



Staffordshire University

—

# **Artificial Intelligence Coursework**

**Boozer Choozer**

**the intelligent decision supporting system**

Steffen Diehl

November 27, 2003

---

**Contents**

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Purpose of the system and its domain knowledge . . . . .	1
<b>2</b>	<b>Knowledge representation of domain knowledge</b>	<b>1</b>
2.1	Knowledge representation - In common . . . . .	1
2.2	Knowledge representation - The Boozer Choozer application . . . . .	1
2.3	Knowledge representation - The Semantic Network . . . . .	2
<b>3</b>	<b>Knowledge representation in the case of Boozer Choozer</b>	<b>3</b>
3.1	Limitations: . . . . .	3
<b>4</b>	<b>Implementation in PROLOG and descriptions</b>	<b>4</b>
4.1	Program Structure . . . . .	4
4.2	Walkthrough with focus on functions . . . . .	5
4.3	Important rules . . . . .	5
4.4	Check functions . . . . .	6
4.5	User Interface . . . . .	6
4.5.1	Walkthrough . . . . .	6
<b>5</b>	<b>Appendix</b>	<b>I</b>
5.1	Testruns . . . . .	II
5.2	Sourcecode . . . . .	V

## List of Figures

1	Semantic Network - with example nodes . . . . .	2
2	Testrun - Search by boozer . . . . .	II
3	Testrun - with a wrong input . . . . .	III
4	Testrun - Search by taste . . . . .	IV

# 1 Introduction

Everybody knows the problem:

You are in a foreign city and want to go out but what pub or club fits your taste?

The Boozer Choozer application is the solution to this problem.

## 1.1 Purpose of the system and its domain knowledge

The purpose and target of the Boozer Choozer system is helping people to find a club, bar, disco or pub in a specific city which fits their taste.

It is an application implemented in PROLOG using SWI-Prolog [1].

The Boozer Choozer is supposed to be an intelligent decision supporting system.

The systems domain knowledge lies in the localization of pubs and knowing their taste and even insider tips for everyone of them.

It is stored in definitions which are made for every object. The classification is done via the city, the boozer itself, its taste and the insider tips for it. Detailed information on that can be found in the next chapter.

## 2 Knowledge representation of domain knowledge

### 2.1 Knowledge representation - In common

The knowledge representation is a keystone in Artificial Intelligence.

Knowledge can be represented in different ways. Networks schemes e.g. semantic networks (SD) are one type of them. Structured representation schemes are e.g. frames.

### 2.2 Knowledge representation - The Boozer Choozer application

The knowledge representation chosen in this project is a semantic network (SN).

A Semantic Network consists of nodes and links.

The nodes represent the concepts which are in this case a city, a boozer, a taste or a tip.

The links represent the relations between the concepts. They are directed to indicate the direction

## 2.3 Knowledge representation - The Semantic Network

of the relationship.

With these definitions it is possible to use the inheritance of semantic networks. By that attributes, defined for a city, can be inherited from all the clubs in this city e.g. the city name for the attribute *liesIn* and the attribute 'is dangerous'.

### 2.3 Knowledge representation - The Semantic Network

The objects or concepts of the Boozer Choozer network are Country, City, Boozer, Taste and Tip. The relations are: *isBoozerIn*, *suitstaste* and *tip* what is visualized in the following diagram.

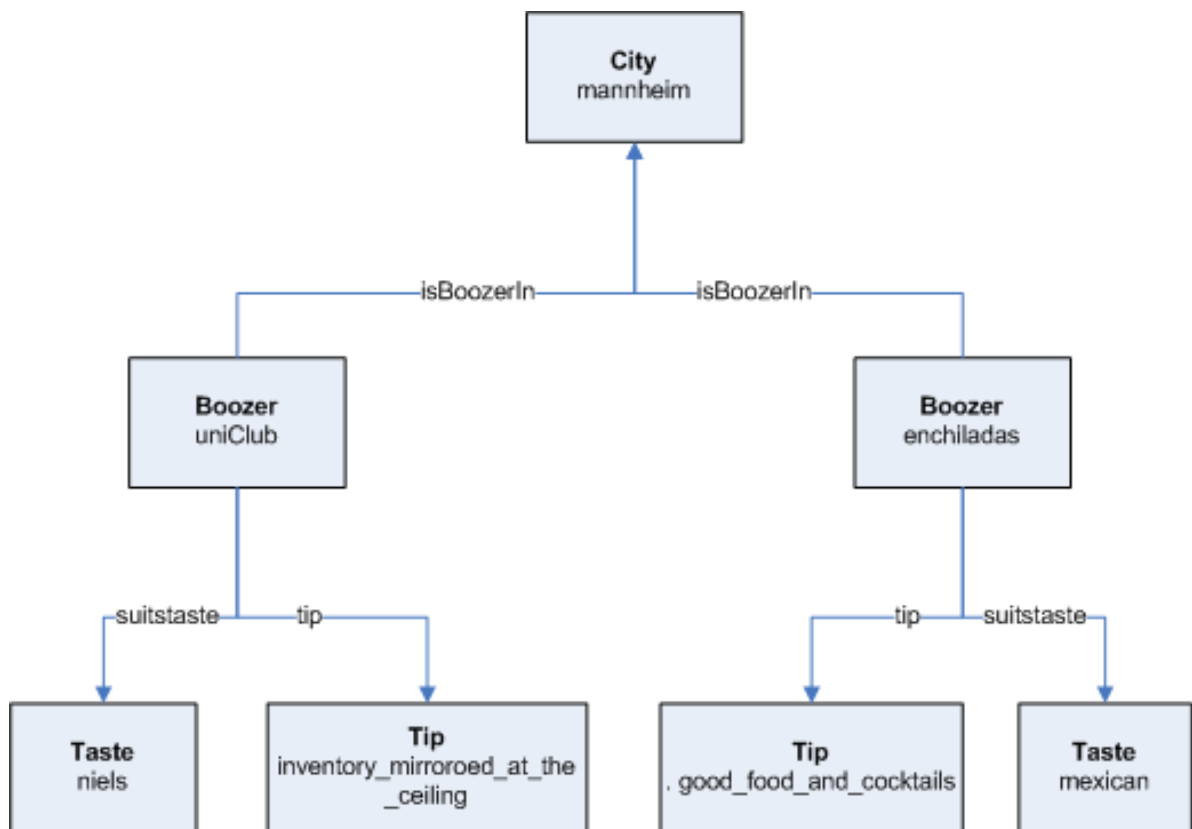


Figure 1: Semantic Network - with example nodes

### 3 Knowledge representation in the case of Boozer Choozer

The Semantic Network is seen as the best representation type for the objects of the Boozer Choozer program and their relations.

Even if there is no standard representation style of semantic networks, the net of the Boozer Choozer will be easy to understand. The relations are explicitly described and obvious.

There are clear hierarchies and relations which no one can misunderstand because everybody knows that a club has to be in a city and has a taste and that there can be a tip for it.

#### 3.1 Limitations:

The system limited in adding new functions respectively rules and questions. These have to be implemented additionally, what can be done by adding them into the main routine which calls all the subroutines or the specific sections of the PROLOG Code like the rules part. Clauses (boozer or additional attributes like a second taste) can be added very easy. They have to be added to the general rules section where all the cities and boozers with their taste and tip are defined. They are included in the search after a new compilation if the code. This can be done very easy because of the clear structure. It is obvious what to add and where.

## 4 Implementation in PROLOG and descriptions

The system was implemented by using the trial and error method combined with learning by doing methods.

There were a lot of versions produced, and a lot of "solutions" discarded. The most time took the check functions to ensure that the input is tested and the application can not be crashed.

### 4.1 Program Structure

The general structure is:

- Initialisation with welcome screen and choice menu
  - Start of the application
- The 2 main routines - *tasteChoose* and *favouriteChoose*
  - Main functions of the application.
  - Use general rules
- City rules
  - Rules to get city input
  - Check input for correctness
- General rules
  - rules used by *tasteChoose* and *favouriteChoose*
  - *tipHint* and *tipReq* for requestin tips to a found boozer
  - *write\_elt* for list output
  - *noRepetition* for removing duplicates from lists
  - the *boozerChoozer* rule
- General definitions
  - *isTasteln* to connect tastes with cities
  - boozer definitions

## 4.2 Walkthrough with focus on functions

This is a walkthrough for one possibility with focus on most important functions:

The start is initialised with `go`. After the `init` method, which displays a welcome screen, the menu is entered. Here a description of the 2 possible choices is printed and the input of the user is requested and processed.

The first option calls first the `getCityInput` method which prints out all cities in the database, requests a choice and checks it for correctness.

Then the `favouriteChoose` method which offers boozers in the chosen city is run. It gets the preferred boozers input, checks if it is valid with `checkBoozersChoice`, finds out its taste with `boozersChoozer` and searches for other boozers with the same taste using `findall`.

Finally it asks the user if he wants a tip for one of the results with `tipHint` and processes this with `tipReq`.

A testrun for the example above can be found in figure 2 in the appendix

## 4.3 Important rules

The rule `boozersChoozer` is the most important one.

It is used to bring together the relations `taste`, `boozers` and `city`.

```
boozersChoozer(Taste, City, Boozers) :- isBoozersIn(Boozers, City),
    suitstaste(Boozers, Taste).
```

The general definitions of the boozers with their rules:

```
isBoozersIn(citrus, mainz).
suitstaste(citrus, classical).
tip(citrus, great_location_and_nice_black_music_on_saturdays).
```

## 4.4 Check functions

By implementing check functions is avoided that a wrong input is processed. There will always be a response and the reoffering off the available choices.

Only input that consists of (small) letters and numbers followed by a comma can be identified wrong. Entering a ,ór something beginning with a capital is not caught.

Rules doing so are:

- *checkCityInput*

Checks if the city is in the definitions.

- *check88*

Guarantees that the taste that was read in is available in the city entered before by the user and requests a new try if not. There is also the way to enter dc for don't care in this part of the Choozer. *Check88* recognizes this and processes in another way that if a taste was chosen.

- *boozerCheck*

Is used to check if a tip was requested or if end was entered.

A testrun for a wrong input of a favourite boozer shown in figure 3 in the appendix.

## 4.5 User Interface

The user interface is clear structured and follows the function of the application. The user is always offered all available options to choose from.

By soing so should be avoided that he gets displeased when he enters 10 tastes that are not available in the database.

### 4.5.1 Walkthrough

A short walkthrough shows the ways through the application.

The user has the choice between searching (A) by boozer or (B) by taste.

In both options the user gets first a list of cities available in the database. The desired city can be chosen.

## 4.5 User Interface

---

- Option A

Choosing option A the user is presented all clubs, discos and bars in the previously chosen city. He can pick one of which he knows the style of.

The Choozer now offers all the boozers in the chosen city which have the same style as the reference location.

- Option B

In B the system presents all the available tastes in this city to the user. Now he can say what kind of style or taste he likes, enter it, and see the boozers found for this demand.

He can even decide that he wants to see all possible styles by choosing the Don't Care about the style option and enter 'dc'. By doing so, the output will include all boozers for the specified city.

At that point the user has in both options one or more results fitting the criteria he entered. Now there is the option to show tips for every result. By typing a name of a boozers the tip is displayed and the application finishes. With the input of 'end' instead of a boozers name the application finishes without providing tips.

A testrun for option B is shown in figure 4 in the appendix

The sourcecode of the whole application version 4.1.0 can be found in the appendix. Because it was not possible to paste it completely into this document because of its width some comments had to be removed. For the code with all comments have a look into the .pl file attached.

---

## 5 Appendix

### References

- [1] <http://www.swi-prolog.org>
- [2] Russel, S., Norvig, P., 1995 *Artificial Intelligence - a modern approach*, New Jersey, Prentice Hall International Editions
- [3] Callear, D., 1994 *Prolog Programming for Students*, London, DP Publications Ltd
- [4] Bratko, I., 2001 *Prolog Programming for Artificial Intelligence*, England, Pearson Education Limited

## 5.1 Testruns



```

SWI-Prolog -- c:/Dokumente und Einstellungen/All Users/Desktop/Prolog/4.1.1.pl
File Edit Settings Run Debug Help
*           have fun!           *
*****

Please choose one of the following options:

-----
'a' followed by '.' for using the- search by BOOZER - function.
You can enter a boozers you like and the Boozer Choozer will search a boozers with the s
ame taste.
-----
OR:
-----
'b' followed by '.' for using the - search by TASTE - function.
You can enter a taste you like and the Boozer Choozer will find a fitting boozers.
-----
|   a.

*****
Choose one of the cities by entering its name followed by a '.'

mainz
london
mannheim

|   mainz.

*****
This list contains all clubs, pubs and bars in mainz.

When you choose one, you will get other locations which fit the taste of your favourit
e

Boozers in town:
alex
pomp
citrus
cubanbar
saosalitos

|   pomp.

These are the boozers having the taste of pomp which is classical

alex
pomp
citrus

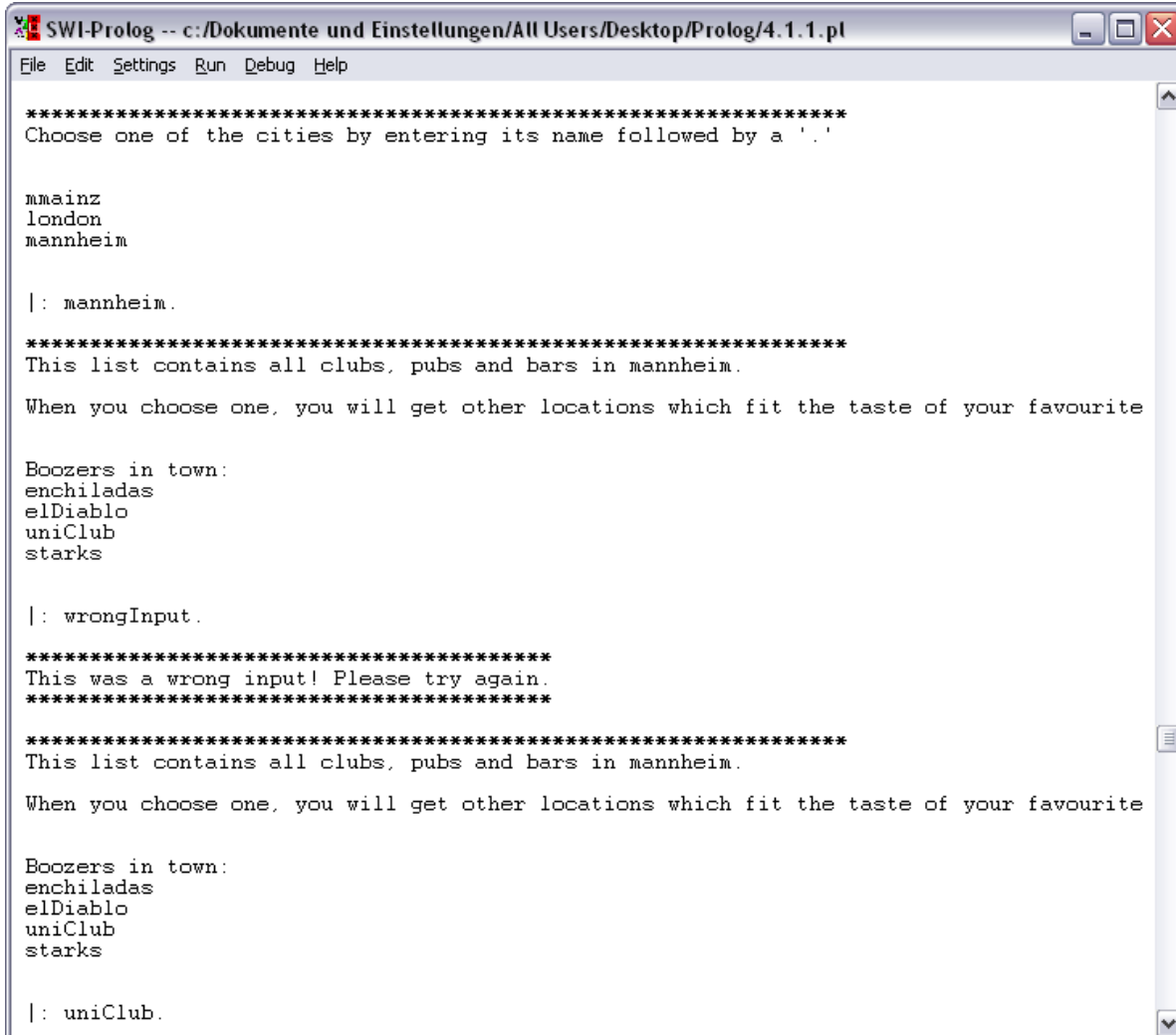
*****
Pssst!!! Hey! Do you want to get some tips to one of the results?
Choose a boozers ...otherwise type 'end'

|   citrus.

*****
Here they come! The interesting (insider)information about citrus
good_food_but_allways_full
  
```

Figure 2: Testrun - Search by boozers

## 5.1 Testruns



```
SWI-Prolog -- c:/Dokumente und Einstellungen/All Users/Desktop/Prolog/4.1.1.pl
File Edit Settings Run Debug Help

*****
Choose one of the cities by entering its name followed by a '.'

mmainz
london
mannheim

|: mannheim.

*****
This list contains all clubs, pubs and bars in mannheim.

When you choose one, you will get other locations which fit the taste of your favourite

Boozers in town:
enchiladas
elDiablo
uniClub
starks

|: wrongInput.

*****
This was a wrong input! Please try again.
*****

*****
This list contains all clubs, pubs and bars in mannheim.

When you choose one, you will get other locations which fit the taste of your favourite

Boozers in town:
enchiladas
elDiablo
uniClub
starks

|: uniClub.
```

Figure 3: Testrun - with a wrong input

## 5.1 Testruns

```

SWI-Prolog -- c:/Dokumente und Einstellungen/All Users/Desktop/Prolog/4.1.1.pl
File Edit Settings Run Debug Help

*****
*                               *
*           !Welcome!           *
*           This is the         *
*           BOOZER CHOOZER      *
*                               *
*           it will help you to find *
* a pub, club or bar, fitting the style you like *
*           in a city you choose. *
*                               *
*           have fun!           *
*****

Please choose one of the following options:

-----
'a' followed by '.' for using the- search by BOOZER - function.
You can enter a boozer you like and the Boozer Choozer will search a boozer with
the same taste.
OR:
-----
'b' followed by '.' for using the - search by TASTE - function.
You can enter a taste you like and the Boozer Choozer will find a fitting boozer.

-----
|: b.

*****
Choose one of the cities by entering its name followed by a '.'

mainz
london
mannheim

|: mainz.

*****
Here is a list of tastes from clubs and pubs in mainz.

You can choose one of them...
...or enter 'dc' for Don't Care and get locations for all tastes

classical
mexican

|: mexican.

Here are the fitting locations in mainz with the taste mexican:

cubanbar
saosalitos

*****
Pssst!!! Hey! Do you want to get some tips to one of the results?
Choose a boozer ...otherwise type 'end'

|: end.■

```

Figure 4: Testrun - Search by taste

## 5.2 Sourcecode

The sourcecode for version 4.2.0

## 5.2 Sourcecode

```

/*-----*/
/* BOOZER CHOOZER */
/*-----*/

/*-----
/*-----*/
/*----- Initialisation with welcome screen and choice menu-----*/
/*-----*/
/*-----*/

/*----- the main routine -----*/

go:--  init ,
      menu.

/*----- show the welcome screen -----*/

init:--  nl, nl, nl,
        write('*****'), nl,
        write('*'), nl,
        write('*'), nl,
        write('*'), nl,
        write('*'), nl,
        ! Welcome!
        This is the

```



## 5.2 Sourcecode

```

nl, nl, nl,
menu.

/*---- Check if Input is a or b ----*/

checkChoose('a').
checkChoose('b').

/*---- Processes input ----*/

choose(a):--   getCityInput(City), favouriteChoose(City).
choose(b):--   getCityInput(City), tasteChoose(City).

/*----
/*-----
/*----- The 2 main routines – tasteChoose and favouriteChoose – searching by entering a favourite boozor OR by a favourite taste ----*/
/*-----
/*-----

/*----
/*-----
/*----- Search by choosing a favourite boozor ----*/
/*-----

favouriteChoose(City):--   findall(Boozor, boozorChoozer(Taste, City, Boozor), BoozorList1),
                           write('*****'), nl,
                           write('This list contains all clubs, pubs and bars in '),
                           write(City).

```

## 5.2 Sourcecode

```

write('.').nl,nl,
write('When you choose one, you will get other locations which fit the taste of your favourite '),nl,nl,
write('Boozers in town:'),nl,
write_elt(BoozerList1),nl,nl,
read(BoozerChoice),
checkBoozerChoice(BoozerChoice,BoozerList1),
boozerChoozer(Taste, City, BoozerChoice),nl,nl,

write('These are the boozers having the taste of '),
write(BoozerChoice),
write(' which is '),
write(Taste),nl,nl,

findAll(Boozer, boozerChoozer(Taste, City, Boozer), BoozerList2),
write_elt(BoozerList2),
tipHint,
tipReq(BoozerList2);
/*---- if the input is not in list ----*/
nl,
write('*****'),nl,
write('This was a wrong input! Please try again. '),nl,
write('*****'),nl,nl,
favouriteChoose(City).

/*---- check if the input is in the list of boozers located in the city ----*/

checkBoozerChoice(BoozerChoice, BoozerList1):- member(BoozerChoice, BoozerList1).

/*----
/*---- Search by choosing a taste ----*/
/*----

```

## 5.2 Sourcecode

```

tasteChoose(City):-
    isTasteIn(TasteList, City), /*---- generate List with all Tastes available in the City ----*/
    nl,nl,
    write('*****
write('Here is a list of tastes from clubs and pubs in '),
write(City),
write(' '),nl,nl,
write('You can choose one of them...'),nl,
write('...or enter 'dc' for Don't Care and get locations for all tastes '),nl,nl,nl,
write_elt(TasteList),nl,nl,

read(Taste),
/*---- go into checking the taste if it is in the TasteList or if it is 'dc' or crap ----*/
check88(Taste, City, TasteList).

/*---- test if taste is in List and proceed - if not check if it is 'dc' ----*/

check88(Taste, City, TasteList):-
    member(Taste, TasteList),
    findall(Boozer, boozerChoozer(Taste, City, Boozer), BoozerList),nl,nl,

    write('Here are the fitting locations in '),
    write(City),
    write(' with the taste '),
    write(Taste),
    write(' '),nl,nl,
    write_elt(BoozerList),

    tipHint,
    tipReq(BoozerList);

```

## 5.2 Sourcecode

```

check99(Taste),
nl,nl,
write('*****
write('You entered dc. The taste is not relevant for you.').
),
findall(Boozer, boozerChoozer(_, City, Boozer), BoozerList),nl,nl,
write('Here are ALL locations in '),
write(City),
write(:),nl,nl,
write_elt(BoozerList),

tipHint,
tipReq(BoozerList).

/*---- checks if Taste is 'dc' ----*/

check99('dc').

/*---- if Taste is not in TasteList / not 'dc' call for a new try because the Taste was a wrong input ----*/

check88(Taste, City, TasteList):-
nl,nl,
write('*****
write('Sorry, but this was a wrong input!'),nl,nl,
write('Have another try...'),nl,nl,nl,
tasteChoose(City).

```

```

/*-----*/
/*-----*/
/*----- City rules -----*/
/*-----*/
/*-----*/

/*----- processing of City input -- request for Input and read it and check it-----*/
getCityInput(City):-- nl,nl,
    checkCityInput(CityInput),
    City=CityInput.
/*----- read in input-----*/

/*----- give back all Cities in Database -----*/
/*----- lists all cities in database -----*/
/*----- remove duplicates -----*/
/*----- output of list -----*/

cityPossibilities(CityListNoRep):-- findall(City, boozerChoozer(-,City,-),CityList),
noRepetition(CityList, CityListNoRep),
write_elt(CityListNoRep),nl,nl.

/*----- Check if there is a boozer in the entered city --- returns true when a the entered string is found-----*/

```

## 5.2 Sourcecode

```

checkCityInput(CityInput):--
    write('*****'),nl,
    write('Choose one of the cities by entering its name followed by a '.''),nl,nl,nl,
    cityPossibilities(CityListNoRep),
    read(CityInput),nl,

    member(CityInput, CityListNoRep).

checkCityInput(CityInput):--
    write('*****'),nl,
    write('The City is not in the database! I'm sorry!'),nl,
    write('Try again.'),nl,nl,nl,

    checkCityInput(CityInput).

/*-----*/
/*-----*/
/*----- general rules -----*/
/*-----*/
/*-----*/

/*----- gives the hint that there are tips in the database for the boozer -----*/

tipHint:--
    nl,

```

## 5.2 Sourcecode

```

write('*****'),nl,
write('Pssst!! Hey! Do you want to get some tips to one of the results?'),nl,
write('Choose a boozer '),
write('... otherwise type ''end'''),nl,nl.

/*---- initializes the tip request check - reads the input and gives it to the check method ----*/

tipReq(BoozerList):- read(BoozerTip),nl,nl,
                    boozerCheck(BoozerTip,BoozerList);
write('*****'),nl,
write('OH NO! Please choose and type one of the boozers listed or type ''end'''),nl,
write('Once again!'),nl,nl,
write('Here comes the list:'),nl,nl,
write_elt(BoozerList),nl,
tipReq(BoozerList).

/*---- check if tips are requested to one of the results ----*/

boozerCheck(BoozerTip, BoozerList):-
                    boozerTip(BoozerTip), /*---- checks if input is end ----*/
                    nl,
                    write('*****'),nl,
                    write('Ok! No Tips needed.'),nl,nl,
                    write('Exiting application.....'),nl,
                    nl,nl,
                    write('Thank you for using the Choozer!'),nl.

/*---- if end was not the input - call the findall tips rule ----*/

```

## 5.2 Sourcecode

```

boozerCheck(BoozerTip, BoozerList):—
    nl, nl,
    write('*****information about *****'), nl,
    write('Here they come! The interesting (insider)information about '),
    write(BoozerTip), nl, nl,
    findall(Tip, tip(BoozerTip, Tip), TipList),
    write_elt(TipList), nl, nl, nl, nl.

/*---- check if input is end ----*/

boozerTip('end').

/*---- List Processing --- writes a list to the screen ----*/

write_elt([]).
write_elt([H|T]):— write(H), nl, write_elt(T). /*---- Writes the Head of a list and calls itself with the Tail ----*/

/*---- repetition removal from lists ----*/

noRepetition([], []).
noRepetition([H|T1], T2):— member(H, T1),!,
                             noRepetition(T1, T2).

noRepetition([H|T1], [H|T2]):— not(member(H, T1)),

```

```

noRepetition(T1, T2).

/*----- */
/*----- */
/*----- CORE ----- */
/*----- */
/*----- */

/*----- THE MOST IMPORTANT PART ----- */

boozerChoozer(Taste, City, Boozer) :- isBoozerIn(Boozer, City),
suitstaste(Boozer, Taste).

/*----- */
/*----- */
/*----- GENERAL DEFINITIONS ----- */
/*----- */
/*----- */

/*----- Tastes available in every city ----- */

isTasteln([mexican, niels, classical], mannheim).
isTasteln([classical_bar, mexican, club], mainz).
isTasteln([classical], london).

```

## 5.2 Sourcecode

```
/*---- DEFINITIONS OF BOOZERS----*/  
  
/*---- Definition of the boozers with taste and tips----*/  
  
/*---- Mainz ----*/  
  
isBoozerIn(alex , mainz).  
suitstaste(alex , classical_bar).  
tip(alex , good_food_but_allways_full).  
  
isBoozerIn(pomp , mainz).  
suitstaste(pomp , classical_bar).  
tip(pomp , very_expensive_but_friendly_people_and_a_very_large_breakfast).  
  
isBoozerIn(citrus , mainz).  
suitstaste(citrus , classical_bar).  
tip(citrus , great_location_and_nice_black_music_on_saturdays).  
  
isBoozerIn(cubanbar , mainz).  
suitstaste(cubanbar , mexican).  
tip(saosalitos , good_cocktails_but_very_small_come_early).  
  
isBoozerIn(saosalitos , mainz).  
suitstaste(saosalitos , mexican).  
tip(saosalitos , good_food_and_cocktails).  
  
isBoozerIn(stars , mainz).
```

5.2 Sourcecode

---

```
suitstaste(stars, club).
tip(stars, black_music_all_night_long).
tip(stars, on_top_of_the_CineStar_cinema).

isBoozerIn(fuenfziggrad, mainz).
suitstaste(fuenfziggrad, club).
tip(fuenfziggrad, black_music_all_night_long_and_gooooo_loocking_girls).

/*---- London ----*/

isBoozerIn(nandos, london).
suitstaste(nandos, classical).
tip(nandos, good_chicken_food_and_drinking_as_much_as_you_want).

/*---- Mannheim ----*/

isBoozerIn(enchiladas, mannheim).
suitstaste(enchiladas, mexican).
tip(enchiladas, good_food_and_cocktails).

isBoozerIn(elDiablo, mannheim).
suitstaste(elDiablo, mexican).
tip(elDiablo, good_food_but_people_are_strange).

isBoozerIn(uniClub, mannheim).
suitstaste(uniClub, niels).
tip(uniClub, inventory_mirroored_at_the_ceiling).

isBoozerIn(starks, mannheim).
suitstaste(starks, niels).
```

5.2 Sourcecode

---

```
suitstaste(starks , classical ).
tip(starks , cool_place_to_be_especially_on_hot_days ).

isBoozerIn(starstower , mannheim ).
suitstaste(starstower , niels ).
suitstaste(starstower , classical ).
tip(starstower , cool_place_to_be_especially_on_hot_days ).
```