Assignment 1

Sarthak Sareen
30761182

Question-1: Problem Solving as Search

In this A* Algorithm has been executed which gives the pattern of the moves that ROBBIE should perform to go from the start node to the goal node,this also calculates the costs of the movement of ROBBIE position after each movement.This algorithm has been implemented made use of a tree. After calculating the costs it also saves the children of the nodes,parentnode operator of the node (which direction to travel from start node to current node),identifier of the node. While Calculating the costs of the movement of the ROBBIE it also calculates some heuristic function to calculate the best possible movement at a particular instance.

Algorithm Used for the Execution

Step 1

Taking out the index of the start node from the map and making as a Node ,calculating its backward cost(g) , the forward cost(h) and also calculating the f=g+h After this it is placed inside the openlist

Step 2

Checking the openlist if it is empty or not. If the open list is empty it will show No path found. and it will stop.

Step 3

Calculating the minimum value from the open list of (f=g+.h) and pooping out from the open list and inserting into the closed list. This node with the minimum f value is assigned as the current node.

Step 4

If the current node is the goal node then calculate the path from the start node to the goal node and stop.

Step 5

If the current node is not the goal node, find the neighbours of the current node while deleting all the neighbours which are present outside the map boundaries. Also deleting the mountain nodes and their adjacent nodes from the valid neighbours.Also deleting the nodes which are already present in the Open and Closed list.

Step 6
Making nodes from the neighbour list and adding the nodes into the open list while calculating its backward cost(g) ,forward cost (h) ,calculating its total cost(f) and saving the node operator also.

Step 7
Return to step 2

<u>Heuristic Function Used:</u>
This function h(x) heuristic function used for this algorithm is called the Euclidean distance. This distance gives the best possible distance from the current node to the goal node.This distance is calculated from the current node and the goal node. Formula used - ((y2-y1)^2 + (x2-x1)^2) where x1 and y 1 are the coordinates of the current node and x2 and y2 are the coordinates of the goal node. x1 ,y1 and x2,y2 are the coordinates of the current node and the goal node used from the map data.
This is admissible because every time the distance is calculated it does not overestimate the distance from the current node and the goal node.
This is proofed by the situation when the current node and the goal node are the same , this function will give the g value as 0 so it does not overestimate the distance.

The heuristic function is monotonic because when we calculate the f value that is the total cost from the start node to the current node, as we move along the path the f value never decreases.

Hence this function proves that it is monotonic and it does not overestimate the distance therefore is the best function to find the F value.

Tie breaking :
In the situation while calculating the minimum value from the list, if the f value comes the same for the two nodes then it will take out the minimum value from the nodes considering its g value.
In the situation when f value,g value and the h value all are the same then it will take minimum according to the insertion of the node(first insert will be taken as minimum first).

Question 2:
First order logic, representation

a) $\forall x(Male(x) \rightarrow \neg(Butcher(x) \wedge Vegetarian(x))$

b) $\forall x \forall y(Male(x) \wedge \neg Butcher(x) \wedge Vegetarian(y) \rightarrow Like(x,y))$

c) $\forall x(Vegetarian(x) \wedge Butcher(x) \rightarrow Female(x))$

d) $\forall x \forall y(Male(x) \wedge Female(y) \wedge Vegetarian(y) \rightarrow \neg(Like(x,y))$

Question 3: Unification

a) P(x, f(x), A, A) and P(y, f(A), y)

The above given expression cannot be unified as both the terms consist of different numbers of expressions. The first expression consists of 4 numbers of terms and the second expression consists of 3 numbers of terms.

Hence because of the unequal number of terms it cannot be unified.

b) P(x, f(x,y), g(x,w)) and P(A, f(w,B), g(w,x))

As both the expressions consist of the same number of terms so it can be unified.

To solve it we will substituting {x / A} -

Therefore we will get-

P(A, f(A,y), g(A,w)) and P(A, f(w,B), g(w,A))

Now substituting {w / A}

We will get this

P(A, f(A,y), g(A,A)) and P(A, f(A,B), g(A,A))

Now substituting {y / B}

We will get this

P(A, f(A,B), g(A,A)) and P(A, f(A,B), g(A,A))

As we compare both the expressions are now unified with the help of substituting the values.

Question 4: Resolution refutation

a) 1- ∀x1 (HIGH-GRADES(x1) → SUCCESSFUL(x1))
2- ∀x2(BRIGHT(x2) ^ WORK-HARD(x2) → HIGH-GRADES(x2))
3- ∀x3(¬BRIGHT(x3) → ¬PASS(x3))
4- ∀x4(¬WORK-HARD(x4) →HAS-FUN(x4))
5- ¬HAS-FUN(JAMES)
6- PASS(JAMES)

b)
As we know that whenever we have this **(**A -> B) we will convert this into (¬A ∨ B) with the help of implication elimination method.

1) ¬ HIGH-GRADES(x1) ∨ SUCCESSFUL(x1)
2) ¬ BRIGHT(x2) ∨ ¬ WORK-HARD(x2) ∨ HIGH-GRADES(x2)
3) BRIGHT(x3) ∨ ¬PASS(x3)
4) - WORK-HARD(x4) ∨ HAS-FUN(x4)
5) - ¬HAS-FUN(JAMES)
6) - PASS(JAMES)

c)
GOAL: SUCCESSFUL(JAMES)
NEGATED GOAL: 7. ¬ SUCCESSFUL(JAMES)
7 and 1: 7: ¬ SUCCESSFUL(JAMES) 1: ¬ HIGH-GRADES(x1) ∨ SUCCESSFUL(x1)
mgu:{x1|JAMES}
resolvent:8. ¬HIGH-GRADES(JAMES)
8 and 2: 8: ¬HIGH-GRADES(JAMES) 2: ¬ BRIGHT(x2) ∨ ¬ WORK-HARD(x2) ∨ HIGHGRADES(x2)

mgu:{x2|JAMES}

resolvent:9. ¬ BRIGHT(JAMES) ∨ ¬ WORK-HARD(JAMES)
9 and 3: 9. ¬ BRIGHT(JAMES) ∨ ¬ WORK-HARD(JAMES) 3. BRIGHT(x3) ∨ ¬PASS(x3)

mgu:{x3|JAMES}

resolvent:9. ¬ WORK-HARD(JAMES) 3. ¬PASS(JAMES)

9 and 3 and 4: 9. ¬ WORK-HARD(JAMES)   3. ¬PASS(JAMES) 4.WORK-HARD(x4) ∨ HASFUN(x4)
mgu:{x4|JAMES}
resolvent:3. ¬PASS(JAMES)   4.HAS-FUN(JAMES)
3 and 4 and 5: 3. ¬PASS(JAMES)   4.HAS-FUN(JAMES) 5. ¬HAS-FUN(JAMES)
resolvent:3. ¬PASS(JAMES)
3 and 6: 3. ¬PASS(JAMES)   6.PASS(JAMES)
resolvent: NIL
Hence Proved James is a successful student

Question 5: Resolution refutation

a)

1) -∀x1(BOY(x1) ∨ GIRL(x1) → CHILD(x1))
2) -∀x2 (CHILD(x2) → GET-DOLL(x2) ∨ GET-TRAIN(x2) ∨ GET-COAL(x2))
3) -∀x3 (BOY(x3) → ¬ GET-DOLL(x3))

4-∀x4 (CHILD(x4) ^ GOOD(x4) → ¬GET-COAL(x4))

b)

1) -(¬BOY(x1) ^ ¬GIRL(x1) ∨ CHILD(x1))

Can be split as :
1.1: ¬ BOY (x1) ∨ CHILD(x1)
1.2: ¬ GIRL (x1) ∨ CHILD (x1)

2) -(¬CHILD(x2) ∨ GET-DOLL(x2) ∨ GET-TRAIN(x2) ∨ GET-COAL(x2))
3) -(¬BOY(x3) ∨ ¬ GET-DOLL(x3))
4) -(¬CHILD(x4) ∨ ¬ GOOD(x4) ∨ ¬GET-COAL(x4))

c)

GOAL: ∀g ((CHILD (y) => ¬ GET-TRAIN (y)) => (BOY (y) => ¬ GOOD (y))
Converting this to clauses
(¬ CHILD (y) ∨ ¬ GET-TRAIN (y)) => (¬ BOY (y) ∨ ¬ GOOD (y))
¬ (¬ CHILD (y) ∨ ¬ GET-TRAIN (y)) ∨ (¬ BOY (y) ∨ ¬ GOOD (y))
(CHILD (y) ∧ GET-TRAIN (y)) ∨ (¬ BOY (g) ∨ ¬ GOOD (y))
Negated Goal:
¬ [(CHILD (y) ∧ GET-TRAIN (y)) ∨ (¬ BOY (y) ∨ ¬ GOOD (y))]
¬ CHILD (y) ∨ ¬ GET-TRAIN (y) ∧ BOY (y) ∧ GOOD (y)
We can split these into 3 as
5.1 {¬ CHILD (y) ∨ ¬ GET-TRAIN (y) }
5.2 { BOY (y) }

5.3 { GOOD(y) }
From 4 and 5.3 we get
(¬CHILD(x4) ⋁ ¬ GOOD(x4) ⋁ ¬GET-COAL(x4)) GOOD(y)
mgu:{x4|y}

6.resolvent: ¬ CHILD (y) ⋁ ¬ GETCOAL (y)
From 6 and 1.1 we get,
¬ CHILD (y) ⋁ ¬ GETCOAL (y) ¬ BOY (x1) ⋁ CHILD(x1)
mgu:{x1|y}
7. resolvent¬ BOY (y) ⋁ ¬ GETCOAL (y)

From 7 and 5.2 we get,
¬ BOY (y) ⋁ ¬ GETCOAL (y) BOY (y)
8. resolvent ¬ GETCOAL (y)

From 5.1 and 1.1 we get,
{¬ CHILD (y) ⋁ ¬ GET-TRAIN (y) } ¬ BOY (x1) ⋁ CHILD(x1)
mgu{x1|y}
9. resolvent ¬ BOY (y) ⋁ ¬ GET-TRAIN (y)

From 9 and 5.2 we get,
¬ BOY (y) ⋁ ¬ GET-TRAIN (y) BOY (y)
10. resolvent ¬ GET-TRAIN (y)
From 3 and 5.2 we get,
(¬BOY(x3) ⋁ ¬ GET-DOLL(x3)) BOY (y)
mgu{x3|y}
11. resolvent ¬ GET-DOLL (y)
From 2 and 8 we get
(¬CHILD(x2) ⋁ GET-DOLL(x2) ⋁ GET-TRAIN(x2) ⋁ GET-COAL(x2)) ¬ GETCOAL(y)
mgu{x2|y}

12. resolvent ¬ CHILD (y) ⋁ GET-DOLL (y) ⋁ GET-TRAIN (y)
From 12 and 10 we get,
¬ CHILD (y) ⋁ GET-DOLL (y) ⋁ GET-TRAIN (y) ¬ GET-TRAIN (y)
13 resolvent ¬ CHILD (y) ⋁ GET-DOLL (y)
From 13 an 11 we get,
¬ CHILD (y) ⋁ GET-DOLL (y) ¬ GET-DOLL (y)
14. resolvent ¬ CHILD (y)

From 14. and 1.1 we get,
¬ CHILD (y) ¬ BOY (x1) ∨ CHILD(x1)
mgu{x1|y}
15. resolvent ¬ BOY (y)

15 and 5.2 we get,
BOY (y) ¬ BOY (y)
Nil (empty)
Hence Proved