

Semantische Sprachverarbeitung

- PROLOG-Crash-Kurs

Übersicht

- Einführung in PROLOG
 - Fakten, Regeln und Anfragen
 - Unifikation
 - Rekursion
 - Listen
 - Operatoren
 - Negation, Identität
 - Backtracking, fail und cut

PROLOG

- Was ist PROLOG?
 - PROgramming in LOGic
 - Deklarative Programmiersprache
- Bestandteile von PROLOG:
 - Ein PROLOG Programm besteht aus
 - Fakten und Regeln (=Wissensbasis)
 - Variablenbindung erfolgt durch Unifikation
 - Resolutionsbeweiser führt Beweise durch

Fakten

- Fakten beschreiben wahre Aussagen in der jeweiligen Domäne:
- James Bond ist langweilig.
- Als Prologklausel:
 - **langweilig(bond)**.
- **langweilig** ist ein einstelliges Prädikat.
- **bond** ist eine Konstante.

Regeln

- Regeln beschreiben logische Implikationen.
- James Bond ist glücklich, wenn er Ü-Eier ißt.
- In PROLOG:

```
gluecklich(bond) :-  
  isst(ue_eier, bond).
```
- Allgemein: <head>:-<body>.
 - Head \square Body oder
 - Head \square Body
 - Mehrere Bedingungen können mit „ \square “ (\square) verknüpft werden.

M. Guhe / F. Schilder - SemSprach
Sommer 2003

Prolog Crashkurs, Seite 5

Anfragen

- Fakten und Regeln werden in der Wissensbasis gespeichert. z.B:

```
langweilig(bond). isst(mars-riegel,  
  bond).
```
- Es können Anfragen an die Wissensbasis gestellt werden.

```
?- langweilig(bond).  
yes  
?- isst(ue_eier, bond).  
no
```
- Ein Faktum ist dann falsch (Ausgabe ist **no**) wenn nichts in der Wissensbasis steht!

M. Guhe / F. Schilder - SemSprach
Sommer 2003

Prolog Crashkurs, Seite 6

Unifikation

- Variablennamen in PROLOG fangen mit einem Großbuchstaben an (z.B. **X** oder **Bond**).
- Variablen erlauben allgemeinere Regeln:

```
gluecklich(X) :-  
  isst(ue_eier, X).
```
- Wenn wir

```
isst(ue_eier, bond)
```

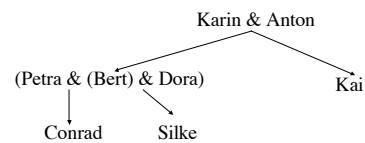
 der Datenbank hinzufügen, können wir die folgende Anfrage stellen.

```
?- gluecklich(X).  
X = bond.  
yes
```
- **X** wird mit **bond** unifiziert.

M. Guhe / F. Schilder - SemSprach
Sommer 2003

Prolog Crashkurs, Seite 7

Beispiel für Rekursion: Vorfahren



- Datenbasis modelliert den Familienbaum:

```
elternteil(karin, bert). elternteil(anton, bert).  
elternteil(karin, kai). elternteil(anton, kai).  
elternteil(dora, silke). elternteil(bert, silke).  
elternteil(petra, conrad). elternteil(bert, conrad).  
female(karin). female(dora). female(petra). female(silke)  
male(bert). male(kai). male(conrad). male(anton).
```

M. Guhe / F. Schilder - SemSprach
Sommer 2003

Prolog Crashkurs, Seite 8

Rekursion

- Relationsketten disjunktiv verknüpft (;) können durch Rekursion abgekürzt werden:

```
vorfahre(X,conrad):-
  elternteil(X,conrad); %% Eltern sind Vorfahren
  (elternteil(X,Y), elternteil(Y,conrad));
%% Großeltern sind Vorfahren
  (elternteil(X,Y), elternteil(Y,Z),
  elternteil(Z,conrad));
%% Urgroßeltern sind Vorfahren      ...
%% Ur-Ur..eltern sind Vorfahren etc.
```

- % Kommentarzeile in Prolog

Rekursive Prädikate schreiben

- **vorfahre(Alt,Jung)**, falls es eine ältere Person **Alt** gibt, die durch eine Kette von **elternteil** Relationen mit einer jüngeren Person **Jung** verbunden ist.
- Keine endliche Axiomatisierung möglich!
- Rekursive Definition:
 - Eltern sind Vorfahren:
vorfahre(Alt,Jung):- elternteil(Alt,Jung).
 - Alle anderen Vorfahren sind Eltern von näheren Vorfahren:
vorfahre(Alt,Jung) :- elternteil(Alt,Mittel), vorfahre(Mittel,Jung).
 - Top-down Beweisstrategie! Erste Regel steht vor zweiter!

Listen I

- Listen sind als rekursive Datenstruktur definiert:
 - Die leere Liste ist eine Liste
 - Ein komplexer Term ist eine Liste, falls der Term aus zwei Teilen besteht: einem Term (<Head>) und einer (Rest)-Liste (<Tail>)
- Leere Liste: []
- Head und Tail werden durch „|“ aufgeteilt: [<Head>|<Tail>]

Listen II

- Eine Liste mit 3 Elementen ist z.B.
`[ue-eier|[milka-schokolade|[mars-riegel|[]]]]`
Die allgemeine Schreibweise in PROLOG ist
`[ue-eier,milka-schokolade,mars-riegel]`
- Listen können wiederum Listen enthalten:
`[1, [1a, 1b, 1c], 2, [2a, 2b]]`
- Das Prädikat **member/2** testet, ob ein Element in einer Liste zu finden ist:
member(A, [A|Rest]).
member(A, [Head|Rest]):-member(A,Rest).

Operatoren

- Operatoren werden benutzt um die Lesbarkeit zu erhöhen:

```
eltern teil(dora, silke), female(dora).
dora ist_mutter_von silke.
```
- Operatoren werden wie folgt definiert:

```
op(10, xfx, ist_mutter_von).
```
- Gewisse Operatoren sind schon vordefiniert:
 - Identität wird mit == überprüft.
 - Der Operator für Negation ist \+

Negation

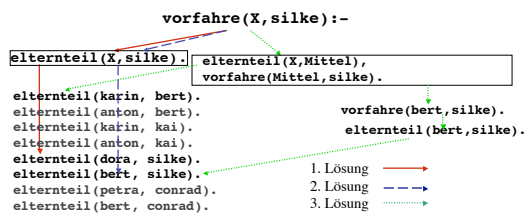
- Wann ist eine Aussage falsch in PROLOG?
 - Wenn die Anfrage nicht mit der Wissensbasis bewiesen werden kann!
 - Ein Faktum ist nicht Teil der Wissensbasis
 - Eine Unifikation mißlingt (**bond=peter**)
- Was passiert wenn eine falsche Aussage negiert wird?

```
(\+ bond=peter)
```
- Wenn nur unvollständiges Wissen vorhanden ist?

```
(\+ isst(mars-riegel, bond))
```

Backtracking und Fail und cut

- Beweise können als Bäume dargestellt werden: z.B. `vorfahre(X, silke).`



Alle Lösungen

- Die Anfrage `vorfahre(X, silke).` ergibt als erste Lösung `dora`.

```
?- vorfahre(X, silke).
X= dora;
X= bert;
X= karin;
X= anton;
no
```
- Backtracking kann auch durch das Prädikat `fail` erzwungen werden.
- Das Prädikat `findall/3` findet alle Lösungen

```
?- findall(V, vorfahre(V, silke), Vorfahren).
Vorfahren = [dora, bert, karin, anton]
```

Zusammenfassung

- PROLOG-Programme bestehen aus
 - Fakten und Regeln
- Anfragen können an die Wissensbasis vom Prolog-System aus gestellt werden.
- Ein Resolutionsbeweiser versucht die Anfrage zu beweisen:
 - Beweisstrategie: top-down, left-right

Quellen und weitere Hilfen

- Literatur:
 - Clocksin, W. & C. Mellish. Programming in Prolog, Springer Verlag.
 - Bratko, I. Programming for Artificial Intelligence. Addison-Wesley.
- Weitere Hilfen:
 - Survival Guide
 - On-line Dokumentation (Aufruf mit manpce)
 - Verzeichnis der Programme:
/home/wsv_1/schilder/SemSprach/