Thursday
4-6-2020

AI Assignment-3

N.V.S.K.Kalyani
17UA1A0584
3rd B.Tech CSE-C

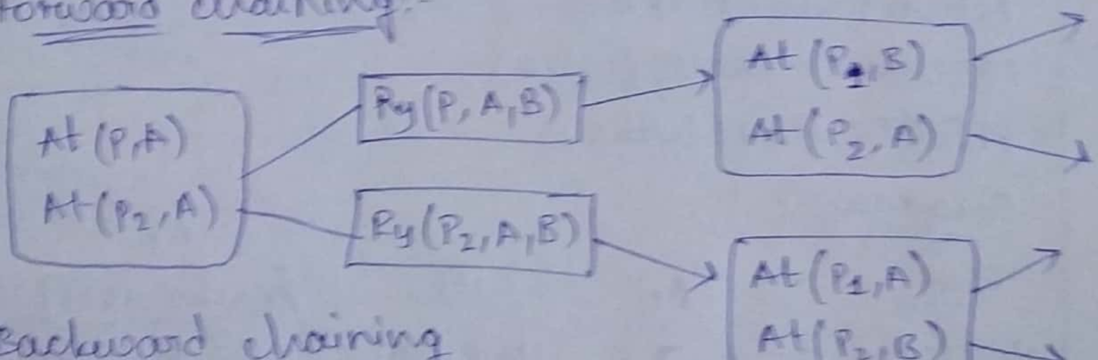
Unit-3

2 marks

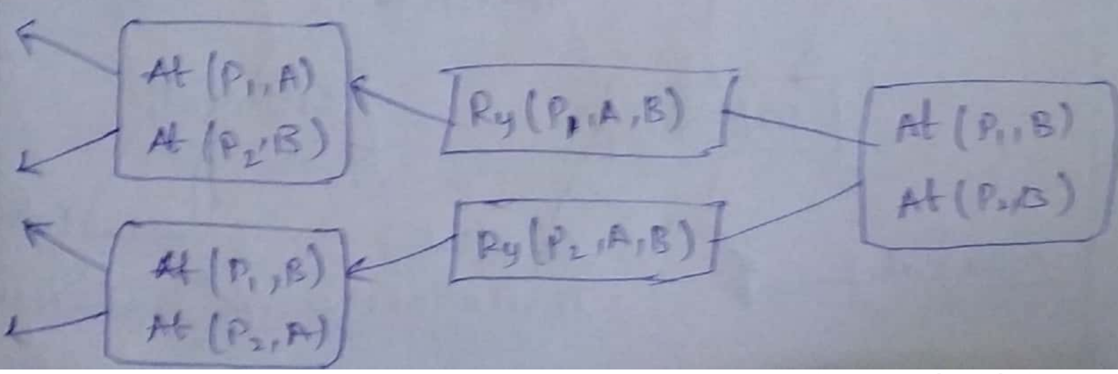
1Ans Algorithm for Air Cargo:-

Init (At(C_1, SFO) ∧ At(C_2, JFK) ∧ At(P_1, SFO) ∧ At(P_2, JFK)
∧ cargo(C_1) ∧ cargo(C_2) ∧ plane(P_1) ∧ plane(P_2)
∧ Airport (JFK) ∧ Airport (SFO)
Goal (At (C_1, JFK) ∧ At(C_2, SFO))
Action (load (C, P, a))
PRECOND: At(c,a) ∧ At(P,a) ∧ cargo(c) ∧ plane(p) ∧ Airport(a)
EFFECT: ¬At(c,a) ∧ In(c,p))
Action (unload (c,p,a))
PRECOND: In(c,p) ∧ At(p,a) ∧ cargo(c) ∧ plane(p) ∧ Airport(a)
EFFECT: At(c,a) ∧ ¬In (c,p)
Action (Fly(p, from, to))
PRECOND: At (c, from) ∧ plane(p) ∧ Airport (from) ∧ Airport (to)
EFFECT: ¬At (p, from) ∧ At(p, to)

2Ans forward chaining:-



Backward chaining



3 Ans Any planning Algorithm that can place two action into a plan without specifying which comes first a partial-order planner.

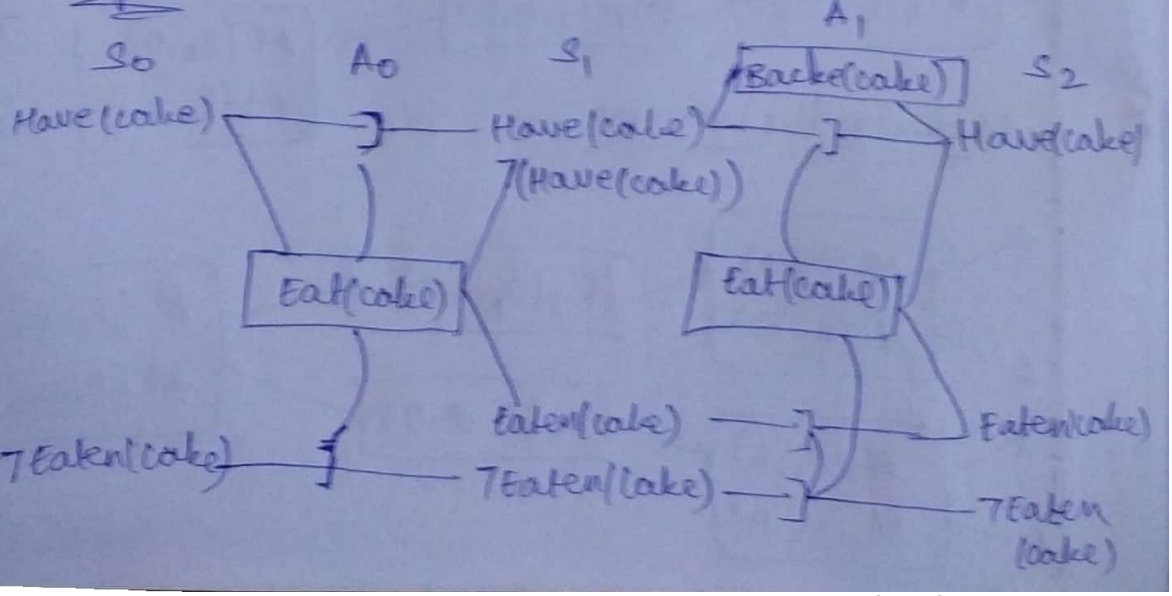
Algorithm:-

Goal (Rightshoe on A leftshoe on)
Init()
Action (Rightshoe. PRECOND: Rightshock on, EFFECT: RS)
Action (RightsOck · EFFECT: Rightsock on)
Action (Leftshoe, PRECOND: Leftsock on, EFFECT: LSsock)
Action (Leftsock, EFFECT: Leftsock on)

4 Ans Algorithm for Have cake & EAT cake:-

Init (Have(cake))
Goal (Have(cake ∧ Eaten(cake))
Action (Eat(cake))
PRECOND: Have(cake)
EFFECT: ¬Have(cake) ∧ Eaten(cake)
Action (Bake(cake))
PRECOND: ¬Have(cake)
EFFECT: Have(cake)

Diagram:-



5AM Algorithm:-

Init (chassis $(C_1) \wedge$ chassis (C_2))
A Engine $(E_1, C_1, 30)$ A Engine $(E_2, C_2, 60)$
A wheels $(W_1, C_1, 30)$ A wheels $(W_2, C_2, 15)$)
Goal (Done $(C_1) \wedge$ Done (C_2))
Action (AddEngine (e,c))
PRECOND: Engine (e,c,d) A chasis (c) A EngineIn (c)
EFFECT: EngineIn (c) A Duration (d)
Action (Addwheels (w,c))
PRECOND: wheels (w,c,d) A chasis (c) A EngineIn (c)
EFFECT: wheelson (c) A Duration (d)
Action (Inspect (c), PRECOND, EngineIn (c) $\wedge$ wheelson (c) $\wedge$ chassis (c),
EFFECT: Done (c) A Duration (10))

10Marks

1AM

Spare tire problem

problem statement

The goal is to have a good spare tire properly mounted into the cars axle, where the intial state has a flat tire on the axle and a good spare tire in the trunk.

* There are just four actions
  → Removing the spare from the trunk.
  → Removing flat tire from axle.
  → Putting spare tire on axle.
  → leaving the car unattended overnight.

* Explanation

Init (At (Flat, Axle) A At (Spare, Trunk))

Goal (At (Spare, Axle))

Action (Remove (Spare, Trunk))

PRECOND: At (Spare, Trunk)

Effect: ¬At (Spare, Trunk) ∧ At (Spare, Ground))

Action (Remove (Flat, Axle))

PRECOND: At (Flat, Axle)

Effect: ¬At (Flat, Axle) ∧ ¬At (Flat, Ground))

Action (puton (Spare, Axle))

PRECOND: At (Spare, Ground) ∧ ¬At (Flat, Axle)

EFFECT: ¬At (Spare, Ground) ∧ At (Spare, Axle)

Action (leave overnight)

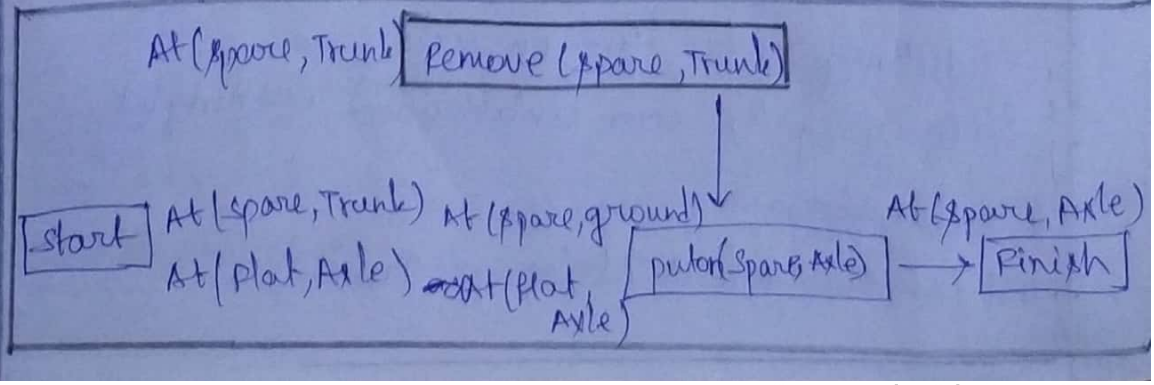
PRECOND: ¬At (Spare, Ground) ∧ ¬At (Spare, Axle) ∧ ¬ At (Spare, Trunk)

EFFECT: A ¬At (Flat, Ground) ∧ ¬At (Flat, Axle)

* Sequence of events for flat tire

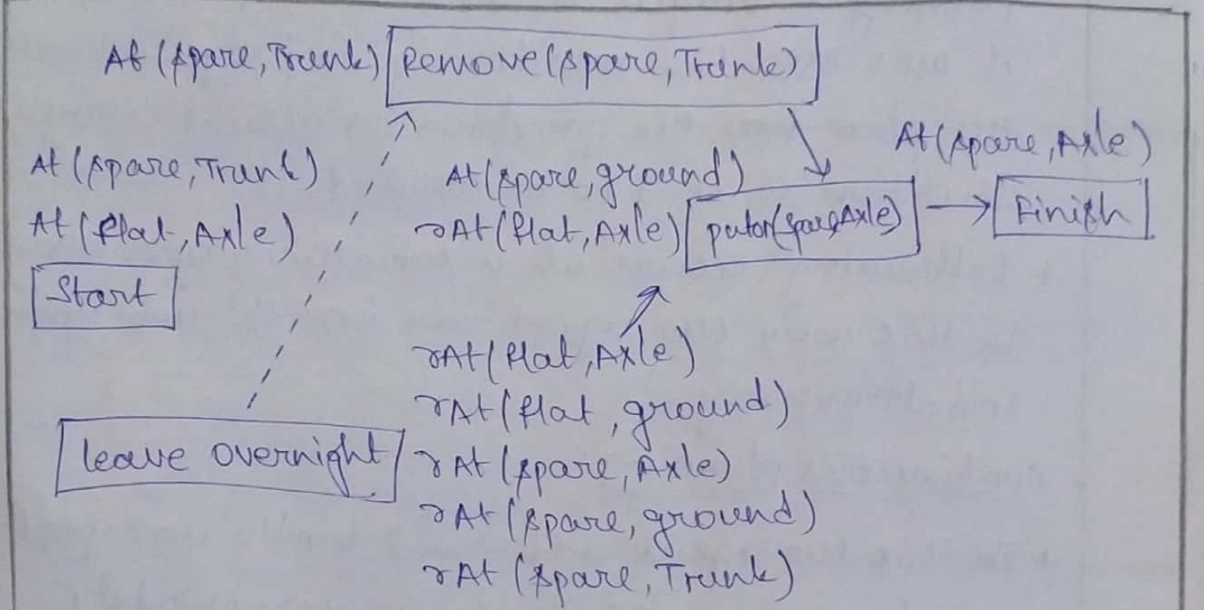
1. Pick the only open precondition At (Spare, Axle) of finish. choose the only applicable action, puton (Spare, Axle).
2. pick the At (Spare, Ground) precondition of puton (Spare, Axle). choose the only applicable action, Remove (Spare, Trunk) to achieve it.

* The resulting plan is



3. pick the ¬At(Flat, Axle) precondition puton(Spare, Axle)

Remove(Spare, Trunk) $\xrightarrow{\text{At(Spare, Ground)}}$ puton(Spare, Axle)



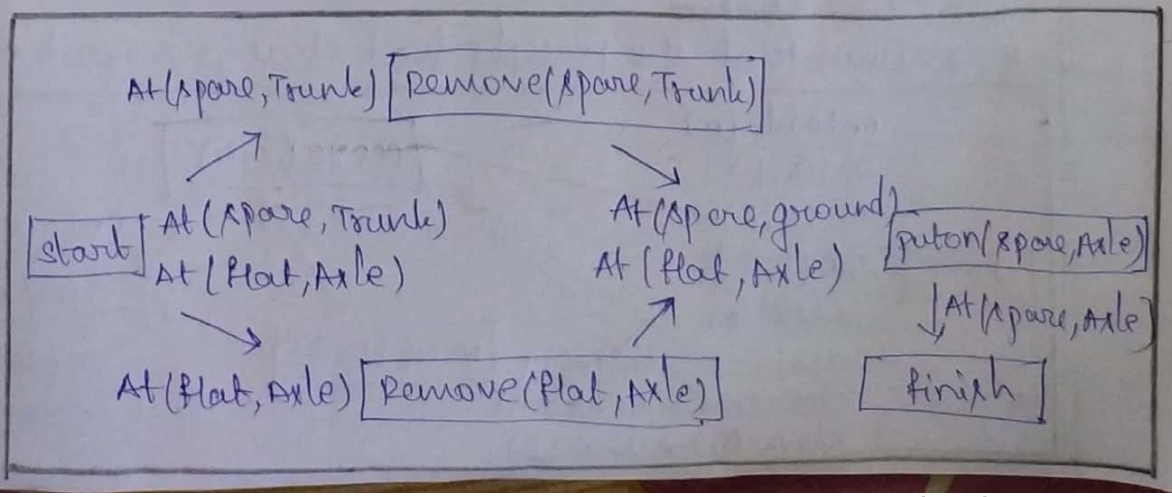
4. The only remaining open preconditions at this point is the At(Spare, Trunk) precondition of the action Remove(Spare, Trunk)

5. consider again pick the At(~~Spare~~ Flat, ~~Trunk~~ Axle) precondition of ~~remove~~ ¬puton(Spare, Axle). we choose Remove(Flat, Axle).

6. Once again pick the At(Spare, Trunk) precondition of Remove(Spare, Trunk) and choose start to achieve it. This time there is no conflicts.

7. Pick the At(Flat, Axle) precondition of Remove(Flat, Axle).

complete - consistent plan:-



2.4

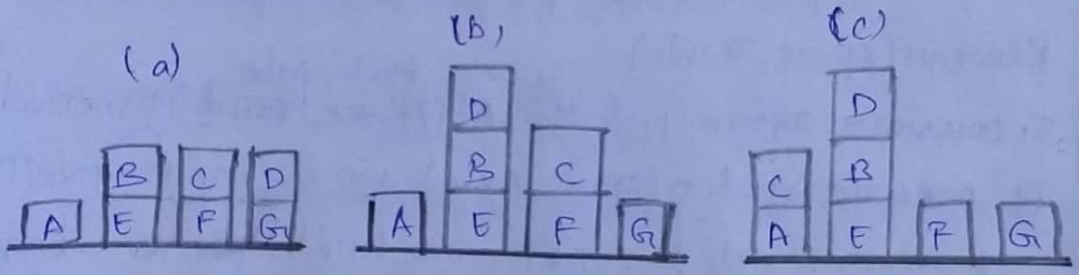
Execution monitoring and Replacing

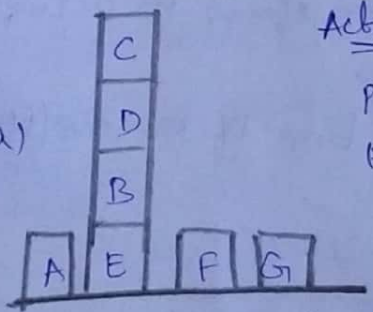
* In the approach, the agent can use any of Preceding planning technique to construct a plan but also it uses execution monitoring to judge whether the plan has the provision for actual current situation or need to be revised.
* Replanning occurs when something goes wrong In this way the agent can handle unbounded Indeterminary.

Continuous planning:

* In this the agent additional handle unexpected events, can revise its plans appropriately.

Ex:- It is a problem from the blocks of world domain





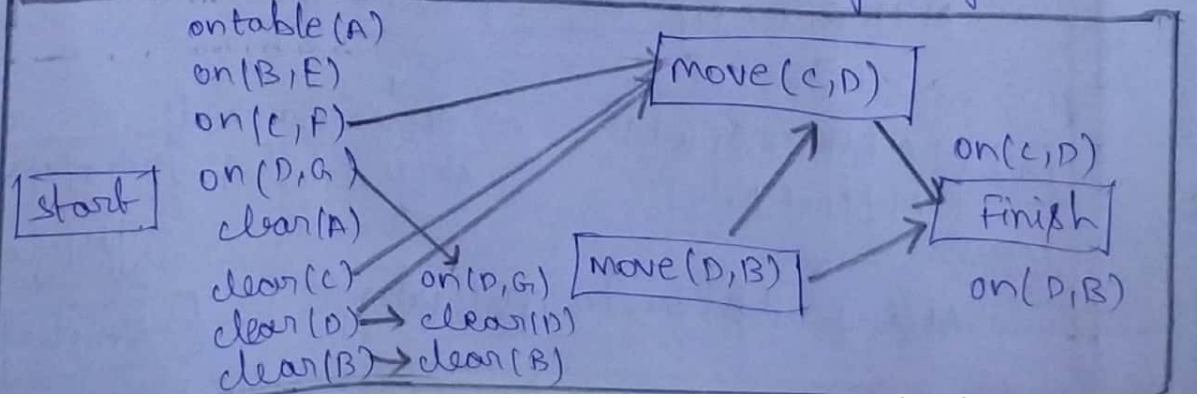
Action schema: Action (move(X,Y))

PRECOND: (clear(Z) ∧ clear(Y) ∧ on(X,X))

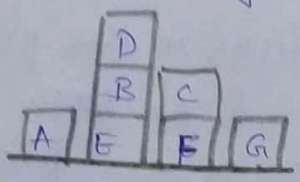
Effect: on(Z,Y) ∧ clear(Z) ∧ on(X,Y) ∧ ¬(clear(Y))

Goal: on(C,D) ∧ on(D,B)

(i) we assume that the percepts don't change & agent construct

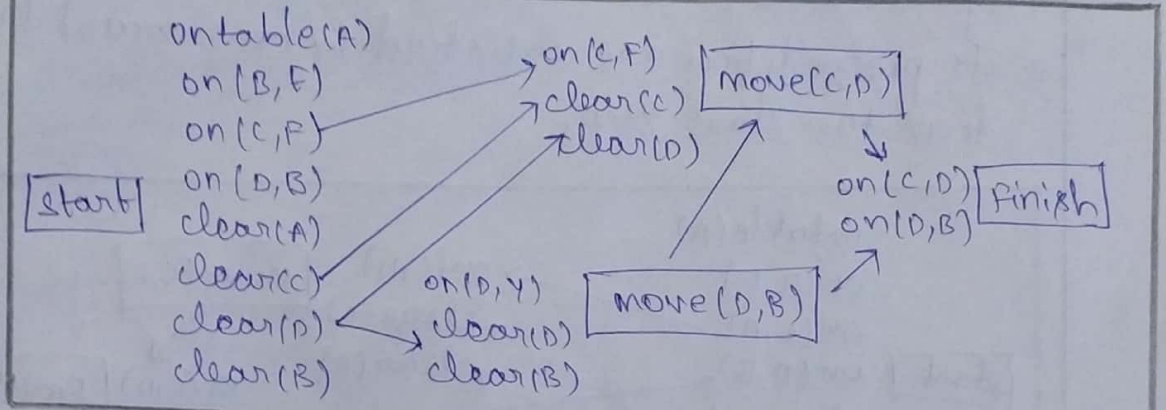


ii, After moving 'D' on 'B', and the world now in shown



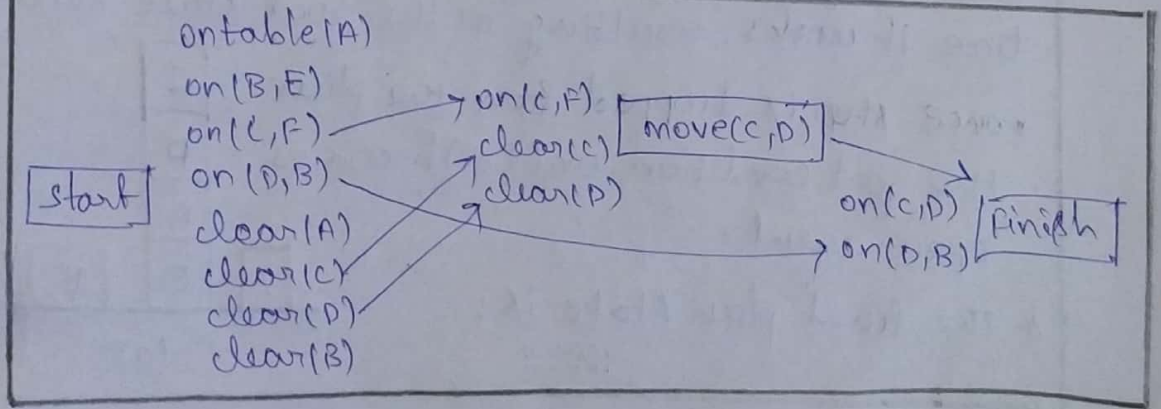
(b)

∴ The new plan is shown below:

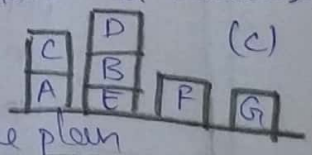


iii, Removing Redundant step:-

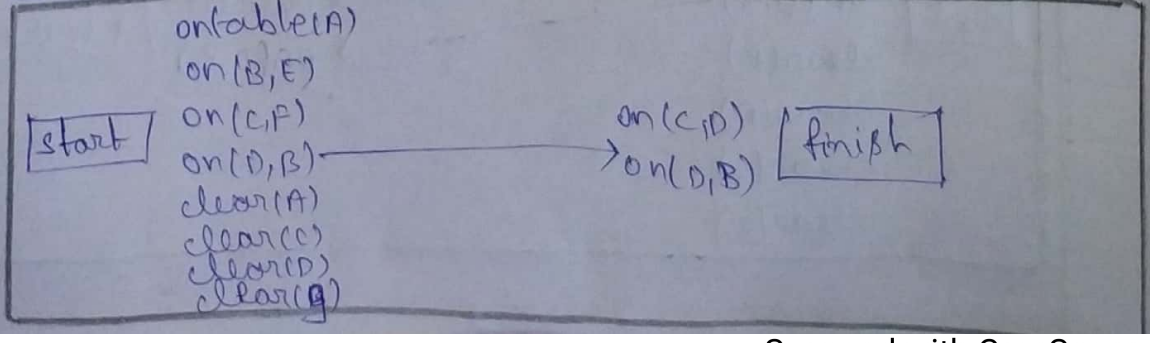
* Once the old causal link from move(D,B) to finish is removed, move(B,B) on longer supplies any causal links at all
* The given flowing plan



iv, After moving 'c' on 'D' is ready to executed because of all its preconditions are satisfied by the start step are necessarily before it, does not conflict with any other link in the plan

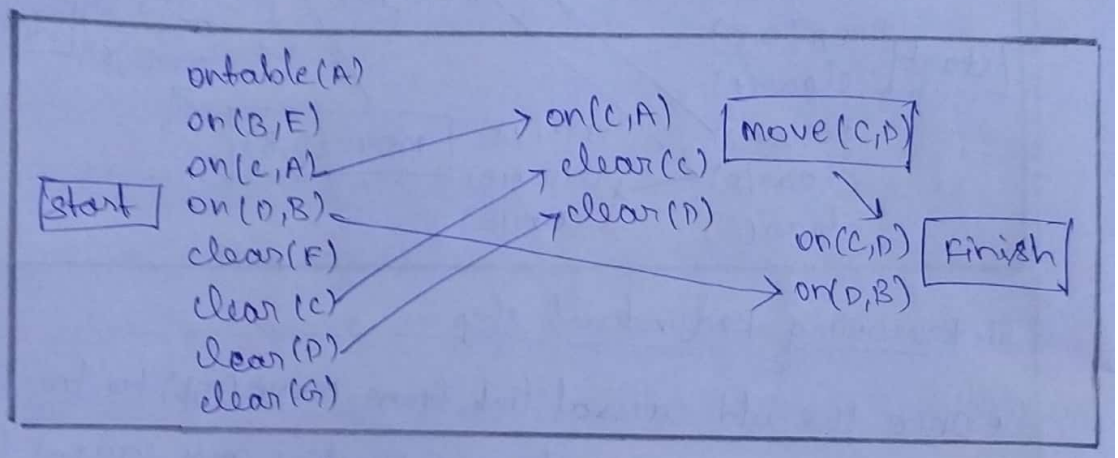


(c)



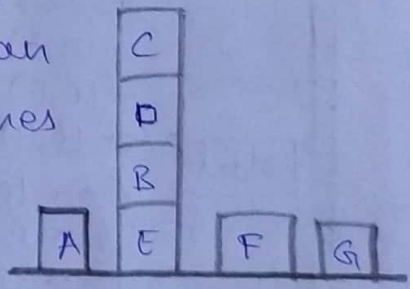
v. Resolving open condition

* Although there are now an actions in the plan this is still an open condition for the finish step.
* The agent decides on plan for the open condition once again, move(C,D) will satisfy the goal condition.
* Its precondition are satisfied by new casual links from the start step.



vi. Re-executed of move(C,D)

* once again move(C,D) is ready for execution. This time it works, resulting in the goal state shown
* once step is dropped from the plan the goal condition on(C,D) becomes open agent.
* The final plan state is:



(a)