

# Künstliche Intelligenz - Übung 2

Adrian Neumann (4140810) und Naja von Schmude (4127652)

8. Mai 2009

## Aufgabe 1

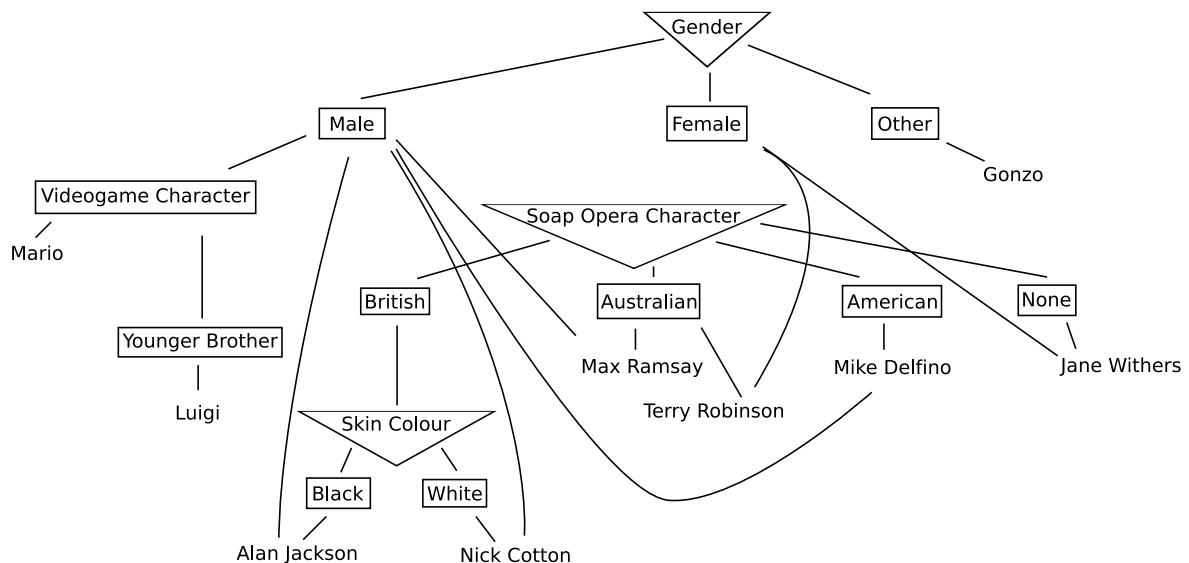
Sir, Kapitel 1 bis 3 erfolgreich gelesen, Sir.

## Aufgabe 2

a)

Als erstes muss natürlich eine Idee für ein Expertensystem vorhanden sein. Nach googeln und Wikipedia findet sich was passendes. Wie wäre es, *fiktive Klempner* zu klassifizieren?

Um die Übersicht zu behalten, wird zunächst per Hand ein Graph gemalt, der die Eigenschaften und Merkmale der einzelnen Charaktere beinhaltet und dann als Vorlage zum Umsetzen in Prolog dient.



Einmal alles zusammengetragen, kann die Wissensdatenbank einfach runtergeschrieben werden.

```
klempner(luigi) :- gender(male), videoGameCharacter, youngerBrother.
klempner(mario) :- gender(male), videoGameCharacter, \+youngerBrother.
klempner(gonzo) :- gender(other).
klempner(max_ramsay) :- gender(male), \+videoGameCharacter, soap_opera(
    australian), skin(white).
klempner(alan_jackson) :- gender(male), \+videoGameCharacter, soap_opera(
    british), skin(black).
klempner(nick_cotton) :- gender(male), \+videoGameCharacter, soap_opera(
    british), skin(white).
```

```

klempner(mike_delfino) :- gender(male), \+videoGameCharacter, soap_opera(
    american), skin(white).
klempner(happy_gilmore) :- gender(male), \+videoGameCharacter, skin(white).
klempner(terry_robinson) :- gender(female), soap_opera(australian).
klempner(jane_withers) :- gender(female), soap_opera(none).

```

Man kann sich jetzt überlegen, wie man das noch vereinfachen kann, und kommt dann darauf, dass man durch geschickten Einsatz von Cut-Operatoren und Umsortieren der Regeln sich ein paar Klauseln sparen kann.

```

klempner(luigi) :- gender(male), videoGameCharacter, youngerBrother,! .
klempner(mario) :- gender(male), videoGameCharacter,! .
klempner(gonzo) :- gender(other).
klempner(alan_jackson) :- gender(male), soap_opera(british), skin(black),! .
klempner(max_ramsay) :- gender(male), soap_opera(australian),! .
klempner(nick_cotton) :- gender(male), soap_opera(british),! .
klempner(mike_delfino) :- gender(male), soap_opera(american),! .
klempner(happy_gilmore) :- gender(male).
klempner(terry_robinson) :- gender(female), soap_opera(australian),! .
klempner(jane_withers) :- gender(female).

```

Als nächstes muss sich überlegt werden, wie man die Daten geschickt präsentiert und am besten dem Benutzer eine Vorauswahl von Antwortmöglichkeiten bereitet. Wir merken, dass wir dabei zwei Fälle unterscheiden müssen: Zum einen gibt es den Fall, wo es nur als Antwortmöglichkeit wahr oder falsch (z.B. `youngerBrother`) gibt, zum anderen gibt es eine Auswahl von gültigen Antwortmöglichkeiten (z.B. `gender`). Dafür braucht es also zwei verschiedene Funktionen.

```

isIta(X) :- known(X,yes),! .
isIta(X) :- ( % 1 < 3 :- (
    \+known(X,_),! ,
    write('Is it a '),
    write(X),
    write('? '),
    read(A),
    asserta(known(X,A)),
    A == yes
) .

askMoar(_,[],_) :- fail.
askMoar(Functor,_,X) :- known(Functor,X),! .
askMoar(Functor,List,X) :- (
    \+known(Functor,_),! ,
    write('Which of these '),
    write(List),
    write(' is its '),
    write(Functor),
    write(' ("none." if none)? '),
    read(A),
    member(A,[none|List]),
    asserta(known(Functor,A)),
    X=A
) .

```

`isIta` fragt dabei nur nach yes oder no und speichert sich dann gleich den übergebenen Wert für weitere Anfragen. Natürlich wird auch zuerst geprüft, ob wir bereits die Antwort kennen. `askMoar` fragt dann natürlich nach einer Antwort aus einer Liste von Antwortmöglichkeiten. Dabei wird auch

das neue Attribut none, zugelassen, wenn keine Möglichkeit der Liste passend ist. Jetzt fehlt nur noch ein Schritt, dass man den Möglichen Attributen die entsprechende Fragestellung zuweise:

```
gender(X) :- askMoar(gender,[male,female,other],X).
videoGameCharacter :- isIta(videoGameCharacter).
youngerBrother :- isIta(youngerBrother).
soap_opera(X) :- askMoar(soap_opera,[australian,british,american],X).
skin(X) :- askMoar(skin,[black,white],X).
```

Jetzt ist das Expertensystem schon fast fertig. Damit es funktioniert, brauchen wir ganz oben noch die Zeile `:- dynamic(known/2)`.. Das Einfügen von `solve(X) :- abolish(known/2), asserta((known(_,_) :- fail)), klemperner(X)` ermöglicht es denn mehrere Anfragen hintereinander zu stellen, so dass dazwischen das gesammelte Wissen bereinigt wird.

b)

Das System wird nun ausgiebig getestet:

```
?- solve(X).
Which of these [male, female, other] is its gender ("none." if none)? male.
Is it a videoGameCharacter ? no.
Which of these [australian, british, american] is its soap_opera ("none." if none)? american.
X = mike_delfino.
```

```
?- solve(X).
Which of these [male, female, other] is its gender ("none." if none)? male.
Is it a videoGameCharacter ? no.
Which of these [australian, british, american] is its soap_opera ("none." if none)? australian
X = max_ramsay.
```

```
?- solve(X).
Which of these [male, female, other] is its gender ("none." if none)? male.
Is it a videoGameCharacter ? no.
Which of these [australian, british, american] is its soap_opera ("none." if none)? british.
Which of these [black, white] is its skin ("none." if none)? none.
X = nick_cotton.
```

Die beiden ersten Test sind wie erwartet, beim letzten kann man sich darüber streiten, ob überhaupt eine Ausgabe erfolgen sollte, wenn man als Hautfarbe none eingibt... Bei explizierter Angabe funktioniert es natürlich wie gewünscht.

```
?- solve(X).
Which of these [male, female, other] is its gender ("none." if none)? male.
Is it a videoGameCharacter ? no.
Which of these [australian, british, american] is its soap_opera ("none." if none)? british.
Which of these [black, white] is its skin ("none." if none)? white.
X = nick_cotton.
```

```
?- solve(X).
Which of these [male, female, other] is its gender ("none." if none)? male.
Is it a videoGameCharacter ? no.
Which of these [australian, british, american] is its soap_opera ("none." if none)? none.
X = happy_gilmore ;
```

false.

?- solve(X).

Which of these [male, female, other] is its gender ("none." if none)? male.

Is it a videoGameCharacter ? no.

Which of these [australian, british, american] is its soap\_opera ("none." if none)? british.

Which of these [black, white] is its skin ("none." if none)? black.

X = alan\_jackson.

?- solve(X).

Which of these [male, female, other] is its gender ("none." if none)? none.

false.

?- solve(X).

Which of these [male, female, other] is its gender ("none." if none)? other.

X = gonzo ;

false.

?- solve(X).

Which of these [male, female, other] is its gender ("none." if none)? female.

Which of these [australian, british, american] is its soap\_opera ("none." if none)? british.

X = jane\_withers.

?- solve(X).

Which of these [male, female, other] is its gender ("none." if none)? female.

Which of these [australian, british, american] is its soap\_opera ("none." if none)? none.

X = jane\_withers.

Hier ist die Ausgabe auch wieder fragwürdig ...

?- solve(X).

Which of these [male, female, other] is its gender ("none." if none)? female.

Which of these [australian, british, american] is its soap\_opera ("none." if none)? australian

X = terry\_robinson.

?- solve(X).

Which of these [male, female, other] is its gender ("none." if none)? male.

Is it a videoGameCharacter ? yes.

Is it a youngerBrother ? no.

X = mario.

?- solve(X).

Which of these [male, female, other] is its gender ("none." if none)? male.

Is it a videoGameCharacter ? yes.

Is it a youngerBrother ? yes.

X = luigi.

Bis auf die zwei etwas unglücklichen Lösungen funktioniert das System gut und zuverlässig. Verbessert könnte es dadurch, dass man z.B. noch Unsicherheiten bei den Eingaben zulässt. Natürlich wäre eine Vergrößerung der Datenbank um weitere Datenbestände sinnvoll, aber um vorzuführen, dass es funktioniert, ist diese Datenbank völlig ausreichend.