Logische Programmierung in Prolog I

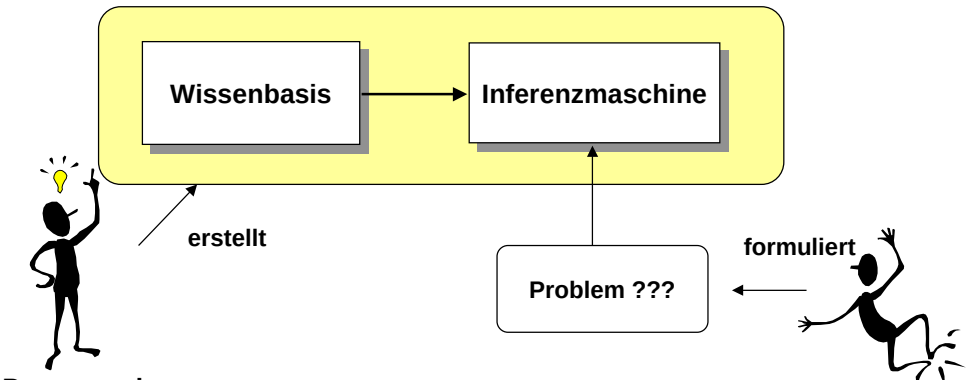
- **Programmiermodell**
- **Grundkonzepte**
- **Sprachelemente von Prolog**
- **Terminologie**

Literatur

- Bratko, I.
  Prolog programming for artificial intelligence. 2nd ed.
  Addison-Wesley, 1990.
- Sterling, L. & Shapiro, E.
  The art of Prolog. MIT Press, 1986.
- Clocksin, W.F. & Mellish, C.S.
  Programming in Prolog, 2nd Ed,
  Springer , 1984.
- Lloyd, J.W.
  Foundations of logic programming. 2nd Ed,
  Springer , 1987.

Einführung

Logikbasierte Programmierung



nicht-algorithmisches, sondern deklaratives Programmieren !!

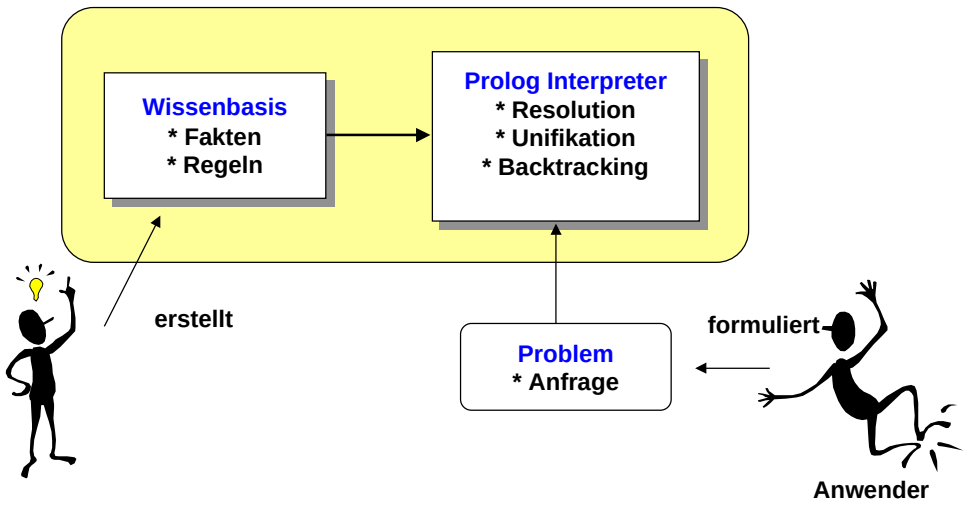
Einführung

Prolog - Einordnung

- **PROLOG**
  - steht für PROgramming in LOGic
  - Logik wird als Programmiersprache benutzt, basiert auf der **Prädikatenlogik**
  - Ist eine Programmiersprache für symbolische, nicht-numersiche Programmierung
  - Ist gut geeignet für Probleme, die Objekte und Relationen zw. Objekten behandeln
  - Syntax der Sprache ist eine modifizierte **Hornklausel-Notation**
- **Entwicklung**
  - Robert Kowalski (Edinburgh): theoretische Entwicklung
  - 1972: erste Implementierung von Colmerauer
  - Mitte der 70er Jahre : effiziente Implementierungen (Warren)
  - 1980: Sprache des japanischen "Fifth Generation Project"

Einführung

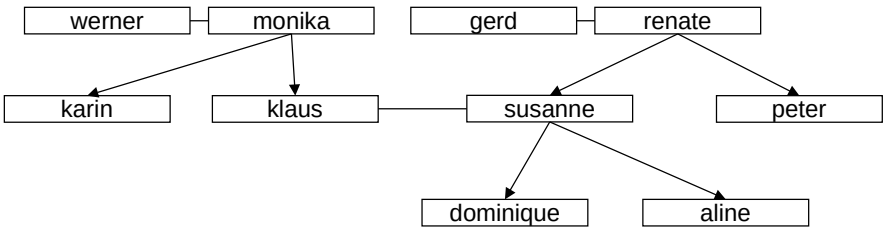
Prolog Programmiermodell



Einführung

Problemwelt

- **Verwandtschaftliche Beziehungen**
  - Wir wollen Fragen in dieser Problemwelt beantworten!
- **Beispiele**
  - "Wer ist die Mutter einer Person X"
  - "Wer ist Großvater einer Person Y"



Grund-konzepte

Tatsachen (Fakten)

Tatsachen

- sind Aussagen über **Objekte**, die als richtig angenommen werden.
- Sie beschreiben
  - Eigenschaften einzelner Objekten
  - Aussagen, die gleichzeitig mehrere Objekte betreffen (Relationen)

Beispiele:

```
weiblich(monika).        maennlich(werner).
weiblich(susanne).       maennlich(klaus).
weiblich(aline).         maennlich(dominique).
weiblich(karin).         maennlich(peter).
weiblich(renate).        maennlich(gerd).
```

```
verheiratet(werner, monika).
verheiratet(gerd, renate).
verheiratet(klaus, susanne).
```

Relationen sind gerichtet

```
istMuttervon(susanne, aline).
istMutterVon(susanne, dominique)
istMutterVon(monika, karin).
istMutterVon(monika, klaus).
istMuttervon(renate, peter).
istMuttervon(renate, susanne).
```


Grund-konzepte

Arbeitsweise von Prolog, Fragen

Prinzip

- Zuerst wird die **Wissensbasis** erstellt und dem Prolog-System mitgeteilt.
- Dann werden **Fragen** (goals, queries) an die Wissensbasis gestellt, deren Richtigkeit bewiesen werden soll.
- Prolog erzeugt die **Antwort**, indem es einen **logischen Beweis** auf der Basis des vorhandenen Wissens durchführt.

Beispiel:

```
?- maennlich(gerd).
yes

?- verheiratet(klaus, susanne).
yes

?- verheiratet(gerd, monika).
no
```

Grund-konzepte

Schlußfolgerungen, Regeln

- **Regeln dienen dazu, um aus bekanntem Wissen neues Wissen herzuleiten.**
- **Beispiel**
  - Vater-Kind-Beziehung
    - "Eine Person V ist Vater eines Kindes K, wenn er mit einer Frau F verheiratet ist und diese Mutter des Kindes K ist."
  -
```
istVaterVon(V,K) :- verheiratet(V,F) ,
                    istMutterVon(F,K).
```
- **Bemerkungen**
  - Alle Voraussetzungen stehen rechts in der Regel.
  - Diese werden durch Komma getrennt.
  - Variablen sind Platzhalter für beliebige konkrete Objekte

Grund-konzepte

Regeln

- **Regeln bestehen aus**
  - einem Bedingungsteil (rechte Seite, Rumpf)
  - und einem Folgerungsteil (linke Seite, Kopf)
- **Eine Regel p :- q, r kann gelesen werden**
  - **Deklarative** Bedeutung im Sinne von WAS
    - IF rechte Seite der Regel THEN linke Seite der Regel
  - **Prozedurale** Bedeutung im Sinne von WIE
    - Um ein Ziel p zu lösen, müssen zuerst die Unterziele q und r gelöst werden
- **Prolog wendet die Regeln rückwärts an**
  - d.h. Prolog startet mit der linken Seite
  - Um zu zeigen, dass die linke Seite gilt, muss gezeigt werden, dass die rechte Seite gilt.
  - backward chaining

Grund-konzepte

Variablen in Fragen

```
mag(georg, karin).
mag(georg, peter).
mag(georg, susi).
mag(susi, wein).
mag(karin, gin).
mag(susi, karin).
mag(susi, peter).

gemFreund(A,B,C) :- mag(A, C),
                    mag(B, C).
```

?- mag (karin, gin).

YES

?- mag(susi, WAS).

WAS = wein ;

WAS = karin

?- gemFreund (georg, susi, WER).

WER = karin ;
WER = peter ;
NO

?- mag (susi, susi)
NO

Es ist nicht beweisbar!

- **Prolog antwortet**
  - mit einer ***Instanziierung*** der Variablen in der Frage
  - ***Weitere*** Instanziierungen können durch ; abgerufen werden

Grund-konzepte

Rekursive Definition von Regeln

- **Prolog erlaubt die rekursive Definition von Regeln**
  - Rekursion ist eine wichtige Programmiertechnik in Prolog
- **Beispiel:**
  - Definition der Vorfahre-Relation mit Hilfe der Relation `elternteil`
  - **Fall1:**
    Für alle X und Z:
    - X ist Vorfahre von Z falls X Elternteil von Z ist
    - `vorfahre(Vorf,X):- elternteil(Vorf,X).`
  - **Fall2:**
    Für alle X und Z:
    - X ist Vorfahre von Z falls es ein Y gibt mit
    - (1) X ist Elternteil von Y und
    - (2) Y ist Vorfahre von Z.
    - ```
      vorfahre(Vorf,X):- elternteil(Vorf,Y),
                         vorfahre(Y,X).
      ```
      ```

Grund-konzepte

Rekursive Definition von Regeln

Beispiel:

- `vorfahre(Vorf,X):- elternteil(Vorf,X).`
- `vorfahre(Vorf,X):- elternteil(Vorf,Y), vorfahre(Y,X).`

```
?-vorfahre(X,aline).
X = klaus;
X = susanne;
X = werner;
X = monika,
X = gerd;
X = renate;
```


Grund-konzepte

Prolog-Programme

- **Prolog-Programme bestehen aus *Klauseln***
  - Fakten und Regeln

```
weiblich(monika).
weiblich(susanne).
weiblich(aline).
weiblich(karin).

verheiratet(werner, monika).
verheiratet(gerd, renate).

istVaterVon(V,K) :- verheiratet(V,F) ,
                    istMutterVon(F,K).
```

- **Anmerkungen**
  - Jede Klausel wird mit einem **Punkt** abgeschlossen.
  - Jede der ersten 4 Klauseln drückt ein **Fakt** über die `weiblich`-Relation aus (Prädikat)
  - Die Anzahl der Argumente wird als **Stelligkeit** der Relation (des Prädikats) bezeichnet.

Grund-konzepte

Universelle Fakten

- **Variablen in Fakten**
  - `mag(susi,X).` „Susi mag alles“
  - `mag(X,X).` „Jedes mag sich selbst“
  - `mag(X,Y).` „Jedes mag alles“
- **Anmerkungen**
  - Gleiche Variable in gleichem Fakt: Gleichheit
  - Gleiche Variable in verschiedenen Fakten: Nichtgleichheit
  - Variablen in Fragen sind existenzquantifiziert:
    `?-mag (susi,X).` **Existiert** etwas, das Susi mag“
  - Variablen in Fakten sind allquantifiziert:
    `mag (susi,X).` „Susi mag **alles**“

Grund-konzepte

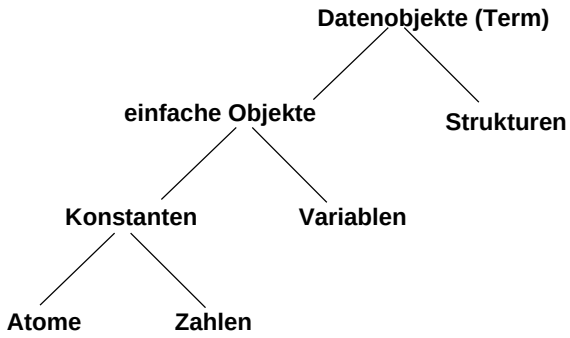
Konjunktiv verknüpfte Fragen

- **Prolog kann verschiedene Fragen kombinieren**
  - Beispiel: Frage: "Wer ist Großvater von Aline?“
  - Vorgehen:
    - (1) Wer ist Vater von Aline? Annahme, es gibt ein Y.
    - (2) Wer ist Vater von Y? Annahme, es gibt ein X.
- **Frage:**
  -
```
?- istVaterVon(Y,aline), istVaterVon(X,Y).
Y = klaus
X = werner
```
- **Ordnung der Teilfragen kann geändert werden, ohne dass sich das Ergebnis ändert.**
- **Beispiel:**
  -
```
?- istVaterVon(X,Y), istVaterVon(Y,aline)
X = werner
Y = klaus
```

Syntax

Objekte in Prolog

- **Objekte**
  - repräsentieren Dinge der Vorstellungswelt oder sind Abstraktionen von realen Gegenständen.
- **Prolog kennt folgende Datenobjekte**
  - Prolog erkennt den Typ eines Objekts an seiner syntaktischen Struktur



Syntax

Atome und Zahlen

- **Atome sind**
  - Zeichenketten aus Buchstaben, Zahlen und dem Zeichen _
  - Atome beginnen mit einem Kleinbuchstaben

    z.B. `renate` `x_25` `frau_holle` `peter`

  - Zeichenketten aus speziellen Zeichen,
    z.B. `<--->` `===>`
  - in einfache Anführungszeichen eingeschlossene Zeichenketten
    z.B. `'Frau Holle'`
  - Spezielle Zeichen: `,` `;` `!` `[ ]`
- **Zahlen in Prolog sind Integer- oder Real-Zahlen, wobei**
  - Real-Zahlen einen Dezimalpunkt enthalten,
    z.B. 2.03
  - Integer-Zahlen nicht,
    z.B. -22, 3, 6

Syntax

Objekte in Prolog

- **Objekte**
  - repräsentieren Dinge der **Vorstellungswelt** oder sind Abstraktionen von realen **Gegenständen**.
- **Terme beschreiben Objekte**
  - **Konstanten** und **Variablen** sind Terme,
  - Ist f ein Funktionssymbol (Funktor) und sind t_1 , ..., t_n Terme, so ist auch $f(t_1, ..., t_n)$ ein Term (**Struktur**).
  - Ein Term heißt **Grundterm**, wenn er keine Variablen enthält.
- **Konstante**
  - Zahlen
  - Atome

Syntax

Variablen

- **Bedeutung von Variablen in Prolog**
  - In prozeduralen Sprachen repräsentieren Variablen Speicherzellen des Rechners. Variablen kann ein Wert zugewiesen werden.
  - In Prolog ist eine Variable ein **Platzhalter** für ein Prolog-Objekt
    - Anstelle einer Variablen kann ein **beliebiges Prolog-Objekt** eingesetzt werden.
    - Bspl: `mag(susi, WAS)`
- **Syntax**
  - Bezeichner einer Variablen beginnt mit Großbuchstabe oder mit Unterstrich "_"
- **Anonyme Variable**
  - Drückt aus, dass an **einer Stelle** ein beliebiges Objekt eingesetzt werden kann und wir uns nicht dafür interessieren.
  - `istEhemann(Person) :- verheiratet(Person, _ )`

Syntax

Strukturen

- **Eine Struktur repräsentiert ein Objekt, das aus mehreren Objekten zusammengesetzt ist.**
- **Aufbau**
  - **Name** der Struktur (Funktor)
  - **Komponenten** der Struktur werden geklammert und durch Komma getrennt
  - Funktor muss ein Atom sein
  - Komponenten können Konstanten oder wiederum Strukturen sein.
- **Beispiele**
  - `datum (24, 11, 1999)`
  - `name (herr, max, mueller)`
  -
```
geboren(name (herr, max, mueller),
        datum (24, 11, 1999))
```


Syntax

Beispiel Datenstruktur: Terme und Fragen

```
buch('The Art of Prolog',author (sterling,shapiro)).
buch('Programming in Prolog',author(clocksin,mellish)).
buch('Foundations of LogicProgramming',author(lloyd)).


?-buch(X,Y).
X= 'The Art of Prolog' Y=author (sterling, shapiro);
X= 'Programming in Prolog' Y=author(clocksin,mellish);
X= 'Foundations of LogicProgramming' Y=author(lloyd);
no

?-buch(X,author(Y,Z)).
X= 'The Art of Prolog' Y=sterling Z=shapiro;
X= 'Programming in Prolog' Y=clocksin Z=mellish;
no
```

Konzepte und Terminologie

Definitionen nach Sterling/Shapiro: Programm, Regel, Prädikat

- Ein **Logikprogramm** ist eine endliche Regelmenge.
- Statt Regel sagt man auch **Horn-Klausel** oder bei Eindeutigkeit auch einfach nur **Klausel.**
- Eine **Regel** hat die Form.

  $A <- B_1, B_2 , ... B_n.$ mit $n >= 0$

  A ist der **Regelkopf** und die B_i's sind der **Regelrumpf.**
  Eine Regel mit $n=0$ wird **Fakt** genannt. A und B_i's werden auch Ziele genannt.
- Ein **Ziel** hat die Form eines Prädikats (Prädikatenlogik 1. Stufe)

  $pred (t_1, t_2, ..., t_m)$ mit $m >=0$

  wobei pred Prädikatnamen, m Stelligkeit und die tj Argumente.
  Prädikate beschreiben Objekte und Beziehungen zwischen diesen Objekten.
- Argumente für Prädikate sind **Terme**; induktiv definiert:
  - eine Konstante ist ein Term,
  - eine Variable ist ein Term,
  - sind $t_1, t_2, ..., t_k$ Terme und ist f ein Funktionssymbol, so ist auch $f (t_1, t_2, ..., t_k)$ ein Term.

Konzepte und Terminologie

Semantik von Regeln und Programmen

- **Konventionen:**
  - hans, h1 Konstante,
  - X, Haus Variable,
  - _ anonyme Variable,
  - mag Prädikatname, plus Funktionssymbol
- **Bedeutung von Regel A <- B1, B2 , ..., Bn:**
  - B1 & B2 & ... Bn - >A,
  - umgangssprachlich „Wenn B1 gilt und B2 gilt und ... und Bn gilt, dann gilt auch A“.
- **Prozedurale Interpretation einer Regel:**
  - „Um A zu beantworten, beantworte B1 und B2 und ... und Bn“.
  - Diese Interpretation ist die Grundlage von PROLOG–Systemen.
- **Eine Frage definiert ein Ziel oder eine Konjunktion von Zielen. Die Ausführung des Programms hat zu zeigen, ob alle Ziele erfüllt werden. Ein Ziel ist erfüllt, wenn:**
  - entweder ein „passendes“ Fakt existiert,
  - oder ein „passender“ Regelkopf existiert und jedes Ziel im Regelrumpf erfüllt wird.