AERO 489 – Selva
Ian Wilhite

2/24/2025

Short questions (10pts)

1) Syntax vs semantic
    a. Syntax is the structure of the notation for the problem. Semantics are the intended meaning of the problem.
    b. The difference between the two can arise when the syntax of the problem can "misunderstand" the intended result, like in the "everybody loves somebody" example in class that results in one "somebody" that everybody loves.
2) Predicate vs Function
    a. A predicate is a Boolean logical statement that evaluates to true or false. A function maps an input to an output and can return different data types. A predicate can be viewed as a type of function that only returns a Boolean.
    b. The IsEven(int num) returns [True, False] and would be considered a **predicate**, because the argument is a boolean.
    c. The SquareNum(int num) returns [int squaredNum], and would be considered a **function** because the return argument is not a boolean.
3) Action schema
    a. The action schema is the notation that describes the outcome of when a given action is applied to an object in a state.
4) Ground atomic sentence
    a. A ground atomic sentence is one that explains a particular relationship or fundamental existence of the world. Ie: Married(John, Jane)
5) Database vs standard FOL semantics
    a. Database semantics assume that information not explicitly given is false.
    b. FOL semantics will not assume anything about fact, and will claim that it could be either true or false.

Problem 1: FOL

a) Tell rules

```
KB_Family = FolKB([

    expr('Married(h, w) ==> Married(w, h)'),

    expr('Child(c, p) ==> Parent(p, c)'),

    expr('(Parent(p, s1) & Parent(p, s2) & Different(s1, s2)) ==> Sibling(s1, s2)'),

    expr('(Married(b, w) & Sibling(w, s)) ==> BrotherInLaw(b, s)'),
    expr('(Married(s, h) & Sibling(h, b)) ==> SisterInLaw(s, b)'),

    expr('Parent(p, c) ==> Ancestor(p, c)'),
    expr('(Parent(p, c) & Ancestor(c, a)) ==> Ancestor(p, a)'),

    expr('(Parent(a, b) & Parent(b, c)) ==> Grandparent(a, c)'),
    expr('(Parent(a, b) & Parent(b, c) & Parent(c, d)) ==> GreatGrandparent(a, d)'),

    expr('(Child(c, h) & Married(h, w)) ==> Child(c, w)')
])
```

b) Tell facts

```
58
59    print(f'adding facts to KB...')
60
61    KB_Family.tell(expr('Married(George, Mum)'))
62    KB_Family.tell(expr('Married(Elizabeth, Phillip)'))
63    KB_Family.tell(expr('Married(Edward, Sophie)'))
64    KB_Family.tell(expr('Married(Andrew, Sarah)'))
65    KB_Family.tell(expr('Married(Anne, Mark)'))
66    KB_Family.tell(expr('Married(Diana, Charles)'))
67    KB_Family.tell(expr('Married(Spencer, Kydd)'))
68
69    KB_Family.tell(expr('Parent(George, Elizabeth)'))
70    KB_Family.tell(expr('Parent(Elizabeth, William)'))
71    KB_Family.tell(expr('Parent(Elizabeth, Harry)'))
72
73    KB_Family.tell(expr('Child(Elizabeth, George)'))
74    KB_Family.tell(expr('Child(Margaret, George)'))
75    KB_Family.tell(expr('Child(Diana, Spencer)'))
76
77    KB_Family.tell(expr('Child(Charles, Elizabeth)'))
78    KB_Family.tell(expr('Child(Anne, Elizabeth)'))
79    KB_Family.tell(expr('Different(Charles, Anne)'))
80    KB_Family.tell(expr('Child(Andrew, Elizabeth)'))
81    KB_Family.tell(expr('Child(Edward, Elizabeth)'))
82    KB_Family.tell(expr('Different(Andrew, Edward)'))
83
84    KB_Family.tell(expr('Child(William, Charles)'))
85    KB_Family.tell(expr('Child(Harry, Charles)'))
86    KB_Family.tell(expr('Different(William, Harry)'))
87    KB_Family.tell(expr('Child(Peter, Anne)'))
88    KB_Family.tell(expr('Child(Zara, Anne)'))
89    KB_Family.tell(expr('Different(Peter, Zara)'))
90    KB_Family.tell(expr('Child(Beatrice, Andrew)'))
91    KB_Family.tell(expr('Child(Eugenie, Andrew)'))
92    KB_Family.tell(expr('Different(Eugenie, Beatrice)'))
93    KB_Family.tell(expr('Child(Louise, Edward)'))
94    KB_Family.tell(expr('Child(James, Edward)'))
95    KB_Family.tell(expr('Different(Louise, James)'))
96
97    KB_Family.tell(expr('Parent(George, Elizabeth)'))
98    KB_Family.tell(expr('Parent(Elizabeth, William)'))
99    KB_Family.tell(expr('Parent(Elizabeth, Harry)'))
100   KB_Family.tell(expr('Parent(Charles, William)'))
101   KB_Family.tell(expr('Parent(Charles, Harry)'))
102   KB_Family.tell(expr('Different(Harry, William)'))
103   KB_Family.tell(expr('Different(Beatrice, Eugenie)'))
104
105   print(f'facts added to KB.')
```

c) Ask
    a. Elizabeth's grandchildren
    b. Eugenie's siblings
    c. Zara's great-grandparents
    d. Eugenie's ancestors

```
[Running] python -u "c:\Users\ianwi\OneDrive\Documents\S4\AERO489\aima-python-61d695b37c6895902081da1f37baf645
creating KB...
KB created.
adding facts to KB...
facts added to KB.
asking Q1
Q1: [{x: Harry}, {x: William}, {x: James}, {x: Louise}, {x: Zara}, {x: Peter}, {x: Beatrice}, {x: Eugenie}]
asking Q2
Q2: [{x: Beatrice}]
asking Q3
Q3: [{x: George}, {x: Mum}]
asking Q4
Q4: []
exiting

[Done] exited with code=0 in 234.011 seconds
```

```
[Running] python -u "c:\Users\ianwi\OneDrive\Docum
creating KB...
KB created.
adding facts to KB...
facts added to KB.
asking Q1
Q1: []
asking Q2
Q2: [{x: Beatrice}]
asking Q3
Q3: [{x: George}, {x: Mum}]
asking Q4
Q4: [{x: George}, {x: Mum}]
exiting

[Done] exited with code=0 in 297.549 seconds
```

Problem 2: Europa Rover

1) Domain:
    a. Requirements
        i. STRIPS is the most common basic notation that the PDDL system will use FOL to model the system.
    b. Predicates (variables to track)
        i. Rover location – the location of the rover on Europa (note: we do not need to track location alpha because it is static and does not move)
        ii. HasBiosample – whether or not the rover has obtained the sample
        iii. HasComponent – whether or not the rover has the component
        iv. BatteryCharged – whether or not the battery is charged

v. Drilled (based on location) – whether or not the current location has been drilled
vi. SoftwareUpdated – whether or not the rover software has been updated
vii. PreheatedDrill – whether or not the drilled has been preheated
viii. AnalysisCompleted – whether or not the analysis is complete
ix. ContactAvailible – whether or not transmission contact is available
x. DataTransmitted – whether or not the data has been transmitted (and the mission has been completed)

c. Actions
i. Charge
ii. Move
iii. Update software
iv. Retrieve components
v. Preheat drill
vi. Drill
vii. Analyze sample
viii. Transmit data

2) See attached files.