Part I: Knowledge based Agents

1. Properties of task environment for Minesweeper world

a. Partially Observable: In Minesweeper world, we are unable to determine the danger i.e. where the mine is present. Therefore, world is partially observable.

b. Stochastic: As the output of the action by the agent depending on the box we click; the environment is stochastic. We do not have any knowledge of mine locations.

c. Sequential: As the next actions depends on the box we click in the present state; the environment is sequential.

d. Static: The environment is static as environment does not change while the agent is deliberating.

e. Discrete: The environment has finite number of discrete states.

f. Single Agent: Only one player is considered while playing minesweeper. Therefore, it is a single agent environment.

g. Known: All possible rules for this environment is already known. Therefore, environment is known.

2. Changes in world if mines could relocate to different unexplored square between moves

Here I'm assuming that when an agent clicks a box and certain numbers are displayed on board. Then they are not changed even if the mine position changes. Only those mines are changed which are not adjacent to the known boxes.

a. Partially Observable: In Minesweeper world, we are unable to determine the danger i.e. where the mine is present. Therefore, world is partially observable.

b. Stochastic: As the output of the action by the agent depending on the box we click; the environment is stochastic. We do not have any knowledge of mine locations.

c. Sequential: As the next actions depends on the box we click in the present state; the environment is sequential.

d. Dynamic: Here the positions of mines are changing in between the agent's actions. Therefore, the environment is dynamic.

e. Discrete: The environment has finite number of discrete states.

f. Single Agent: Only one player is considered while playing minesweeper. Therefore, it is a single agent environment.

g. Known: All possible rules for this environment is already known. Therefore, environment is known.

3. Consider a variation of this world that contains a drone

a. Partially Observable: In Minesweeper world, we are unable to determine the danger i.e. where the mine is present. Therefore, world is partially observable.

b. Non-Deterministic: When we click a box with a drone, then it throws our agent to randomly selected square. No probability can be assigned to this that's why environment is non-deterministic.

c. Sequential: As the next actions depends on the box we click in the present state; the environment is sequential.

d. Static: Here the positions of mines are not changing in between the agent's actions. Therefore, the environment is static.

e. **Discrete:** The environment has finite number of discrete states.
f. **Single Agent:** Only one player is considered while playing minesweeper. Therefore, it is a single agent environment.
g. **Known:** All possible rules for this environment is already known. Therefore, environment is known.

Part II: Logic

4. Following knowledge base is given

$B \land C \Rightarrow A$ $\quad E \land F \Rightarrow A$ $\quad D \Rightarrow B$ $\quad F \land H \Rightarrow B$ $\quad E \Rightarrow C$

$H \Rightarrow D$ $\quad E$ $\quad G \Rightarrow F$ $\quad C \Rightarrow G$

a. Give a model of the knowledge base

One model is when all the items given are **true**

b. Give an interpretation that is not a model of the knowledge base:

E=False is an interpretation this is not a model of our knowledge base

c. Give two atoms that are logical consequences of the knowledge base

E and C are two atoms that are logical consequences of our knowledge base

d. Give two atoms that are not logical consequences of the knowledge base.

D, H and B are not logical consequences of our knowledge base.

5. Show all the possible resolutions for the following pairs of clause:
    a. A, -A V B

       Clause 1 : {A}
       Clause 2 : {-A,B}

       {B} ………..Using Resolution Principle

       Hence, we can derive B from two clauses

    b. A V B, -A V –B

       Clause 1: {A,B}
       Clause 2 : {-A,-B}

       {A,-A} or {B,-B} ….. At a time only one variable can be resolved

       Hence, we can derive A Λ -A and B Λ -B from these two clauses

    c. –X V Y, X V –Y V Y

       X V –Y V Y can be writtern as X as Y V –Y will always be True

       Clause 1: {-X, Y}
       Clause 2 : {X ,Y}

       {Y} ………..Using Resolution principle

       Hence, we can derive Y from these two clauses

Part III: FOL & Inferences

6. Represent the following knowledge base in first order logic. Use the predicates:
   - fast(y) • tasty(y) • has-cheese(y) • dieting(x)
   - likes (x, y) • eats (x, y) • hungry(x) • picky(x) •UMBC_Student(x)

where arguments x have the domain of all people, and arguments y have the domain of all food.

a) Anyone who is hungry and not dieting will not be picky.

$$\forall_x[hungry(x) \land - dieting(x) \Rightarrow -picky(x)]$$

b) Everyone who is picky only eats tasty food

$$\forall_x \forall_y[(picky(x) \land tasty(y) \Rightarrow eats(x, y)]$$

c) A person eats food if and only if they like it and are hungry.

$$\exists_x \forall_y[eats(x, y) \Leftrightarrow likes(x, y) \land hungry(x)]$$

d) A hungry person likes food that comes quickly

$$\exists_x[hungry(x) \Rightarrow \forall_y(likes(x, fast(y)))]$$

e) No-one who is dieting eats food with cheese.

$$\forall_x \forall_y[dieting(x) \Rightarrow eats(x, -(has - cheese(y)))]$$

f) Every UMBC student likes tasty food

$$\forall_x \forall_y[UMBC\_Student(x) \Rightarrow likes(x, tasty(y))]$$

g) Val is a UMBC student.

UMBC_Student(Val)

h) James ate chicken.

eats (James, chicken)

i) Val is dieting and did not eat pizza

dieting(Val) Λ *– eats (Val, pizza))*

j) James is picky

picky(James)

7. Convert the KB to conjunctive normal form.

a) Anyone who is hungry and not dieting will not be picky.
$\forall_x[hungry(x) \wedge -dieting(x) \Rightarrow -picky(x)]$
Step 1: Removing implication
$\forall_x - (hungry(x) \wedge - dieting(x)) \vee -picky(x)$
Step 2: Using DeMorgan's Law
$-[\forall_x - \left(hungry(x) \wedge - dieting(x)\right) \vee -picky(x)]$
$\exists_x[(hungry(x) \wedge - dieting(x)) \wedge picky(x)]$
Step 3: Removing Quantifiers
$hungry(x) \wedge - dieting(x) \wedge picky(x)$

b) Everyone who is picky only eats tasty food
$\forall_x \forall_y[(picky(x) \wedge tasty(y) \Rightarrow eats(x, y)]$
Step1: Removing Implications
$\forall_x \forall_y[-((picky(x) \wedge tasty(y)) \vee eats(x, y)]$
Step2: Using DeMorgan's Law
$-[\forall_x \forall_y[-((picky(x) \wedge tasty(y)) \vee eats(x, y)]]$
$[\forall_x \forall_y[(\left(picky(x) \wedge tasty(y)\right) \wedge - eats(x, y)]]$
Step3: Removing Quantifiers
$(\left(picky(x) \wedge tasty(y)\right) \wedge - eats(x, y)]$

c) A person eats food if and only if they like it and are hungry.
$\exists_x \forall_y[eats(x, y) \Leftrightarrow likes(x, y) \wedge hungry(x)]$
Step1: Removing bidirectional implication using bidirectional implication rule.
$\exists_x \forall_y[eats(x, y) \Rightarrow likes(x, y) \wedge hungry(x)] \wedge [likes(x, y) \wedge hungry(x) \Rightarrow eats(x, y)]$
Step2: Removing implication using implication rule
$\exists_x \forall_y[-eats(x, y) \vee likes(x, y) \wedge hungry(x)] \wedge [-(likes(x, y) \wedge hungry(x)) \vee eats(x, y)]$
Step3: Applying DeMorgan's Law
$-[\exists_x \forall_y[-eats(x, y) \vee likes(x, y) \wedge hungry(x)] \wedge [-(likes(x, y) \wedge hungry(x)) \vee eats(x, y)]]$
$-[\forall_x \exists_y[eats(x, y) \wedge - (likes(x, y) \wedge hungry(x))] \vee [likes(x, y) \wedge hungry(x) \wedge - eats(x, y)]]$
Step4: Dropping Quantifiers
$[eats(x, y) \wedge - (likes(x, y) \wedge hungry(x))] \vee [likes(x, y) \wedge hungry(x) \wedge - eats(x, y)]]$

d) A hungry person likes food that comes quickly
$\exists_x[hungry(x) \Rightarrow \ \forall_y(likes(x, fast(y)))\ ]$
Step1: Removing implication using implication rule from the equation
$\exists_x \forall_y - \big(hungry(x)\big) \vee (likes(x, fast(y)))\ ]$
Step2: Applying DeMorgan's Law
$-[\exists_x \forall_y - \big(hungry(x)\big) \vee (likes(x, fast(y)))\ ]]$
$\exists_x \forall_y [\big(hungry(x)\big) \Lambda - (likes(x, fast(y)))\ ]]$
Step3: Removing Quantifiers
$[\big(hungry(x)\big) \Lambda - (likes(x, fast(y)))\ ]]$

e) No-one who is dieting eats food with cheese.
$\forall_x \forall_y [dieting(x)\ \Rightarrow eats(x, -(has - cheese(y)))]$
Step 1: Removing implication using implication rule from the equation
$\forall_x \forall_y [-dieting(x)\ \vee eats(x, -(has - cheese(y)))]$
Step 2: Using DeMorgan's Law
$-[\forall_x \forall_y [-dieting(x)\ \vee eats(x, -(has - cheese(y)))]]$
$[\forall_x \forall_y [dieting(x)\ \Lambda - eats(x, -(has - cheese(y)))]]$
Step3: Removing Quantifiers
$[dieting(x)\ \Lambda - eats(x, -(has - cheese(y)))]$

f) Every UMBC student likes tasty food
$\forall_x \forall_y \big[UMBC_{Student(x)} \Rightarrow \ likes\big(x, tasty(y)\big)\big]$
Step1: Removing implication using implication rule from the equation
$\forall_x \forall_y \big[-UMBC_{Student(x)} \vee \ likes\big(x, tasty(y)\big)\big]$
Step2: Using DeMorgan's Law
$-[\forall_x \forall_y \big[-UMBC_{Student(x)} \vee \ likes\big(x, tasty(y)\big)\big]]$
$\forall_x \forall_y \big[UMBC_{Student(x)}\ \Lambda\ - likes\big(x, tasty(y)\big)\big]]$
Step3: Dropping quantifiers
$\big[UMBC_{Student(x)}\ \Lambda\ - likes\big(x, tasty(y)\big)\big]]$

g) Val is a UMBC student
UMBC_Student(Val)

h) James ate chicken
eats (James, chicken)

i) Val is dieting and did not eat pizza
dieting(Val) Λ – *eats (Val, pizza))*

j) James is picky
picky(James)

8. We wish to prove that *hungry(Val) ⇒ eats (Val, chicken)*

Converting to CNF

Step1: Removing implications

$-(hungry(Val) \vee eats(Val, chicken)$

Step2: Using DeMorgan's Law

$-[-(hungry(Val) \vee eats(Val, chicken)]$

$[(hungry(Val) \Lambda - eats(Val, chicken)]$

---

9. Add the negated goal to the KB, and use forward chaining to prove that it is true. Show your proof as a series of sentences to be added to the KB. (Denote new sentences with letters starting after k.) You must clearly show which sentences are used to produce each new sentence.

$\forall_x \forall_y [UMBC\_Student(x) \ \Rightarrow \ likes(x, tasty(y))$] .... (1)

Putting value Val for x

$\forall_y [UMBC\_Student(Val) \ \Rightarrow \ likes(val, tasty(y))$]

$\exists_x \forall_y [eats(x, y) \Leftrightarrow likes(x, y) \ \Lambda \ hungry(x)]$ ..... (2)

Putting Val for x in equation 2

$\forall_y [eats(Val, y) \Leftrightarrow likes(Val, y) \ \Lambda \ hungry(Val)]$

$\forall_x \forall_y [(picky(x) \Lambda tasty(y) \ \Rightarrow \ eats(x, y)]$ ...... (3)

Putting Val for x in equation (3)

$\forall_x \forall_y [(picky(Val) \Lambda tasty(y) \ \Rightarrow \ eats(Val, y)]$

We know that eats(James, chicken)

eats(Val, chicken)

We know that $\exists_x [hungry(Val) \Rightarrow \ \forall_y (likes(Val, fast(y)))$]

Thus we deduce from above equation : hungry(Val) -> eats(Val, chicken)

---

10. For two of the sentences in the KB, give a 1-2 sentence explanation of how those sentences are a poor representation of the real world.

Sentences "Everyone who is picky only eats tasty food" & "Every UMBC Student likes tasty food" show poor representation. They lack universal patterns of relationship among objects in knowledge base.

Reference:

[1] Russell, Stuart, Peter Norvig, and Artificial Intelligence. "A modern approach." *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs* 25 (1995): 27