CS 380 Artificial Intelligence, Spring 2019
Sect: 002
Homework 1 - AI Beginnings

by JL Popyack

Due Date: Wednesday, April 17, 2019, 11:55 PM

PURPOSE:

This assignment provides introductory experiences with production systems, agents and working with Python and LISP programming, plus a view of current practices in artificial intelligence.

THE ASSIGNMENT:

This assignment consists of two sections:

1. Written problems to be turned in.
2. A Python and/or LISP programming assignment to be completed.

Reading:

- *From the text*:
    - Chapters 1 (Introduction), 2 (Artificial Intelligence)

Click here to return to top of this page.

1. **Written Problems to be turned in**:
   You should create a PDF with your answers to these questions. For Problem A, some of your answers will also be submitted to an online form, so that we may assemble a class bibliography of current events in AI.
    A. **Current Events - Artificial Intelligence and Learning Systems *(20 points)*** Visit the ACM Tech News archives (`https://technews.acm.org/archives.cfm`). Find a current article (i.e., appearing sometime since January 1, 2018) which deals with current events/research/products, etc. in the field of Artificial Intelligence.
        - Write a paragraph summarizing the main points of the article.
        - In a separate paragraph, you should discuss the *societal impact* of the technologies discussed in the article, *including benefits and harmful effects*. Topics you might consider include:
            - How do you think things will advance in the next ten years?
            - Is there a potential to apply the technology to application domains other than

the current use?
- What risks exist in the use of such technology? How might it be misused for nefarious purposes? What inadvertent side effects might be realized? Is there potential for distaster?
- Speculate on how this technology can be used most effectively in the home, workplace and entertainment industry of the future.

Some of these questions may not be applicable for the article you chose. However, your paragraph should examine both potential benefits and harmful harmful effects of the technology.
- Write a description of 1-2 sentences that informs a potential reader what the article contains. You will use this description as part of your submission to an online form, along with title, author, date and location of the article (see "What to Submit", below):

B. **Production Systems *(40 points - 20 points each)*:** Specify a knowledge base, rules (precondition/action pairs), initial state, state representation, and a goal/termination condition for the following problems. Be as explicit as possible, providing a specific data representation for the states and rules and showing how the results of applying a rule to a state are determined. ***You do not need to solve the problem, or specify an algorithm for solving the problem; possible algorithms for choosing between applicable rules will be covered later***:
   a. *Losing Your Marbles:* Each of three baskets contains a certain number of marbles. You may move from one basket into another basket as many marbles as are already there, thus doubling the quantity in the basket that received the marbles. You must find a sequence of moves that will yield the same number of marbles in the three baskets (or decide that no such sequence exists).
   b. The *N Queens* problem is to place *N* queens on an *N*×*N* chessboard, in such a way that they are "mutually nonthreatening", that is, none can attack any other. A chess queen may move from one cell on the board to any other cell which is either either in the same row, same column, or along the same diagonal.

C. ***Chapter 2 (Intelligent Agents) - 10 pts***: For each of the following agents, develop a PEAS description of the task environment (as in Figures 2.4-2.5 from the text), and characterize the task environment (as in Figure 2.6):
   i. an intelligent thermostat (e.g., the Nest Programmable Thermostat (`https://nest.com/thermostat/inside-and-out/#explore-your-nest`)). You may wish to include features not available for the *Nest*. If so, identify these.
   ii. intelligent television console - the problems here are a dizzying number of channels and choices, numerous people in the household with different viewing preferences, etc.

Click here to return to top of this page.

2. **Programming Assignment - Search/Functional Programming Beginnings *(30 points)* :**

The purpose of this exercise is to develop some of the functions necessary to solve pegboard problems, by formulating them as production systems. For the next several assignments, you will be writing a *Python* program to solve this problem, using various search strategies. ***For extra credit, you may additionally write a LISP program to solve it.*** For either language, you should use a functional programming approach and representation as described below (e.g., we will use Python list representations which closely match their LISP counterparts). For this assignment, you do not have to implement a solution strategy, other than the "flail wildly" search strategy, which is used to demonstrate that your routines work.

Pegboard problems are single-player games played on a grid (or *pegboard*), in which moves are made by successively jumping and removing pegs from the pegboard. A peg can jump an adjacent peg if there is a slot adjacent to that peg in the opposite direction - horizontally, vertically, or diagonally. After a peg has been jumped, it is removed from the board (and possibly eaten). A typical objective of this problem is to begin with a full pegboard from which one peg has been removed, and determine a sequence of jumps which will result in one peg remaining, perhaps in the position from which the first peg was removed. A popular form of this problem involves a rectangular pegboard with 4 slots on a side (see below), which proves to be a challenging problem for a human being.

State Representation	**Example**

`F E D C`
`B A 9 8`
`7 6 5 4`
`3 2 1 0`
	`X O X X`
	`O O X O`
	`X O X X`
	`X X X X`

FEDC BA98 7654 3210	1011 0010 1011 1111

The knowledge base for this problem consists of a *m×n* grid of slots. The state *s* will be a partially filled-in grid. As shown above, each slot can be represented by a single bit, and therefore a *m×n* grid can be represented by *mn* bits. In particular, the *4×4* grid shown in the example can be represented by a single 16-bit unsigned integer.

A rule *r* can be characterized by the attributes *(jumper, goner, newpos)*, which respectively refer to the position of a peg that is about to jump (*jumper*), the position of the peg it jumps over (*goner*), and the new position of the jumper (*newpos*). The rule is defined by the following action and preconditions:

- *Action*: change values in state *s* of *jumper* position to 0, *goner* position to 0, and *newpos* position to 1.
- *Precondition*: values of *jumper, goner, newpos* positions are respectively *1*, *1*, and *0*.

For instance, in the example depicted above, the peg in position *C* can jump over the peg in position *9* and land in position *6*. The current state is 1011001010111111, a binary number having the decimal value *45759*. The rule *r* can be represented by the binary numbers 0001000000000000 (which has the decimal value $2^{12}=4096$), 0000001000000000 ($=2^9=512$) and 0000000001000000 ($=2^6=64$) . The result of applying the rule is placing a peg in position *6* (adding 2^6) and removing those in positions *9* and *C* (subtracting 2^9 and 2^{12}). The result therefore is 1010000011111111, which has the decimal value *41215=45759+64-4096-512*.

Using this representation, write the following statements and functions, and test them using Python, showing your output.

a. Assign a *m×n* grid filled with pegs in all but position *9* to **`initialState`**.
b. Write a function **`goal(state)`** which returns **`True`** if **`state`** equals the state with exactly 1 peg, in position *9*.
c. Write a function **`applyRule(rule,state)`** which returns the value of applying a rule to a given state. This does not change the value of **`state`**.
d. Write a function **`precondition(rule,state)`** which returns **`True`** if the given **`rule`** may be applied to **`state`**, that is, the preconditions are satisfied. For instance, given the value of **`state`** from the example above, the precondition for rule **`[C,9,6]`** is satisfied (and **`True`** is returned) because slots *C*, *9* and *6* are consecutive slots along a diagonal, and there are pegs in slot *C* and slot *9* and no peg in slot *6*.
e. Write a function **`applicableRules(state)`** which calls **`precondition`** and returns a *list* of all possible rules that may be applied to the current state, where rules have the form described above. This should be a list of all possible rules which satisfy the preconditions for the given state. For instance, for the value of state in the example above, **`applicableRules`** should return **`[[C,D,E], [4,5,6], [3,7,B], [0,4,8], [2,5,8], [C,9,6], [4,9,E], [0,5,A]]`** (in some order).
f. Write a function **`describeState(state)`**, which prints the partially filled in grid corresponding to **`state`**.
g. Write a function **`describeRule(rule)`**, which explains the meaning of the given rule, e.g.,
The peg in slot 4 jumps over the peg in slot 9 and lands in slot E.
h. Test these primitives by writing a routine **`flailWildly(state)`**, which repeatedly tests **`goal(state)`** and if a goal has not been reached, determines all applicable rules for the current state, chooses one randomly and applies it. At each step, describe the current state, describe each applicable rule, and describe which rule has been chosen. If no rules are applicable, flailing stops with failure.

Click here to return to top of this page.

WHAT TO SUBMIT:

All homework for this course must be submitted electronically using Bb Learn and Google Forms. ***Do not e-mail your assignment to a TA or Instructor!*** If you are having difficulty with your Blackboard account, ***you*** are responsible for resolving these problems with a TA, an Instructor, or someone from IRT, before the assignment it due. It is suggested you complete your work early so that a TA can help you if you have difficulty with this process.

For this assignment, you must submit:

- to Bb Learn:
  - A PDF document with your answers to the "Written problems to be turned in".
  - Your Python source code, written documentation for your program, and results of your testing.
  - (Extra Credit) LISP source code for the programming exercise

- a Google Form (click here) :
    - information about the article you read, including title, author(s), date, URL and a 1-2 sentence summary description.

Strongly recommended. We strongly recommend using a compression utility (such as 7Zip - download a *free* demo) so that you can compress your files into a single file (with a **`.zip`** extension) and just upload it, rather than go through the tedious and error-prone process of uploading each file separately.

Click here to return to top of this page.

ACADEMIC HONESTY:

You must compose all program and written material yourself, including answers to book questions. All material taken from outside sources must be appropriately cited. If you need assistance with this aspect of the assignment, see a consultant during consulting hours.

Click here to return to top of this page.