t_i unifizierbar sind. Man erweitere σ um die Substitutionen, die sich aus der Unifikation der s_i und t_i ergeben.

Beispiel 3.19. Es seien X, Y, Z Variablensymbole.

Sind die Terme $f(X, h(Y), Y)$ und $f(g(Z), Z, a)$ unifizierbar?

Regel	Ungelöste Unifikationen	σ
	$f(X, h(Y), Y) ? f(g(Z), Z, a)$	$\{\}$
(d)	$X ? g(Z), h(Y) ? Z, Y ? a$	$\{\}$
(c)	$h(Y) ? Z, Y ? a$	$\{X/g(Z)\}$
(c)	$Y ? a$	$\{X/g(h(Y)), Z/h(Y)\}$
(b)		$\{X/g(h(a)), Z/h(a), Y/a\}$

Resolution in PL1

Bemerkung 3.7. Die Begriffe *Klausel*, *Klauselform* und *Literal* werden in PL1 analog zur Aussagenlogik definiert.

Bemerkung 3.8. Wir dehnen die Unifikation auf atomare PL1-Formeln aus.

Zwei atomare Formeln $P(s_1, \ldots, s_n)$ und $P(t_1, \ldots, t_n)$ sind unifizierbar gdw. s_i und t_i für $i = 1, \ldots, n$ unifizierbar sind.

Definition 3.24. Es seien K_1, K_2 PL1-Klauseln. Die Klausel R heißt *PL1-Resolvente* von K_1 und K_2 gdw. folgendes gilt:

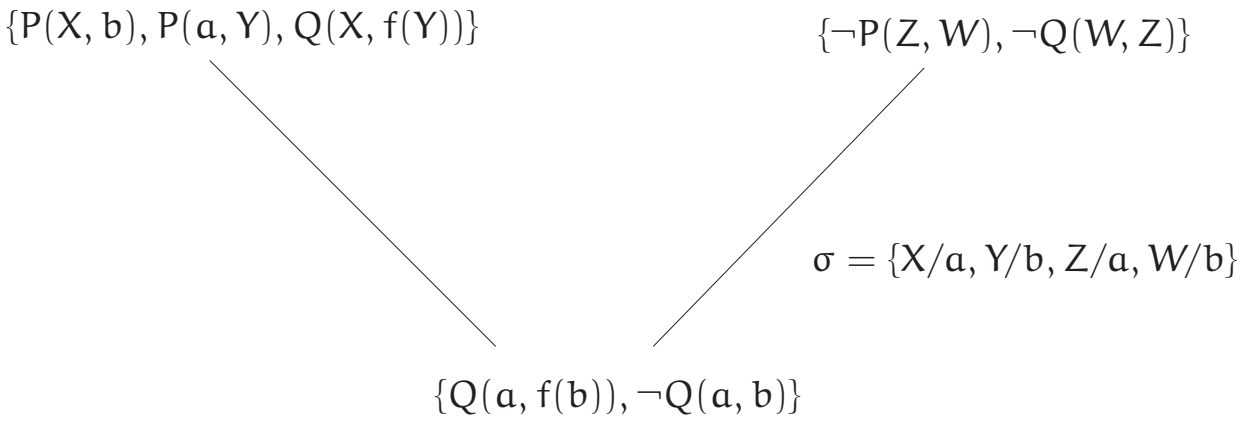
(a) K_1 und K_2 haben keine gemeinsamen Variablen.

(b) Es gibt positive Literale $A_1, \ldots, A_k \in K_1$, ein negatives Literal $\neg A \in K_2$ und einen allgemeinsten Unifikator σ von $A, A_1, \ldots A_k$.

(c) R hat die folgende Form:

$$\sigma((K_1 \setminus \{A_1, \ldots, A_k\}) \cup (K_2 \setminus \{\neg A\}))$$

Beispiel 3.20. Darstellung der Resolution für PL1:



Beispiel 3.21. Überführung einer PL1-Formel in Klauselform.

$$\forall X((\forall Y P(X,Y)) \rightarrow \neg(\forall Y Q(X,Y) \rightarrow R(X,Y)))$$

Für den zweiten an Y gebundenen Allquantor ersetzen wir Y durch Z:

$$\forall X((\forall Y P(X,Y)) \rightarrow \neg(\forall Z Q(X,Z) \rightarrow R(X,Z)))$$

Der Junktor $\rightarrow$ wird eliminiert:

$$\forall X(\neg(\forall Y P(X,Y)) \vee \neg(\forall Z \neg Q(X,Z) \vee R(X,Z)))$$

Die Negationen werden ganz nach innen gezogen:

$$\forall X(\exists Y \neg P(X,Y)) \vee (\exists Z Q(X,Z) \wedge \neg R(X,Z)))$$

Quantoren nach aussen ziehen:

$$\forall X \exists Y \exists Z(\neg P(X,Y) \vee (Q(X,Z) \wedge \neg R(X,Z)))$$

Eliminierung von $\exists Y$ durch Skolemfunktion f:

$$\forall X \exists Z(\neg P(X, f(X)) \vee (Q(X, Z) \wedge \neg R(X, Z)))$$

Eliminierung von $\exists Z$ durch Skolemfunktion g:

$$\forall X(\neg P(X, f(X)) \vee (Q(X, g(X)) \wedge \neg R(X, g(X)))$$

Jetzt kann man den Quantor weglassen:

$$\neg P(X, f(X)) \vee (Q(X, g(X)) \wedge \neg R(X, g(X))$$

De Morgan liefert:

$$(\neg P(X, f(X)) \vee Q(X, g(X))) \wedge (\neg P(X, f(X)) \vee \neg R(X, g(X)))$$

Darstellung als zwei Klauseln:

$$\{\neg P(X, f(X)), Q(X, g(X))\} \text{ und } \{\neg P(X, f(X)), \neg R(X, g(X))\}$$

Umbenennung der Variablen so, daß eine Variable in höchstens einer Klausel auftritt:

$$\{\neg P(X, f(X)), Q(X, g(X))\} \text{ und } \{\neg P(Y, f(Y)), \neg R(Y, g(Y))\}$$

Beispiel 3.22. Resolution in PL1:

1. Es gibt einen Patienten, der alle Ärzte mag:

$$\exists X Patient(X) \wedge \forall Y(Arzt(Y) \rightarrow Mag(X, Y))$$

2. Kein Patient mag einen Quacksalber:

$$\forall X Patient(X) \rightarrow \forall Y(Quaksalber(Y) \rightarrow \neg Mag(X, Y))$$

Daraus wollen wir schließen, daß ein Arzt kein Quaksalber ist.

$$\forall Y(Arzt(Y) \rightarrow \neg Quaksalber(Y))$$

Wir formen 1. um:

$$\begin{aligned} & \exists X Patient(X) \wedge \forall Y(Arzt(Y) \rightarrow Mag(X, Y)) \\ \equiv \quad & \exists X Patient(X) \wedge \forall Y(\neg Arzt(Y) \vee Mag(X, Y)) \\ \equiv \quad & \exists X \forall Y(Patient(X) \wedge (\neg Arzt(Y) \vee Mag(X, Y))) \end{aligned}$$

Skolemisierung liefert:

$$\begin{aligned} & \forall Y(Patient(a) \wedge (\neg Arzt(Y) \vee Mag(a, Y))) \\ \equiv \quad & Patient(a) \wedge \forall Y(\neg Arzt(Y) \vee Mag(a, Y)) \end{aligned}$$

Daraus ergeben sich die Klauseln:

$$K_1 : \{Patient(a)\} \text{ und } K_2 : \{\neg Arzt(Y), Mag(a, Y)\}$$

Wir formen 2. um:

$$\begin{aligned} & \forall X Patient(X) \rightarrow \forall Y(Quaksalber(Y) \rightarrow \neg Mag(X, Y)) \\ \equiv \quad & \forall X \neg Patient(X) \vee \forall Y(\neg Quaksalber(Y) \vee \neg Mag(X, Y)) \\ \equiv \quad & \forall X \forall Y \neg Patient(X) \vee \neg Quaksalber(Y) \vee \neg Mag(X, Y) \end{aligned}$$

Daraus ergibt sich mit Umbenennung die Klausel:

$$K_3 : \{\neg Patient(X), \neg Quaksalber(Z), \neg Mag(X, Z)\}$$

Negation der Hypothese:

$$\begin{aligned} & \neg\forall Y(Arzt(Y) \rightarrow \neg Quaksalber(Y)) \\ \equiv \quad & \neg\forall Y(\neg Arzt(Y) \vee \neg Quaksalber(Y)) \\ \equiv \quad & \exists Y\neg(\neg Arzt(Y) \vee \neg Quaksalber(Y)) \\ \equiv \quad & \exists Y(Arzt(Y) \wedge Quaksalber(Y)) \end{aligned}$$

Skolemisierung liefert die beiden Klauseln:

$$K_4 : \{Arzt(b)\} \text{ und } K_5 : \{Quaksalber(b)\}$$

Ableitung:

Klauseln	Resolvente	Unifikator	Nummer
K_2, K_4	$\{Mag(a, b)\}$	$\sigma = \{Y/b\}$	K_6
K_6, K_3	$\{\neg Patient(a), \neg Quaksalber(b)\}$	$\sigma = \{X/a, Z/b\}$	K_7
K_5, K_7	$\{\neg Patient(a)\}$		K_8
K_1, K_8	$\square$		q.e.d.