Relationen

Programmiersprachen,
Datenbanken,
Partitionen und
etwas mehr

Einführung in die Algebra
24.10.2005

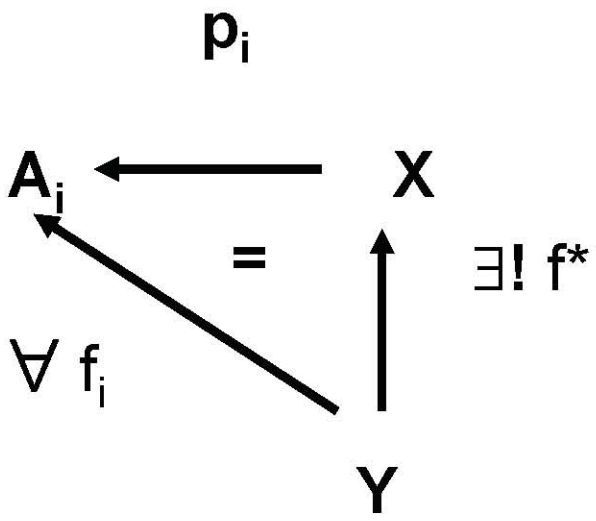
Produkt von Mengen

Sei $(A_i, i \in I)$ **eine Familie von Mengen.**

Eine Menge X zusammen mit einer Familie von Abbildungen p_i: X → A_i heißt Produkt der $(A_i, i \in I)$, wenn folgendes gilt:

Zu jeder Menge Y und jeder Familie von Abbildungen $f_i : Y \rightarrow A_i$ existiert genau eine Abbildung $f^*: Y \rightarrow X$ mit

$$p_i^* \, f^* = f_i$$



Produkt und Tupelbegriff

- weitere zweistellige Grundoperation der Mengenlehre:

$$A \times B = \{ (a, b) \mid a \in A \text{ und } b \in B \}$$

(kartesisches) Produkt

- verallgemeinerte Produktbildung für n Mengen ($n \geq 2$) :

$$A1 \times \ldots \times An = \{ (a1, \ldots, an) \mid ai \in Ai \}$$

- Elemente des Produkts von n Mengen heissen (n)-Tupel.

- spezielle Bezeichnungen für Tupel:
  - n = 2: Paare
  - n = 3: Tripel
  - n = 4: Quadrupel

Relationen

Definition

Eine **n-stellige Relation R** ist eine Teilmenge des karthesischen Produktes von n Mengen $A_1, A_2,...,A_n$:

$$R \subseteq A_1 \times A_2 \times ... \times A_n$$

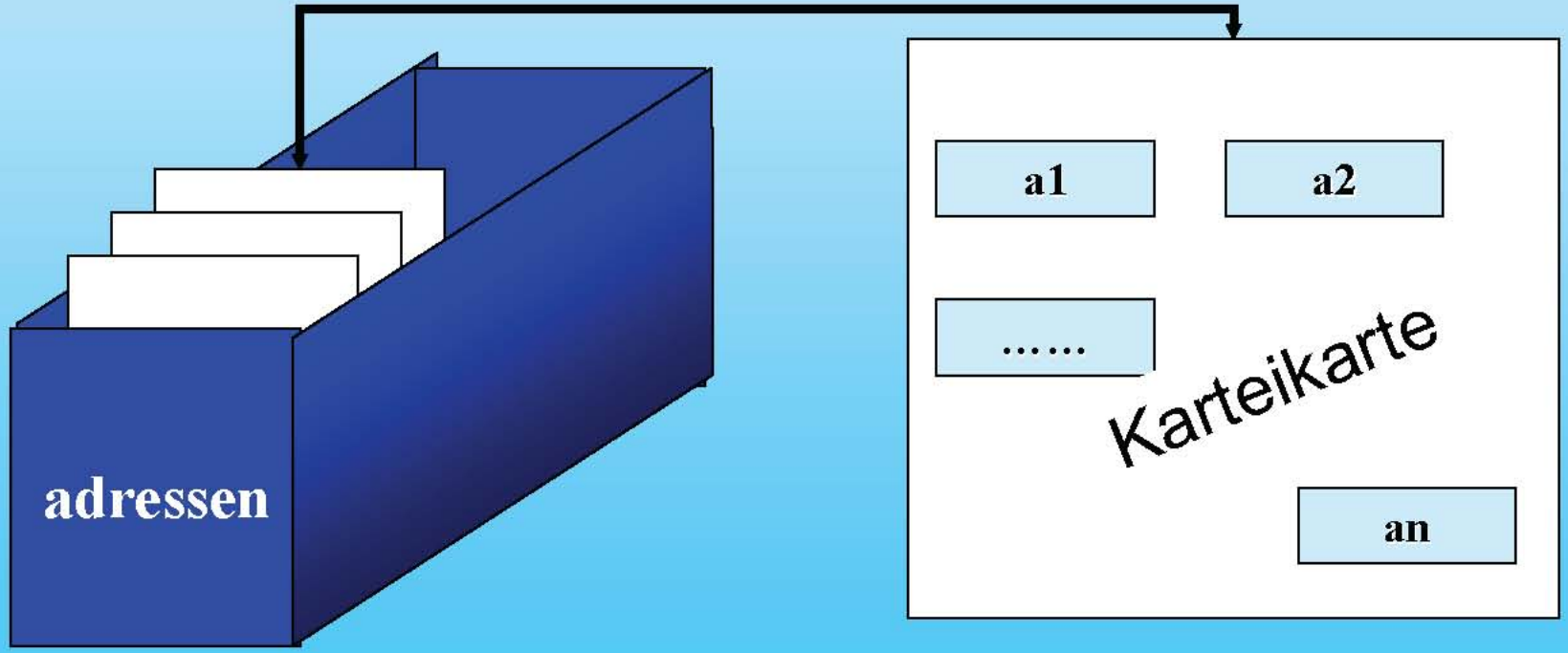
Schreibweise:

$$(a_1, a_2,...,a_n) \in R \leftrightarrow R\ (a_1, a_2,...,a_n)$$

Relationen
Beispiele

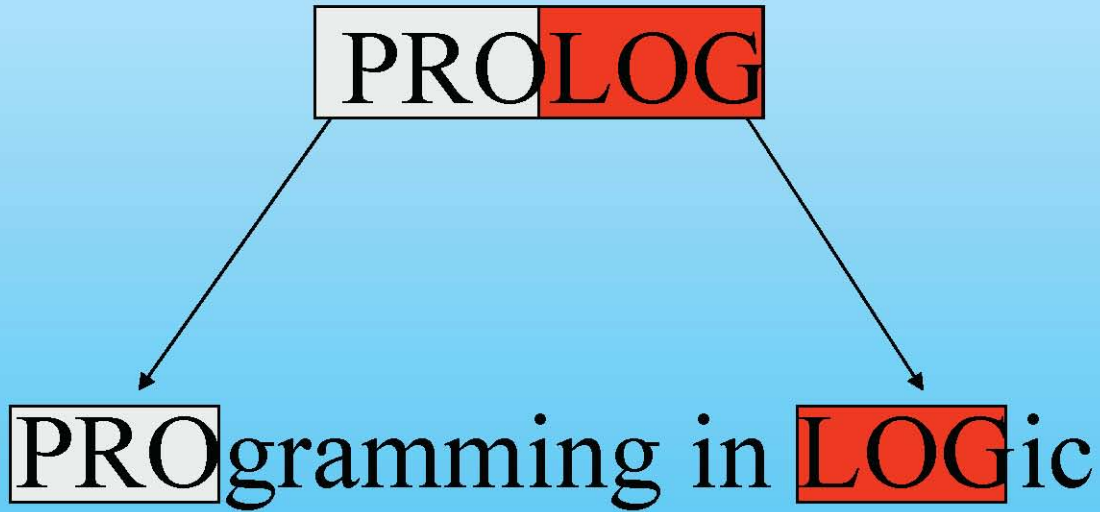
1. Datenbanken: Tabelle

   Beispiel: adressen(a1,a2,…,an)



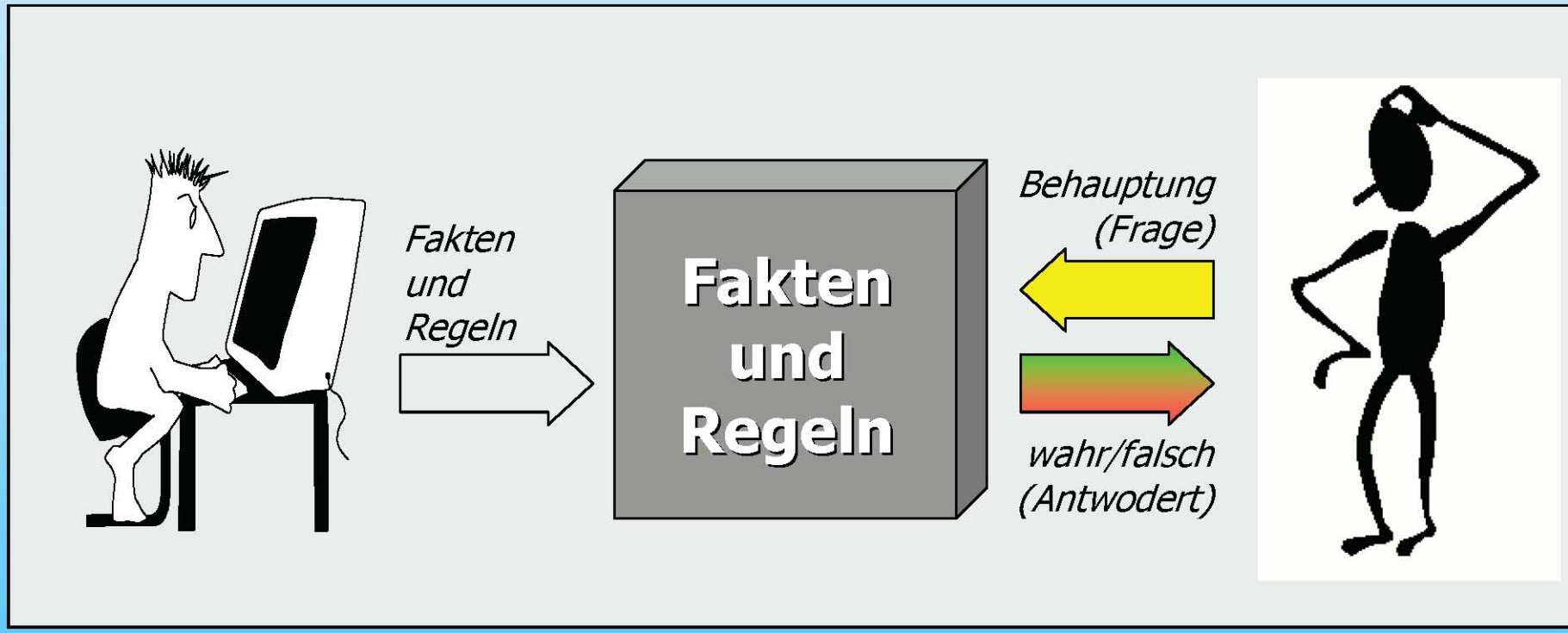
Relationen
Beispiele

Für was steht PROLOG?



PROLOG

PROgrammieren in LOGig



**Prolog-
Progammierer**

**Prolog-
Programiersystem**

Benutzer

Anwendungsgebiete

- Expertensystem (Diagnosesystem)
- Relationale Datenbanken
- mathematische Logik
- abstrakte Problemlösungen
- Simulieren des menschlichen Sprachverstehens
- automatischer Entwurf
- Lösung von symbolischen Gleichungen
- Analyse biochemischer Strukturen
- zahlreiche Gebiete der KI

Geschichte der Logischen Programmierung

Die Geschichte der Logischen Programmierung ist nicht sehr lang.
Die theoretischen Grundlagen wurden in den 70er Jahren erarbeitet. Der erste Prolog-Interpreter wurde 1972 von Alain Colmerauer in Marseilles geschrieben.
Erst Anfang der 80er Jahre kamen die ersten kommerziellen Prolog-Interpreter auf den Markt.
Durch die Wahl von PROLOG als Sprache der Rechner der 5. Generation bei einem jap. Forschungsprojekt gelang der weltweite Durchbruch.

Programmierparadigmen

1. Imperatives Programmieren
   Funktionales Programmieren
   Deklaratives Programmieren
   Objektorientiertes Programmieren

1. Imperatives Programmieren

Es wird beschrieben, **WIE** ein bestimmtes Problem gelöst werden soll.

- Assembler
- ADA
- BASIC
- C / C++
- COBOL
- FORTRAN
- Java
- Modula
- PASCAL
- Perl
- PL/1
- Simula
- Smalltalk
- und viele mehr...

2. Funktionales Programmieren

Das Programm ist eine Folge von Funktionen.

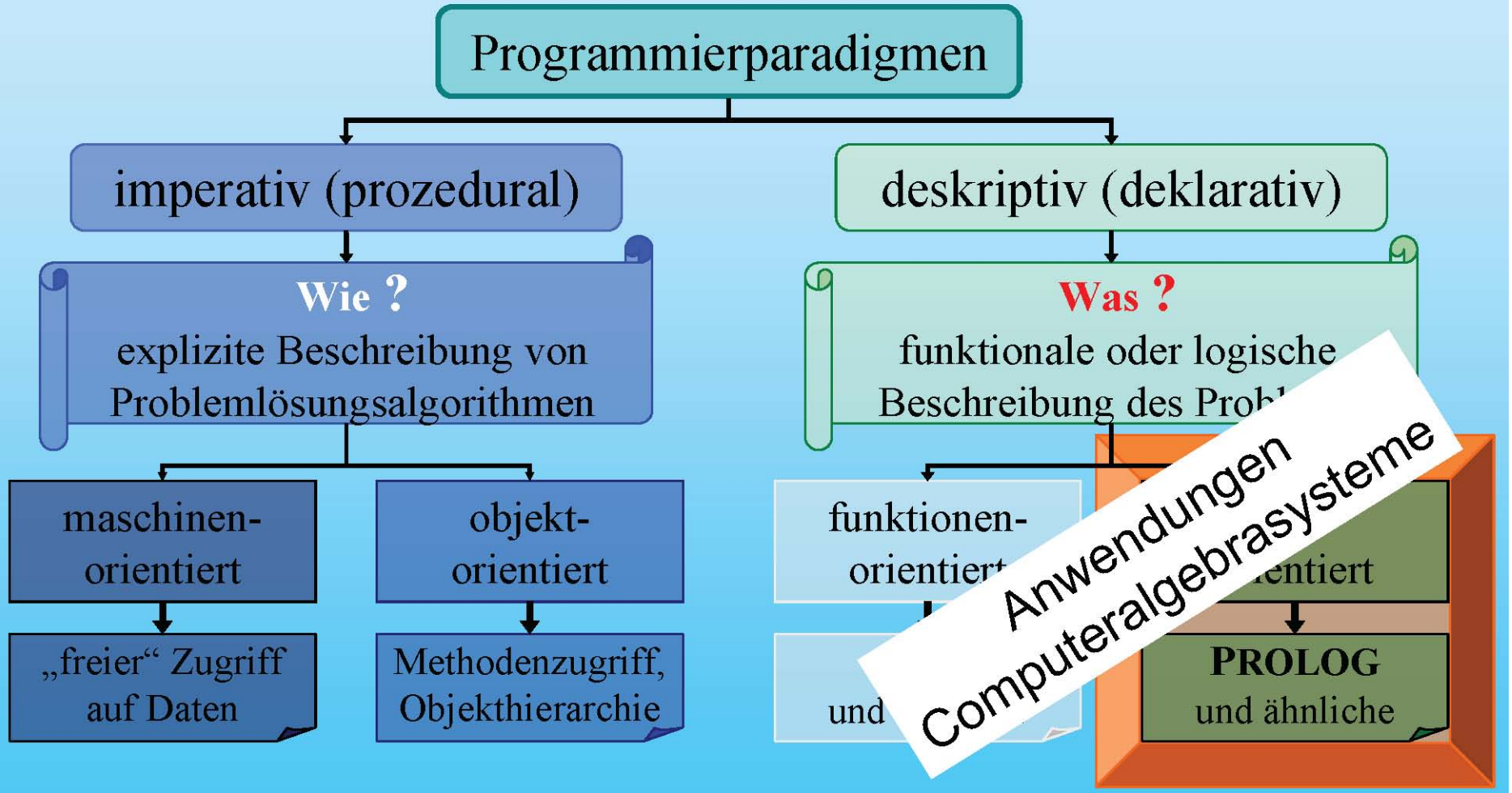
- Lisp
- Logo
- Haskell
- ML
- Hope
- Scheme
- Concurrent Clean
- Erlang
- NESL
- Sisal
- Miranda

3. Deklaratives Programmieren

Es wird beschrieben, **WAS** das Problem ist, nicht jedoch wie dieses zu lösen ist. Die Lösung muß der Computer finden.

- Prolog
- Goedel
- Escher
- Elf
- Mercury

Logische Programmierung
Einordnung des logischen Paradigmas



4. Objektorientiertes Programmieren

Wird in Verbindung mit den drei vorhergehenden Paradigmen verwendet.

Imperativ:

- Smalltalk
- Java
- Eiffel
- C++
- Object Pascal

Funktional:

- XLISP
- Haskell
- andere...

Deklarativ:

- Bestimmte Versionen von Prolog
- und andere...

Nahezu alle neuen Programmiersprachen sind objektorientiert!

Programmlängen-Vergleich

Programmiersprache	Programmgröße in Quellseiten
Fortran	36
Cobol	25
Ada	24
PL/I	22
C	22
Pascal	20
Basic	19
MProlog	9

PROLOG

PROgrammieren in LOGig

Prologprogramm

- **Fakten**
- **Regeln**
- **Anfragen**

$p(a_1,\ldots,a_n).$

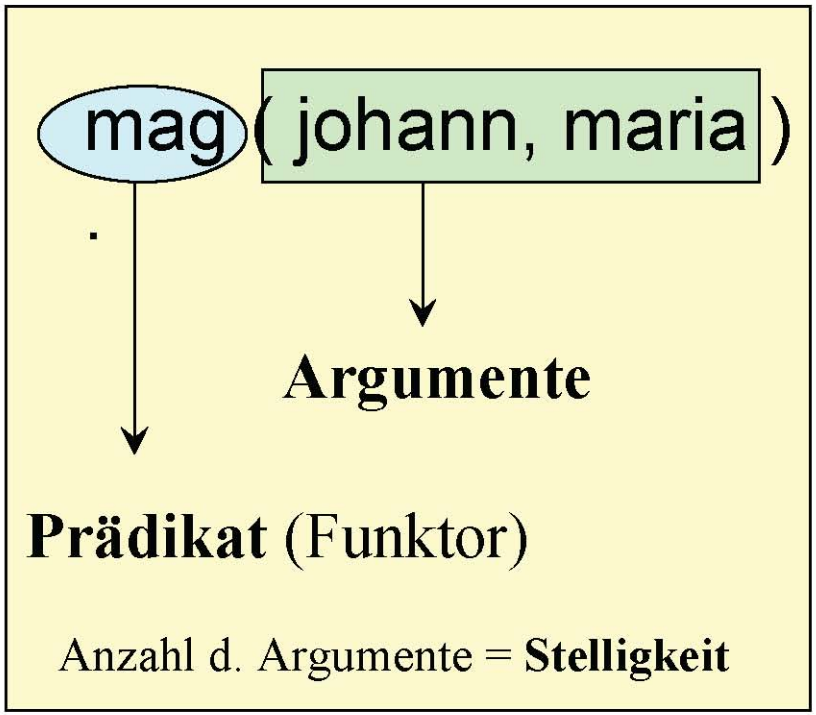
- p ist der Name des Fakts
- $a_1, \ldots, a_n$ sind die Argumente des Fakts

Fakten 1

Beispiele:

Schreibweise in Prolog:	Natürliche Bedeutung:
• ‘die Sonne scheint‘.	• Die Sonne scheint.
• es_regnet.	• Es regnet.
• mensch(sokrates).	• Sokrates ist ein Mensch.
• männlich(daniel).	• Daniel ist männlich.
• mag(johann,maria).	• Johann mag Maria.
• besitzt(johann,gold).	• Johann besitzt Gold.
• vater(hans, gabriel).	• Hans ist der Vater von Gabriel.

Fakten 2



Die Reihenfolge von johann und maria ist **nicht** egal, d.h. es ist ein Unterschied ob man mag(johann,maria) oder mag(maria,johann) schreibt. Ersteres könnte z.B. „Johann mag Maria“, das andere „Maria mag Johann“ bedeuten.

Prolog weiß nicht, was ein Fakt bedeutet. Deshalb muß man selber festlegen was z.B. mag(johann,maria) bedeutet.

- Ein Fakt kann beliebig viele Argumente haben.
- Ein Fakt muß mit einem Punkt beendet werden.
- Ein Fakt muß mit einem Kleinbuchstabe oder ‘ beginnen.

PROLOG

PROgrammieren in LOGig

Prologprogramm

PROgrammieren in LOGig

- **Fakten**
- **Regeln**
- **Anfragen**

Regel

b1 und b2,..und bn → f

Wenn b1 und b2,..,und bn gelten,dann gilt auch f.

Prologschreibweise

f :- b1,b2,…,bn.

Regeln 2

Die linke Seite (vor :-) ist wahr, wenn die rechte Seite bewiesen werden kann.

Ein Komma zwischen zwei Fakten auf der rechten Seite entspricht einer logischen UND-Verknüpfung. Ein Semikolon entspricht einer ODER-Verknüpfung. Es kann geklammert werden.

Die UND- bzw. ODER-Verknüpfungen können auch bei Fragen angewendet werden.

Mit not kann verneint werden.

Die Regeln werden auch Horn-Klauseln genannt.

Prologprogramm

- **Fakten**
- **Regeln**
- **Anfragen**

Gelten die Fakten
p1, p2,…, pn ?

Prologschreibweise
? p1, p2,…, pn

Variablen

Wissensbasis:

```
besitzt(johann, gold).
besitzt(johann, buch).
mag(johann, maria).
mag(johann, fisch).
```

Um festzustellen, was John besitzt, kann man die Frage besitzt(johann,X) stellen. X ist dabei eine Variable. Prolog antwortet dabei mit folgendem:

X = gold
X = buch

_ ist eine anonyme Variable. Bei dem Beispiel oben, würde _ anstatt X folgendes bedeuten: Besitzt John etwas? (Antwort: yes).

Variablen beginnen mit einem Großbuchstaben oder Unterstrich .

Fakten

Sokrates ist ein Mensch.

•Regeln

Alle Menschen sind sterblich.

Anfragen

Ist Sokrates sterblich?

Prologschreibweise

mensch(sokrates).

mensch(X) → sterblich(X)

sterblich(X):- mensch(X).

? sterblich(sokrates)

Expertensysteme
(= wissensbasierte Systeme)

"Ein Expertensystem ist ein Computerprogramm, welches Spezialwissen und Schlußfolgerungsfähigkeiten von menschlichen Experten in einem begrenzten Aufgabengebiet nachbildet und somit Problemstellungen mit einer einem Experten vergleichbaren Leistung löst."

Expertensysteme

Prinzipieller Aufbau

Benutzer :
Experte …. Laie

Dialogkomponente

Erklärungskomponente

Inferenzkomponente

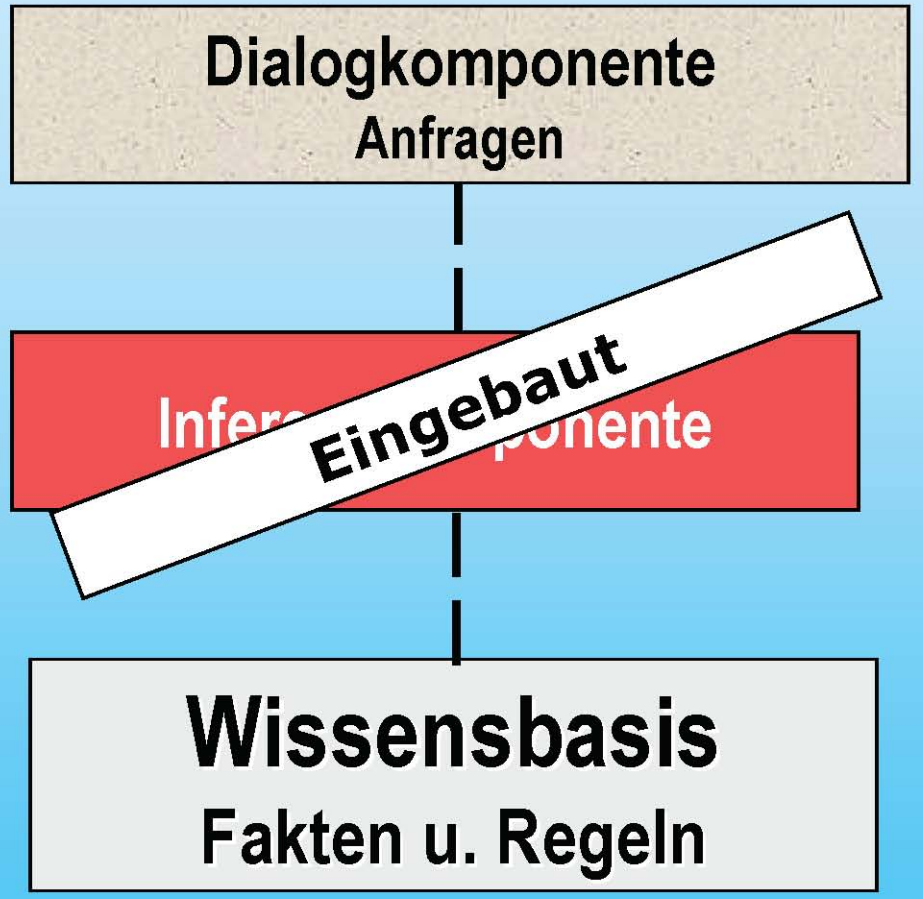
Wissensbasis

Wissenserwerbskomponente

Datenbank

Experte

Prologprogramm <-> Expertensystem



Arithmetik 1

Beispiel: Berechnen der Fakultät

```
fak(0,1).

fak(N,X) :-        N > 0,   M is N - 1,   fak(M,Y),   X is N * Y.
```

```
FUNCTION fak(N : Integer) : Integer;
BEGIN
        IF N = 0 THEN fak := 1
        ELSE IF N > 0 THEN fak := N * fak(N-1);
END;
```

Aufruf:

?- fak(0,N).

N = 1

?- fak(6,N).

N = 720

Arithmetik 2

Beispiel: Berechnen der Fibonacci Folge

```
fib(0,1).
fib(1,1).
fib(N,X) :- N1 is N - 1, N2 is N - 2, fib(N1,X1), fib(N2,X2), X is X1 + X2.
```

Aufruf:

?- fib(11,N).

N = 144

?- fib(4,N).

N = 5

Arithmetik 3

+ - * /	Addition, Subtraktion, Multiplikation, Division
mod	Modulo
// ^	Gleitzahl-Division, Potenzierung
( )	Priorität
is	Zuweisen eines arith. Ausdruckes
> <	größer, kleiner
=> =<	größer gleich, kleiner gleich (zuerst = dann > !)
=:=	gleich (arithmetisch)
=\=	ungleich (arithmetisch)

Infix: **(4 + 1)*4** Postfix: ***(4,+(4,1))**

A, B und C stehen vor Gericht.

- A sagt aus, dass B lügt.
- B sagt aus, dass C lügt.
- C sagt aus, dass A und B lügen.

Wer lügt, wer sagt die Wahrheit?

Prologprogramm

```
ist_Lügner(wahr,lügt).
ist_Lügner(lügt,wahr).
beide_lügen(wahr,lügt,lügt).
beide_lügen(lügt,wahr,lügt).
beide_lügen(lügt,lügt,wahr).
beide_lügen(lügt,wahr,wahr).
```

```
?- ist_Lügner(A,B),
   ist_Lügner(B,C),
   beide_lügen(C,A,B).
```

Die Lösung eines Kriminalfalles

```
person( heinrich, 25, m , fotograph).
person( heinrich, 25, m , gaernter).
person( ivon,     22, w , kindermaedchen ).
person( berthold, 55, m , fahrer ).
person( johann,   25, m , taschendieb).

hatte_Beziehungen_zu(ivon,johann).
hatte_Beziehungen_zu(ivon,berthold ).
hatte_Beziehungen_zu(susanne,johann).

ermordet_mit(susanne,harter_Gegenstand).
ermordet(susanne).
.....
```

```
.....

moerder(Moerder) :-
        person(Moerder,_,_,_),
        ermordet(Ermordete),
        verdaechtig(Moerder),
        befleckt_mit(Moerder,Substanz),
        befleckt_mit(Ermordete,Substanz).
```

Relationen

Binäre Relation

Definition

Eine binäre Relation R ist eine Teilmenge des karthesischen Produktes zweier Mengen A und B:

$$R \subseteq A \times B$$

Für

$$(a,b) \in R$$

schreibt man auch

$$aRb \text{ oder } R(a,b)$$

Binäre Relationen
Beispiele

a) die Ordnungsrelation $\leq$ auf $\mathbb{N}$ und $\mathbb{R}$,

b) die Enthaltenseinsbeziehung zwischen Mengen: $A \subseteq B$,

c) die Ähnlichkeit von $(n \times n)$-Matrizen A and A': $\exists\ S$ such that $A' = S^{-1}AS$

d) die Kongruenz $(\mod n)$ auf $\mathbb{Z}$, z.B. $6 \equiv 21 (\mod 5)$.

Binäre Relationen und Graphen

Eine binäre Relation R kann durch einen Graphen veranschaulicht werden in dem jedes Tupel (a,b) als Kante zwischen den Knoten a und b interpretiert wird.

$(a,b) \in R$  a  b

Umgekehrt entspricht jede Kante (a,b) eines Graphen die zwei Knoten a und b verbindet einer Relation R für die

aRb als „a ist mit b direkt verbunden“ interpretiert werden kann.

Reflexivität

Eine binäre Relation $R \subseteq S \times S$ ist reflexiv, wenn jedes Element von S zu sich selbst in Relation steht:

$$\forall x: x \in S: xRx$$

zB: die Relation „hat dieselbe Mutter wie“ ist reflexiv

Reflexive Relationen können durch einen Graphen modelliert werden, bei dem alle Knoten Schlaufen haben.

Symmetrie

Eine binäre Relation $R \subseteq S \times S$ ist symmetrisch, wenn aus xRy auf yRx geschlossen werden kann:

$$\forall\ x,y: x,y \in S: xRy \Rightarrow yRx$$

zB: die Relation „ist verheiratet mit“ ist symmetrisch

Symmetrische Relationen entsprechen ungerichteten Graphen.

Transitivität

Eine binäre Relation $R \subseteq S \times S$ ist transitiv, wenn aus xRy und yRz auf xRz geschlossen werden kann:

$$\forall\, x,y,z: x,y,z \in S: (xRy \wedge yRz) \Rightarrow xRz$$

zB: die Relation „ist Vorfahre von“ ist transitiv

Äquivalenzrelationen

Definition:

Eine Äquivalenzrelation auf einer Menge M ist eine binäre Relation ~ auf M, die

- **reflexiv,**
- **symmetrisch und**
- **transitiv ist.**

Definition Äquivalenzklasse

Sei ~ eine Äquivalenzrelation auf M und a ∈M. Dann heißt die Menge der zu a äquivalenten Elemente die Äquivalenzklasse von a.

$$\bar{a} := \{x \in M \mid a \sim x\} \subseteq M$$