CT Questions & Answers - answers.pdf

2022

1. (a) Comment on: The aim of Artificial Intelligence is to make machine intelligent. 6

   (b) What are the different types of Artificial Intelligence systems? Discuss briefly. 8

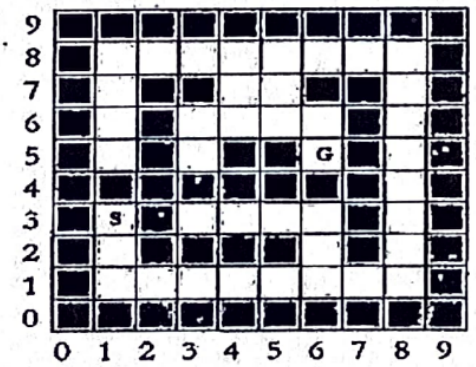
   (c) Discuss on Turing Test in connection to intelligence of a machine. Is there any limitation of this test? - Discuss. 6

c) (i) One limitation of Turing test is when the interrogator asks a question that's difficult for a human to solve, and thus even if the machine is super-intelligent and gives a correct answer, the interrogator can't judge its intelligence, if it doesn't have human-like action, ie it cannot measure intelligence that is beyond the ability of humans. Turing test is a one-sided test as a machine could be considered intelligent without knowing enough about humans to act like one

(ii)The interpretation of the test is subjective and depends on the judgement of the human evaluators. Different evaluators may have different opinions on whether a machine has passed the test, leading to inconsistencies in the results.

2. (a) You have read various search algorithms. How do you evaluate their performances? Briefly discuss on it. 5

   (b) Consider the problem of Maze search with the following figure. 15



In this problem, an agent is trying to traverse a maze from the starting point S to the goal point G. In each step, the agent can move in one of the four compass directions; each move costs 1 unit. The agent always considers alternative moves in the following order:

1. Move North
2. Move East
3. Move South
4. Move West

Apply A* search algorithm to solve this problem. Number the squares in the order the agents visits the squares, starting with 0 at the starting point. You do not need to re-expand nodes already visited; this means that you can "jump" from the current node to the next node in the queue.

Calculate the path cost on the basis of one cost unit per move. The heuristics to use is the function of the difference between the horizontal position of the current node and the goal node, plus the difference in the vertical position of the current node [xn,yn] and the goal node [xg,yg] and it is: $h([x,y]) = (|xg - xn|) + (|yg - yn|)$

For this problem,

- Mark the sequence in which the nodes are visited in the maze.
- Draw the corresponding search tree.
- Fill in the table with the information about the search trace. (initial one is S, and you may need to insert the rows into the table)

Current Node	Path Cost	Heuristic	f-cost	Queue
S				

2 a > Optimality , Completeness , Time Complexity , Space Complexity

---

3. (a) Suppose a given graph is weighted one. In each time while examining a node from OPEN list, the least cost one is selected and expanded. What is this search algorithm known as? Discuss briefly on its advantages and disadvantages. 5

   (b) Justify the follwing statements alongwith suitable reasoning.

   (i) General Graph Search Algorithm is applicable for a wide variety of search processes. 5

   (ii) Bidirectional search procedures are applicable for all kinds of problems. 5

   (iii) A* search is a combination of past and future. 5

---

4. (a) Consider the following game tree in which static scores are all from first player's point of view.

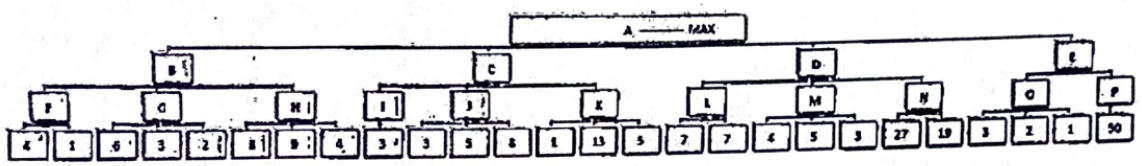
   (i) Which path will be chosen if Minimax algorithm is used?

   (ii) Which branches will be pruned if α-β pruning algorithm is used? 3+7

   (The static evaluation scores from left to right are:

   4 1 6 3 2 8 9 4 3 3 5 8 1 13 5 7 7 4 5 3 27 19 3 2 1 50)

   Show all intermediate values



(b) Compare the performances of Minimax and Alpha beta pruning algorithm. Write down the aspects for comparison (include at least time, space, and correctness aspects). 5

(c) Discuss on how AND-OR tree can be utilized for two person game playing tasks. 5

4.a. Similar to ct question

4.b.

Minimax	Alpha-beta
adversarial algorithm provides an optimal move for the player assuming that opponent is also playing optimally.	adversarial algorithm like minimax but prunes branches which are not necessary to search
No pruning of branches, entire tree is searched	Pruning is done, entire tree is not searched
Avg time complexity = O(b^d)	Avg time complexity = O(b^(d/2))
Worst time complexity = O(b^d)	Worst time complexity = O(b^d)
Space complexity is O(bd) as it does not store results of sub-trees	Space complexity is O(bd/2) as it stores results of sub-trees to prune sibling branches

4.c. The AND-OR GRAPH (or tree) is useful for representing the solution of problems that can solved by decomposing them into a set of smaller problems, all of which must then be solved. This decomposition, or reduction, generates arcs that we call AND arcs. One AND arc may point to any number of successor nodes, all of which must be solved in order for the arc to point to a solution. Just as in an OR graph, several arcs may emerge from a single node, indicating a variety of ways in which the original problem might be solved. This is why the structure is called not simply an AND-graph but rather an AND-OR graph (which also happens to be an AND-OR tree)
http://aimaterials.blogspot.com/p/and-or-graph.html

And-Or trees can be utilized in two-person game playing tasks to represent the search space of possible game moves and outcomes. In an And-Or tree, nodes represent game states, and edges represent game moves. Each node is labeled with a player identifier, indicating which player has the move in that state.
The And nodes in the tree represent the logical conjunction of the moves that the current player can make, while the Or nodes represent the logical disjunction of the moves that the opposing player can

make in response. The leaves of the tree represent terminal states, where the game is over and the outcome is known.
The use of And-Or trees allows for efficient search and evaluation of the game tree, as the search can be pruned when a certain outcome is determined to be unfavorable or impossible. In game playing tasks, the goal is typically to find the best move to make in a given game state, given a certain evaluation function that assigns scores to each game outcome.
And-Or trees can be used in various game playing tasks, such as chess, checkers, and Go. In these games, the search space is enormous, and a brute-force search is often infeasible. And-Or trees allow for a more efficient and effective search, allowing players to make better decisions and improve their gameplay.
Overall, And-Or trees are a powerful tool for representing and searching game trees in two-person game playing tasks, enabling efficient search and evaluation of game outcomes.

—----------

5.(a) Why do we require 'unification'? 3

(b) Find the *mgu* of the following: 5
{ Q(h(x,y), w), Q(h(g(v), a), f(v)), Q(h(g(v)), f(b)) }

(c) Convert the following *wff* into clause form. 6
$\forall x\,(f(x) \Rightarrow \exists y\,(m(x,y))) \wedge \forall x \forall z\,((a(x,z) \vee b(x,z)) \wedge c(z,3))$

(d) What is Skolemization? Eliminate Existential quantifier from the following WFF: 3+3

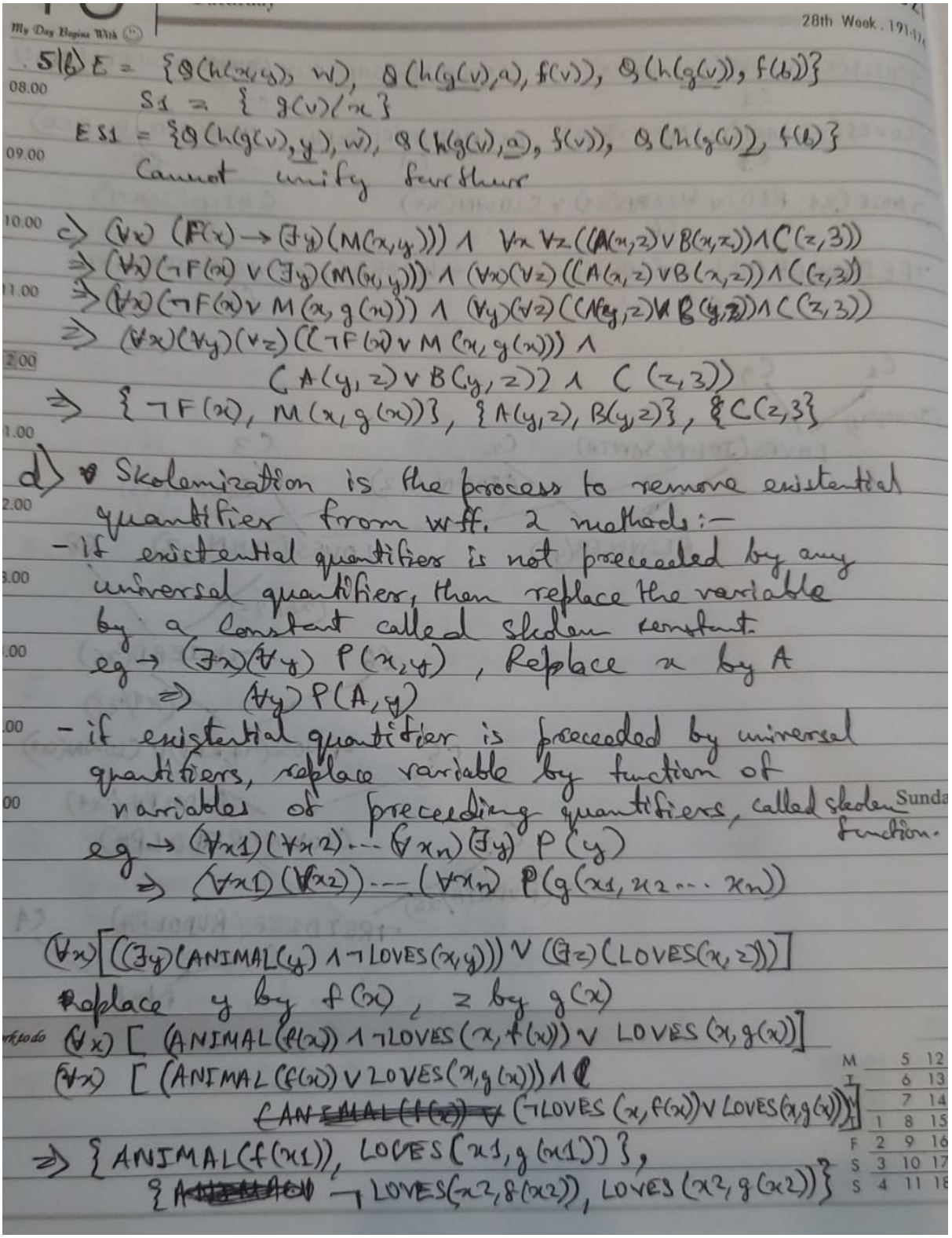
$$\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z\ Loves(z,x)]$$

—----------

5a
Unification is a fundamental operation in artificial intelligence and plays a crucial role in automated reasoning, natural language processing, and various other AI applications. Here are three reasons why unification is required in AI:

1. Inference: Unification is used to match two or more terms that have variables, allowing for the generation of new knowledge from existing knowledge. For example, if we know that "all cats are mammals" and "Fluffy is a cat," we can infer that "Fluffy is a mammal" by unifying the variable "cat" with the term "Fluffy." Unification is essential for automated reasoning systems to generate new knowledge from existing knowledge and make inferences.
2. Pattern Matching: Unification is also used in pattern matching, a technique used in natural language processing and other applications to identify and extract patterns from data. For example, in a text-processing application, we may use unification to match a pattern like "a noun followed by a verb" to identify sentence structures.
3. Planning: Unification is used in AI planning systems to match the preconditions and effects of actions. In planning, the goal is to find a sequence of actions that achieve a particular objective. Unification is used to match the preconditions of an action with the current state of the world and to unify the effects of the action with the desired state of the world. Unification helps to identify actions that can be executed and the changes that will result from those actions.

In summary, unification is required in AI because it enables automated reasoning, pattern matching, and planning, which are essential for various AI applications.



5 d i

Skolemization is a technique in mathematical logic and automated theorem proving that involves replacing existential quantifiers in a formula with Skolem functions, which are functions that return a witness for the existential quantifier based on the values of the universally quantified variables in the formula. The resulting formula is equisatisfiable with the original formula, meaning that if one is true, the other is also true. Skolemization is often used as a preprocessing step in automated theorem

proving to eliminate existential quantifiers and simplify formulas. It is named after the Norwegian mathematician Thoralf Skolem.

6. (a) Is it true that resolution refutation always terminates either by finding a contradiction or by failing to find a contradiction? Provide reasons in support of your answer. 4

(b) What is Refutation tree? Discuss on any two control strategies (along with their pros and cons) for performing resolution refutation. 2+6

(c) What is Fuzzy set? Draw the differences between Crisp set and Fuzzy set. 3+5

---

7. (a) What is classification? How can you use Genetic algorithm for classification? 2+4

(b) Discuss on the pros and cons of mutation operator used in GA . 4

6a

There are two possibilities for how the Resolution algorithm terminates: via a refutation step, or via clause saturation.

In the first case, there is a step for which $c_1 = x$ and $c_2 = \overline{x}$, in which case P and Q are empty, and hence the resolvent $P \vee Q$ is the **empty clause**, and is denoted by F, since it is logically equivalent to `false`. Such a step is called a **refutation step**, and the algorithm is said to have performed a **refutation proof**. In this case $\mathcal{C}$ is unsatisfiable since, by Proposition 1, any assignment that satisfies $\mathcal{C}$ must also satisfy a resolvent (in this case F) derived from $\mathcal{C}$.

In the second case, for each possible pair of clauses $c_1 = P \vee x$, and $c_2 = Q \vee \overline{x}$, it is the case that $P \vee Q$ is subsumed by a member of $\mathcal{C}$, in which case no new resolvents (including F) can be added to $\mathcal{C}$. In this case $\mathcal{C}$ is said to be **saturated**. In the next section we prove that saturated instances must be satisfiable. Conversely, if $\mathcal{C}$ is satisfiable, then, by Proposition 1, it cannot derive the empty clause. Moreover, it must become saturated since there are at most 3^n different possible resolvents (why?).

https://home.csulb.edu/~tebert/teaching/lectures/528/resolution/resolution.pdf

6.b. A resolution refutation can be represented as a refutation tree (within the derivation graph) where 2 nodes (clauses) are joined (resolved) to a third node (resolvant) which is essentially their simplification and it may have a root node labelled by NIL.

A control strategy for a refutation system is said to be complete if its use results in a procedure that will find a contradiction (eventually) whenever one exists

There are 5 control strategies:-

1. breadth-first strategy - all of the first-level resolvents are computed first, then the second-level resolvents, and so on. {A first-level resolvent is one between two clauses in the base set; an i-th level resolvent is one whose deepest parent is an (i-1)-th level resolvent.) The breadth-first strategy is complete, but it is grossly inefficient.

2. set-of-support refutation - is one in which at least one parent of each resolvent is selected from among the clauses resulting from the negation of the goal wff or from their descendants (the set of support). It can be shown that a set-of-support refutation exists whenever any refutation exists and, therefore, that the set of support can be made the basis of a complete strategy. The strategy need only guarantee to search for all possible set-of-support refutations (in breadth-first manner, say). Set-of-support strategies are usually more efficient than unconstrained breadth first ones.
3. unit-preference strategy - modification of the set-of-support strategy in which, instead of filling out each level in breadth-first fashion, we try to select a single-literal clause (called a unit) to be a parent in a resolution. Every time units are used in resolution, the resolvents have fewer literals than do their other parents. This process helps to focus the search toward producing the empty clause and, thus, typically increases efficiency.
4. linear-input - each resolvent has at least one parent belonging to the base set.linear-input form refutation is one in which each resolvent has at least one parent belonging to the base set.There are cases in which a refutation exists but a linear-input form refutation does not; therefore, linear-input form strategies are not complete.
5. ancestry-filtered - each resolvent has a parent that is either in the base set or that is an ancestor of the other parent. Thus, ancestry-filtered form is very much like linear form. It can be shown that a control strategy guaranteed to produce all ancestry-filtered form proofs is complete.

6c A fuzzy set is a type of set in which the membership of an element is not a binary value (0 or 1) but rather a value on a continuous scale between 0 and 1. The membership function of a fuzzy set is used to determine the degree of membership of an element in the set.
Here is an example of a fuzzy set:
Let X be the set of all integers between 1 and 10. The fuzzy set A on X can be defined as follows:

- The membership value of the element 1 is 0.3
- The membership value of the element 2 is 0.5
- The membership value of the element 3 is 0.7
- The membership value of the element 4 is 0.9
- The membership value of the element 5 is 1.0
- The membership value of the element 6 is 0.9
- The membership value of the element 7 is 0.7
- The membership value of the element 8 is 0.5
- The membership value of the element 9 is 0.3
- The membership value of the element 10 is 0.1

The main differences between a crisp set and a fuzzy set are:

1. Membership values: In a crisp set, the membership of an element is either 0 or 1, while in a fuzzy set, the membership is a value on a continuous scale between 0 and 1.
2. Boundaries: Crisp sets have clear boundaries between the elements that belong to the set and those that do not, while fuzzy sets have fuzzy boundaries, with elements that partially belong to the set.

3. Precision: Crisp sets are precise, and there is no ambiguity in the membership of an element, while fuzzy sets are imprecise, and the degree of membership can be interpreted in different ways.

7(a) classification is a predictive modeling problem where the class label is anticipated for a specific example of input data. It is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data.

a)2)
—-
Define the problem: Same as before, define the binary classification problem with input features X and binary class labels y.

Encoding: Instead of using a binary encoding, we need to use a real-valued encoding scheme. Each potential solution (individual) in the GA population will now be represented as a real-valued chromosome, where each component represents the weight or importance assigned to a specific feature. For example, if there are five features, the chromosome could be a real-valued vector of length 5, with each component indicating the weight of the corresponding feature.

Initialization: Generate an initial population of individuals randomly or using a heuristic initialization method. Each individual represents a potential solution to the classification problem. For instance, we can randomly create a population of 50 individuals, each with a real-valued vector of length 5.

Fitness evaluation: Evaluate the fitness of each individual by applying the encoded feature weights to a classification algorithm. Train the classifier using the weighted features and measure its performance using an evaluation metric such as accuracy or F1-score. Higher fitness values indicate better classification performance.

Selection: Select individuals from the population based on their fitness values. The selection process remains the same as in the binary case.

Genetic operators:

a. Crossover: Perform crossover (recombination) between selected individuals to generate offspring. For real-valued encodings, a common crossover operator is the arithmetic crossover. In this operator, the corresponding components of two parents are combined using a weighted average to create new solutions.

b. Mutation: Introduce random changes to the genetic material (chromosomes) of selected individuals. For real-valued encodings, mutation can involve perturbing the values of the chromosome components. This can be done by adding a small random value or by applying a mutation operator specific to the problem domain.

Replace the population: Replace the current population with the new offspring generated through crossover and mutation.
f
Termination criteria: Determine the termination criteria for the GA, such as a maximum number of generations, reaching a desired fitness level, or a stagnation condition. If the termination criteria are not met, go back to step 4.

Extract the best solution: Once the GA terminates, extract the best individual (solution) from the final population. This corresponds to the optimal feature weights that achieved the highest fitness. In our example, the real-valued vector with the best fitness value represents the feature weights.

Apply the best solution: Use the extracted solution to classify new, unseen instances. Apply the same encoding and decoding process used in the GA to determine the class labels of the test instances. Use the weighted features and a classification algorithm to make predictions on new data.

—---

(b) Pros:
Mutation operator guarantees variation, hence every possible solution will be checked which guarantees that optimal solution will be reached, if sufficient number of generations are explored
Prevents early convergence at a suboptimal solution (local minima)

Cons:
The quality of a solution depends on parameters like mutation probability, number of generations. Hence, for less generations solution might not be optimal.

Also, mutation operator increase randomness, hence might not converge if the mutation rate is high.

Advantages:

Exploration of new solutions: The mutation operator introduces random changes to the genetic material, allowing the algorithm to explore new regions of the search space. This can be beneficial when the population is stuck in local optima and needs a random perturbation to escape.

Maintaining diversity: Mutation helps maintain diversity within the population by introducing new genetic material. This is particularly important in preventing premature convergence, where the algorithm gets stuck in a suboptimal solution.

Overcoming genetic drift: Genetic drift refers to the loss of genetic information over generations due to random sampling. Mutation helps counteract this effect by continuously introducing new genetic material, ensuring that valuable traits are not lost over time.

Disadvantages:

Premature convergence: Although mutation can help avoid premature convergence in some cases, it can also hinder convergence towards the global optimum. Excessive mutation rates may disrupt the convergence process and prevent the algorithm from reaching the optimal solution.

Search inefficiency: Random changes introduced by mutation can lead to inefficient search patterns. If the mutation rate is too high, it can result in a random walk through the search space, making it difficult to converge towards the optimal solution.

Loss of good solutions: Mutation is a random operator, and it can modify or eliminate already good solutions in the population. This can result in the loss of valuable genetic material and slow down the overall progress of the algorithm.

Tuning the mutation rate: Finding an optimal mutation rate is crucial in genetic algorithms. Setting the mutation rate too low may lead to slow exploration and limited diversity, while setting it too high can disrupt convergence and hinder the algorithm's performance.

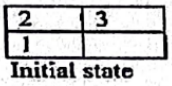
---

8. Answer either (a) or (b).

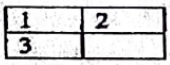
(a) Consider the 3-puzzle problem shown in Fig. below:

Possible operators (in order) are: up, down, left, right. Assume that repeated states are not detected.

Draw search tree using BFS. Would DFS find the goal? How many nodes would be generated if IDS is used starting with depth increment of one? 10



Initial state



Final state

(b) Consider the following facts:

Every child loves Santa.
Everyone who loves Santa loves any reindeer.
Rudolph is a reindeer, Rudolph has a red nose.
Anything which has a red nose is weird or is a clown.
No reindeer is a clown.
John does not love anything that is weird.

---Prove that "John is not a child" using resolution. 10

---

08 JULY Thursday 28th Wee

My Day Begins With

8|b) 1. Every child loves Santa.

$(\forall x)\,(\text{CHILD}(x) \rightarrow \text{LOVES}(x, \text{SANTA}))$

$\Rightarrow (\forall x)\,(\neg\text{CHILD}(x) \vee \text{LOVES}(x, \text{SANTA}))$

$\Rightarrow \{\neg\text{CHILD}(x_1), \text{LOVES}(x_1, \text{SANTA})\}$

2. Everyone who loves Santa loves any reindeer

$(\forall x)\,(\text{LOVES}(x, \text{SANTA}) \rightarrow (\forall y)(\text{REINDEER}(y) \wedge \text{LOVES}(x, y))$

$\Rightarrow (\forall x)(\forall y)(\neg\text{LOVES}(x, \text{SANTA}) \vee (\text{REINDEER}(y) \wedge \text{LOVES}(x, y)))$

$\Rightarrow (\forall x)(\forall y)((\neg\text{LOVES}(x, \text{SANTA}) \vee \text{REINDEER}(y)) \wedge (\neg\text{LOVES}(x, \text{SANTA}) \vee \text{LOVES}(x, y)))$

$\Rightarrow \{\neg\text{LOVES}(x_2, \text{SANTA}), \text{REINDEER}(y_1)\}, \{\neg\text{LOVES}(x_3, \text{SANTA}), \text{LOVES}(x_3, y_2)\}$

3. Rudolph is a reindeer, Rudolph has a red nose

$\Rightarrow \text{REINDEER}(\text{RUDOLPH}) \wedge \text{NOSE}(\text{RUDOLPH}, \text{RED})$

$\Rightarrow \{\text{REINDEER}(\text{RUDOLPH})\}, \{\text{NOSE}(\text{RUDOLPH}, \text{RED})\}$

4. Anything which has red nose is wierd or clown

$(\forall x)(\text{NOSE}(x, \text{RED}) \rightarrow (\text{WIERD}(x) \vee \text{CLOWN}(x)))$

$\Rightarrow (\forall x)(\neg\text{NOSE}(x, \text{RED}) \vee \text{WIERD}(x) \vee \text{CLOWN}(x))$

$\Rightarrow \{\neg\text{NOSE}(x_4, \text{RED}), \text{WIERD}(x_4), \text{CLOWN}(x_4)\}$

5. No reindeer is clown.

$(\forall x)(\text{REINDEER}(x) \rightarrow \neg\text{CLOWN}(x))$

$\Rightarrow (\forall x)(\neg\text{REINDEER}(x) \vee \neg\text{CLOWN}(x))$

$\Rightarrow \{\neg\text{REINDEER}(x_5), \neg\text{CLOWN}(x_5)\}$

6. John does not love anything that is wierd

$(\forall x)(\text{WIERD}(x) \rightarrow \neg\text{LOVE}(\text{JOHN}, x))$

$\Rightarrow (\forall x)(\neg\text{WIERD}(x) \vee \neg\text{LOVE}(\text{JOHN}, x))$

$\Rightarrow \{\neg\text{WIERD}(x_6), \neg\text{LOVE}(\text{JOHN}, x_6)\}$

5. JOHN is not a child

$\therefore \neg\text{CHILD}(\text{JOHN})$

7S = CHILD(JOHN)

M T W T

8.b.

2021
28th Week . 190-175

JULY
Friday

09

My Day Begins With

$\neg CHILD(x1) \vee LOVES(x1, SANTA)$ — C1

$\neg LOVES(x2, SANTA) \vee$ ~~LOVES~~ $REINDEER(y1)$ — C2

$\neg LOVES(x3, SANTA) \vee LOVES(x3, y2)$ — C3

$REINDEER(RUDOLPH)$ — C4

$NOSE(RUDOLPH, RED)$ — C5

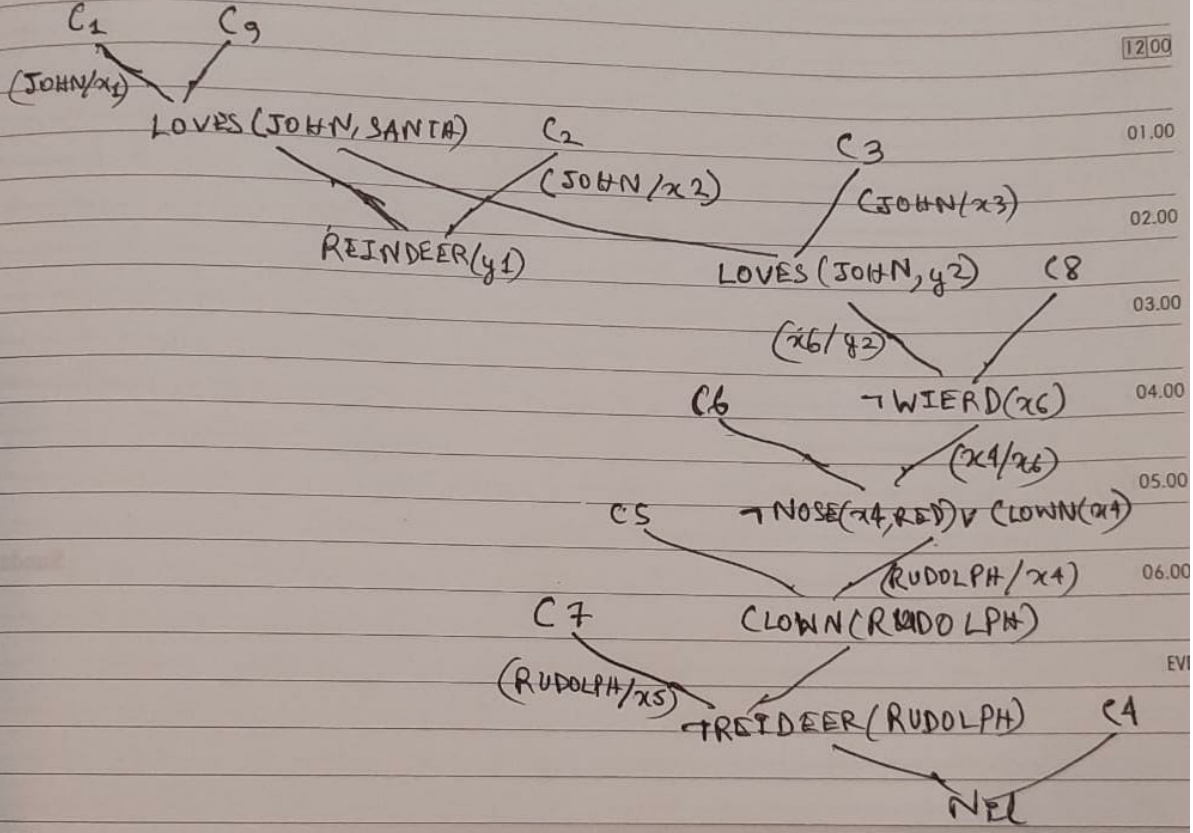
$\neg NOSE(x4, RED) \vee WIERD(x4) \vee CLOWN(x4)$ — C6

$CHILD(JOHN)$ — C9

$\neg REINDEER(x5) \vee \neg CLOWN(x5)$ — C7

$\neg WIERD(x6) \vee \neg LOVE(JOHN, x6)$ — C8

08.00
09.00
10.00
11.00
12.00
01.00
02.00
03.00
04.00
05.00
06.00



GUST

0	2	9	16	23
1	3	10	17	24
	4	11	18	25
	5	12	19	26
	6	13	20	27
	7	14	21	28
	8	15	22	29