Artificial Intelligence

Assignment-1

Arnav Singh
2021019

Theory Assignment

1. Logic (5 marks): The universe either will simply exist as it is or end in a heat death. If there was no big bang then the universe simply existed. If and only if the universe is expanding then there was a big bang. If the universe is expanding and accelerated then it will end in a heat death.

   (a) Write the sentences using logical connectives.(1)

   (b) Write the contra-positive of the sentence using logical connectives. (1)

   (c) What can be inferred and not inferred from the statement. (1)

   (d) Draw the And-OR graph (2)

(a) Write the sentences using logical connectives.

First, let's divide the paragraph into multiple statements, which will be conjoined later using logical connectives.

P. The universe will simply exist as it is
Q. The universe simply existed
R. The universe will end in a heat death
S. There was a big bang
T. The universe is expanding
U. The universe is accelerated
~S. There was no big bang

Consequently, the sentences are:-

(i) **P ∨ R** (The universe either will simply exist as it is or end in a heat death)

(ii) ~S -> Q (If there was no big bang, then the universe simply existed)

(iii) T <-> S (If and only if the universe is expanding, then there was a big bang)

(iv) (T ∧ U) -> R (If the universe is expanding and accelerated, then it will end in a heat death)

Truth value of the para= (P v R) ∧ (~S->Q) ∧ (T<-> S) ∧ ((T ∧ U) -> R)

(b) Contrapositives are defined for implications
For the explicit implications, they are:

(ii) ~Q -> S (If the universe doesn't simply exist, there was a big bang)
(iii) ~S <-> ~T (If and only if there was not a big bang, the universe is not expanding)
(iv) ~R -> (~T ∨ ~U) (If the universe will not end in a heat death, then the universe is not expanding or not accelerated)

We can convert P ∨ R to ~P -> R, and its contrapositive is ~R -> P. It can be written as ' If the universe will not end in a heat death, then the universe will simply exist as it is'

(c)

Inference:

Assumptions: All the sentences in the para are true

If the universe is not expanding, then the universe simply existed.

This can be inferred by considering the different values of T and S, For T <-> S to be true. Now, if T=False, so should S=False.

Also, ~S -> Q should be true. Plugging in the value of S, True->Q Should be true, this only happens if Q=True.

So, if S is false, Q must be true.

Also, If the universe will not end in a heat death and there was not a big bang and the earth is not expanding, the universe will simply exist as it is. As in the same case when T and S are both false, The fourth sentence will always be true, because it will morph into False->R, which is true regardless of the value of R. so R could be False, but since PVR=True, P must be true in that case.

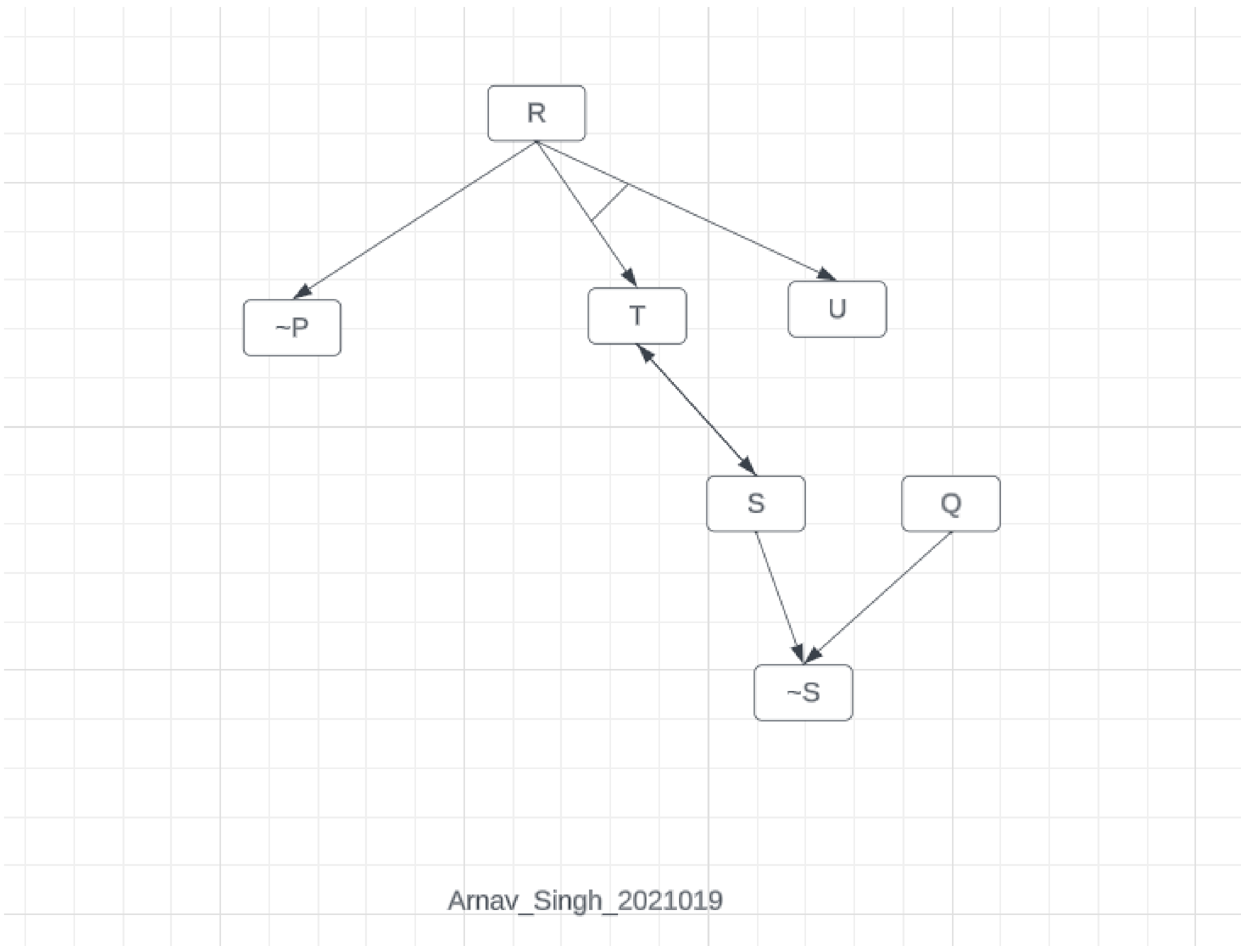
Not Inference:

Assumptions: All the sentences in the para are true

The universe will not simply exist as it is, or the universe will not end in a heat death.

This statement is false, considering that it is the negation of Statement, which has been assumed to be true. So, It can't be inferred From the givens statements. (since it negates the truth of our assumptions)

(d)

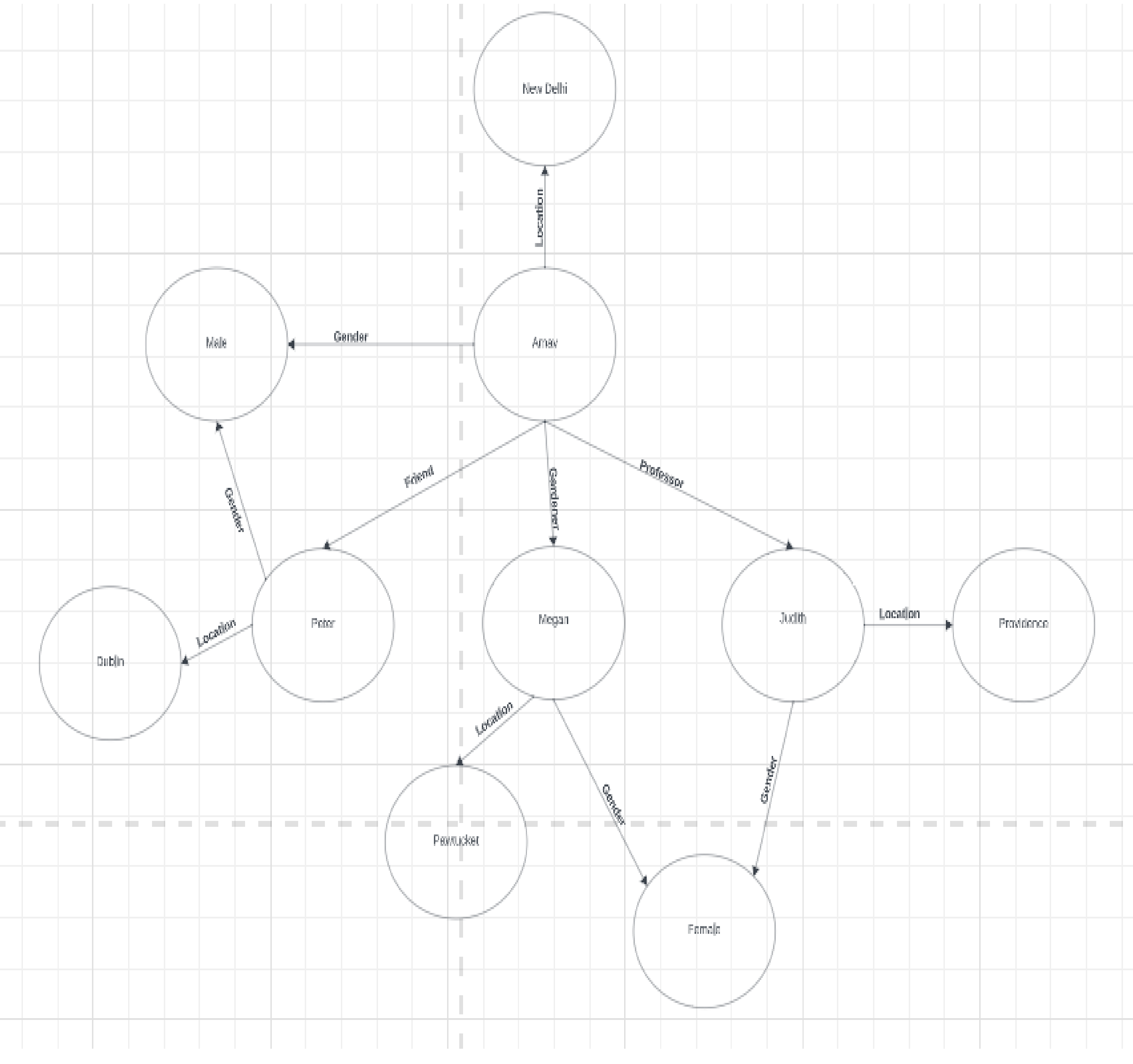


The arrows here are used to indicate that the node with the arrowhead imply the node on the other end. Unless a line is used to connect two arrows on the same level (which indicates and relationship), the nodes on their ends are in an or relationship. (the wikipedia representation is used)

2. Semantic Network (5 marks) : Construct a Semantic network based on the relation between three people and yourself.(2) Include the information of their :

   (a) Location (1)
   (b) Gender (1)

   Explain inheritance and multiple inheritance using examples from your Network. (2)



Inheritances

Arnav is a Male

Arnav lives in New Delhi

Arnav is Indian

Peter is a Male

Peter lives in Dublin

[Peter is Arnav's friend]

Megan is a Female

Megan lives in Pawtucket

[Megan is Arnav's gardener]

Judith is a Female

Judith lives in Providence

[Judith is Arnav's Professor]

These relationships are inheritances. Peter is a Male, and as a result, he inherits the properties of the Male Class (He has XY chromosomes). However, a class can inherit from multiple sources, Peter also lives in Dublin, by virtue of which he is a citizen of a democratic nation and can vote and elect his leaders, etc.

To put it simply, Multiple inheritance is when a person inherits traits from multiple classes, in this example, each individual inherits traits from multiple classes (location, gender, ethnicity), etc.

Acc. to Norvig:-

"Categories serve to organize and simplify the knowledge base through inheritance. If we say that all instances of the category Food are edible, and if we assert that Fruit is a subclass of Food and Apples is a subclass of Fruit, then we can infer that every apple is edible. We say that the individual apples inherit the property of edibility, in this case, from their membership in the Food category."

"Inheritance becomes complicated when an object can belong to more than one category
or when a category can be a subset of more than one other category; this is called multiple inheritance."

($\mathit{Extra\ credit}$: 3 marks) Show that the proof by resolution approach for propositional logic is sound and complete.

To Show:

The proof by resolution method for proposition logic is:-
(i) Sound: (if proof by res. is true entailment from premises to result)

Acc. to the lecture slides:-

Soundness: The *soundness* of the resolution rule can be seen easily by considering the literal ℓ_i that is complementary to literal m_j in the other clause. If ℓ_i is true, then m_j is false, and hence $m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n$ must be true, because $m_1 \vee \cdots \vee m_n$ is given. If ℓ_i is false, then $\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k$ must be true because $\ell_1 \vee \cdots \vee \ell_k$ is given. Now ℓ_i is either true or false, so one or other of these conclusions holds—exactly as the resolution rule states.

P	R	S	(PvS)	(PvS) v (~RvS)
T	T	T	T	T
T	F	F	T	T
T	T	F	T	T
T	F	T	T	T
F	T	T	T	T
F	F	F	F	F
F	T	F	F	F
F	F	T	T	T

Since, their truth tables match (which proves that there is entailment from premises to conclusion) ,. i.e. (P v S) is equivalent to (P∨R) ∨ (~R ∨ S) ensures that proof by res. is sound prop. logic.

((P v S) is true whenever (P∨R) ∨ (~R ∨ S) is)

(ii) Completeness:

Proof by resolution for propositional logic is complete because It can output each and every true statement in the system. (if entailment from premises to result exists, we can prove it)

Proof by Resolution

- Example to motivate resolution of one literal at a time

```
function PL-RESOLUTION(KB, α) returns true or false
  inputs: KB, the knowledge base, a sentence in propositional logic
          α, the query, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
  new ← { }
  while true do
     for each pair of clauses Ci, Cj in clauses do
        resolvents ← PL-RESOLVE(Ci, Cj)
        if resolvents contains the empty clause then return true
        new ← new ∪ resolvents
     if new ⊆ clauses then return false
     clauses ← clauses ∪ new
```

Figure 7.13 A simple resolution algorithm for propositional logic. PL-RESOLVE returns the set of all possible clauses obtained by resolving its two inputs.

As in the above example from lecture slides, we can prove completeness after assuming that it can be true for some Values of boolean variables. Now, we can assume that its cnf Will resolve into ~ actual solution. if, we're unable to simplify the cnf into ~actual solution, we will have to accept that the cnf resolved into the actual solution.

<u>Coding</u>

Now, the way my recommendation algorithm works is that it takes inputs from the user as to their rating of a certain category, and its subcategories (and multiplies each by the app's own ratings of a certain place). Normalises it so as to ensure that each category contributes equally to the final rating, selects the place which is most suitable for the user, and returns it, along with feedback, should it be added by the user.

The knowledge base is a dictionary of len 50, which further contains dictionaries, which represent a place and its company rating.

The add_destination funcn simply adds a new location's dictionary to the kb, and add_feedback function adds a location's feedback to an array of arrays, which is indexically matched to the knowledge base.

(Both the app's ratings and the user's ratings are multiplied to increase accuracy. For example, if I rate museum visits as 10/10, but the app's rating of museums in town is 3/10, my resulting experience would be 30/100 (0.3 instead of 1). This suggests that even though I personally enjoy going to museums, the app indicates that the museum is in bad shape. Consequently, my expected experience is worse than it would have, were the place's museums better kept)
Below is a diagrammatic explanation of the algorithm:-