Assignment Three: Logic and Planning
Due: 1:00pm, Wednesday 18 November 2015.

This can be done in groups of size 1, 2 or 3. Working alone is not recommended. A group of size \( n \) can choose any \( n + 2 \) questions out of questions 1 to 6. Every group should do question 7. All members of the group need to be able to explain the group’s answer. Please look at all of the questions, as the exam will assume that you have thought about all of the questions.

Submit your answers in class or in box #26 in room X235. You must submit a written response to the questions; it is this written response that will be marked. Use proper sentences in your answer. Code, AILog and/or AIspace representations should be submitted via handin using the directory cs322/assign3 in case the markers want to look at it. Use the cover sheet provided on the course web page.


Ask questions on Piazza. Feel free to answer them too. Note that only questions 3 and 6 involve programming.

**Question One**

Consider the domain of house plumbing represented in Figure 1. In this figure, \( p_1, p_2 \) and \( p_3 \) are water pipes. \( p_1 \) is the pipe coming in from the main water supply. \( t_1, t_2 \) and \( t_3 \) are taps and \( d_1, d_2 \) and \( d_3 \) are drainage pipes. The other labels should be obvious.

Suppose we have the following atoms:
- \( \text{pressurised}_{p_i} \), is true if pipe \( p_i \) has mains pressure in it. \( p_1 \) is always pressurized. Other pipes are pressurized if they are connected to a pressurized pipe through an open tap.
- \( \text{on}_{t_i} \) is true if tap \( t_i \) is on.
- \( \text{off}_{t_i} \) is true if tap \( t_i \) is off.
- \( \text{wet}_b \) is true if \( b \) is wet.
- \( \text{flow}_c \) is true if water is flowing through component \( c \).
- \( \text{plugged}_c \) is true if component \( c \) has the plug in.
- \( \text{unplugged}_c \) is true if component \( c \) doesn’t have the plug in.

Assume the taps and plugs have been in the same positions for one hour; you don’t need to consider the dynamics of turning on taps and inserting and removing plugs.

You can either use AILog or the definite clause interpreter in Python. The file http://www.cs.ubc.ca/~poole/cs322/2015/as3/plumbing.ailog contains a AILog axiomatization for how water can flow down drain \( d_1 \) if taps \( t_1 \) and \( t_2 \) are on and the bath is unplugged. [http://www.cs.ubc.ca/~poole/cs322/2015/as3/plumbing.py](http://www.cs.ubc.ca/~poole/cs322/2015/as3/plumbing.py) contains a Python representation (and is also in the Python distribution).

(a) Finish the axiomatization for the sink in the same manner as the axiomatization for the bath.

(b) What information would you expect the occupant of the house to be able to provide that the plumber can’t? Change the axiomatization so that such information is asked of the user of the system.
(c) Axiomatize how the floor is wet if the sink overflows or the bath overflows. They overflow if the plug is in and water is flowing in. You may invent new atoms as long as you give their intended interpretation.

(d) Suppose a hot water system is installed to the left of tap $t1$. This has another tap in the pipe leading into it, and supplies hot water to the shower and the sink (there are separate hot and cold water taps for each). Add this to your axiomatization. You don’t need to model water temperature. Give the denotation for all atoms you invent.

You need to hand in a complete listing of your program, including the intended interpretation for all symbols used and a trace of the AILog or Python session or to show it runs.

Question Two

This question considers using integrity constraints and consistency-based diagnosis in an agent that interacts with various information sources on the web. To answer a question, the agent will ask a number of the information sources for facts. However, information sources are sometimes wrong. It is useful to be able to automatically determine which information sources may be wrong when a user gets conflicting information.

In this question explores how integrity constraints and assumables can be used to determine what errors are present in different information sources.

This question uses meaningless symbols such as $a$, $b$, $c$, …, but in a real domain there will be meaning associated with the symbols, such as $a$ meaning “there is skiing in Hawaii” and $z$ meaning “there is no skiing in Hawaii”, or $a$ meaning “butterflies do not eat anything” and $z$ meaning “butterflies eat nectar”. We will use meaningless symbols in this question because the computer does not have access to the meanings and must simply treat them as meaningless symbols.

Suppose the following information sources and associated information are provided:

Source $s_1$: Source $s_1$ claims the following clauses are true:

$$a \leftarrow h.$$  
$$d \leftarrow c.$$
Source \(s_2\): Source \(s_2\) claims the following clauses are true:

\[
e \leftarrow d.
\]
\[
f \leftarrow k.
\]
\[
z \leftarrow g.
\]
\[
j.
\]

Source \(s_3\): Source \(s_3\) claims the following clause is true:

\[
h \leftarrow d.
\]

Source \(s_4\): Source \(s_4\) claims the following clauses are true:

\[
a \leftarrow b \land e.
\]
\[
b \leftarrow c.
\]

Source \(s_5\): Source \(s_5\) claims the following clause is true:

\[
g \leftarrow f \land j.
\]

Yourself: Suppose that you know that the following clauses are true:

\[
false \leftarrow a \land z.
\]
\[
c.
\]
\[
k.
\]

Not every source can be believed, because together they produce a contradiction.

(a) Code the knowledge provided by the users into AILog syntax using assumables. To use a clause provided by one of the sources, you must assume that the source is reliable.

(b) Use the program to find all of the conflicts about what sources are reliable. (To find conflicts in AILog, ask \(false\), and then use \(more\).) Either use AILog or you can go the proof by hand.

(c) Which single source(a), if unreliable, could account for a contradiction (assuming all other sources were reliable)? Explain how, and explain the significance.

(d) What are the minimal diagnoses? Explain how you got these, and explain the significance of them.

**Question Three**

Extend either the top-down interpreter or the bottom-up Python interpreter at to allow for integrity constraints and assumables.

(a) Implement a function that takes in a knowledge base and a set of assumable and returns the set of the minimal conflicts.
(b) Find minimal diagnoses. This should take in the minimal conflicts and return the minimal diagnoses. You need to demonstrate that your code works on some test cases.

This does not neet to change the definition of clauses or of a knowledge base (but you can add “Assumable” in the same way as “Askable” in implemented). If you do this question, you can use your code instead of AILog to implement the previous questions. Note that we did not cover the bottom-up interpreter in class, so you will need to read the textbook to determine how to do this.

**Question Four**

(a) Change the representation of the delivery robot world [Example 6.1] so that the robot cannot carry both mail and coffee at the same time. Test it on an example that gives a different solution than the original representation.

(b) Suppose the robot cannot carry both mail and coffee at the same time, but the robot can carry a box in which it can place objects (so it can carry the box and the box can hold the mail and coffee). Suppose boxes can be picked up and dropped off at any location. Give the STRIPS representation for the resulting problem (including the actions, the start state and the goal state) and test it at least on the problem of starting from the lab with mail waiting, the robot must deliver coffee and the mail to Sams office.

You must either use the Python planning code or do it by hand, showing how a planner would get the solution you claim. What you hand in must both give the answer, and evidence that it works.

**Question Five**

For this question, use the delivery robot world of the textbook. To test your answer, you can use (but don’t need to change) the supplied searching and planning code.

- Define a heuristic function for the forward planner that takes into account the goal and the state. Show that using the heuristic means fewer nodes are expanded.

- Define a heuristic function for the regression planner that takes into account the goal and the node. Show that using the heuristic means fewer nodes are expanded.

**Question Six**

Implement loop detection in the Python implementation of regression planning. This is an extension of loop detection in graphs: there is a loop if the current subgoal (the body of the answer clause) is a subset of the subgoal of a node on the current path.

You must demonstrate that it works either with the examples in the existing code base or with an example you create yourself.

**Question Seven**

For each question, specify how long you spend on it, and what you learned. How was the work in the team allocated? Was the question reasonable? (This questions is worth marks, so please do it!)