CS 227: Homework #3

Assigned: Friday May 6th.
Due Dates: Complete Question Answering Steps 1-4 of Problem 1 by **May 17th** (no submission required on this day). Final submission is on Thursday**, May 19th at 5pm.**
Important note: You may do **problem 1** in **groups of two**. Feel free to use Piazzza to find project partners. If you work in a group, only submit one report for the group and write both names clearly on your report. Also to help us with grading, **do not staple** problem 1 to your individual submissions for the rest of the homework (problem 2 and 3).

Problem 1: Using Logic Programming to Represent Real World Knowledge

The goal of this problem is to represent real world knowledge for reasoning and answering questions. For this exercise, we will use a logic programming environment called Flora that is based on F-Logic. Since logic programs are more expressive than structured descriptions, many more different forms of knowledge can be stated. We will use the California Driver's Handbook as the source of the knowledge to be represented. The focus of the reasoning task will be on knowledge that can be objectively and operationally used by the driver of a vehicle (examples of this will be given later in the handout). We will test the knowledge by using two sets of questions: a training set and a novel set. This will force us to develop a general representation and discourage customization of representation to a question.

Overview of the project

For this assignment, we will reduce the task of answering questions to answering queries using a logic program.

An electronic version of the California Driver's Handbook is available at http://www.dmv.ca.gov/pubs/dl600.pdf.

You will formally represent the content of one page from the Driver's handbook. Specifically, you will represent **page 28 starting at the section titled *Speed Limit***. You will test your representation using a set of training questions that you will develop yourself. You must complete your KB development by May 17th. On May 17th, we will release five novel questions

that were not previously known to you. You must test your KB on the novel question set without making any updates to the KB. You will then report on your score, and failure analysis of your knowledge base.
You will develop the knowledge base in the following steps:

- Design the knowledge representation on paper.
- Represent the knowledge in FLORA
- Encode the knowledge that you generated into FLORA
- Test your KB using sample questions that you will develop yourself
- Test your KB on the novel questions

To get you started, a complete step-by-step worked out example of representing knowledge and testing it by asking questions is enclosed in this handout in appendix 1. We have also provided the Flora file for this example that you can load and tryout. We recommend that you start early so that you can give yourself enough time to complete the assignment. As always please post questions on Piazzza if any part is unclear to you.

Flora Setup

1. Download FLORA from the following website: https://sourceforge.net/projects/flora/files/FLORA-2/0.96.1%20%28Callistephus%20Daintiness%29/
2. The downloaded zip file contains a compiled version for Windows and the source code for Linux and Mac that needs to be compiled. The instructions for installing and running the program for each platform are as follows:

Windows:

1. Unzip the florabundle.zip file at the root of any drive, for example, C: or D:, etc. (Please note we strongly recommend using a drive root location. Please ask for help from a TA if you have to choose a different location.)
2. You should then see a directory such as c:\XSBFLORABUNDLE.
3. Go to the directory c:\XSBFLORABUNDLE\
4. In a DOS shell, type runflora.bat, and it will bring up the FLORA command prompt program (there is no graphical user interface associated with this distribution).

Mac & Linux:

1. Unzip the distribution. A folder named XSBFLORABUNDLE will be created. (Make sure the pathname to the location where you place the zip file does not have any spaces in its name)
2. Open a terminal and perform the following steps to compile the source code.
3. Type cd XSBFLORABUNDLE/XSB/build
4. Type ./configure
5. Type ./makexsb
6. Next, cd XSBFLORABUNDLE/flora2
7. Type ./makeflora all ../XSB/bin/xsb
8. From the same directory, run flora by typing ./runflora. This should start the FLORA command prompt interface (there is no graphical user interface associated with this distribution).

Flora Documentation

Flora documentation is available in the directory
XSBFLORABUNDLE/flora2/docs/manual.pdf

Sections 2-3 and 5-10 should be sufficient to get you started.

Question Answering Steps

1. For each sentence appearing on the assigned page, analyze it to identify knowledge that is specific and can be operationally used by the driver of a vehicle. You need not represent any sentences that do not contain such knowledge. You will obviously make some modeling decisions to make these choices. Explain and justify whatever choices you make. See the sample solution for concrete examples.
2. For all the necessary sentences that need to be represented, formalize them using logic programming rules as illustrated in the sample solution in the Appendix.
3. Encode this knowledge using Flora.
4. Test the knowledge by posing questions. Devise any questions that you think would be necessary to test the knowledge that you have just created. Record the answers returned by Flora in response to your questions.
5. Test the knowledge using novel questions that will be released on May 17th. You must not change the KB during this phase. Record the answers returned by Flora.

6. Perform a failure analysis for any questions that your KB failed to answer. Identify knowledge and inference pieces that prevented the question from being answered. Suggest ways in which you could have made your KB more robust and complete for answering novel questions.

Please see the Appendix for a sample illustration of the process and a complete worked out solution for a few paragraphs.

Submission & Deliverables

You need to submit:

1. By May 19th, 5PM, submit the following on paper:
    a. Your paper design of the knowledge base you constructed. For each sentence, identify your decision about whether it needed to be represented. For each represented sentence, include the classes, relations, and relevant rules.
    b. A trace of answers to the reference questions as well as the novel questions
    c. Failure analysis for the novel questions

Evaluation and Grading

Your work will be evaluated and graded based on the coverage and clarity of your knowledge presentation, thoroughness with which you tested the KB, an analysis of your results such as why your system answers some question and why it fails on others, how you might improve your representation after looking at the novel question performance, and your adherence to the no changes to the KB during evaluation rule. **The score received on the novel questions will be irrelevant for grading.**

Extra and beyond: Importance of this project

If we have a complete representation of the driving law as suggested in this exercise, one can formulate interesting questions against the knowledge base which cannot be easily answered without the KB. For example:

1. Is the law complete and consistent? If people drive by exactly following the law, can they get into a situation where they get into an accident?
2. Is the law contradictory? Are there any inconsistencies or situations

where the driving law gives us conflicting guidance on what should be done?

3. How can the law be improved? What sections of the law are too vague to be useful?

Answering such questions is outside the scope of this exercise. However they are good candidates for a research project.

Problem 2: Answer Set Programming

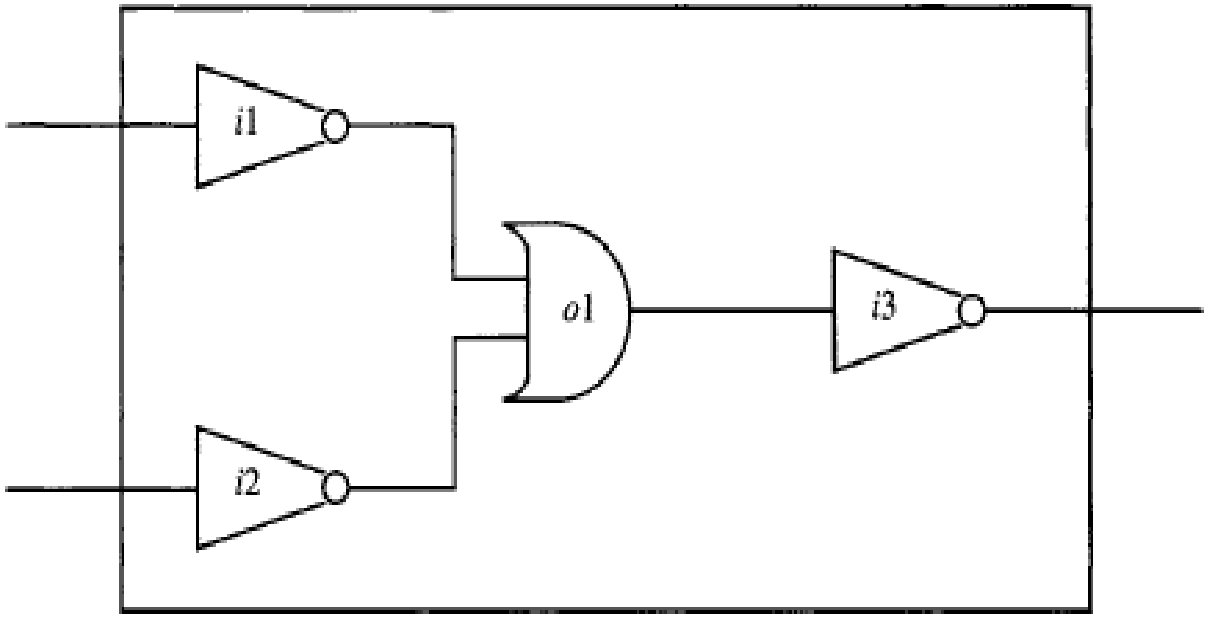
Compute the answer sets of the following programs:
Assume that ***naf*** represents negation as failure, and ¬ represents classical negation.

a) p(a) ← naf p(b) /\ ¬p(c)
p(b) ← naf p(a) /\ ¬p(c)
¬ p(X) ← naf p(X)

b) p ← naf q
q ← naf p
r ← naf s
s ← naf r
¬s ← q

#

Problem 3: Abductive Reasoning



Consider the binary circuit for logical AND depicted in the Figure above where *il*, *i2*, and *i3* are logical inverters and *ol* is an OR gate.

a) Write sentences describing this circuit--its components, connectivity, and normal behavior.
b) Write a sentence for a fault model saying that a faulty inverter has its output the same as its input.
c) Assuming the fault model and that the output is 1 given inputs of 0 and 1,

generate the three abductive explanations for this behavior.

d) Generate the three consistency-based diagnoses for this circuit under the same conditions.
e) Compare the abductive and consistency-based diagnoses and explain informally why they are different.

Appendix
Sample Solution for Problem 1

As an example, we will consider the text starting from page 26 of the Handbook. We will analyze three paragraphs from this page, and then show the formalization. We have added line numbers to the paragraph for ease of reference.

Laws and RULEs of the Road
RIght-of-way rules
General Information
1. Right-of-way rules, together with courtesy and common sense, help to promote traffic safety. 2. It is important to respect the right-of-way of others, especially pedestrians, motorcycle riders, and bicycle riders. 3. Never assume other drivers will give you the right-of-way.

This paragraph introduces the importance of the Right of way rules. While concepts such as *courtesy* and *common sense* can be understood and used by humans, their explicit definitions are not in the handbook, and therefore, will not be represented. The concept of *Right of Way* is very important for driving, and is also not explicitly defined anywhere in the handbook. Since it is operationally relevant for Driving, we will represent it. The second sentence can be viewed as an integrity constraint for driving in that one should never undertake an action that violates the right-of-the-way of others. The third sentence implicitly states that just because you have the right-of-the-way, do not assume that it is safe to take a driving action permitted by it.

Wikipedia defines right-of-the-way as *The general principle that establishes who has the right to go first is called "right of way", or "priority".*

Let us begin defining our ontology. We will show the Flora definitions as we introduce terms and rules.

We define the root class *Thing*. We define *Event* and *Entity* as two subclasses of *Thing.*

Event :: Thing.
Entity :: Thing.

We define *DrivingSituation* as a subclass of *Event*. We define *DrivingEntity* as a subclass of *Entity.* We define *SpatialEntity* as a subclass of *Entity* to denote a geographical location.

DrivingSituation :: Event.
DrivingEntity :: Entity.
SpatialEntity :: Entity.

We define *Action* as a subclass of *Event*, and *Drive* as a subclass of *Action*. We define the relation *agent* with domain *Action* and range *Entity*.

*Action :: Event [agent *=> Entity].*
Drive :: Action.
IllegalDrive :: Drive.

We define the relation *participant* with the domain of *Event* and range of *Entity*. We define the relation *rightOfTheWay* as a relation with domain *DrivingSituation* and range *DrivingEntity*.

*Event :: Thing [participant *=> Entity].*
*DrivingSituation :: Event [rightOfTheWay *=> DrivingEntity].*

To formalize the second sentence, we introduce the notion of *IllegalDrive*. If a driving entity takes a driving action in a situation in which it does not have the right of the way it is an instance of Illegal Drive. We define the relation *follows* with the domain of *Event* and range of *Event* to denote an event that immediately follows the other.

IllegalDrive :: Drive.
*Event :: Thing [participant *=> Entity, follows *=> Event].*

```
?D : IllegalDrive :-
  ?E1 : Entity,
  ?E2 : Entity,
  ?D  : Drive [ agent -> ?E1, follows -> ?S],
  ?S:DrivingSituation [participant -> ?E1,
                      participant -> ?E2,
                        rightOfTheWay -> ?E2],
  not ?E1 :=: ?E2.
```

This rule uses the negation as failure symbol ``not'', and the equality symbol ``:=:''.
As we write each rule, we test it by creating an artificial situation and ensuring that it gives us the result we want. Each rule should be tested in isolation, and also in a way that it interacts with other rules in the knowledge base. We will give a test case for each rule as we introduce it. For the above rule, consider the situation:

```
P1 : DrivingEntity.
P2 : DrivingEntity.
D1 : Drive [ agent -> P1, follows -> S1].
S1 : DrivingSituation [participant -> P1,
                    participant -> P2,
                        rightOfTheWay -> P2]
```

Then we pose a query as follows:

```
flora2 ?- D1:IllegalDrive.
Yes
```

The expected answer of ``Yes'' confirms the intended behavior of this rule.

We formalize the third sentence by stating that having the right of the way is a necessary but not sufficient condition for a safe and legal driving action.

```
SafeDrive :: Drive.

?S [ rightOfTheWay -> ?E1] :-
    ?S : DrivingSituation [participant -> ?E1],
    ?E1 : DrivingEntity,
    ?_ : SafeDrive [follows -> ?S, agent -> ?E1].
```

To test this rule, we introduce the following situation:

S2:DrivingSituation [participant -> P3].
P3: DrivingEntity.
D2: SafeDrive [follows -> S2, agent -> P3].

And pose the following query:
flora2 ?-
S2[rightOfTheWay -> ?x].
flora2 ?-
?x = P3
1 solution(s) in 0.0160 seconds
Yes

This confirms that the rule functions as expected.

Let us now consider the second paragraph in the same section:

> **4. Respecting** the right-of-way of others is not limited to situations such as yielding to pedestrians in crosswalks, or watching carefully to ensure the right-of-way of bicyclists
> and motorcyclists. 5. Motorists must **respect** the right-of-way of others by not violating traffic laws such as failing to stop at a stop sign or traffic light, speeding, making unsafe lane changes, or illegal turns. 6. Statistics show that right-of-way violations cause a high percentage of injury collisions in California.

This paragraph is a source of many useful terms for the driving domain. Sentence 4 introduces the notion that the right of the way is not just limited to yielding to others when needed, but it also means obeying the law as suggested in sentence 5. Sentence 6 is a general statement highlighting the importance of following the right-of-the-way law. From this paragraph, we will only represent the terms introduced in sentence 4, and represent the

sentence 5 as a driving rule.
We begin by defining the terms introduced in this paragraph.

We define the class *Person*, and define *Pedestrian* as its subclass. We define *Bicyclist, Motorist*, and *MotorCyclist* as subclasses of *DrivingEntity* and *Person.*

Person :: Entity.
Pedestrian :: Person.

Bicyclist :: (DrivingEntity, Person).
Motorist :: (DrivingEntity, Person).
MotorCyclist :: (DrivingEntity, Person).

We define the class *Crosswalk* and *StreetCorner* as a subclass of *SpatialEntity*. We define *TrafficSign* as a subclass of *SpatialEntity*, and define *StopSign* and *TrafficSign* as its subclasses.

CrossWalk :: SpatialEntity.
StreetCorner :: SpatialEntity.

TrafficSign :: SpatialEntity.
StopSign :: TrafficSign.
TrafficSignal :: TrafficSign.

We define *FailToStopAtTrafficSign, Speeding*, MakingIllegalTurn and *MakingUnsafeLaneChange* as subclasses of *IllegalDrive.*

FailToStopAtTrafficSign :: IllegalDrive.
Speeding :: IllegalDrive.
MakingIlliegalTurn :: IllegalDrive.
UnsafeLaneChange :: IllegalDrive.

We define a relation v*iolatesRightOfTheWay* with domain *Drive*, and range *Entity.*
*Drive :: Action [violatesRightOfTheWay *=> Entity].*

To formalize the intent of sentence 5, we say that any *IllegalDrive* violates the right of the way of others that are also on the road.

```
?I [ violatesRightOfTheWay -> ?Q] :-
  ?_P : Entity,
  ?Q : Entity,
  ?S : DrivingSituation [participant -> ?_P,
                         participant -> ?Q],
  ?I : IllegalDrive [agent -> ?_P,
                     follows -> ?S],
  not  ?_P :=: ?Q.
```

We test this rule by introducing the following situation:

```
P4 : Person.
P5 : Person.
S3 : DrivingSituation [ participant -> P4, participant -> P5].
I1 : FailToStopAtTrafficSign [agent -> P4, follows -> S3].
```

We pose the following query:

```
flora2 ?- I1 [violatesRightOfTheWay -> ?x].
?x = P5

1 solution(s) in 0.0310 seconds
Yes
```

An answer of ``Yes'', and the correct binding for the query variable confirms the correct functioning of this rule.
It is not necessary to represent sentence 6 as it does not provide operational/actionable knowledge for a driver.

Next, we consider the 6th paragraph on page 26.

> • **Respect** the right-of-way of pedestrians. Always stop for any pedestrian crossing at corners or other crosswalks, even if the crosswalk is in the middle of the block and at corners with or without traffic lights, whether or not the crosswalks are marked by painted lines.

?S [rightOfTheWay -> ?P] :-
?C : CrossWalk,
?P : Pedestrian,
?S : DrivingSituation [location -> ?C, participant -> ?P].

To test this rule, we introduce the following situation:
C1 : CrossWalk.
P6 : Pedestrian.
S4 : DrivingSituation [location -> C1, participant -> P6].

And then pose the following query:
flora2 ?- S4 [rightOfTheWay -> ?x].
?x = P6

1 solution(s) in 0.0000 seconds
Yes

The correct answer confirms the encoding of this rule.

Sample Test Questions

We will now consider a few example questions that exercise more than one rule at a time. The intention of such questions is to test the overall question answering ability of a knowledge base so that it can handle novel questions that are not known ahead of time.
Consider the question: ``A car and pedestrian are stopped at a corner. If the car proceeds whose right of the way will it violate?''
To answer this question, the system first needs to determine that the pedestrian has the right of the way, and by noticing that the car violates the right of the way conclude that it is an illegal drive, from which it can conclude that this drive must violate the right of the way of the pedestrian.

P7 : Car.
P8 : Pedestrian.
C3 : StreetCorner.
S6 : DrivingSituation [location -> C3,
participant -> P8,
participant -> P7].
D3 : Drive [agent -> P7,
follows -> S6].

We pose the query:

flora2 ?- D3 [violatesRightOfTheWay -> ?x].
?x = P8

1 solution(s) in 0.0160 seconds
Yes

It returns the expected answer. Let us now consider another question: ``If I do not stop at a traffic sign, do I violate somebody's right of the way?''
P9: Entity.
P10 : Entity.
S7 : DrivingSituation [participant -> P9,
participant -> P10].
F1:FailToStopAtTrafficSign [agent -> P9,
follows -> S7].

Answering this question requires using inheritance to conclude that *FailToStopAtTrafficSign* is an *IllegalDrive*, and then using a rule to conclude that this action must violate the right of the way of the person who might be at the same location. We can pose the query as:

flora2 ?- F1 [violatesRightOfTheWay -> ?x].
?x = P10

1 solution(s) in 0.0000 seconds
Yes

This returns the expected result.