gradeup

Planning

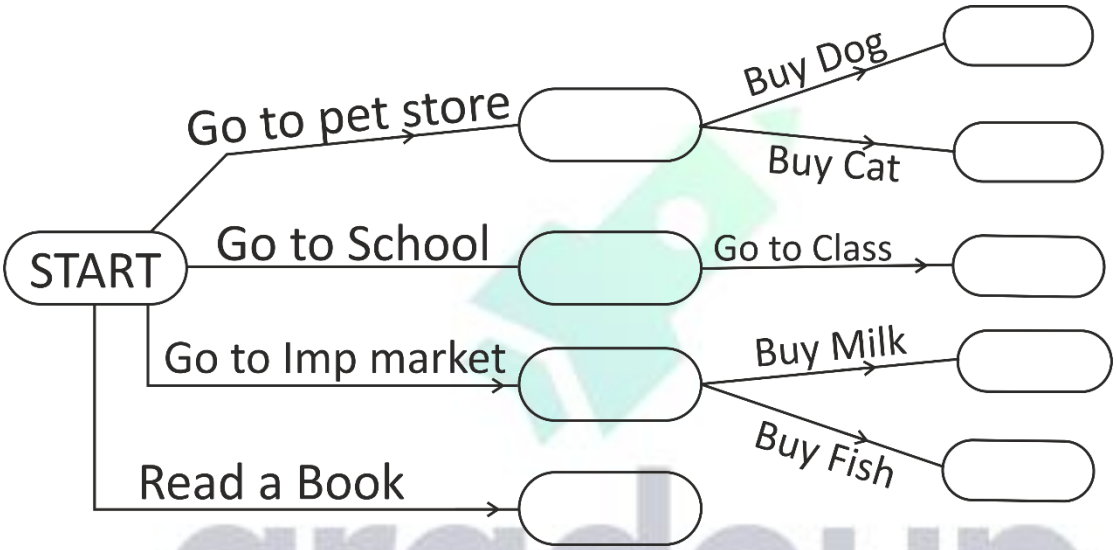
Prep Smart. Stay Safe. go Gradeup

gradeup.co

Planning in AI

It deals with combined goals where goals can be one or more may be needed to done in some sequence, which goals needs to do next and depending on state. Which goal will be possible next. will be some questions a agent have to work out.

Using search algorithm without planning can be complex. So some sort of planning is needed.



Initial state is home:

Goal state is getting milk and fish.

If you use searching algorithm without planning then branching will be huge and search is adequate here.

Agent must consider action sequences starting from initial state. Before we can purchase anything, have to go to supermarket but the agent does not know this.

Planning is needed to find a sequence of action that achieve a goal.

Divide and conquer is required in which an efficient planner should consider problem decomposition it should work on sub goals independently.

Some deduction of how planning works using preconditions, we should know that getting to grocery store is a precondition of buying the milk.

Factors that should be considered while defining planning are:

- Define states
- Define goals
- Define preconditions
- Define post conditions

We can use specification languages like STRIPS or ADL to describe a system.

Another possibility to describe system is the planning domain definition language (PDDL). It describes a system using a set of pre conditions and post conditions. PDDL is a domain definition language which is supported by most planners.

STRIP or ADL is used to program in a way that takes the pre condition into consideration. So STRIPS is a language and ADL is another language that is considered while dealing with algorithms and programming using planning functionality and planning algorithms.

Planning can be executed using 2 algorithm:

1. Forward state space search
2. Backward state space search

Example of how preconditions can be used for planning. They are first defining the initial state of a cargo agent who manages loading and unloading and scheduling of supplies for cargo.

The problem can be defined by 3 actions:

LOAD UNLOAD FLY

Possible initial states and preconditions:

At (C_1, SFo) $\wedge$ At (C_2, JFK) $\wedge$ At(P_1, SF0)

$\wedge$ At (P_2, JFK) $\wedge$ Cargo (C_1) $\wedge$ Cargo (C_2)

$\wedge$ Plane (P_1) $\wedge$ Plane (P_2) $\wedge$ Airport (CFK)

($\wedge$ Airport (SFo))

(At (C_1, JFK)) $\wedge$ At (C_2, SFo))

(load C_1 to JFK & C_2 at SFo)

Action:(load (c,p,a),[load cargo in plane at)

Pre-condition: At (c,a) ^ At(P,a) ^ cargo(c) ^ plane(p) ^ airport(a) (load cargo at airport , plane is at airport)

Effect: ~At(c,a) ^ In(c,p) (cargo will be at this airport using 'At' function & cargo will be in place using 'IN; function)

Action(unload (c,p,a))

Precondition: In (c,p) ^ At(p,a) ^ (argo(c)) ^ place(p) ^ Airport(a)

Effect: At(c,a) ^ ~ In(c,p))

Action(Fly(P_1 from, to)

Precondition: At (p,from) ^ plane(p) ^ airport(from) ^ airport(to)

Effect: ~At (P,from) ^ at(p,to)

We can fly if cargo is in specific plane. This is how you define a planning function so planning can be defined by 3 function load, unload, fly.

Predicates means:

IN means: cargo is in plane

At means: plane is at airport

When at then plane don't' have cargo available to reload or use.

LOAD (C1, P1, SFO), fly (P1, C1, JFK)

LOAD (C2, P2, JFK), fly (P2, JFK, SFO)

Components of planning:

1. Choose the best rule for applying based on the best available heuristics.
2. Apply the chosen rule for computing the new problem state.
3. Detect when a SO1 has been found.
4. Detect deed ends so that they can be abandoned & systems effort is - directed in more finite direction.
5. Detect when an almost correct SO1 has been found.

Linear planning: planning or scheduling of project management tasks where distance is a significant factor in the project.

Example of projects include roads, rail, pipeline, and transmission line. It consider both time/ factor of task and location factor.

Basic idea is work on one goal until completely solved before moving on to the next goal.

No inter leaving of goal achieved.

Non linear planning:

Problems such as this one require sub problems to be worked on continuously. Here parallel execution is possible:

Goal stalk planning: use to handle interactive compound goals.

Start: On (B,A) ^ ONTABLE(A) ^ ONTABLE(C) ^ ONTABLE(D)

Goal: ON(C,A) ^ ON(B,D) ^ ONTABLE(A) ^ ONTABLE(D)

We split the problem into 4 sub problems. 2 are already solved and true

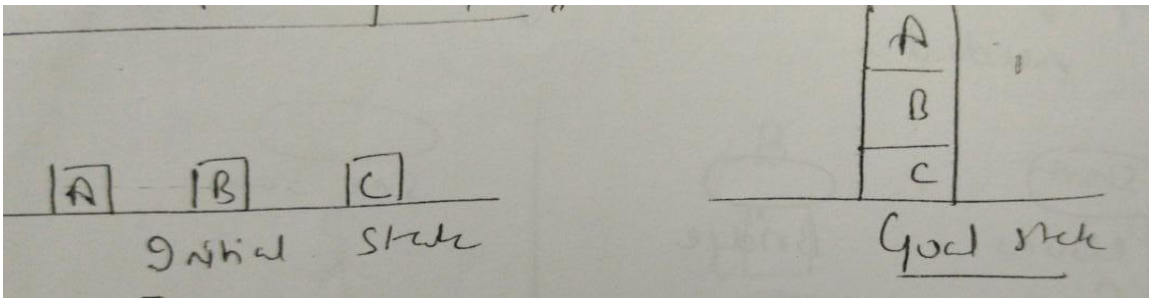
UNTABLE(A)

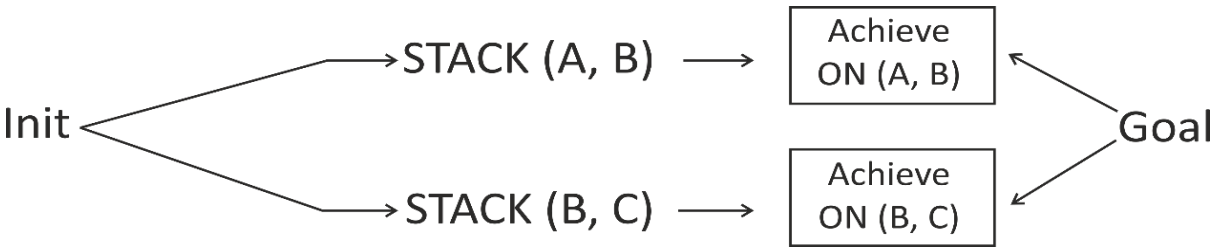
UNTABLE(D)

Hierarchic cell planning: it is an automated planning in which dependency among actions can be given in the form of networks.

Compound task is decomposed into simple tasks.

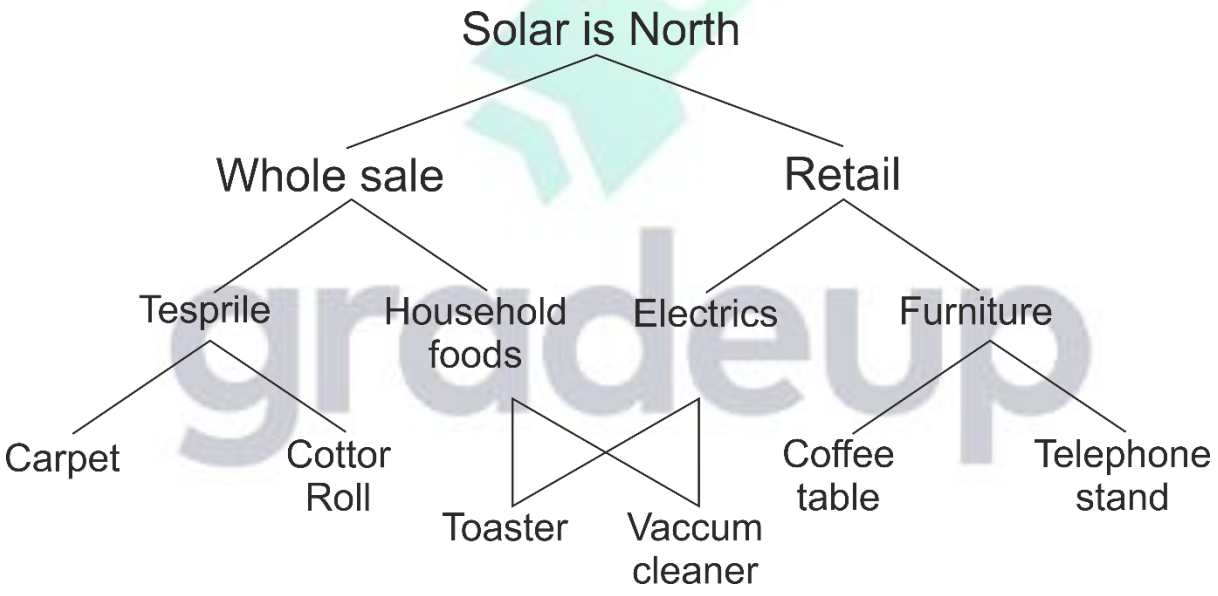
Non-linear planning example:





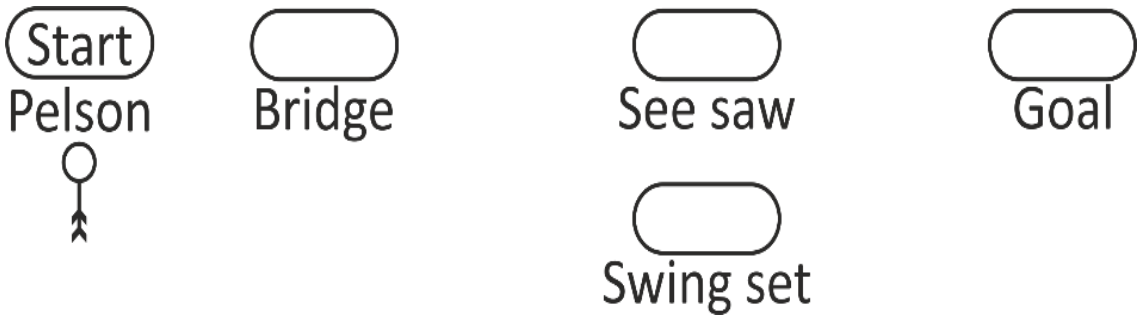
Hierarchy planning example:- organizing thing high to low

Example: human body- system of organs made up of individual organs , composed of tissue , the cells then organs.



Partial ordering Planning: ordering of actions is partial. Also it does not specify which action will come first when 2 actions are processed.

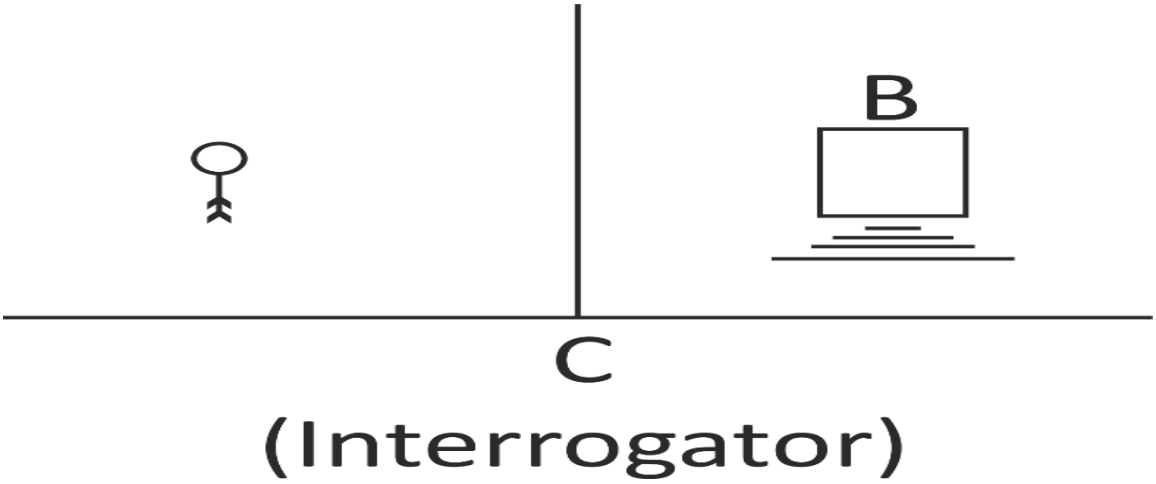
It specifies the ordering between actions when needed.



Person has to travel from start to goal. Bridge must be traversed before see saw and swing set. See saw and swing can be traversed in any order. It is specified only if necessary.

STRIPS: standard research institute problem solver is a planning technique that works by executing a domain and problem to find a goal. You do this by providing objects, actions, preconditions and effects. Once the world is defined, you define a problem set. Problem consist of initial state and a goal condition. STRIPS can search all possible states starting from initial one, executing various actions until it reach goal.

Common language for writing STRIPS is PDDL (planning domain definition language). It lets you to write code in English words which can be clearly read and understood.



Player C, the interrogator is given the task of trying to determine which player A or B is a computer and which is a human. Interrogator ask questions both of them. If interrogator is useable to distinguish the answers provided by both human and computer then computer pass the test and computer is considered as intelligent as human. Computer is trying to make interrogator guess wrongly so as to be indistinguishable from human as much as possible.

gradeup SUPER

Gradeup UGC NET
Super Superscription

Features:

1. 7+ Structured Courses for UGC NET Exam
2. 200+ Mock Tests for UGC NET & MHSET Exams
3. Separate Batches in Hindi & English
4. Mock Tests are available in Hindi & English
5. Available on Mobile & Desktop

Gradeup Super Subscription, Enroll Now

gradeup.co