Einführung in PROLOG

1.Einfache Beispiele

Es gibt nur drei Konstrukte in PROLOG. **Fakts (Fakten)**, **rules (Regeln)** und **queries**. Eine Sammlung von Fakten und Regeln nennt man **knowledge base.** Beim Programmieren in PROLOG geht es fast nur darum knowledge bases zu beschreiben. Wie benutzt man ein PROLOG-Programm? Indem man queries eingibt, also indem man Fragen über die Informationen stellt, die in der knowledge base stehen.

1.1 Knowledge Base 1(KB1)

musik(klassik).
musik(rock).
musik(schlager).
musik(country).
laut(rock).

Beachten Sie bitte, dass die Namen *klassik, rock,schlager, country* und die Eigenschaften *musik* und *laut* kleingeschrieben sind. Das ist wichtig. Später sehen wir, warum.

Wie benutzen wir nun KB1? Indem wir queries starten! Nachdem wir die Datei KB1.ecl complieliert haben, können wir z.B. fragen, ob Rock laut ist:

?- laut(rock).

Wir drücken auf RUN und PROLOG wird antworten:

yes

, da dieser Fakt in KB1 explizit enthalten ist (Das Fragezeichen und der Strich sind übrigens der prompt vom Eclipse-PROLOG. Wir geben nur die Querie und einen Punkt ein)
Auf

?- laut(klassik)

erhalten wir

no

, denn über diesen Fakt steht keine Aussage in KB1. Die Queries

?- laut(hiphop)

wird ebenfalls mit „no“ beantwortet, da PROLOG keine Informationen über hiphop hat, schließt es, dass der Fakt nicht aus KB1 abgeleitet werden kann.

?- schnell(country)

führt sogar zu einem abort, da die Eigenschaft schnell überhaupt noch nicht definiert wurde.

1.2. Knowledge base 2 (KB2)

```
gitarrenlastig(grunge).
mag(sam,Musik) :-  gitarrenlastig(Musik).
```

KB2 enthält nur einen Fakt und eine Regel. Regeln enthalten Informationen, die bedingt war sein können. Sam mag Musik, **wenn** es gitarrenlastige Musik ist. Beachten Sie, dass Musik groß geschrieben ist, denn es ist eine Variable in dieser Regel.
Wenn wir jetzt fragen:

```
?- mag(sam,grunge).
```

wird PROLOG mit "yes" antworten, zwar steht dieser Fakt nicht in KB2, aber aus der Rule **mag(sam,Musik) :- gitarrenlastig(Musik).** und dem Fakt **gitarrenlastig(grunge).** kann PROLOG mithilfe von Resolution schließen, dass **mag(sam,grunge).** Wie das genau funktioniert betrachten wir später.

1.3. Knowledge base 3 (KB3)

```
musik(klassik).
musik(rock).
musik(schlager).
musik(country).
laut(rock).
mag(sam,Etwas):-
        musik(Etwas),
        laut(Etwas).
mag(sam,schlafen).
```

Unsere erste Regel über die Vorlieben von Sam hat hier zwei Aussagen in ihrem Körper, oder um mit PROLOG zu sprechen, zwei goals. Entscheidend ist das Komma zwischen **musik(Etwas)**und **laut(Etwas).** Es entspricht einer logischen *Konjunktion*, also einem **und** in PROLOG. Also mag Sam etwas nur, wenn es Musik ist **und** wenn es laut ist.
Also ergibt:

```
?- mag(sam,country).
```

ein "No", da KB3 nicht die Information enthält, **laut(country).**
Die beiden Aussagen sind ein Beispiel für eine *Disjunktion*, da sie beide Aussagen über die Sachen machen, die Sam mag. Er mag also laute Musik **oder** Schlafen.
Regeln, die den gleichen Kopf haben werden also durch *oder* verknüpft.
Eine andere Möglichkeiten eine *Disjunktion* auszudrücken wäre:

```
mag(sam,Musik) :-
        schnell(Musik);
        laut(Musik).
```

Sie sehen also, PROLOG hat etwas mit Logik zu tun. Ein **:-** bedeutet Implikation, ein **,** Konjunktion und ein **;** (oder zwei Regeln mit der gleichen Folgerung) Disjunktion. Außerdem spielt ein klassisches logisches Beweisgesetz (Resolution) eine große Rolle im PROLOG Programmieren. Und tatsächlich: PROLOG bedeutet Programmieren in Logik.

1.4. Knowledge base 4 (KB4)

```
chinesisch(chow_mein).
chinesisch(chop_suey).
chinesisch(suess_sauer).
mag(sam,chips).
mag(sam,chow_mein).
mag(peter,chow_mein).
mag(qu,chop_suey).
```

Eine sehr langweilige Knowledge base. Interessant sollen hier die queries sein. Wir möchten gerne erstmalig Variablen benutzen.

?- chinesisch(X).

Das X ist eine Variable (Alle Wörter mit großen Anfangsbuchstaben sind das) Die query fragt also PROLOG, sag mir, welches Essen, das Dir bekannt ist, chinesisch ist.
Dazu arbeitet sich PROLOG durch KB4 von oben nach unten durch und versucht die Information chinesisch(X) mit den Informationen von KB4 in Übereinstimmung zu bringen. PROLOG bindet dann (instaziiert) X und chow_mein und antwortet uns:

X = chow_mein

Nicht nur, dass es Informationen über chinesisches Essen in KB4 gibt, (kein simples „yes“) uns wird sogar eine Instanziierung, bzw. Variablen-Bindung gezeigt.
Aber Variablen können für verschieden Dinge stehen, und es gibt Information über weiteres chinesiches Essen in KB4, denn Eclipse schreibt „More“ in die Ausgabe. Wir drücken auf die Taste „More“. Dieses entspricht also der querie: *Gibt es noch weiteres?* Antwort:

X = chop_suey

Nach einer weiteren Nachfrage:

X = suess_sauer

Nun steht kein more mehr in der Ausgabe, da es keine weiteren Ergebnisse gibt..

Versuchen wir eine kompliziertere Querie:

?- mag(sam,X), chinesisch(X).

Also *gibt es Essen, dass Sam mag **und** das chinesisch ist?* Ja, Prolog kann es finden:

X = chow_mein

Variablen an Informationen in einer Knowledge base zu binden ist der Kern von PROLOG.

1.5. Knowledge base 5 (KB5)

liebt(othello,desdemona).
liebt(desdemona,jago).
liebt(romeo,julia).
liebt(julia,romeo).

eifersuechtig(X,Y) :- liebt(X,Z), liebt(Z,Y).

Hiermit wird ein Konzept von eifersuechtig definiert. Das ist eine generelle Aussage.
Also:

?- eifersuechtig(othello,P).

gibt es ein Individuum P, so daß Othello eifersuechtig auf P ist?

P = jago

Aufgabe:

1. Betrachten Sie KB5.
Gibt es weitere eifersüchtige-Individuen in KB5?
Mit welcher Querie würde uns PROLOG alle eifersüchtigen-Individuen in KB5 nennen?

2.Repräsentieren Sie folgendes in PROLOG:

a) Rincwind ist ein Zauberer
b) Zwerge und Elben mögen sich nicht
c) Wenn zwei verheiratet sind, mögen sie sich.
d) Otto mag niemanden, der ihn dämlich nennt und Engländer ist.
e) Hobbits essen alles, was nahrhaft oder schmackhaft ist

3. Wir haben folgende knowledge base:

```
wizard(ron).
hasWand(harry).
quidditchPlayer(harry).
wizard(X) :- hasBroom(X),hasWand(X).
hasBroom(X) :- quidditchPlayer(X).
```

Was antwortet PROLOG auf folgende Queries?

a) **wizard(ron).**
b) **witch(ron).**
c) **wizard(hermione).**
d) **witch(hermione).**
e) **wizard(harry).**
f) **wizard(Y).**
g) **witch(Y).**