INDRAPRASTHA INSTITUTE of INFORMATION TECHNOLOGY DELHI

CSE643 – Artificial Intelligence

Monsoon 2021 session

End-sem exam

Max marks: 50 (will be scaled down to 25 marks) **02-Dec-2021, 9AM to 12PM**

Submission deadline: 02-Dec-21 at 12:00 PM

INSTRUCTIONS:

You will have to create a PDF file with your answers, name the file as AI-EndSem-<Name>-<RollNo> and upload it on the classroom page by 12:00 pm. In the answer sheet write or type your name and roll number. In case you choose to have hand-written answers then those pages can be scanned and uploaded (make sure that it is clearly readable). Make sure it has the name and roll number on it.

Q1: **(6 marks)**

Represent the knowledge given below in FOPL and convert to CNF and then using resolution refutation (write resolution steps and draw the graph) determine whether Dhari is frustrated or not. (note: Sambar is a type of Indian deer).

Every lion chases some sambar. Every sambar that jumps and runs is smart. No lion catches any smart sambar. Any lion that chases some sambar but does not catch it is frustrated. If all sambars are smart then all lions are frustrated. Dhari is a lion. Tej is a sambar and jumps and runs.

Answer

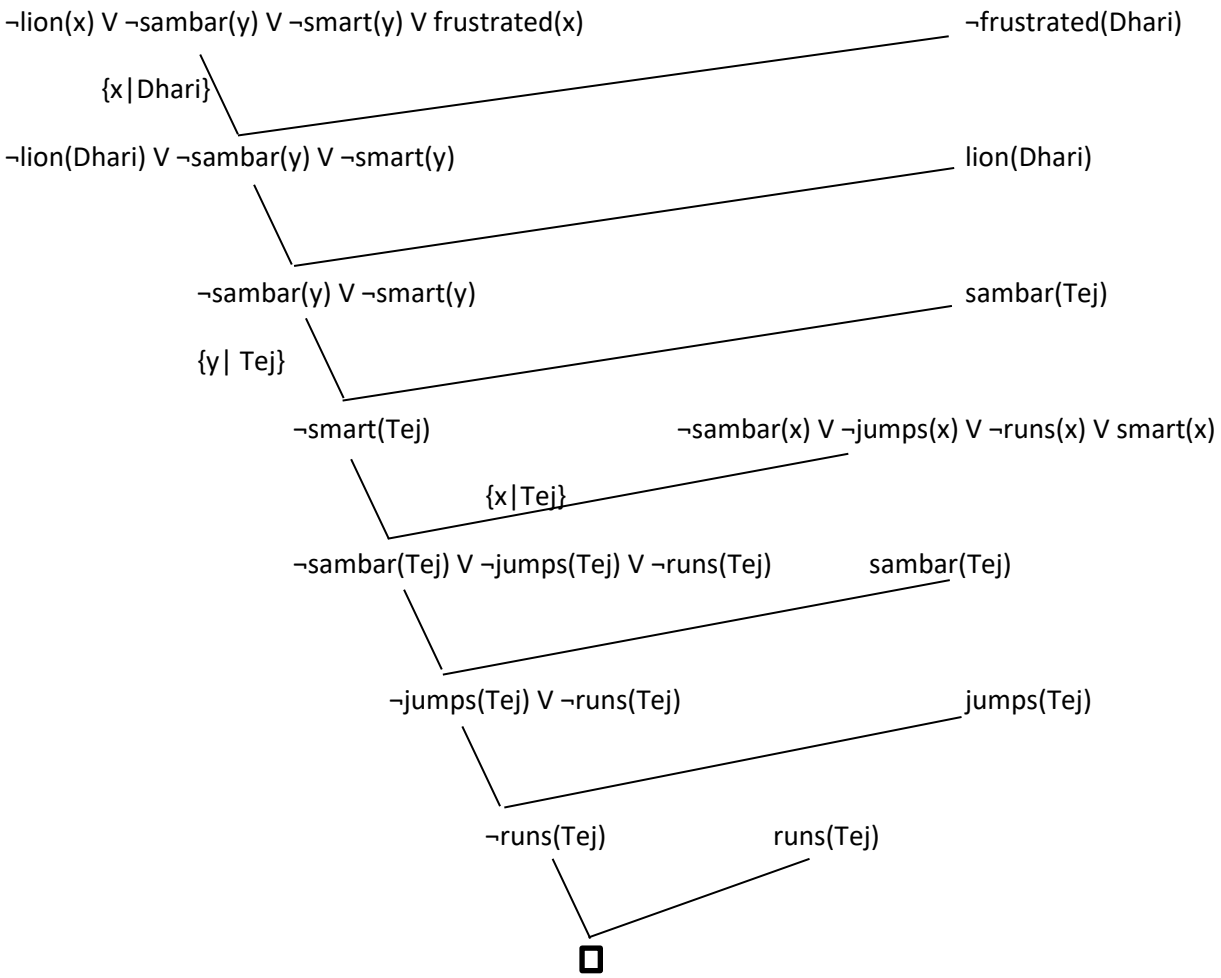
1. $\forall x \exists y$ (lion(x) $\wedge$ sambar(y) $\rightarrow$ chase(x,y))
2. $\forall x$ sambar(x) $\wedge$ jumps(x) $\wedge$ runs(x) $\rightarrow$ smart(x)
3. $\forall x \forall y$ (lion(x) $\wedge$ sambar(y) $\wedge$ smart(y) $\rightarrow$ $\neg$catch(x,y))
4. $\forall x$ (lion(x) $\wedge$ $\exists y$ (sambar(y) $\wedge$ chase(x,y) $\wedge$ $\neg$catch(x,y)) $\rightarrow$ frustrated(x))
5. $\forall x \forall y$ (lion(x) $\wedge$ sambar(y) $\wedge$ smart(y) $\rightarrow$ frustrated(x))
6. lion(Dhari)
7. sambar(Tej)
8. jumps(Tej)
9. runs(Tej)

In CNF

10. $\forall x$ (lion(x) $\wedge$ sambar(C) $\rightarrow$ chase(x,C)) – introducing skolem constant C for existential quantifier

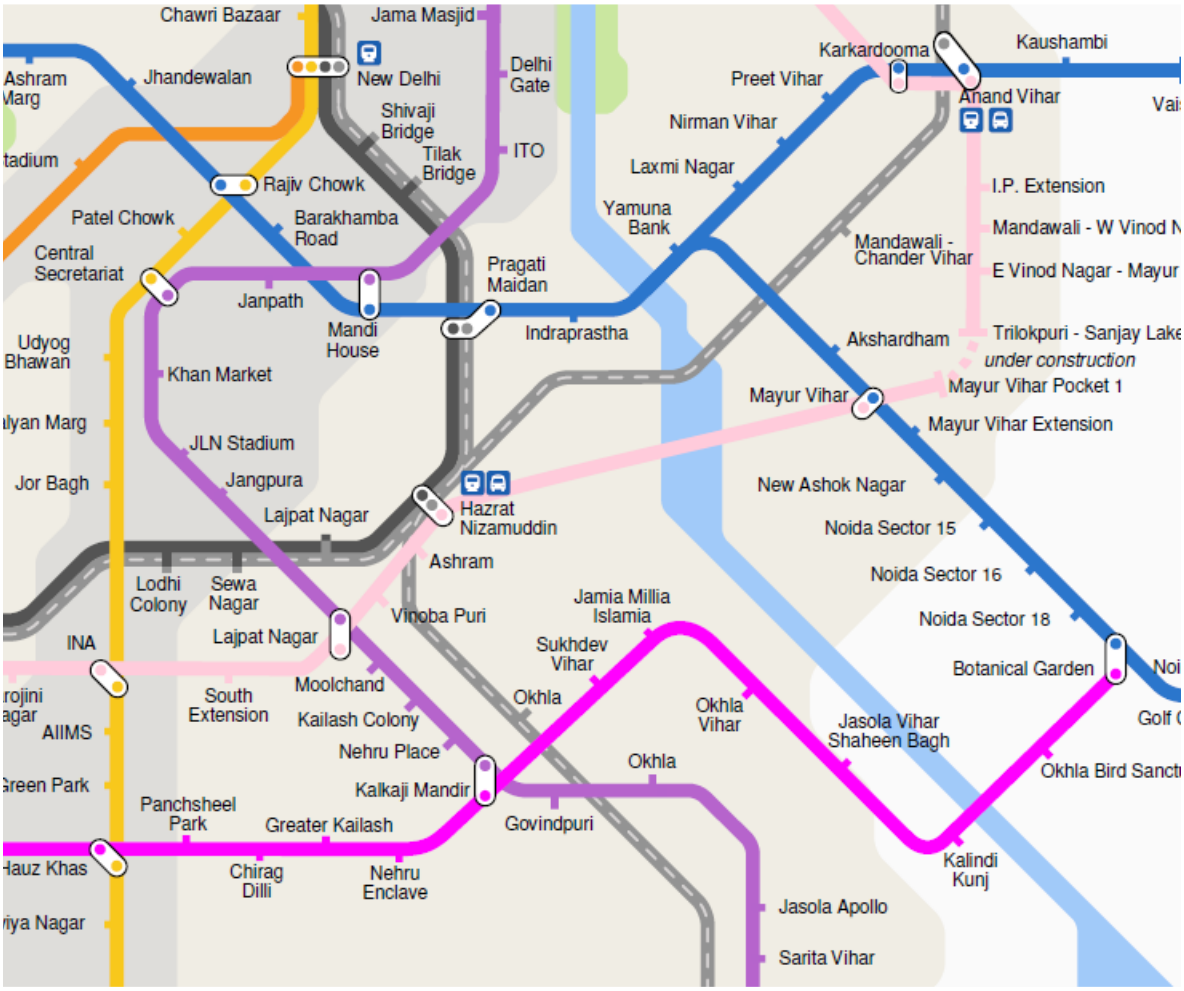
11. ¬lion(x) V ¬sambar(C) V chase(x,C) – (10) in clause form
12. ¬sambar(x) V ¬jumps(x) V ¬runs(x) V smart(x)
13. ¬lion(x) V ¬sambar(y) V ¬smart(y) V ¬catch(x,y)
14. ∀x (lion(x) Λ sambar(D) Λ chase(x,D) Λ ¬catch(x,D) → frustrated(x)) – introducing skolem constant
15. ¬lion(x) V ¬sambar(D) V ¬chase(x,D) V catch(x,D) V frustrated(x) – (14) in clause form
16. ¬lion(x) V ¬sambar(y) V ¬smart(y) V frustrated(x)
17. Hypothesis is frustrated(Dhari). Assume negation of hypothesis, that is ¬frustrated(Dhari). Add it to clauses.
18. From 16 and 17 we get ¬lion(Dhari) V ¬sambar(y) V ¬smart(y) with unification of {x|Dhari} and resolution
19. From 6 and 18 we get ¬sambar(y) V ¬smart(y) – from resolution
20. From 7 and 19 we get ¬smart(Tej) -- from unification {y| Tej} and resolution
21. From 12 and 20 we get ¬sambar(Tej) V ¬jumps(Tej) V ¬runs(Tej) -- from unification {x|Tej} and resolution
22. Using facts 7, 8, 9 and 21 we get a contradiction, thus the negation of the hypothesis is FALSE and so the hypothesis frustrated(Dhari) is TRUE.

Resolution Graph



Q2: **(6 marks)**

You are building a system to tell users how to travel between two locations on Delhi Metro as quickly and as cheaply as possible for the metro map given below. Assume that Delhi Metro has a flat cost of Rs 30/- for any metro train ride from starting station to ending station irrespective of where you get off. If you change to an interconnecting train then you have to pay Rs 30/- again. Formulate this as a A* search problem of going from any given station to another station. Assume some heuristic and show that it is admissible and consistent.



Answers

We define the cost g(n) of getting to a node n as g(n) = 30 * Metrosboarded + NumSt, where Metrosboarded is the number of metro trains boarded so far, and NumSt is number of stations from Start station to get to n.

We define our heuristic function h(n) as: h(n) = Stns(n) + 30*Int(n), where Stns(n) is the minimum number of stations between any given station 'n' and the destination (goal) station. And Int(n) is the

minimum number of interchange metro trains that need to be boarded to get to the destination (goal) station – if the given station and goal station are on the same line then Int(n) = 0 else it is the minimum number of trains to board to get to goal station.

A heuristic h(n) is admissible if, for every node n, it underestimates the cost to the goal, that is $h(n) \leq h^*(n)$; basically a lower bound on $h^*(n)$, where $h^*(n)$ is the cost of the optimal path to the goal. Since our heuristic is taking Stns(n) as the minimum number of stations between station 'n' and destination(goal) station, it will always be a lower bound.

A heuristic h(n) is consistent if, for every node n and every successor n' of n, the estimated cost of reaching the goal from n is no greater that the step cost of getting to n' plus the estimated cost of reaching the goal from n', i.e. $h(n) \leq c(n, n') + h(n')$. Since our heuristic is taking both Stns(n) and Int(n) as minimum number of stations and minimum number of interchange metro trains that need to be boarded between start and goal, the heuristic is also consistent.

Initial State: Start station

Actions: Calculate h(n) from Stns(n) and Int(n). Calculate f(n) = g(n) + h(n), where g(n) = 30 * Metrosboarded + NumSt, where Metrosboarded is the number of metro trains boarded so far, and NumSt is number of stations from Start station to get to n. Expand the minimum f(n) node.

State space: Stations travelled and Metro trains boarded.

Path: From Start to Destination station.

Goal test: Does CLOSED have Destination station or not. If yes Halt else expand node from OPEN and go back to Actions.

Q3: **(6 marks)**

You have to create a rule-based system for representing the following rules of IIITD on Academic load. Using the concepts you have learnt so far, represent this knowledge using rule-based reasoning and show where forward-chaining and backward-chaining can occur.

> In general, students will take courses as per the program. Normal load for the first and second year students is 16-20 credits and for the third and fourth year students is 16-22 credits.
>
> When a student registers for more credits than the stipulated normal load, it is called the overload. First year students cannot take any overload. Second year students having a CGPA of more than 8.0 can take an overload of up to 2 additional credits over 20 credits. Third and fourth year students having a CGPA of more than 8.0 can take an overload of up to 2 additional credits over 22 credits.

When the student registers for less than 16 credits in a semester, it is called the underload. A student may be allowed, with permission of the DOAA an underload of a maximum of 4 credits below 16 credits.

Students under academic warning may not be permitted any overload. Such students may be advised by DOAA to take an underload.

Students can avail underload below 12 credits (as low as 4 credits) only in the eighth semester of their B.Tech. program. Students wanting to avail internship (industrial, academic, research, etc.) without taking semester leave can do so under this clause provided they satisfy the below criteria:

Eligibility to apply for internship at the end of Semester 6:

(1) Must have completed 126 credits.
(2) Must have done 4 cr of SG and CW credits in addition to 126 credits.
(3) Must have completed all core courses.
(4) Must have been left with 4 cr of IP/IS/UR/Online courses/BTP.

Eligibility to go for internship at the end of Semester 7

(1) Must have completed 148 credits.
(2) Must have done 4 cr of SG and CW credits in addition to 148 credits.
(3) Must have completed all core courses and other graduation requirements pending a 4 cr IP/IS/UR/Online course/BTP.
(4) Must have been left with 4 cr of IP/IS/UR/Online courses/BTP.

Answers

Rule 1: IF (student_year is first or second year) AND (student_credit is >= 16 and <=20 credits) THEN student_load(normal).

Rule 2: IF (student_year is third or fourth year) AND (student_credit is >= 16 and <=22 credits) THEN student_load(normal).

Rule 3: IF student_year is first year THEN overload_allowed(no).

Rule 4: IF student_year is second year AND student_CGPA > 8.0 AND student_credit = 20 credits AND student_wants_more_courses THEN overload_allowed(yes) AND student_credit_max = 22 credits

Rule 5: IF (student_year is third year or fourth year) AND student_CGPA > 8.0 AND student_credit = 22 credits AND student_wants_more_courses THEN overload_allowed(yes) AND student_credits_max = 24 credits

Rule 6: IF student_credit < 16 THEN student_load(underload).

Rule 7: IF student_load(underload) AND student_has_DOAA_permission THEN student_credit_min = 12.

Rule 8: IF student_under_academic_warning THEN overload_allowed(no).

Rule 9: IF student_under_academic_warning and student_has_DOAA_permission THEN underload_allowed(yes).

Rule 10: IF student in BTech 8^{th} semester AND student wants to do less courses then student_credits_min = 4.

Rule 11: IF student wants to avail internship AND student does not take semester leave AND student at end of 6^{th} semester AND student credits >= 126 AND student done 4 cr of SG and CW AND core courses is subset of student completed courses AND student left with 4 cr of IP/IS/UR/Online courses/BTP THEN student eligible to apply for internship.

Rule 12: IF student wants to avail internship AND student does not take semester leave AND student at end of 7^{th} semester AND student credits >= 148 AND student done 4 cr of SG and CW AND student completed courses is all core courses AND student has completed all graduation requirements AND student left with 4 cr of IP/IS/UR/Online courses/BTP THEN student eligible to apply for internship.

Forward-chaining occurs in Rule 1, Rule 2, Rule 3, and Rule 8.

Backward-chaining occurs in rest of the rules.

Q4: **(4 marks)**

a) Suppose we have a dataset of n examples and in a Univariate regression we hypothesize the learning function as $h_w(x) = \ln(w^2 x_i)$. Assuming that the true output is represented by y_i, derive the gradient descent update for w. (Note: d/dx of ln(x) = 1/x)

b) For the Boolean function given below, is it possible to have a single unit neural network? If so, what will it be? If not, then what is the reason?

X_1	X_2	Y
1	1	0
0	0	0
1	0	1
0	1	0

Answers

a) The L2 loss function is defined as: Loss(h_w) = $\sum_{i=1}^{n}\left(y_i - h_w(x_i)\right)^2$. Thus: Loss(h_w) = $\sum_{i=1}^{n}\left(y_i - ln(w^2 x_i)\right)^2$. The gradient descent of the of the weight w would be:

w = w - $\alpha \frac{\delta Loss(h_w)}{\delta w}$. That is w = w - $\alpha \frac{\delta}{\delta w}(\sum_{i=1}^{n}\left(y_i - ln(w^2 x_i)\right)^2)$ where α is the learning rate.

Using the summation rule of differentiation,

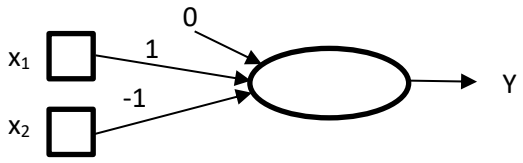
$\frac{\delta}{\delta w}\left(\sum_{i=1}^{n}\left(y_i - ln(w^2 x_i)\right)^2\right) \equiv \sum_{i=1}^{n}\frac{\delta}{\delta w}\left(y_i - ln(w^2 x_i)\right)^2)$

Using the chain rule of differentiation,

$\equiv \sum_{i=1}^{n} 2 * \left(y_i - ln(w^2 x_i)\right) * \frac{-2wx_i}{w^2 x_i} \equiv \sum_{i=1}^{n} 2 * \left(y_i - ln(w^2 x_i)\right) * \frac{-2}{w} \equiv \sum_{i=1}^{n}\frac{-4}{w} * \left(y_i - ln(w^2 x_i)\right)$

Thus, the gradient descent update for w is w ← w - $\alpha(\sum_{i=1}^{n}\frac{-4}{w} * \left(y_i - ln(w^2 x_i)\right))$ where α is the step size.

b) Yes, it is possible to have a single unit neural network for this. It will be as follows where $h_w(x)$ is hypothesized as ($w_1x_1 + w_2x_2 > 0$) with the weights given below:



Q5: **(4 marks)**

Describe the objective of Inductive logic programming with dataset characteristic, goal of ILP and stopping criterion for the induction process. Use an example to elucidate. Answer based on what has been taught in the course.

Answer

Objective of Inductive Logic Programming is to learn a generalized hypothesis from given background knowledge and a set of examples, in such a way that the generalized hypothesis entails all the positive examples and none of the negative examples.

Dataset characteristic

1. B is the background knowledge that is provided and consists of logical clauses (FOPL). These are the seed facts that are provided for the learning process.

2. E^+ is the set of positive examples from which the hypothesis should learn to generalize.
3. E^- is the set of negative examples which the hypothesis should learn not to cover in its generalization.
4. $B \wedge E^- \not\models \square$ E^- are the negative examples and these examples should not derive a contradiction with the background knowledge.
5. $B \not\models E^+$ Background knowledge should not already subsume the positive examples

Goal of ILP

1. On the basis of B , E^+ and E^- learn a generalized hypothesis $H = \{C_1 \vee C_2 \vee \ldots \vee C_n\}$ to classify positive and negative examples correctly in such a way that it generalizes and works correctly on the test data.
2. Start with empty H.
3. Take the most constrained positive example in E^+ and create clause C_x with a new literal instead of a constant in that example, such that C_x covers that positive example with variables in place of constants.
4. Test if C_x covers a negative example, if yes drop and go to Step 3 until all positive examples are covered.
5. If C_x does not cover a negative example, add to H. Go to Step 3 until all positive examples are covered.

Stopping criterion

1. $B \wedge H \models E^+$ E^+ are the positive examples and the hypothesis H learnt together with the background knowledge B should logically imply all positive examples.
2. $B \wedge H \wedge E^- \not\models \square$ E^- are the negative examples and the hypothesis H learnt together with the background knowledge B should not derive a contradiction.

Example

Given B: father(Ram, Vinay), father(Vinay, Shyam), father(Ram, Rajiv), father(Rahul, Hari)

And E^+ : grandfather(Ram, Shyam)

And E^- : grandfather(Ram, Hari)

H is learnt as: grandfather(X,Y) :- father(X,Z), father(Z, Y).

Q6: **(4 marks)**

Describe at a high level how Meta-learning is implemented and the three types of Meta-learning techniques that can used. Answer based on what has been taught in the course.

Answer

Meta-learning is learning to learn a function given input and output datasets. Basically, it is a process of improving a learning algorithm over multiple learning episodes.

A conventional ML improves model predictions over multiple data instances. During base learning, a learning algorithm learns the task to classify data instances given in a dataset with respect to an objective. That is given $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ it learns a function f_w to predict y_i given x_i, that is we try to *min loss(D | f_w)*.

During meta-learning, we are given many such D_i datasets and the function f_{wi} it has learnt together with the weights w_i. Meta-learning tries to learn from a training set $(D_i, f_{wi} \mid w_i)$ to find optimal weights such that $loss(D_{test} \mid w_j)$ is minimal.

Types of Meta-learning

1. Metric-based meta learning: In this type of meta-learning we try to minimize the distance between two dataset samples. A Siamese network is used for this learning to detect the similarity between two datasets. Based on the similarity the weights can be learnt.
2. Optimization-based meta learning: We try to find the best possible weights by choosing n examples at random, do some learning using D_{train} to update w, and then compute loss on D_{test} and then change the initial weights w_0.
3. Model-based meta learning: We try to model it as a sequence of dataset points, $\{D_1,\ldots,D_n\}$ and its corresponding weights, we try to learn weights for a new dataset point D_{n+1}. A Recurrent Neural Net can be used for this kind of meta learning.

Q7: **(5 marks)**

i) What is reinforcement learning? What is learnt in reinforcement learning? Why is it required?
ii) What is Trustworthiness in AI systems? Why is it required?

Answers

i)
   a. Reinforcement learning is learning a model of the appropriate behaviour towards a goal based on rewards for actions to achieve that goal through trial-and-error interactions with the environment.
   b. The optimal value function is learnt in reinforcement learning based on the rewards.
   c. Reinforcement learning is required when we do not have a model or data about the environment and the only way to learn is to interact with the environment.

ii)
   a. Trustworthiness in an AI system is the measurable belief that we can have in an AI system A, such that A will perform given task S dependably for a specified period, P, within a trust context T.

b. Trustworthiness is required for us to determine how much we can rely and be confident that the AI system would perform as per our expectations and norms.

Q8: **(5 marks)**

An intelligent robot in a car manufacturing firm needs to devise a plan to achieve the goal of having to fix the wheel assembly for cars such as axles, disc brakes, wheels, fastener bolts and wheel caps, starting from an initial, empty state. The disc brakes can be fixed only after the axles are fixed, the wheels can be fixed only after the brakes have been fixed, the fastener bolts can be fixed only after the wheels are fixed, and the wheel cap can be fixed only after the bolts. There is difference between front axle and back axle, similarly there is difference between front brakes and back brakes.

i) Devise a plan using Action, Precondition, Add and Delete effects. Assume that axle, disc brakes, wheels, fastener bolts, wheel caps are available.

ii) Draw the plan graph.

Answers

Action: Put-Front-Axle

Precondition: nothing-on-front-axle ^ available(Front-Axle) ^ ¬car-has(Front-Axle)

Add: car-has(Front-Axle)

Delete: available(Front-Axle) ^ ¬car-has(Front-Axle)

Action: Put-Front-Brakes

Precondition: car-has(Front-Axle) ^ available(Front-Disc-Brakes) ^ ¬car-has(Front-Disc-Brakes)

Add: car-has(Front-Disc-Brakes)

Delete: available(Front-Disc-Brakes) ^ ¬car-has(Front-Disc-Brakes)

Action: Put-Back-Axle

Precondition: nothing-on-back-axle ^ available(Back-Axle) ^ ¬car-has(Back-Axle)

Add: car-has(Back-Axle)

Delete: available(Back-Axle) ^ ¬car-has(Back-Axle)

Action: Put-Back-Brakes

Precondition: car-has(Back-Axle) ^ available(Back-Disc-Brakes) ^ ¬car-has(Back-Disc-Brakes)

Add: car-has(Back-Disc-Brakes)

Delete: available(Back-Disc-Brakes) ^ ¬car-has(Back-Disc-Brakes)

Action: Put-Wheels-On(Front)

Precondition: car-has(Front-Disc-Brakes) ^ ¬car-has(Front-wheels)

Add: car-has(Front-wheels)

Delete: ¬car-has(Front-wheels)

Action: Put-Wheels-On(Back)

Precondition: car-has(Back-Disc-Brakes) ^ ¬car-has(Back-wheels)

Add: car-has(Back-wheels)

Delete: ¬car-has(Back-wheels)

Action: Put-Fastener-Bolts(Front)

Precondition: car-has(Front-wheels) ^ ¬car-has(Front-fastener-bolts)

Add: car-has(Front-fastener-bolts)

Delete: ¬car-has(Front-fastener-bolts)

Action: Put-Fastener-Bolts(Back)

Precondition: car-has(Back-wheels) ^ ¬car-has(Back-fastener-bolts)

Add: car-has(Back-fastener-bolts)

Delete: ¬car-has(Back-fastener-bolts)

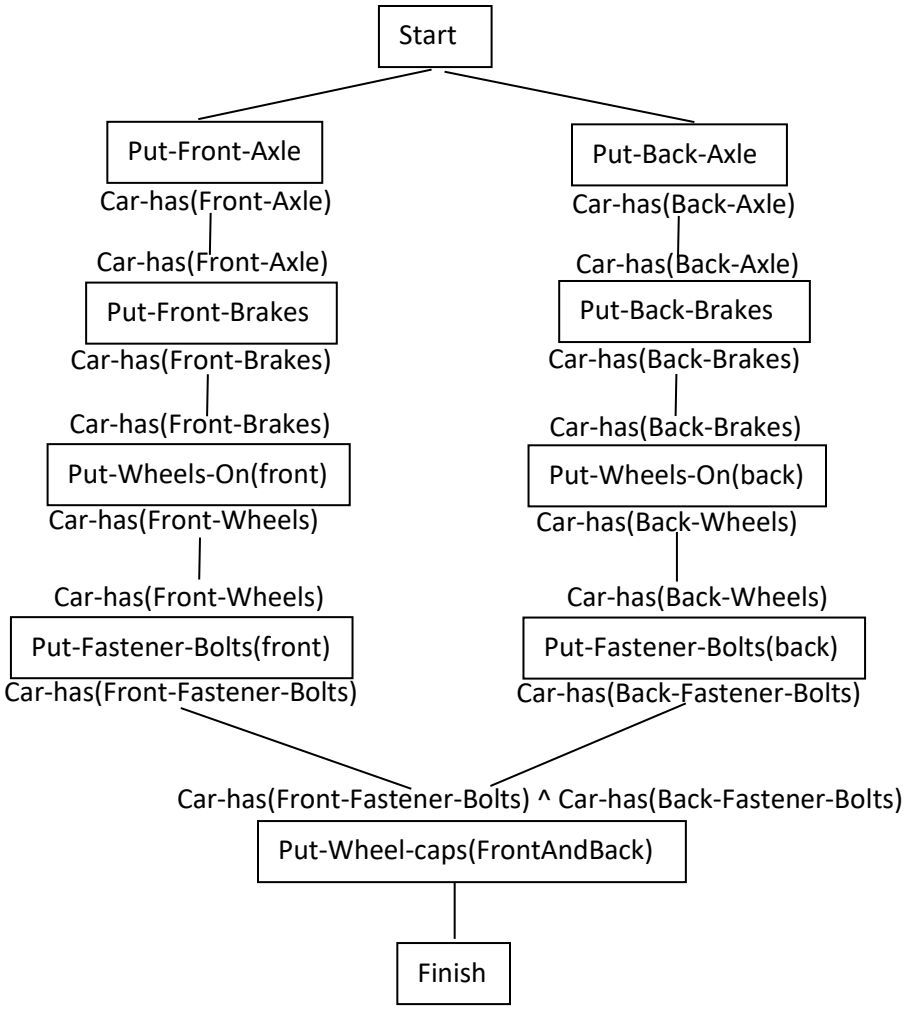
Action: Put-Wheel-Caps(FrontAndBack)

Precondition: car-has(Front-fastener-bolts) ^ car-has(Back-fastener-bolts) ^ ¬car-has(wheel-caps)

Add: car-has(wheel-caps)

Delete: ¬car-has(wheel-caps)

Plan Graph

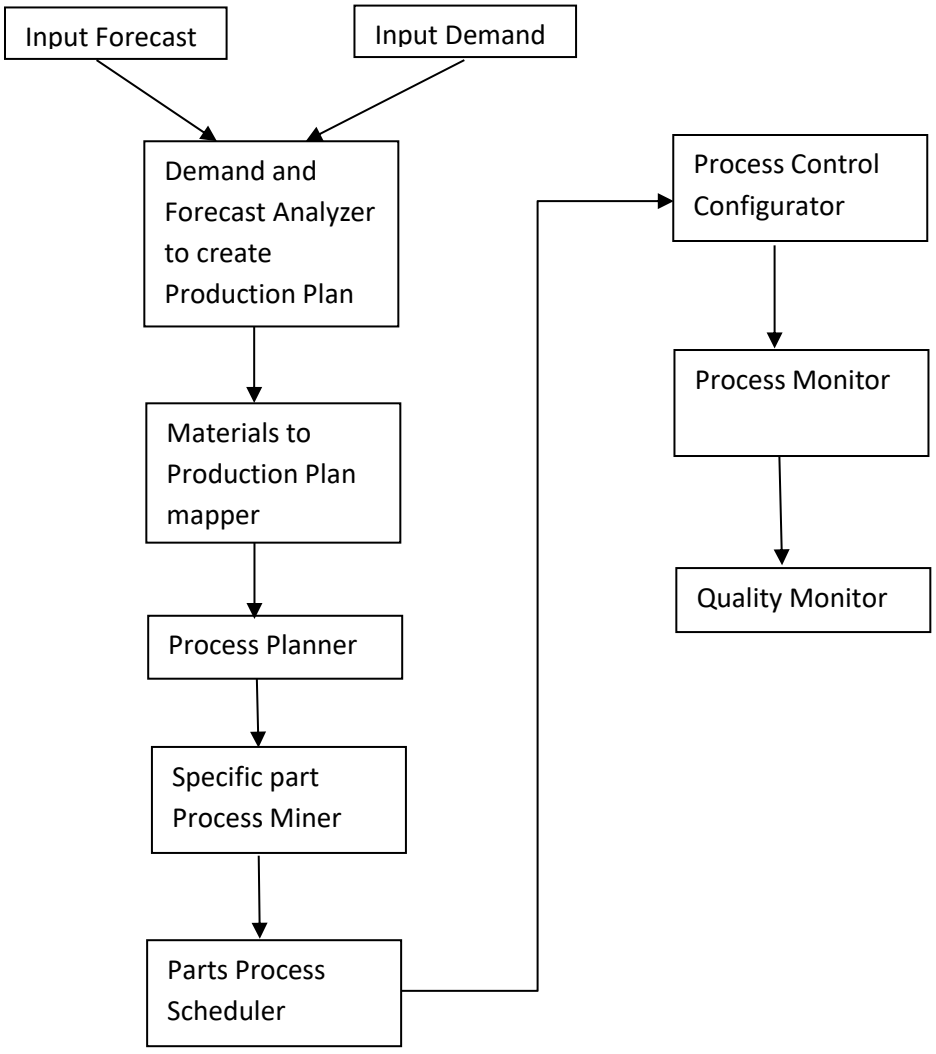


Q9: **(10 marks)**

Sundaram Fasteners Limited manufactures various parts, like rotors, gears, shock absorber components, bushes, chassis fasteners, engine fasteners, transmission shafts, bevel gear and pinion, crankshaft sprocket, turbine shafts etc. With same raw material one or more parts can be manufactured depending on the demand and forecast. Suppose you have been asked to design an Intelligent Production scheduling and monitoring system for Sundaram Fasteners Limited, explain where all will you use Search, Logic, Knowledge representation (explicit and implicit), Reasoning (explicit and implicit), Planning, and Machine Learning in such a system? Draw an AI architecture of the system and explain each module with one example.

Answers

Vehicle Parts: { Rotors, Gears, Shock absorber components, Bushes, Chassis fasteners, Engine fasteners, Transmission shafts, Bevel gear and pinion, Crankshaft sprocket, Turbine shafts }. We create the architecture given below.



Description of each Module

1. Demand and Forecast Analyzer to create Production Plan
   This module analyzes Input Demand and Input Forecast to create a Production Plan. Since these inputs are based on various market forces, the demand and forecast will be estimates with some probability / score. From such estimates we try to construct the production plan of the various parts. Each part can also vary based on size, shape and type that is required. It may so happen that some parts are always manufactured together, for example, gears and bushes may need to be manufactured together. Since it will be good to use prior history of demand-forecast

to production plan, we use *machine learning* models to find the best possible production plan that can possibly meet the demand/ forecast.

2. Materials to Production Plan mapper
   Once the Production Plan is determined, we use Search and Logic with explicit rules to decide which materials are required for each part in the Production Plan. For example, we may have rules to produce gears as follows:
   if stress on gear high and difficult to produce gear then material-required = cast_steel
   if high toughness and high strength required then material-required = plain_carbon_steel
   if high tooth strength and low tooth wear required then material-required = alloy_steel
   etc.
   This helps plan the production process by determining if the materials are available.

3. Process Planner
   This module creates the plan to produce the parts based on the previous module. It uses Planning algorithms to check for Preconditions, Actions, and Post-conditions, the inter-relationships to achieve the Production Plan with Material plan. For example, it may create a plan as START → Precondition: Gear stress determined ^ Gear materials available ; Action: Add Part to Manufacture Order; Delete: Gear materials available by consumed amount → FINISH

4. Specific part Process Miner
   This module will extract out all Process specific aspects for a particular part from various history recorded, since there may be changes / similarities in conditions and situations. It will use Case Based reasoning and find similar cases. For example: CBR may be used to decide that 90-grade aluminum can be used instead of 98-grade aluminum since some material is out-of-stock.

5. Parts Process Scheduler
   This part will schedule the parts production based on explicit rules. For example, If Rotors to be produced then Produce Rotor-bars and Shorting-rings first and then Produce Rotor-core and Rotor-winding.

6. Process Control Configurator
   This part will set the process control parameters based on machine learning. For example, for Rotor-shafts it will predict that the pressure needs to be 90psi with probability 0.9 etc.

7. Process Monitor
   This module will be based on explicit and implicit reasoning. There will be parameters that decide if the manufacturing process is going as per plan. Thus, there will be explicit reasoning to determine if there are any major violations, while implicit reasoning will determine if any possible cause of concern may arise. For example, if the rotor is being manufactured using materials that could possibly heat-up then implicit reasoning may derive that heat-treatment needs to be used to achieve hardness and overall strength.

8. Quality Monitor

   This module will determine the quality of the part manufactured and will be based on the image capture of the part that is produced as well as its test report. Thus, the image processing submodule will be based on deep learning to identify defective parts from defect identification images, and the test report will be processed through an explicit rule-based system to check if the test of the part falls within the acceptable range.