COMP 348: Principles of Programming Languages
Assignment 1 on Logical Programming

Summer 2020, sections AA and AB

May 3, 2020

Contents

1 General Information

Date posted: Monday May 4th, 2020.

Date due: Monday May 18th, 2020, by 23:59[1].

Weight: 5% of the overall grade.

2 Introduction

In this assignment you will be practicing logical programming using Prolog language. This assignment consists of three parts: 1) Fact representation, 2) Queries, Unification, and Resolution, and 3) Implementation Exercises.

Note that Part 1 of the assignment is due on May 10th and is to be submitted separately. See section 5 for more details.

3 Ground rules

You are allowed to work on a team of 3 students at most (including yourself). Each team should designate a leader who will submit the assignment electronically. ONLY one copy of the assignment is to be submitted.

This is an assessment exercise. You may not seek any assistance from others while expecting to receive credit. **You must work strictly within your team).** Failure to do so will result in penalties or no credit.

4 Your Assignment

Your assignment is given in three parts, as follows. 1) Fact representation, which is to be submitted separately, 2) Queries, Unification, and Resolution, and 3) Implementation Exercises, all of which will be demoed by the whole team.

[1]see submission notes (deadline for 1 is Sunday, May 10th at Noon)

4.1 Fact Representation

Provide a knowledge-base of clauses specifying your class schedule in prolog, as demonstrated in the following.

```
team([<<student-ids>>...]).
student_info('<<student-id>>', '<<first-name>>', '<<last-name>>').
takes_course('<<student-id>>', '<<course-name>>',
    '<<course-number>>', '<<course-section>>').
course_schedule('<<course-name>>', '<<course-number>>',
    '<<course-section>>', '<<dow>>', '<<from-time>>', <<to-time>>').
```

- *dow* is day of week: 'sun', 'mon', 'tue', 'wed', 'thu', 'fri', and 'sat'.
- *from-time* and *to-time* are class start and end in military format, i.e. 0845 (8:45am), 1445 (2:45pm), etc.
- *course-name*, *course-number*, and *course-section* are all in lower case: e.g.. `comp 348 aa`

Below is a sample knowledge-base for a group of two: John Doe and Jane Doe. Note that in this question, using single quote (') is mandatory.

```
team(['4000123', '4000228']).
student_info('4000123', 'John', 'Doe').
student_info('4000228', 'Jane', 'Doe').
takes_course('4000123', 'comp', '348', 'aa').
takes_course('4000123', 'comp', '348', 'aaae').
takes_course('4000228', 'comp', '348', 'ab').
takes_course('4000228', 'comp', '348', 'abaf').
takes_course('4000228', 'comp', '346', 'cc').
course_schedule('comp', '348', 'aa', 'mon', '1445', '1715').
course_schedule('comp', '348', 'aa', 'wed', '1445', '1715').
course_schedule('comp', '348', 'aaae', 'mon', '1345', '1405').
course_schedule('comp', '348', 'aaae', 'wed', '1345', '1405').
course_schedule('comp', '348', 'ab', 'tue', '1315', '1545').
course_schedule('comp', '348', 'ab', 'thu', '1315', '1545').
course_schedule('comp', '348', 'abaf', 'tue', '1615', '1705').
course_schedule('comp', '348', 'abaf', 'thu', '1615', '1705').
course_schedule('comp', '346', 'cc', 'tue', '1830', '2100').
course_schedule('comp', '346', 'cc', 'thu', '1830', '2100').
```

John is taking one course (`comp 348 aa` + the tutorial `aaae`) and Jane is taking two courses (`comp 348 ab` + the tutorial `abaf` we well as `comp 346 cc` with no tutorial). Note that both courses are scheduled on multiple days.

Q 1. Provide the knowledge-base of ALL courses for ALL members of the team. Create a file named **'schedule-studid.pl'** where studid is the id of the leader of the team. Include all clauses in the file. Write every clause in a single line. One file per group. Submit this separately (see submission notes).

Q 2. Run the following query and report the result:

```
team(X), member(S, X),
findall(S, takes_course(S, _, _, _), LL),
length(LL, NN),
write(S), write(' has only taken '), write(NN),
write(' courses and tutorials in summer 2020.'), nl, fail.
```

Is the above information correct?

The above must be submitted in stage 1 (see submission notes for details).

Q 3. Create below mentioned rules to query the designation information from your knowlege-base.

Hint: Use the cut operator where applicable.

```
all_sections(CNAM, CNUM, L) :- ...
/* L contains a list of all sections of course CNAME, CNUM,
i.e. calling all_sections('comp', '348', L) will result in L=['aa', 'ab'];
     no duplicates */

has_taken(S, [CNAM|[CNUM|[SEC|[]]]]) :- ...
/* true if student S takes the course CNAM CNUM SEC,
   e.g. takes('4000123', ['comp', '348', 'aa']) */

has_taken2(S, [CNAM|[CNUM|[]]]) :- ...
/* true if S takes any sections of the course CNAM CNUM,
   e.g. takes('4000123', ['comp', '348']) */

all_subjects(S, L) :- ...
/* L contains all the courses subjects that have been
taken by student S, i.e. ['comp', 'soen']; no duplicates */

all_courses(S, L) :- ...
/* L contains all the courses that have been taken by
   student S, i.e. all_courses('4000123', L) will result in
   L=[['comp', '348', 'aa'], ['comp', '348', 'ab']] */

all_courses2(S, L) :- ...
/* similar to all_courses but without section info;
   no duplicates */
```

Q 4. Modify the code in Q2 to count only the courses regardless of their sections (i.e. `comp 348 aa` and `aaae` will be counted as one)

Q 5. Compare the result for `all_courses2('4000123', L)` vs. `all_courses2(4000123, L)`. Explain the difference.

4.2 Queries, Unification, and Resolution

Q 6. Unification: Indicate which of the following pairs of terms can be unified together? If they can't be unified, please provide the reason for it. In case of error, indicate the error. If they can be unified successfully, wherever relevant, provide the variable instantiations that lead to successful unification. (Note that '=' indicates unification)

1. `food(bread, X) = Food(Y, soup)`
2. `Bread = soup`
3. `Bread = Soup`
4. `food(bread, X, milk) = food(Y, salad, X)`
5. `manager(X) = Y`
6. `meal(healthyFood(bread), drink(milk)) = meal(X,Y)`
7. `meal(eat(Z), drink(milk)) = [X]`
8. `[eat(Z), drink(milk)] = [X, Y | Z]`
9. `f(X, t(b, c)) = f(l, t(Z, c))`
10. `ancestor(french(jean), B) = ancestor(A, scottish(joe))`
11. `meal(healthyFood(bread), Y) = meal(X, drink(water))`
12. `[H|T] = [a, b, c]`
13. `[H, T] = [a, b, c]`
14. `breakfast(healthyFood(bread), egg, milk) = breakfast(healthyFood(Y), Y, Z)`
15. `dinner(X, Y, Time) = dinner(jack, cook( egg, oil), Evening)`
16. `k(s(g), Y) = k(X, t(k))`
17. `equation(Z, f(x, 17, M), L*M, 17) = equation(C, f(D, D, y), C, E)`
18. `a(X, b(c, d), [H|T]) = a(X, b(c, X), b)`

Q 7. Queries: Assume we have the following database in a Prolog program:

```
course(hit_transfer, mechanical).
course(web_design,computer).
course(design_methods, fine-arts).
course(poetry, literature).
course(leadership, management).
course(biology,medicin).
lab_number(mechanical,15).
lab_number(fine_arts,10).
lab_number(X, Z) :-course(X, Y), lab_number(Y, Z).
field(mechanical,engineering).
field(computer, engineering).
field(fine-arts, art). field(literature, social).
field(management, buisiness).
field(X, Y) :- course(X, Z), field(Z, Y).
student(anna, hit_transfer).
student(daniel, hit_transfer).
student(adrian, web_design).
student(ava, design_methods).
student(jack, poetry).
student(lee, leadership).
student(X, Y) :- field(Z, Y), student(X, Z).
student(X):- student(X,_).
```

Determine the type of each of the following queries (ground/non-ground), and explain what will Prolog respond for each of these queries (write all the steps of unifications and resolutions for each query)?

1. ? `field(hit_transfer,engineering).`
2. ? `lab_number(fine_arts,X).`
3. ? `field(computer, literature).`
4. ? `course(X,Y).`
5. ? `student(adrian).`
6. ? `student(anna, engineering).`

7. `? student(X, engineering).`

8. `? student(X,fine-arts), course(fine_arts, Y).`

9. `? field(_,X).`

10. `? lab_number(_,X), field(X,Y).`

11. `? lab_number(X,15), field(X,Y).`

12. `? student(X), !, student(X,_).  % note to cut here`

13. `? student(X), student(X,_), !.`

14. `? course(X,_), \+ student(_,X). % \+ is for negation (not)`

Q 8. Unification, Resolution, and Backtracking: Assume, you are working with the following knowledge base:

```
house_elf(dobby).
witch(hermione).
witch(mcGonagall).
witch(rita_skeeter).
wizard(dobby).
magic(X):- house_elf(X).
magic(X):- wizard(X).
magic(X):- witch(X).
```

Write the details of steps of search (unification, resolutions, and back tracking) and also the answer for each of the following queries:

```
? magic(Hermione).
? magic(hermione).
```

4.3 Implementation Exercises

Q 9. Assume, you are working with the following knowledge base::

```
family(
  person(john, cohen, date(17,may,1990), unemployed),
        person(lily, cohen, date(9,may,1990), unemployed),[] ).
family(
  person(john, armstrong, date(7,may,1988), unemployed),
  person(lily, armstrong, date(29,may,1961), unemployed),[]).
family(
  person(eric, baily, date(7,may,1963), works( bbc, 2200)),
  person(grace, baily, date(9,may,1965), works( ntu, 1000)),
  [ person(louie, baily, date(25,may,1983), unemployed) ] ).
family(
  person(eric, baily, date(7,may,1963), works( acc, 21200)),
  person(grace, baily, date(9,may,1965), works(ntnu, 12000)),
  [ person( louie, baily, date(25,may,1983), unemployed) ]).
family(
  person(eric, fox, date(27,may,1970), works(bbc, 25200)),
  person(grace, fox, date(9,may,1971), works(ntbu, 13000)),
  [ person(louie, fox, date(5,may,1993), unemployed) ]).
family(
  person(tom, cohen, date(7,may,1960), works( bcd, 15200)),
  person(ann, cohen, date(29,may,1961), unemployed),
  [ person(pat, cohen, date(5,may,1983), works(bcd, 15200)),
    person(jim, cohen, date(5,may,1983), works(bcd, 15200))]).
family(
  person(bob, armstrong, date(12,oct,1977), works(ntnu, 12000)),
  person(liz,armstrong, date(6,oct,1975), unemployed),
  [ person(bob, armstrong, date(6,oct,1999), unemployed),
    person(sam,armstrong, date(8,oct,1998), unemployed) ]).
family(
  person(tony, oliver, date(7,may,1960), works( bbc, 35200)),
  person(anny, oliver, date(9,may,1961), unemployed),
  [ person(patty, oliver, date(8,may,1984), unemployed),
    person(jimey, oliver, date(5,may,1983), unemployed) ]).
```

```
family(
  person(jack, fox, date(27,may,1940), unemployed),
  person(jane, fox, date(9,aug,1941), works( ntu, 13050)),
  [ person(andy, fox, date(5,aug,1967), works(com, 12000)),
    person(kai, fox, date(5,jul,1969), unemployed) ]).

husband(X) :- family( X, _, _).
wife(X) :- family( _, X, _).
child(X) :- family( _, _, Children), member(X, Children).
exists(Person) :- husband(Person); wife(Person); child(Person).
dateofbirth(person(_, _, Date, _), Date).
salary(person(_, _, _, works(_, S)), S).
salary(person(_, _, _, unemployed), 0).
```

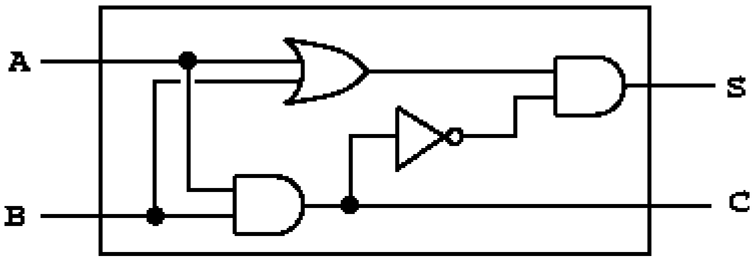
a) Write a prolog rule `totalIncome/2` to compute the total income of a family.

b) Write a prolog query to print total income of each family.

c) Write a prolog query to print family details of each family that has income per family member less than 2000.

d) Write a prolog query to print family details of each family where children's total income is more than their parents.

Q 10. The following circuit represents a half-adder where A and B are the inputs, and S and C represent the Sum and Carry bits, respectively.



a) Implement the above circuit in prolog.

b) Write the query to calculate the outputs for S and C for A = 0 and B = _. Provide the output.

Q 11. Write a Prolog query with arity 2 to return the first n numbers of a Lucas sequence in a list.

`https://brilliant.org/wiki/lucas-numbers/`

The Lucas sequence has the same recursive relationship as the Fibonacci sequence, where each term is the sum of the two previous terms, except that the first two numbers in the sequence are: 2, and 1. The first few elements of the sequence are: $2, 1, 3, 4, 7, 11, 18, ...$

5 What to Submit

This assignment is submitted in two stages.

Stage 1 - Fact Representation

The first part of the assignment (1) is submitted by **Sunday, May 10th at Noon**. It has to be completed by ALL members of the team in one submission file. The knowledge-base file must contain all the facts for all the members of the team, all merged in one single file. You submission includes one single file namely **'schedule-studid.pl'** where studid is the id of the leader of the team. Failure to submit may result in 0 mark for the whole assignment for the whole team. Make sure there is no syntax error. Make sure the query in (2) succeeds. Test before submission. Do not include the query nor result of 2 in this submission.

Stage 2 - Whole Assignment

The whole assignment (including 1 and 2) is submitted by the due date under the corresponding assignment box. Your instructor will provide you with more details.

Submission Notes

Note: The following applies to stage 2 (Whole Assignment Submission).
Clearly include the names and student IDs of all members of the team in the submission. Indicate the team leader.

<u>IMPORTANT:</u> You are allowed to work on a team of 3 students at most (including yourself). Any teams of 4 or more students will result in 0 marks for all team members. If your work on a team, ONLY one copy of the assignment is to be submitted. You must make sure that you upload the assignment to the correct assignment box on Moodle. No email submissions are accepted. Assignments uploaded to the wrong system, wrong folder, or submitted via email will be discarded and no resubmission will be allowed. Make sure you can access Moodle prior to the submission deadline. The deadline will not be extended.

Naming convention for uploaded file: Create one zip file, containing all needed files for your assignment using the following naming convention. The zip file should be called a#_studids, where # is the number of the assignment, and studids is the list of student ids of all team members, separated by (_). For example, for the first assignment, student `12345678` would submit a zip file named `a1_12345678.zip`. If you work on a team of two and your IDs are `12345678` and `34567890`, you would submit a zip file named `a1_12345678_34567890.zip`. Submit your assignment electronically on Moodle based on the instruction given by your instructor as indicated above:

`https://moodle.concordia.ca`

Please see course outline for submission rules and format, as well as for the required demo of the assignment. A working copy of the code and a sample output should be submitted for the tasks that require them. A text file with answers to the different tasks should be provided. Put it all in a file layout as explained below, archive it with any archiving and compressing utility, such as WinZip, WinRAR, tar, gzip, bzip2, or others. You must keep a record of your submission confirmation. This is your proof of submission, which you may need should a submission problem arises.

6 Grading Scheme

Q1-Q2	2 marks
Q3-Q5	5 marks
Q6	5 marks
Q7	5 marks
Q8	5 marks
Q9	10 marks
Q10	3 marks
Q11	5 marks

Total: 40 marks.

References

1. swi-prolog: `https://swish.swi-prolog.org`

2. Lucas sequence: `https://brilliant.org/wiki/lucas-numbers/`

3. Half-Adder: `http://www.circuitstoday.com/half-adder-and-full-adder`