Knowledge Representation & Reasoning

COMP7021(Spring 2021)



Mohit Mehndiratta, 20622275

Master of Data Science

23/10/2021

# Report – Case Study (Coloured Cubes)

## Problem Domain Scenario:

Consider 11 cubes, out of which 3 cubes are red in color, 3 cubes are blue in color,

3 cubes are green in color and 2 cubes are yellow in color. They are stacked randomly

on each other on a table and creating a single stack. We must perform 2 actions. One is "put" in which a robot must put a cube one at a time either on the table if there is an empty space and none of the cube placed matches the color of that cube, otherwise put the cube on the same color of the cube. Another action is "remove", in which a robot has to remove the cube which is yellow in color. Thus, in the final step we should have 3 stacks of cubes of colors red, blue, and green.

Below is the transition diagram which depicts above scenario:

R1

Y1

B1

G1

B2

G2

R2

Y2

R3

G3

B3

B3

G2

R2

B2

R3

Transition

R1

B1

G3

G1

T0 (Start). TF (Final) table

Please note that we have denoted colours as names like r1 for first red cube, y1 for first

Yellow cube etc.

## Complete System Description:

Formal Description - Signature:

cube = {r1,y1,b1,g1,b2,g2,r2,y2,b3,r3,g3}

loc = {r1,y1,b1,g1,b2,g2,r2,y2,b3,r3,g3,t}

inertial fluent on(C1,L1) which states Cube C1 is on location L1

defined fluent above(C1,L1) which states Cube C1 is above location L1

action = put(C1,L1) which states put Cube C1 at location L1, remove(C1) which states

remove the cube C1.

n = 11

step = 0..n

System Description:

Causal Law:

put(C1, L1) causes on(C1, L1)

remove(C1) causes removed(C1)

State Constraints:

-on(C1,L1) if not on(C1,B1)

-on(C1,L1) if remove(C1)

-on(C1,L2) if on(C1,L1), L1 != L2

-on(C2,C) if on(C1,C), C1 != C2

above(C1,L1) if on(C1,L1)

above(C1,L1) if on(C1,C2), on(C2,L1)

Executability Conditions:

impossible put(C1,L1) if on(C2,C1), loc(L1)

impossible put(C1,C2) if on(C3,C1), on(C2,C4)

impossible put(C1,L1) if on(C1,L2), on(C2,L1), C1 != C2

## Clingo Program:

%% Signature

%% Statics

%% define cubes

cube(r1). cube(y1). cube(b1). cube(g1). cube(b2). cube(g2).

cube(r2). cube(y2). cube(b3). cube(r3). cube(g3).

%% define locations

loc(r1). loc(y1). loc(b1). loc(g1). loc(b2). loc(g2).

loc(r2). loc(y2). loc(b3). loc(r3). loc(g3). loc(t).

%% Intertial and defined fluents

fluent(inertial,on(C1,C2)) :- holds(on(C1,C2),I).

fluent(defined,above(C1,C2)) :- holds(above(C1,C2),I).

%% Setting step numbers

#const n=11.

step(0..n).

%%% cube not on location L at step 0, CWA

-holds(on(C1,L1), 0) :- not holds(on(C1,L1), 0), cube(C1), loc(L1).

%% Encoding SD

%% Causal Law

%% Putting cube B on location L at step I causes B to be on L at step I+1

holds(on(C1,L1), I+1) :- occurs(put(C1,L1), I).

%% Removing C1 cube will cause removed C1 cube.

removed(C1) :- occurs(remove(C1), I), cube(C1).

%% State Constraints

%% if removal of a cube occur at step I, then it would not hold any

%% relation at step I+1

-holds(on(C1,L1), I+1) :- occurs(remove(C1), I), cube(C1), loc(L1).

%% A cube cannot be in two location at once

-holds(on(C1,L2), I) :- holds(on(C1,L1), I), L1 != L2, loc(L2).

%% Only one cube can be directly on top of one another.

-holds(on(C2,C),I) :- holds(on(C1,C),I), C1 != C2, cube(C), cube(C2).

%% above(C1,L1) if on(C1,L1)

holds(above(C1,L1),I) :- holds(on(C1,L1),I).

%% above(C1,L1) if on(C1,C2), on(C2,L1)

holds(above(C1,L1),I) :- holds(on(C1,C2),I), holds(on(C2,L1),I).

%% Executability conditions:

%% It is impossible to move an occupied cube:

-occurs(put(C,T),I) :- holds(on(C1,C),I), loc(T).

-occurs(put(C,C2),I) :- holds(on(C1,C),I),

holds(on(C2,C3),I),

C1 != C2, C != C3.

%% It is impossible to move a cube onto

%% an occupied cube:

-occurs(put(C1,C),I) :- holds(on(C1,L),I),

holds(on(C2,C),I),

C1 != C2, C != t.

%% CWA for defined fluent

-holds(F,I) :- fluent(defined,F), not holds(F,I), step(I).

%% CWA for removed

-removed(B) :- not removed(B), cube(B).

%% Inertia rules

%% Inertia rule 1: Anything that holds at step I, will also hold at step I+1 as long as no evidence shows its opposite.

holds(F,I+1) :- fluent(inertial,F), holds(F,I), not -holds(F,I+1), I<n.

%% Inertia rule 2: Anything that does not hold at step I, will also not hold at step I+1, as long as no evidence shows its opposite.

-holds(F,I+1) :- fluent(inertial,F), -holds(F,I), not holds(F,I+1), I<n.

%% Initial Configuration

holds(on(g3,t), 0). holds(on(r3,g3), 0). holds(on(b3,r3), 0).

holds(on(y2,b3), 0). holds(on(r2,y2), 0). holds(on(g2,r2), 0).

holds(on(b2,g2), 0). holds(on(g1,b2), 0). holds(on(b1,g1), 0).

holds(on(y1,b1), 0). holds(on(r1,y1), 0).

%% Encoding action

occurs(put(r1,t),0).

occurs(remove(y1),1).

occurs(put(b1,t),2).

occurs(put(g1,t),3).

occurs(put(b2,b1),4).

occurs(put(g2,g1),5).

occurs(put(r2,r1),6).

occurs(remove(y2),7).

occurs(put(b3,b2),8).

occurs(put(r3,r2),9).

occurs(put(g3,g2),10).

#show holds/2.

#show -removed/1.

#show removed/1.

## Query Evaluations:

1. Show all the states for holds:

The Output below depicts every information of cubes placed at each state.

Output:

Text

Description automatically generated

The above output contains information like:

holds(on(r3,r2),11), holds(on(g3,g2),11), holds(on(b3,b2),11), holds(on(r2,r1),11), holds(on(g2,g1),11), holds(on(b2,b1),11), holds(on(r1,t),11), holds(on(g1,t),11), holds(on(b1,t),11).

This exactly, is the 11th and final stage of our desired outcome.

1. Show the ones which are removed:

The output below shows all the yellow cubes which were removed.

Output:

Text

Description automatically generated

From this output, we can see that y1 and y2 (yellow) cubes are removed.

1. Show the ones which are not removed:

The output below, shows all the cubes which are not removed.

Output:

Text

Description automatically generated

From this output, we can see that all the cubes except the “yellow” were not removed.