Künstliche Intelligenz - Übung 8

Julian Dobmann, Kai Kruschel

Aufgabe 1 Meta-Interpreter

Die einfachste Form einen Meta-Interpreter zu schreiben, ist es einfach das Systemprädikat `call/1` in ein selbst definiertes Prädikat zu kapseln.

```
cheato(A):-call(A).
```

Dieser "Interpreter" kuckt allerdings seine Argumente nichtmal an, sonder leitet sie einfach an den eigentlichen Prolog-Interpreter weiter und lässt diesen die ganze Arbeit machen.

Das Prädikat `solve/1` soll wenigstens Konjunktionen und Disjunktionen selbst bearbeiten können. Dazu muss man einen Anker, die Behandlung von Prädikaten und von Kon- und Disjunktionen definieren:

```
% Anker
solve(true).

% Konjunktionen
solve((A,B)) :-
    solve(A), solve(B).

% Disjunktionen
solve((A;B)) :-
    solve(A);
    solve(B).

% Prädikate
solve(A) :-
    clause(A, Body),
    solve(Body).
```

Dieser Interpreter benutzt nur das Systemprädikat `clause/2`, welches für ein *gültiges* Prädikat den Körper diese Prädikats zurückgibt, und bei ungültigen Prädikaten failt.

Aufgabe 1 Expertensysteme/Wissensbasis

a)

Um den `;` bzw. Oder-Operator in Wissensdatenbanken zu unterstützen, muss die `prove/2` Klausel erweitert werden, um nicht nur UND-Verknüpfungen, sondern auch ODER-Verknüpfungen zu unterstützen.

```
prove(true,_) :- !.

% augmentation for or-clause support in knowledge database
prove((Goal;Rest),Hist) :-
    prove(Goal,[Goal|Hist]);
    prove(Rest,Hist).
% end augmentation

prove((Goal,Rest),Hist) :- !,
    prov(Goal,[Goal|Hist]),
    prove(Rest,Hist).
prove(Goal,Hist) :-
    prov(Goal,[Goal|Hist]).
```

Jetzt sind Definitionen folgender Form in der Wissensdatenbank möglich:

```
bird(mallard) :-
family(duck),           % different rules for male
voice(quack),
head(green); head(mottled_brown); head(turquoise).
```

b)

siehe angehängte Datei `cars.nkb