AI ASSIGNMENT — Knowledge Representation, Reasoning and Planning

Saumil Lakra, 2021097

THEORY

1 G: green light

Y: yellow light

R: red light

Condition 1:

$$(G \vee Y \vee R) \wedge \neg(G \wedge Y) \wedge \neg(G \wedge R) \wedge \neg(Y \wedge R)$$

Condition 2:

S_i : traffic light state ~~at~~ initially

S_{i+1} : traffic light state ~~off~~ after 1 cycle

$$\cancel{G_{S_i}} \quad G_{S_i} \wedge \neg G_{S_{i+1}} \longrightarrow Y_{S_{i+1}} \quad \text{— a}$$

at state s_i green light is there. At state s_{i+1}, green light goes off & Y is switched on.

Similarly for rest of the lights.

$$Y_{S_i} \wedge \neg Y_{S_{i+1}} \longrightarrow R_{S_{i+1}} \quad \text{— b}$$

$$R_{S_i} \wedge \neg R_{S_{i+1}} \longrightarrow G_{S_{i+1}} \quad \text{— c}$$

$$\Rightarrow a \wedge b \wedge c$$

Condition 3
After 3 cycles, the ~~colour of~~ light goes off.

$$(G_{s_i} \wedge G_{s_{i+1}} \wedge G_{s_{i+2}}) \longrightarrow \neg G_{s_{i+3}} \quad a$$

$$(Y_{s_i} \wedge Y_{s_{i+1}} \wedge Y_{s_{i+2}}) \longrightarrow \neg Y_{s_{i+3}} \quad b$$

$$(R_{s_i} \wedge R_{s_{i+1}} \wedge R_{s_{i+2}}) \longrightarrow \neg R_{s_{i+3}} \quad c$$

$$\Rightarrow \quad a \wedge b \wedge c$$

2. C, color palette : $\{c_1, c_2, \ldots, c_k\}$
   N: set of nodes
   R: set of edges

1. 2 connected (adjacent) nodes don't have same color. (AXIOM 1)

$$\forall n_1, n_2 \in N,\ Edge(n_1, n_2) \rightarrow \forall c \in C,\ \neg(color(n_1, c) \wedge color(n_2, c))$$

2. Exactly 2 nodes are allowed to wear yellow. (AXIOM 2)

$$\Big(\exists n_1, n_2 \in N\ color(n_1, y) \wedge color(n_2, y) \wedge n_1 \neq n_2 \wedge \forall n \in N \wedge (n = n_1) \wedge (n \neq n_2)\Big) \rightarrow \neg(color(n, y))$$

y: yellow
color(n, d) means node n has color d.

3. Starting from any red node, you can reach a green node in no more than 4 steps. (AXIOM 3)

route(e_1, e_2, s) : Route from a node e_1 to e_2 with s ~~or~~ less no: of steps.

$\forall n \in N \ color(n, R) \longrightarrow (\exists n_2 \in N \ color(n_2, G) \wedge route(n, n_2, 4))$

R: red color
G: green color.

4. For every color in the palette, there is atleast one node with this color (AXIOM 4)

$\cancel{\forall c \exists n \in N \ color(n,}$
$\forall t \in C \ \exists n \in N \ color(n, t)$

5. The nodes are divided into exactly |C| disjoint non-empty cliques (AXIOM 5).

~~Any 2 nodes of same color, they must be connected~~

$\cancel{\forall n_1, n_2 \in N, \forall c \in C (color(n_1, c) \wedge color(n_2, c) \wedge n_1 \neq n_2) \rightarrow Edge(n_1, n_2) \wedge Edge(n_2, n_1)}$

$\forall n_1, n_2 \in N, \forall t \in C \ (color(n_1, t) \wedge color(n_2, t) \wedge n_1 \neq n_2) \longrightarrow (Edge(n_1, n_2) \wedge Edge(n_2, n_1))$

3

Whoever can read is literate.

~~R(t) x can~~
$R(t)$: t can read
$L(t)$: t is literate

PL $\Rightarrow$ $R \longrightarrow L$
FOL $\Rightarrow$ ~~$\forall t (R(t)$~~ $\forall t (R(t) \longrightarrow L(t))$

Dolphins unfortunately are not literate

$D(t)$: t is dolphin
$L(t)$: t is literate.

PL $\Rightarrow$ $D \longrightarrow \neg L$
FOL $\Rightarrow$ $\forall t (D(t) \longrightarrow \neg L(t))$

Some dolphins are intelligent

from above parts $D(t)$ is used.
$I(t)$: t is intelligent

PL $\Rightarrow$ ~~$D \wedge I$~~ $D \wedge I$
FOL $\Rightarrow$ $\exists x (D(x) \wedge I(x))$

Some who are intelligent cannot read.

$I(t)$ & $R(t)$ from above questions.

$PL \Rightarrow I \wedge \neg R$

$FOL \Rightarrow \cancel{\ } \exists t \, (I(t) \wedge \neg R(t))$

There exists a dolphin who is both intelligent & can read but for every intelligent dolphin, if it can read, it must be that it is not literate.

Predicates used here from above questions.

$\cancel{PL \Rightarrow (D \wedge I \wedge R) \wedge \forall x((D(x) \wedge I(x) \wedge R(x)) \longrightarrow \neg L(x)}$

$PL \Rightarrow (D \wedge I \wedge R) \wedge \forall x\big(D(x) \wedge I(x) \wedge R(x)) \longrightarrow \neg L(x)\big)$

$\cancel{FOL \Rightarrow \exists t} \, ($

$FOL \Rightarrow \exists t\,(D(t) \wedge I(t) \wedge R(t)) \wedge \forall x\Big((D(x) \wedge I(x) \wedge R(x) \; \cancel{\rightarrow} \; \longrightarrow \neg L(x)\Big)$

__/__/__

~~Re~~ Resolution of 4th statement.

~~Pre~~ PL ⇒ CNF

$R \rightarrow L \Rightarrow \neg R \vee L$

$D \rightarrow \neg L \Rightarrow \neg D \vee \neg L$

$D \wedge I$

~~I ∧ ¬R~~

To prove, $I \wedge \neg R$ is true

We ~~take its~~ we negate it.

$= \neg(I \wedge \neg R)$

$= \neg I \vee R$

Available set of clauses:

$\{(\neg R \vee L), (\neg D \vee \neg L), (D), (I)\}$

Resolution

$\neg I \vee R \qquad \neg R \vee L$

$\neg I \vee L \qquad \neg D \vee \neg L$

$\neg I \qquad I$

empty clause

This confirms our contradiction, ~~implying (1) has to false~~

__/__/__

Resolution of 5^{th} statement.

$$(D \wedge I \wedge R) \wedge \forall x((D(x) \wedge I(x) \wedge R(x) \longrightarrow \neg L(x))$$

negation of above statement.

$$(\neg D \vee \neg I \vee \neg R) \vee \exists x(D(x) \wedge I(x) \wedge R(x) \wedge L(x))$$

in this we can see that there exists someone who ~~is~~ can read but is not literate, this contradicts $R \longrightarrow L$ (statement 1)

This confirms our contradiction.

COMPUTATIONAL

2. Reasoning:

- **Brute-Force Approach:**
  Memory used for direct_route_brute_force: 0.03125 MiB
  Time taken for direct_route_brute_force: 0.6642169952392578 seconds
  Steps taken for direct_route_brute_force: 4804
  Test direct_route_brute_force (2001, 2005): Pass

- **FOL Library-Based Reasoning:**
  Memory used for query_direct_routes: 0.0002 MiB
  Time taken for query_direct_routes: 0.6582179069519043 seconds
  Steps taken for query_direct_routes: 4
  Test query_direct_routes (2001, 2005): Pass

Brute force approach is less efficient since it has O(n) complexity and requires 4804 steps to reach the solution. For this particular test case, their time taken is approximately equal, brute force approach taking more time than FOL pydatalog approach. Brute force uses more memory as compared to FOL method because it relies on dictionary structures instead of linear search(O(n)), incase of brute force.

3. Planning:

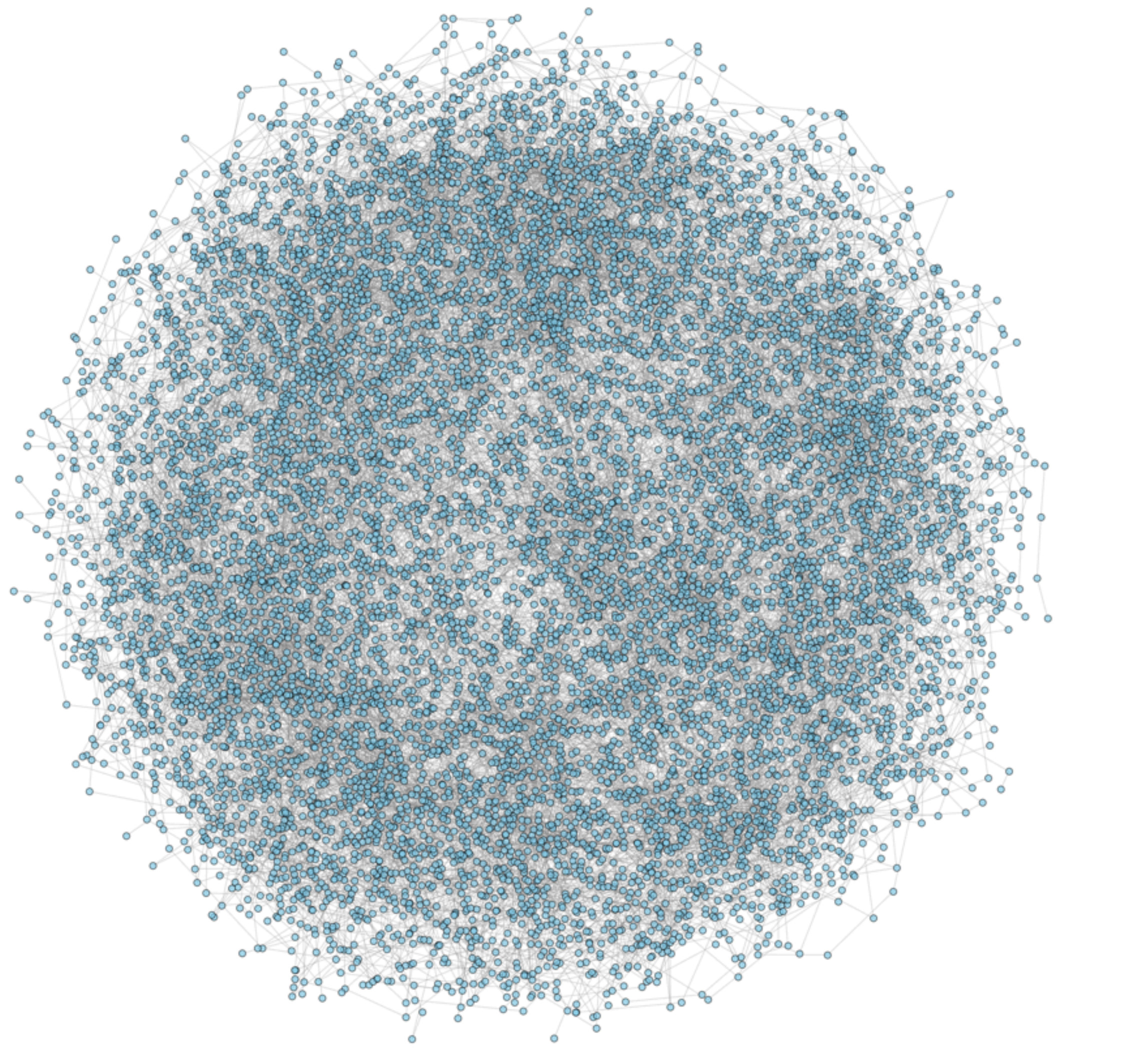
- **Forward Chaining:**
  Memory used for forward_chaining: 0.015625 MiB
  Time taken for forward_chaining: 0.6186718940734863 seconds
  Steps taken for forward_chaining: 4
  Test forward_chaining (22540, 2573, 4686, 1): Pass

- **Backward Chaining:**
  Memory used for backward_chaining: 0.0625 MiB
  Time taken for backward_chaining: 0.6909177303314209 seconds
  Steps taken for backward_chaining: 2
  Test backward_chaining (22540, 2573, 4686, 1): Pass

Both the methods take approximately equal time, with backward chaining taking more time than forward chaining. Both of the methods and memory efficient with backward chaining taking slightly more memory than forward chaining, this is because intermediate caching is used for goal based querying. In terms of steps taken backward chaining is more efficient.

In all the above methods, to get non zero memory a dummy is being used.

Graph Representation of route_to_stops mapping using Plotly



Here,

- Stops are nodes
- Routes are the edges

High resolution png is attached with the assignment