



Staffordshire University
—
Artificial Intelligence

Coursework
—
The Dating Application

Florian Seidel

November 28th, 2003

Content

CONTENT	2
1 INTRODUCTION	3
1.1 Purpose of the system	3
1.2 Knowledge Domain	3
1.3 Target User	4
1.4 Developing the Application	4
2 KNOWLEDGE REPRESENTATION	4
2.1 In General	4
2.2 Semantic Networks	5
2.2.1 Advantages	5
2.2.2 Limitations	5
2.3 The Dating Application	6
2.3.1 The Semantic Network	6
2.3.2 The Reason	6
3 IMPLEMENTATION IN PROLOG	7
3.1 Program Structure	7
3.1.1 Knowledge Base	7
3.1.2 Main Functions	7
3.1.3 Auxiliary Functions	8
3.2 User Interface	9
3.3 Extending the program	9
4 TEST RUNS	10
4.1 Test Run No. 1	10
4.2 Test Run No. 2	11
4.3 Test Run No. 3	12
5 SOURCE CODE	13

1 Introduction

Sometimes, when you walk into a cafe you see a man sitting there all by himself with a red rose in the chest pocket of his jacket, smiling at every woman that passes the door. He is most likely waiting for his ‘blind date’. And as I believe, it must be quite a horrible situation for this man, not knowing what the other person looks like or who it could be.

Wouldn’t it be much nicer and much more relaxing, if you knew what to expect of your date? For this reason, I decided to develop an intelligent “dating application” that helps you to find a person looking for a date, who you know matches your ‘taste’.

1.1 Purpose of the system

So the purpose of the system is actually exactly that. It aims to get lonely people in touch with one another.

1.2 Knowledge Domain

The knowledge domain of the Dating Application lies in matching the user’s preferences with the profiles of people in the database.

The user has to enter his/her gender that tells the system, which gender he/she is looking for. Possible entries are either ‘male’ or ‘female’. The date’s gender can be determined, since the application is designed for heterosexual users only. (This is not meant to be discriminating against anybody not heterosexual. The application is extensible, so this might be a feature of the second version.)

Getting the user’s preference of age, the system can determine in which age category is bearable for the user to meet with, since it is unlikely that there is a person registered with exactly the age the user prefers. The people’s ages are grouped into the following categories:

- ‘young’ for 18- to 24-year-olds,
- ‘grown-up’ for 25- to 34-year-olds,
- ‘mid-aged’ for 35- to 49-year-olds,
- ‘experienced’ for 50- to 69-year-olds, and
- ‘wise’ for 70- to 105-year-olds.

Finally the user has to enter the hair colour his/her date should have and then a person that matches all these requirements is determined by the system. The hair colour has to be chosen from a list, displaying all the hair colours registered in the system.

The output of the system is the name of the date and his/her telephone number, so the user can get in touch with him/her right away.

The system is intelligent is so far that some degree of reasoning is implemented within the search mechanism. For example, if the user states to be male, the system reasons that the user is looking for a woman, since men seek women by nature. In a more schematic and PROLOG-similar way this look like:

is(user,male) **AND** seeks(male,female) → seeks(user,female).

1.3 Target User

The target users are heterosexual men and women between the age of 18 years and 105 years.

Underage people are not included in this search program to prevent paedophiles finding out information about possible victims. Secondly, sexual intercourse among underage people is only allowed with their parents' permission, which cannot be verified by this program.

Still it might seem unusual that people up to the age of 105 years are included in the search, but since the elderly shall not be discriminated against in any way, they should have the possibility to enter their data into the knowledge base as well as anybody else.

1.4 Developing the Application

The Application is programmed in PROLOG with SWI-PROLOG.

SWI-PROLOG is a free software PROLOG compiler, downloadable for free at <http://www.swi-prolog.org/>.

2 Knowledge Representation

2.1 In General

Knowledge representation is a key stone in the field of Artificial Intelligence. The knowledge and the knowledge domain that a system is dealing with can be expressed in several different ways.

Semantic networks, predicate logic, frames, and rule-based systems are possible knowledge representation schemes, each of which has some advantages but also limitations.

Also a hybrid representation scheme can be chosen, which involves a mixture of those techniques.

2.2 Semantic Networks

A semantic network consists of nodes, which represent concepts in the real world, and directed links, which represent relationships between concepts. Those directed links can describe general relations like similarities, specific relations indicating a hierarchy, property relations highlighting typical characteristics of a concept such as the gender of a user, or even more complex relations such as a user preferring a hair colour.

2.2.1 Advantages

The main advantages of semantic networks over alternative knowledge representation schemes are

- that they are easy to understand,
- that they are good at representing factual knowledge,
- that they show inheritance and infer through links,
- their flexibility in adding new nodes,
- that they are closer to human information storing, and
- that they reduce search time as nodes are connected to related nodes.

2.2.2 Limitations

The downside of semantic networks is that

- it is not a standardised knowledge representation technique,
- exceptions of inheritance are difficult to handle,
- sequential and temporal facts are difficult to express, and
- thus it is difficult to handle procedural knowledge or negations.

2.3 The Dating Application

2.3.1 The Semantic Network

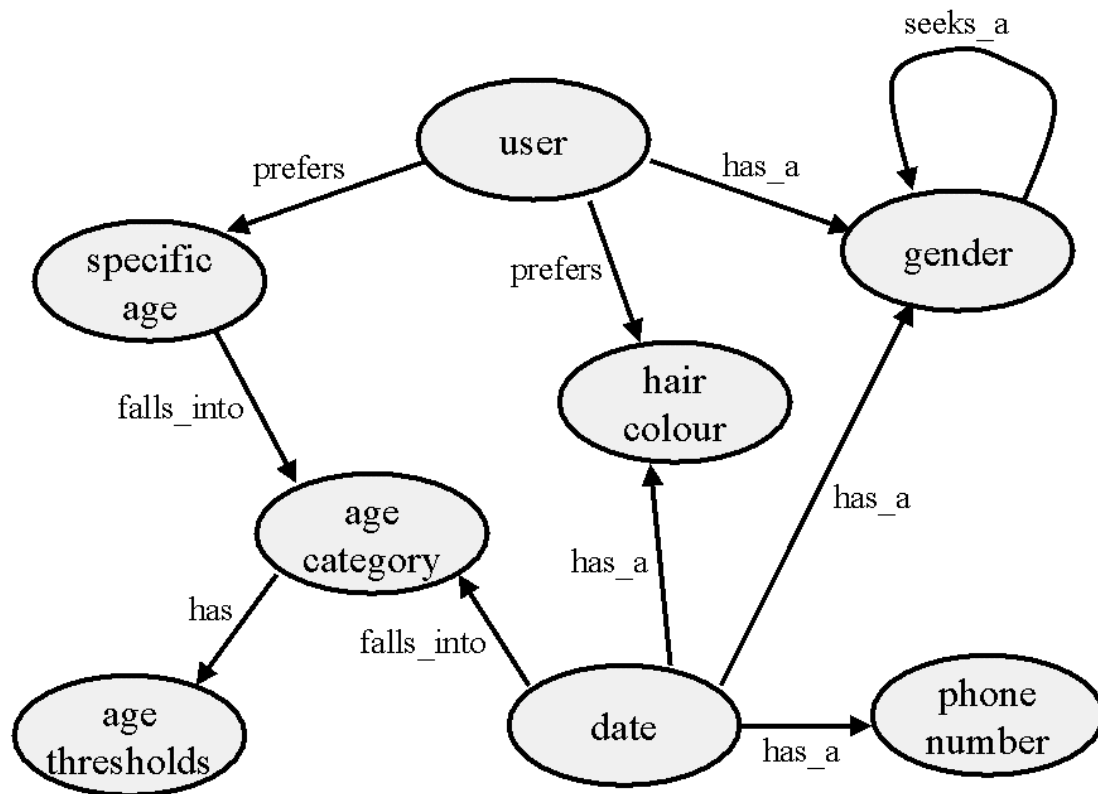


Figure 1: Semantic Network

2.3.2 The Reason

The main reason why I chose the semantic network for the Dating Application is that it is good at representing factual knowledge. Since in the

As two additional, even though less important arguments for the semantic network in this case were on the one hand the easy way of adding nodes and thus expanding the network and accordingly extending the program, and on the other hand the simplicity of the semantic network itself, as it helped me to stay focussed as I was a newcomer to programming in PROLOG.

3 Implementation in PROLOG

3.1 Program Structure

The program can be divided into three main sections:

- the knowledge base,
- the main functions, and
- the auxiliary functions.

3.1.1 Knowledge Base

The knowledge base obviously relates most closely to the semantic network, as it is the schematic representation of the domain knowledge.

The knowledge base defines the people, who are registered as ‘available dates’ in the system. It defines a date’s names, gender, age category, hair colour, and phone number.

It also defines that men seek women as dates and vice versa and sets the thresholds for the age categories, which are needed to assign the specific age a user is looking for to an age category, which the registered people belong to.

3.1.2 Main Functions

The main functions include all functions that are called by the core function `go`. These are also the only functions that interact with the user.

`go` is the function that needs to be executed to start up the application.

`presentWelcomeFrame` displays a nice-looking “Welcome” frame to give the application a good ‘first impression’ in the eye of the user.

`getGenderInput` then asks the user, if he/she is male or female. If the user answers this question correctly, the sought gender of the date is determined, then made persistent in `TargetGender` and afterwards the program would exit `getGenderInput`. But if the user does not enter ‘male’ or ‘female’, the system would display the appropriate error message and then call `getGenderInput` again.

`getAgeInput` asks the user how old his/her date should approximately be allowing a range from 18 to 105 years old. If the user entry is actually a number and if it is within this range, the method calculates in age category the entered age lies and stores the entry in the

variable `AgeCategory`. If the entry was bad, another appropriate error message is displayed and the function calls itself again.

`getHairColourInput` asks the user what hair colour he/she would prefer his/her date to have. The special feature about this function is that it displays a list of all so far registered hair colours, from which the user has to choose one. This list is handled by the auxiliary function `writeHairColours` as described below. Again if the input is one of the registered hair colours, the function stores the value and exits `getHairColourInput`. If the user did not manage to enter one of the correct hair colours, he will be displayed an according error message and the function calls itself again.

`presentResults` presents the results. What a surprise! To do so, it receives determined gender, age category and hair colour of the date and passes this data on to the auxiliary function `findResults`. Getting back a list from `findResults` called `Results` containing all the names of the people that matched the search criteria, `presentResults` summarizes the search criteria, outputs the names and their phone numbers in a neat format and then exits `presentResults`. If `findResults` returned an empty list, `presentResults` informs the user about the disappointing news and exits `presentResults` as well.

The `go` function is rounded up by a nice “Thank you for using...” phrase and the program is done.

3.1.3 Auxiliary Functions

`writeHairColours` first of determines all the hair colours that are registered in the knowledge base and puts them into the list `AllHairColours`. This list is then cleaned from duplicates by the auxiliary function `removeRep` (see below) and then writes all the elements of the cleaned list using the `write_elt` auxiliary function (see below).

`removeRep` receives a list that is most likely to have duplicate entries. `RemoveRep` removes those duplicate entries and stores only one of each entry in a second list, which it returns again.

`findResults` is the most important of the auxiliary functions, as it pulls together all the data derived from the user entries. It gets the name of the first person in the knowledge base that matches the desired age category, checks if this person has the desired hair colour and if that matches it checks if this person also has the desired gender. If all the data matched with one person in the knowledge base, the `findResults` returns this person’s name. If one of the arguments failed, it returns an empty variable and thus no result.

`write_elt` is a small auxiliary function that prints out the elements of a list vertically.

`writeGender` prints out either ‘man’ if the gender is determined as ‘male’ and writes ‘woman’ if the gender is ‘female’.

`writeNames` prints out the list of names that resulted from `findResults` and adds each person’s phone number after the person’s name combined in a nice layout.

3.2 User Interface

In SWI-PROLOG (having consulted the `.pl` file):

Enter “`go.`”. That should start the Dating Application performing the first steps described in section 3.1.2.

While enjoying the program, keep in mind to finish each entry with a full stop, e.g. when asked for my gender, I would have to answer “male.” <ENTER> for SWI-PROLOG to understand what I am saying.

Else than that, the user interface work very intuitive and user has to follow only the questions and answers on the screen. Using some common sense, handling this program is a piece of cake! 😊

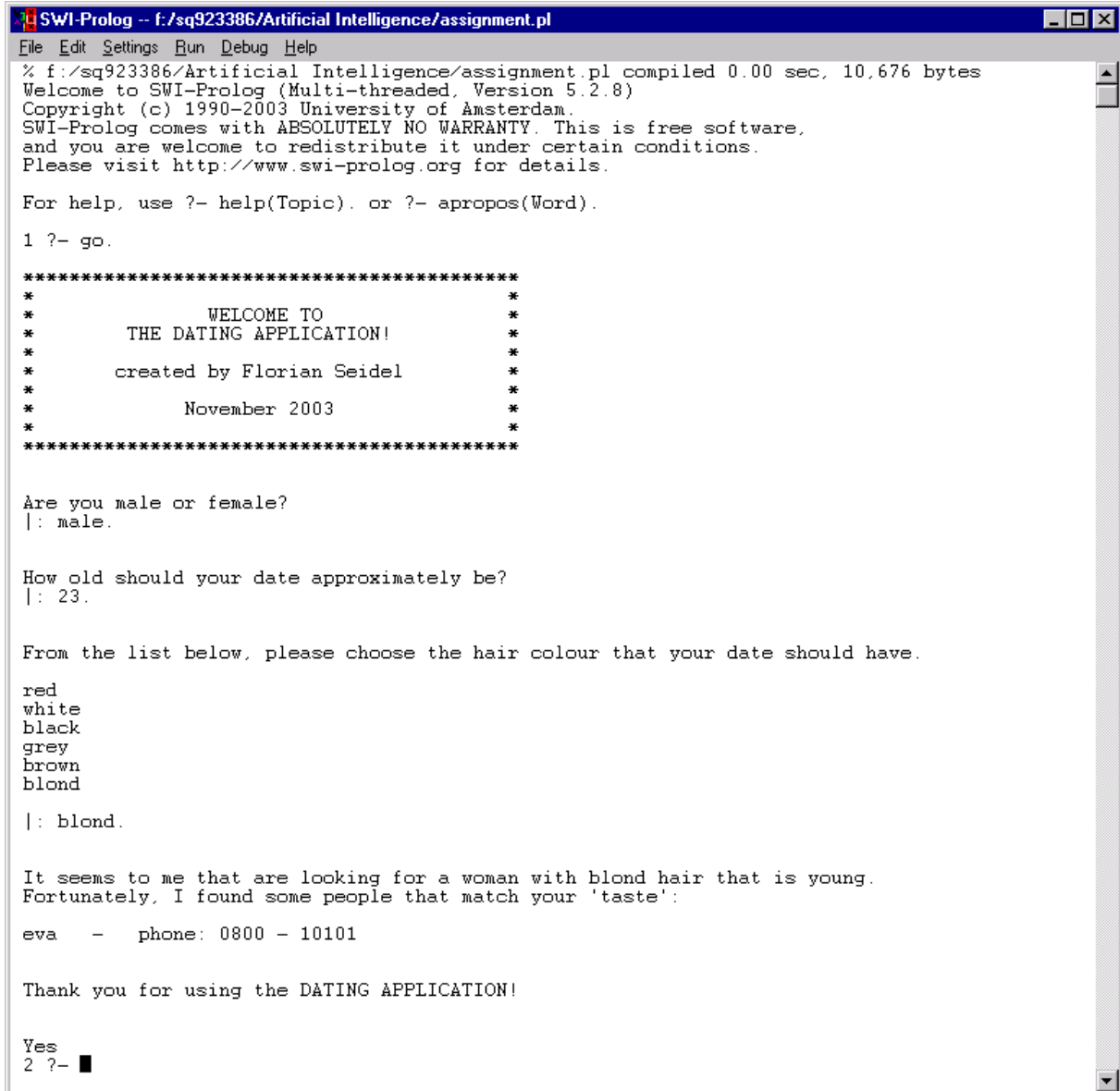
3.3 Extending the program

To add more people to the knowledge base, simply add another rule for `gender`, `age`, `hairColour`, and `phone` in the same way the other rules are defined.

To add more criteria to define your ‘perfect’ date, you have to create a new rule for each criterion defining it for every person already in the knowledge base. Additionally you have to implement a function handling the user input and add this function to the `go` routine.

4 Test Runs

4.1 Test Run No. 1



```

SWI-Prolog -- f:/sq923386/Artificial Intelligence/assignment.pl
File Edit Settings Run Debug Help
% f:/sq923386/Artificial Intelligence/assignment.pl compiled 0.00 sec, 10,676 bytes
Welcome to SWI-Prolog (Multi-threaded, Version 5.2.8)
Copyright (c) 1990-2003 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- go.

*****
*                               *
*           WELCOME TO         *
*     THE DATING APPLICATION!   *
*                               *
*   created by Florian Seidel   *
*                               *
*           November 2003      *
*                               *
*****

Are you male or female?
|: male.

How old should your date approximately be?
|: 23.

From the list below, please choose the hair colour that your date should have.

red
white
black
grey
brown
blond

|: blond.

It seems to me that are looking for a woman with blond hair that is young.
Fortunately, I found some people that match your 'taste':

eva - phone: 0800 - 10101

Thank you for using the DATING APPLICATION!

Yes
2 ?- █
  
```

Figure 2: Test Run 1

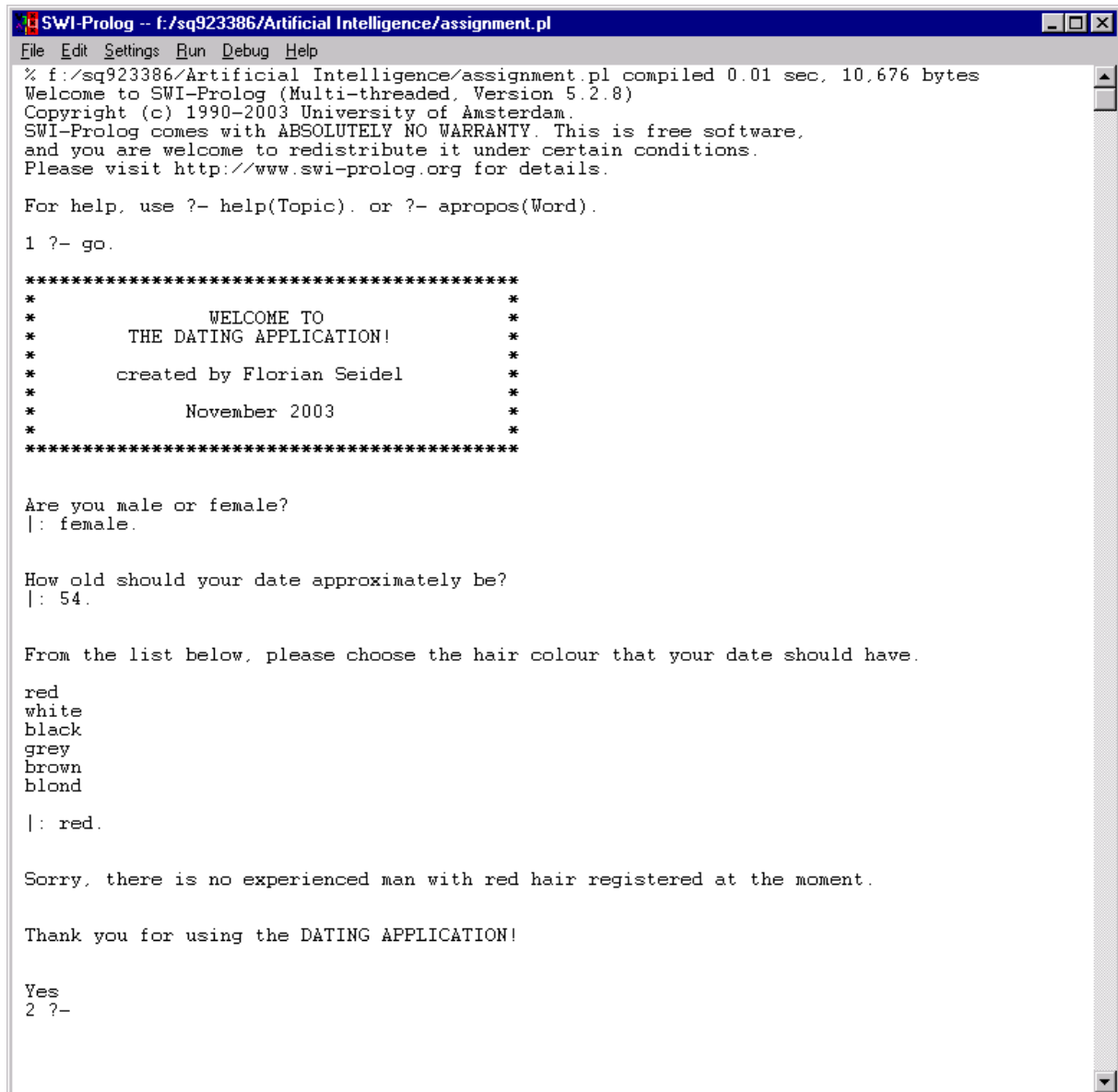
The first test run (see Figure 2, above) shows a perfect example of a man looking for a blond woman in her mid-twenties.

After the `go.` command is called, the welcome frame is displayed and the user claims to be a man when asked for his gender. Afterwards he specifies that he is looking for a 23-year-old woman with blond hair.

Since Eva in this case matched all the criteria (woman, young and blond), her name and her phone number is displayed.

The Dating Application then finishes with its standard closing phrase.

4.2 Test Run No. 2



```

SWI-Prolog -- f:/sq923386/Artificial Intelligence/assignment.pl
File Edit Settings Run Debug Help
% f:/sq923386/Artificial Intelligence/assignment.pl compiled 0.01 sec, 10,676 bytes
Welcome to SWI-Prolog (Multi-threaded, Version 5.2.8)
Copyright (c) 1990-2003 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- go.

*****
*                                     *
*           WELCOME TO               *
*      THE DATING APPLICATION!       *
*      created by Florian Seidel     *
*           November 2003            *
*                                     *
*****

Are you male or female?
|: female.

How old should your date approximately be?
|: 54.

From the list below, please choose the hair colour that your date should have.
red
white
black
grey
brown
blond
|: red.

Sorry, there is no experienced man with red hair registered at the moment.

Thank you for using the DATING APPLICATION!

Yes
2 ?-
  
```

Figure 3: Test Run 2

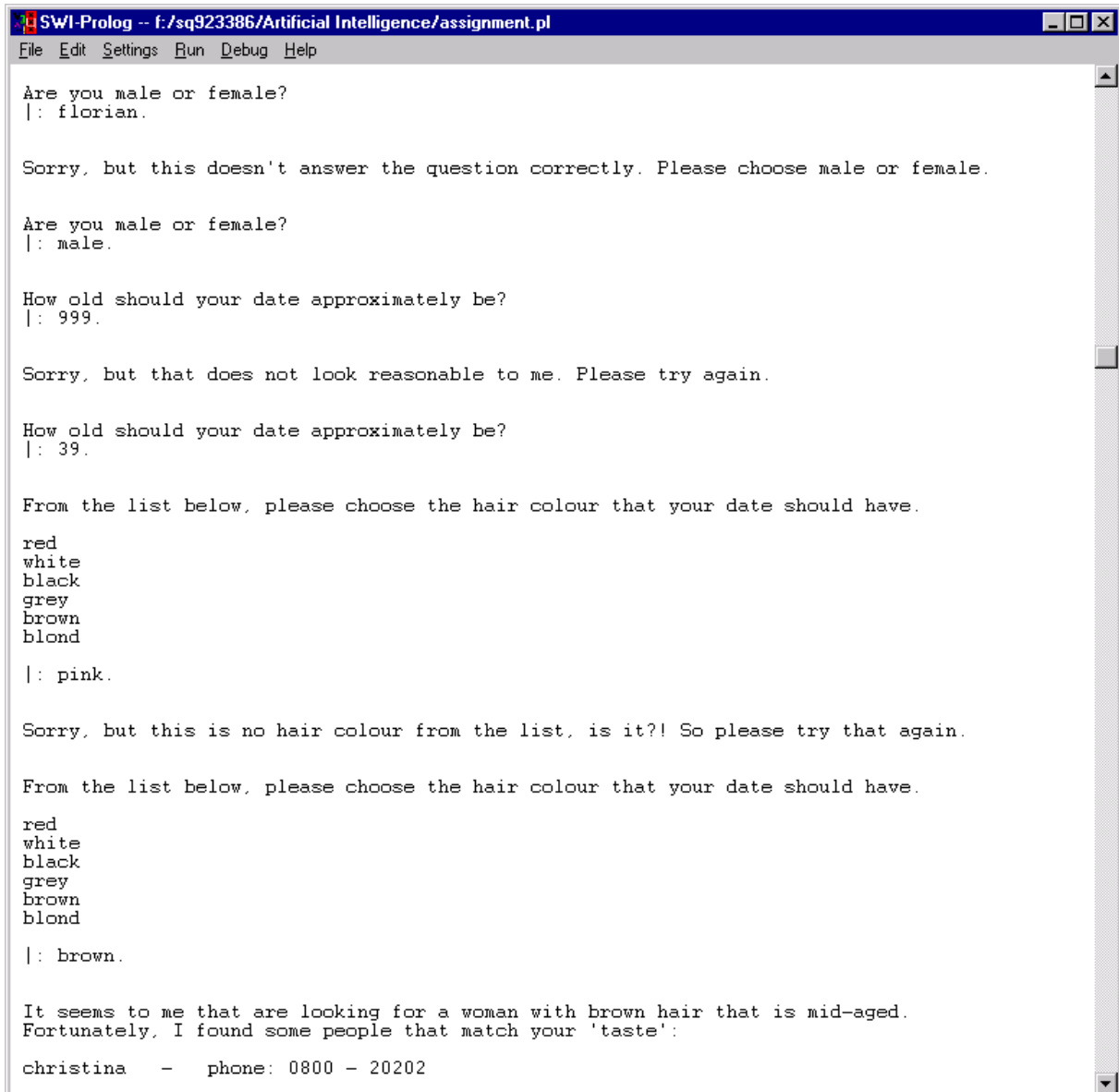
In the second test run (see Figure 3, above) another search without error is performed, but this time the search was not successful.

In this case a woman is looking for a 54-year-old gentleman with red.

Since these quite unusual facts do not result in a successful search, the user gets the disappointing message that there is no man with red hair in the age range of the “experienced” registered in the knowledge base.

The Dating Application then finishes with its standard closing phrase.

4.3 Test Run No. 3



```
SWI-Prolog -- f:/sq923386/Artificial Intelligence/assignment.pl
File Edit Settings Run Debug Help

Are you male or female?
|: florian.

Sorry, but this doesn't answer the question correctly. Please choose male or female.

Are you male or female?
|: male.

How old should your date approximately be?
|: 999.

Sorry, but that does not look reasonable to me. Please try again.

How old should your date approximately be?
|: 39.

From the list below, please choose the hair colour that your date should have.
red
white
black
grey
brown
blond

|: pink.

Sorry, but this is no hair colour from the list, is it?! So please try that again.

From the list below, please choose the hair colour that your date should have.
red
white
black
grey
brown
blond

|: brown.

It seems to me that are looking for a woman with brown hair that is mid-aged.
Fortunately, I found some people that match your 'taste':

christina - phone: 0800 - 20202
```

Figure 4: Test Run 3

In the third test run (see Figure 3, above) all the error the error messages are triggered.

In this case, the user entered the programmer's first name, when asked for his/her gender, which triggered the first error message and called the first query function again. Then he entered his correct gender and the program proceeded with the next question.

Then the user pretended to be looking for a 999-years-old date, which is non-sense and thus triggered the second error message. Then he entered the valid age of 39 and the program proceeded with the next question.

Then the user wanted his date to have pink hair, although there is nobody registered with pink hair, so the system displayed the third error message and asks again. Then he selected the hair colour brown from the list and the program proceeded with the results.

Finally the user gets Christina's phone number and the Dating Application finishes with its standard closing phrase.

5 Source Code

```

/*****/
/*                               The Knowledge Base                               */
/*****/

/*  Defines each person's gender.  */

    gender(eva, female).
    gender(christina, female).
    gender(victoria, female).
    gender(lisa, female).
    gender(kelly, female).
    gender(james, male).
    gender(richard, male).
    gender(david, male).
    gender(tony, male).
    gender(george, male).

/*  Defines each person's age.  */

    age(eva, young).
    age(christina, mid-aged).
    age(victoria, grown-up).
    age(lisa, experienced).
    age(kelly, wise).
    age(james, experienced).
    age(richard, wise).
    age(david, grown-up).
    age(tony, mid-aged).
    age(george, young).

/*  Defines each person's hair colour.  */

    hairColour(eva, blond).
    hairColour(christina, brown).
    hairColour(victoria, red).
    hairColour(lisa, blond).
    hairColour(kelly, white).
    hairColour(james, black).
    hairColour(richard, grey).
    hairColour(david, blond).
    hairColour(tony, brown).

```



```

write('*          November 2003          *'), nl,
write('*          *'), nl,
write('*****'), nl,

/* Ask for the user's gender. */

getGenderInput(TargetGender):-
    nl,nl,
    write('Are you male or female?'),nl,
    read(GenderInput),
    seeks(GenderInput,TargetGender);
    nl,nl,
    write('Sorry, but this doesn\'t answer the question correctly.
        Please choose male or female. '),nl,
    getGenderInput(TargetGender).

/* Asks for the age the date should have. */

getAgeInput(AgeCategory):-
    nl,nl,
    write('How old should your date approximately be?'), nl,
    read(Age),
    integer(Age),
    ageThresholds(AgeCategory,LowerThreshold,UpperThreshold),
    Age >= LowerThreshold,
    Age < UpperThreshold;
    nl,nl,
    write('Sorry, but that does not look reasonable to me.
        Please try again. '),nl,
    getAgeInput(AgeCategory).

/* Asks for the hair colour that the date should have. */

getHairColourInput(HairColour):-
    nl,nl,
    write('From the list below, please choose the hair colour that
        your date should have. '),nl,
    writeHairColours,
    read(HairColour),
    hairColour(_,HairColour);
    nl,nl,
    write('Sorry, but this is no hair colour from the list, is it?!
        So please try that again. '),nl,
    getHairColourInput(HairColour).

/* Present the results to the user. */

presentResults(Gender,Age,HairColour):-
    findall(Name, findResults(Name,Gender,Age,HairColour),Results),
    not(Results = []),
    removeRep(Results, ResultsRemRep),
    nl,nl,
    write('It seems to me that are looking for a '),
    writeGender(Gender),

```

```

write(' with '),
write(HairColour),
write(' hair that is '),
write(Age),
write('. '),nl,
write('Fortunately, I found some people that match
      your \'taste\': '),nl,nl,
writeNames(ResultsRemRep),nl,nl;
nl,nl,
write('Sorry, there is no '),
write(Age),
write(' '),
writeGender(Gender),
write(' with '),
write(HairColour),
write(' hair registered at the moment. '),nl,nl,nl.

/*****
/*                               The Auxiliary Functions                               */
*****/

/*  Writes a list of all registered hair colours.  */

writeHairColours:-
    findall(HairColour, hairColour(_, HairColour), AllHairColours),
    removeRep(AllHairColours, AllHairColoursRemRep),
    nl, write_elt(AllHairColoursRemRep), nl, !.

/*  Removes entry repetitions in a list.  */

removeRep([], []).
removeRep([H| T1], T2):-
    member(H, T1),
    removeRep(T1, T2).
removeRep([H| T1], [H| T2]):-
    not(member(H, T1)),
    removeRep(T1, T2).

/*  Finds the names of the people that match the user's criteria.  */

findResults(Name, Gender, Age, HairColour):-
    age(Name, Age),
    hairColour(Name, HairColour),
    gender(Name, ResultGender),
    Gender == ResultGender.

/*  Writes elements of a list in vertical order.  */

write_elt([]).
write_elt([H|T]):- write(H), nl, write_elt(T).

```

```
/* Writes the fitting term for the fitting gender. */

writeGender(Gender):-
    Gender == male,
    write('man');
    Gender == female,
    write('woman').

/* Writes the list of names and their telephone numbers. */

writeNames([]).
writeNames([H|T]):-
    write(H),
    write(' - phone: 0800 - '),
    phone(H,Phone),
    write(Phone),nl,
    writeNames(T).
```