3rd instance → {small, Red, Cotton}

$S_2 = \{$

MOD-3

KNOWLEGIDE AND REASONING.

REASONING AGENTS:

- Goal based Agent is a type of reasoning agent.

Any knowledge based agent should know about the world, reason about the world, act upon the world.

↓

try to choose

There are two abstraction levels:

Knowledge level - what the agent knows, agent goals are

Symbol level.

what symbols the agent manipulates & how?

function KB-agent (percept) ← P/P

↳ returns an action

Static: KB, a knowledge base

t, a counter, initially 0, indicating time

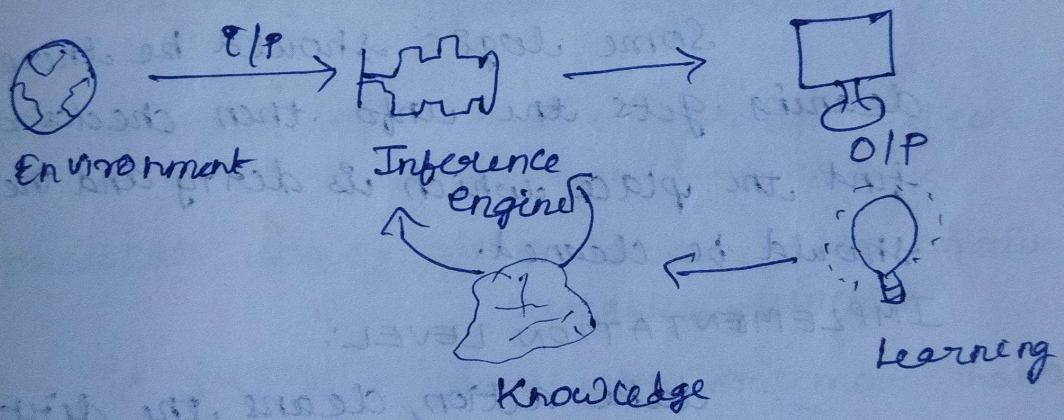
TELL(KB, make-PERCEPT-SENTENCE (percept, t))

action ← ASK(KB, MAKE-ACTION-QUERY(t))
TELL(KB, MAKE-ACTION-SENTENCE(action, t))
t ← t+1
return action.

Here I/P: percept
O/P: action.

- Represent states, actions, etc
- Incorporate new percepts
- update internal representations of the world.
- Deduce hidden properties of the world
- deduce appropriate actions.

KNOWLEDGE BASED AGENT:



KNOWLEGDE BASE:

Acts as a repository of facts, rules, processes and other necessary data that the agent uses to guide its decision making process.

Past experience also added.

INFERENCE ENGINE:

Agent's Component that applies logical rules to the knowledge base to deduce additional information.

OPERATIONAL MECHANISM:

Environment perception
Knowledge base model
Inference engine app
Action determination
Learning from outcome

LEVELS OF KNOWLEDGE BASED AGENT:

KNOWLEDGE LEVEL:

The agent knows where the map is, where the floor is where the cleaning materials are.

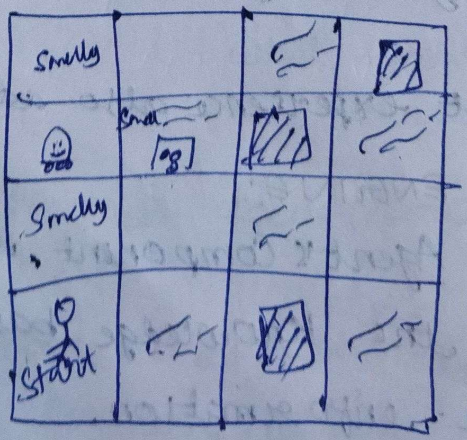
LOGICAL LEVEL:

Some logic should be there for cleaning. gets the info. then checks and find the place which is dirty and then should be cleaned.

IMPLEMENTATION LEVEL:

take action, cleans the dirt on the floor and as a result the agent gets clean floor which is the goal state.

WUMPUS WORLD:



A - gent
B - breeze
G - litter, old
P - pit
OK - safe
S - Stench
V - Visited
W - Wumpus.

LOGIC:-

A formal system for expressing knowledge about a domain consisting of syntax set of legal sentences. and a semantic interpretation of legal sentences.

A proof system - a set of axioms (theorem / rules.) plus rules of inference for deducing sentences from a knowledge base. → deriving some conclusion

Why do we need logic:-

Problem solving agents were very inflexible: hard code for every possible state.

Search is almost always exponential in the number of states.

The agent needs to update the knowledge.

Cannot infer unobserved information.

TYPES:-

- Propositional logic - deals with specific objects either true or false.
- predicate logic - allows statement to contain variables, function
- fuzzy - deals with statement that are somewhat vague, (will be some criteria)
  Eg:- paint is grey
  sky is cloudy
- probability - deals with statements that are possibly true, such as whether I'll win lottery next week
- temporal - deals with about John was a student at crescend for 4 years.

- Modal - deals with statements about belief or knowledge such as Mary believes that John is married to Sue.

PROPOSITIONAL LOGIC:

simpler.

All the statements are made by propositions.

either true or false.

SYMBOLS:- $\neg, \wedge, \vee, \Rightarrow$ - connectives

P, Q, R - propositional symbols.

True or false.

2M SYNTAX:

Sentence → atomic sentence | complex sentence.

atomic → propositional symbol, T, F

complex → ¬ Sentence

& (Sentence $\wedge$ Sentence)

" $\vee$ "

" $\Rightarrow$ "

SEMANTICS:-

1) It is not cloudy and raining

¬ cloudy $\wedge$ raining

2) It is raining

PL:- Raining.

3) It is neither cloudy nor Raining.

¬ Cloudy $\vee$ ¬ Rainy.

4) It is cloudy, so it may be raining

Cloudy $\Rightarrow$ raining.

A	B	¬A	A∧B	A∨B	A⇒B
T	T	F	T	T	T
T	F	F	F	T	F
F	F	T	F	F	T
F	T	T	F	T	T

FIRST ORDER LOGIC: . PREDICATE LOGIC:

First order logic is declarative, compositional and more expressive than propositional logic.

Always FOL contains,

Objects: people, houses, nos, colors

Relations: - sled, around, prime, brother of

functions: father of, best friend, one more than.

SYNTAX:

Sentence → Atomic sentence
| (sentence connective Sentence)
| Quantifier Variable.. sentence
| ¬ sentence.

Atomic sentence → predicate (Term, ...) |
Term = Term

Term → Function (Term, ...)
| Constant
| Variable

Connective → ⇒ | ∧ | ∨ | ⇔

Quantifier → ∀ | ∃

Constant → A | X1 | Mary | ...

Variable → a | x | s | ...

predicate → Before | has color | Raining | ...

function → Mother | left leg.

2M

QUANTIFIERS:

Each quantifier defines a variable for the duration of the following expression.

TYPES:

Universal Quantifier.

∀ < variables < Sentance >

Everyone at Crescent is smart:

$\forall x \; At(x, Cre) \Rightarrow Smart(x)$

- Existential quantifier:

$\exists$ <Variables> <Sentence>

Someone at Cres is ~~not~~ Smart.

$\exists x \; At(x, Cres) \wedge Smart(x)$

$\exists x \; \forall y \; loves(x, y)$

There is a person who loves everyone in the world.

$\forall y \; \exists x \; loves(x, y)$

Everyone in the world is loved by atleast one person.

The following simple facts represents it using First Order Logic:

(1) Marcus was a Man.

Man(Marcus)

↳ constant should come in bracket

(2) Marcus was a pompiean.

pompiean(Marcus)

3) All pompieans were Romans

$\forall x : pompieans(x) \Rightarrow Romans(x)$

4) Caesar was a ruler

ruler(Caesar)

5) All romans were either loyal to Caesar or hated him.

$\forall x : Romans(x) \Rightarrow loyal(Caesar) \vee hated(Caesar)$

6) Everyone is loyal to someone.

~~$\forall x : \exists x$~~ $\forall x : \exists y : \text{loyal to}(x, y)$

7) People only try to assassinate rulers they are not loyal to.

~~$\forall x \exists y$~~ $\forall x \forall y : \text{person}(x) \wedge \text{ruler}(y) \wedge \text{tryassassinate}(x, y) \rightarrow \neg \text{loyal to}(x, y)$

8) Marcus tried to assassinate Caesar.

tryassassinate (Marcus, Caesar)

~~5.ii)~~

1), 2), 3) $\rightarrow$ Marcus was a Roman

4) $\rightarrow$ Caesar was a ruler.

7), 8) $\rightarrow$ Marcus is not loyal to Caesar.

$\neg$ loyal to(Marcus, Caesar)
↑ 7, substitution
person(Marcus) ∧
ruler(Caesar) ∧
try assassinate (Marcus, Caesar)
↑
person (marcus)

REPRESENTING IS-A RELATIONSHIP:-

Is A and Instance supports inheritance and play important role

1) Joe is a musician.

Musician(Joe)
isa(Joe, Musician)

2) Marcus was a man.

Man(Marcus)
Isa(Marcus, Man)
instance (Marcus, Man)

3) All pompeians were Romans)

$\forall x$: Pompeians(X) $\Rightarrow$ Roman(X)

ruler(Pompeian, Roman)
Is a (pompeian, Roman)

COMPUTABLE FUNCTIONS AND PREDICATES:.

All the simple facts can be expressed as combination of individual poredicates such as as try assassinate(Marcus, Caesar).

Eg:. No Mortal lives longer than 150 years

$\forall x : \forall t_1 : \forall t_2 : mortal(x) \wedge born(x, t_1) \wedge gt(t_2 - t_1, 150) \rightarrow dead(x, t_2)$

x is born in t_1 t_2 → Current year.

If someone dies, then he is dead at all times.

INFERENCES IN PREDICATE LOGIC:.

① Modus Ponens → $p \Rightarrow q, P \Rightarrow q$: q is true.
Modus Tollens.

MODUS PONENS:.

S-1: If I'm sleepy I will go to bed.

Sleepy ⇒ Go to bed

S-2:. I'm sleepy.

∴ The inference there is I go to bed.

$$p \rightarrow q, p \Leftrightarrow q.$$

MODUS TOLLENS:.

$$p \rightarrow q, \neg q \Leftrightarrow \neg p.$$

Eg:.

Statement-1:. If I'm sleepy, I go to bed

Statement-2:. I'm not going to bed.

Conclusion:- I'm not sleeping

② AND ELIMINATION:.

$$\frac{\alpha \wedge \beta}{\alpha}, \quad \alpha \wedge \beta = \alpha, \beta$$

Eg:. Marcus was a man and a pompiean.

Man(Marcus) ∧ pompiean(Marcus)

S-1:. Man(Marcus)

S-2:. pompiean(Marcus)

③ CHAIN RULE:.

$P \Rightarrow Q \wedge Q \Rightarrow R \quad P \to Q, Q \to R \Leftrightarrow P \to R.$

$\therefore P \Rightarrow R$

Eg:- Marcus was a man and all men are Mortal

Marcus is Mortal.

$\forall x$: Man(x) ⇒ Mortal(x)

Mortal(Marcus)

4) SUBSTITUTIONAL METHOD:.

Substitution of variables by ground term.

Subst(θ/V), P)

Result of SUBST({John/x}, king(x) ∧ Greedy(x) ⇒ Evil(x)): king(John) ∧ Greedy(John) ⇒ Evil(John)

S1:- A greedy king is an evil king

S2:. John is a king.

S1:. $\forall x$ king(x) ∧ Greedy(x) ⇒ Evil(x)

S2:. king(John)

SUBST({John/x}, S1)

5) UNIVERSAL INSTANTIATION (UI)

Every instantiation of a universally qualified sentence is entailed by it:

$$\frac{\forall v \; \alpha}{\text{Subst}\{v/g\}\,\alpha}$$

for any variable v and ground term g

(subst(x, y) = substitution of y by x)

Eg:- $\forall x \; \text{King}(x) \wedge \text{Greedy}(x) \Rightarrow \text{Evil}(x)$

$\text{King}(\text{John}) \wedge \text{Greedy}(\text{John}) \Rightarrow \text{evil}(\text{John})$

$\text{king}(\text{Richard}) \wedge \text{Greedy}(\text{Richard}) \Rightarrow \text{evil}(\text{Richard})$

$\text{King}(\text{Father}(\text{John})) \wedge \text{Greedy}(\text{Father}(\text{John})) \Rightarrow \text{Evil}(\text{Father}(\text{John}))$

6) EXISTENTIAL INSTANTIATION

Eg: There exists a crown on John's head.

$\exists x \; \text{crown}(x) \wedge \text{on head}(x, \text{John})$

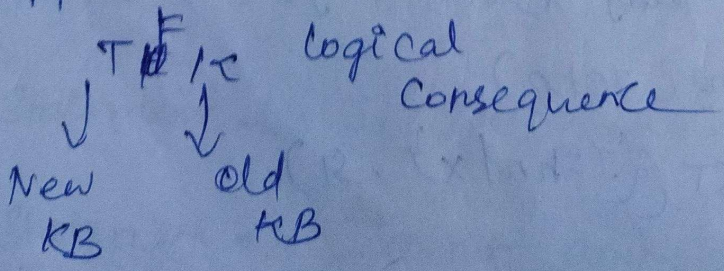
$\text{Crown}(C_1) \wedge \text{on Head}(C_1, \text{John})$

2M provided C_1 is a new constant symbol, called a skolem constant

REDUCTION CONTD.:

A ground sentence is entailed by a new KB if entailed by the original KB

All the logics applied in old KB can be applied to the new KB.



LIFTING AND UNIFICATION:

LIFTING:

They only make those substitutions that are required to allow particular inferences to proceed.

Eg: Generalised Modeus ponens.

UNIFICATION:

$\alpha = \{x/John, y/John\}$ works

unify $(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

	p	q
①	Knows(John, x)	Knows(John, Jane)
②	Know(John, x)	knows(y, OJ)
③	kno "	knows(y, Mother(x))
④	"	knows(x, OJ)

Predicate and no. of arguments should match.

① unify $(p, q) = \theta = \{x/Jane\}$

② $\theta = \{y/John \quad x/OJ\}$

③ $\theta = \{y/John \quad x/Mother(y)\}$

④ $\theta = \{x/John \quad x/OJ\}$ X

fails.

INFERENCE RULES:-

① Idempotent:-
$P \wedge P \Leftrightarrow P$
$P \vee P \Leftrightarrow P$

② Commutative:-
$P \vee Q \Leftrightarrow Q \vee P$
$P \wedge Q \Leftrightarrow Q \wedge P$

③ Associative:-
$P \wedge (Q \wedge R) \Leftrightarrow (P \wedge Q) \wedge R$
$P \vee (Q \vee R) \Leftrightarrow (P \vee Q) \vee R$

④ Demorgan's Rule:-
$\neg(P \vee Q) \Leftrightarrow \neg P \wedge \neg Q$
$\neg(P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$

⑤ Distributive:-
$P \vee (Q \wedge R) \Leftrightarrow (P \vee Q) \wedge (P \vee$
$P \wedge (Q \vee R) \Leftrightarrow (P \wedge Q) \vee (P \wedge$

⑥ Implication elimination:-
$P \rightarrow Q \Leftrightarrow \neg P \vee Q$

8) RESOLUTION:-

Consider the facts in KB.

① All people who are graduating are happy.

2) All happy people smile.

3) Someone is graduating

Apply the resolution theorem in the above KB and find solution for

"Is Someone Smiling?"

Convert all the statements / all the facts in the KB into clausal form.

STEP-1:- Clausal form

Negate the goal state and convert the result to a clausal form and add it to the set of clauses, already obtained.

Converting the facts to predicate logic.

$\forall x$ graduating $(x) \rightarrow$ happy (x)

$\forall x$ happy $(x) \rightarrow$ Smile (x)

$\exists x$ graduating

Goal State :- Someone is smiling.

$\exists x$ Smile (x) ← Goal State clausal form.

Convert the predicate logic to clausal form.

a) Eliminate the implications and biconditionals if any.

b) Move negation inverse ~~invert~~ inwards.

c) Standardise the Variable. (Renaming)

d) ~~Columnization~~, Skolemization. Existential instantiation.

e) Drop the universal quantifier.

f) Distributes And over OR and OR over AND

a) F1 :- $\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta$

$\forall x$ graduation $(x) \rightarrow$ happy (x)

CNF:- $\forall x \neg$ graduation $(x) \vee$ happy (x)

F2:- $\forall x$: happy $(x) \rightarrow$ smile (x)

CNF:- $\forall x \neg$ happy $(x) \vee$ smile (x)

F3 :- $\exists x$ graduating (x)

CNF3 :- $\exists x$ graduating (x)

F4 :- $\exists x$ Smile (x)

CNF4 :- $\exists x$ Smile (x)

b)

CNF1:- $\forall x \neg graduating(x) \vee happy(x)$

for this problem negation inwards is not necessary.

But you may have,

$\forall x : \neg [young(x) \wedge graduating(x)) \vee happy(x)$ ↳ demorgan's law

CNF1:- Same as before

$\forall x \neg graduating(x) \vee happy(x)$

Same as before all the CNF's

c) Same as before

d) Eliminate $\exists$

CNF1:- Same

CNF2:- Same

CNF3:- $\exists x\, Graduating(x)$

Eliminating Existential quantifier and add the column constant graduating

CNF3:- ~~Smile~~ (x_1)

CNF4:- $Smile(x_1)$

e)

CNF1:- $\neg graduating(x) \vee happy(x)$

CNF2:- $\neg happy(x) \vee smile(x)$

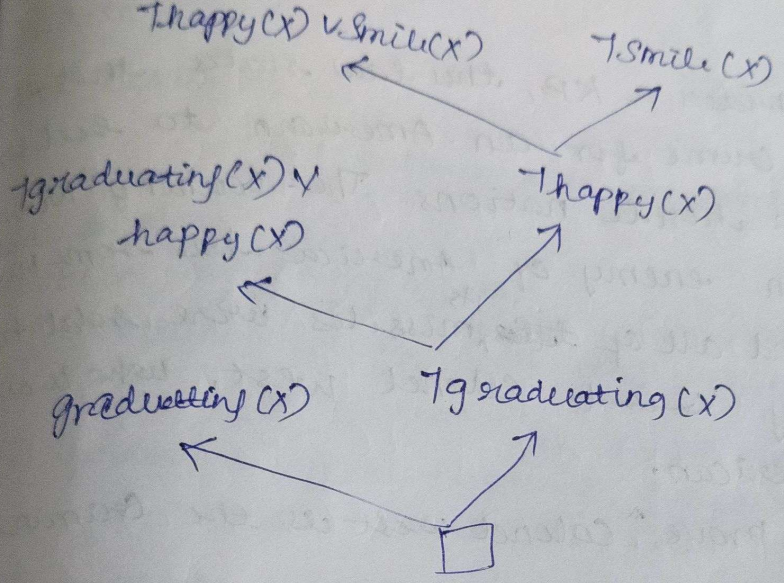
CNF3:- $graduating(x_1)$

CNF4:- $Smile(x_1)$

STEP-2:

Apply the resolution theorem to derive a conclusion for "Is someone smiling"

Now we have to devide the class one class will be the negation of the goal state



contradiction occurs.

STEPS:—

→ Hence our assumption ¬Smile(x) is wrong ∴ we conclude that "someone is smiling"

→ Select two clauses, one is negation of the goal state and the other is ~~clause~~ clausal form of fact in KB.

→ Resolve them together and check if the resultant is the empty clause.

→ Then the contradiction has been found.

→ If it is not the empty clause then add it to the set of clauses and apply the resolution procedure again.

REPRESENTING KNOWLEDGE USING RULES:

- Procedural Knowledge - consists of controlled info that has the instruc. to use the know.
- Descriptive Knowledge :- set of facts will describe about the objects in universe, descriptive in nature.

2 METHODS.

① FORWARD CHAINING / BACKWARD CHAINING:

Consider a KB, this law states that it is a crime for an American to sell weapons to hostile nations. The country "NONO" an enemy of America has some missiles and all of ~~these~~ its missiles were sold to it by ~~Kennet~~ Colonel west, who is an American.

Prove "Colonel west is the Criminal".

F-1 :-

MOD-4

PLANNING:-
The task of coming up with a sequence of actions that will achieve a goal is called planning

TYPES:

CLASSICAL:
We consider only environments that are fully observable, deterministic, finite, static and discrete.

NON-CLASSICAL:
partially observable or stochastic environment and ~~involves~~

PROBLEM PLANNING STEPS:.
① overwhelmed by irrelevant actions.
(book example)
② finding a good heuristic function.
trying to get the optimal Solution.
③ problem decomposition.
dividing the problems into sub-problems.
eg: tour planning

LANGUAGE OF PLANNING A PROBLEM:.
STRIPS - Stanford Research Institute Problem Solver.

COMPONENTS:.
→ Representation of States
→ " " goals
→ " " Actions

An action schema has three parts:.
A Action name & parameter
& precondition
+ effect

STRIPS EXAMPLE:-

AIR CARGO TRANSPORT

An air cargo transport problem involving loading and unloading cargo onto and off of planes and flying it from one place to another (place). The problem can be defined with three actions: load, unload, fly.

States:-

In(C, P)

C- cargo P-plane. is inside

At(X, a) X- Object X (either plane or cargo) is at airport a.

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))

Actions:-

① Load(C, P, a)

precond: At(C, a) ∧ At(P, a) ∧ Cargo(c) ∧ plane(p) ∧ Airport(a)

effect: (¬At(C, a) ∧ In(C, P))

Here the cargo will not be at the airport

the cargo will be at the plane.

② (Unload(C, P, a)) A(P, a) plane(P)

precond: In(C, P) ∧ Cargo(c) ∧ ~~airport(a)~~

effect: At(C, a) ∧ ¬In(C, P)

(3) Action(fly(p, from, to))

Precond: A(p, from) ∧ plane(p) ∧ Airport(from) ∧ Airport(to)

effect: ¬At(p, from) ∧ At(p, to)

Planning solution:

load(C1, P1, SFO)
load(C2, P2, JFK)
fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
unload(C1, P1, JFK)
unload(C2, P2, SFO)

COMPONENTS OF A PLANNING SYSTEM:

* Choose the best rule to apply the next rule based on the available guess.
* Apply the chosen rule to calculate the new problem definition.
* Find out when a soln. has been formed
* Detect dead ends.
* Find out when a near-perfect soln. is formed.

(4) TYPES OF PLANNING IN AI:

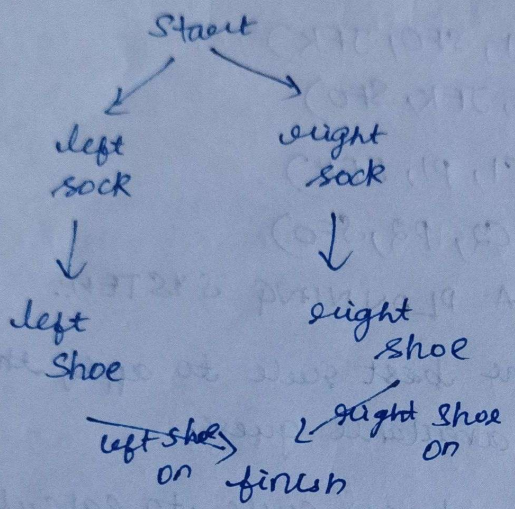
- Forward State Space planning (FSSP)
- Backward State Space planning (BSSP)
  - Process is called regression.
  - Subgoals should also be checked for consistency.

PLAN IN PARTIAL ORDERING:

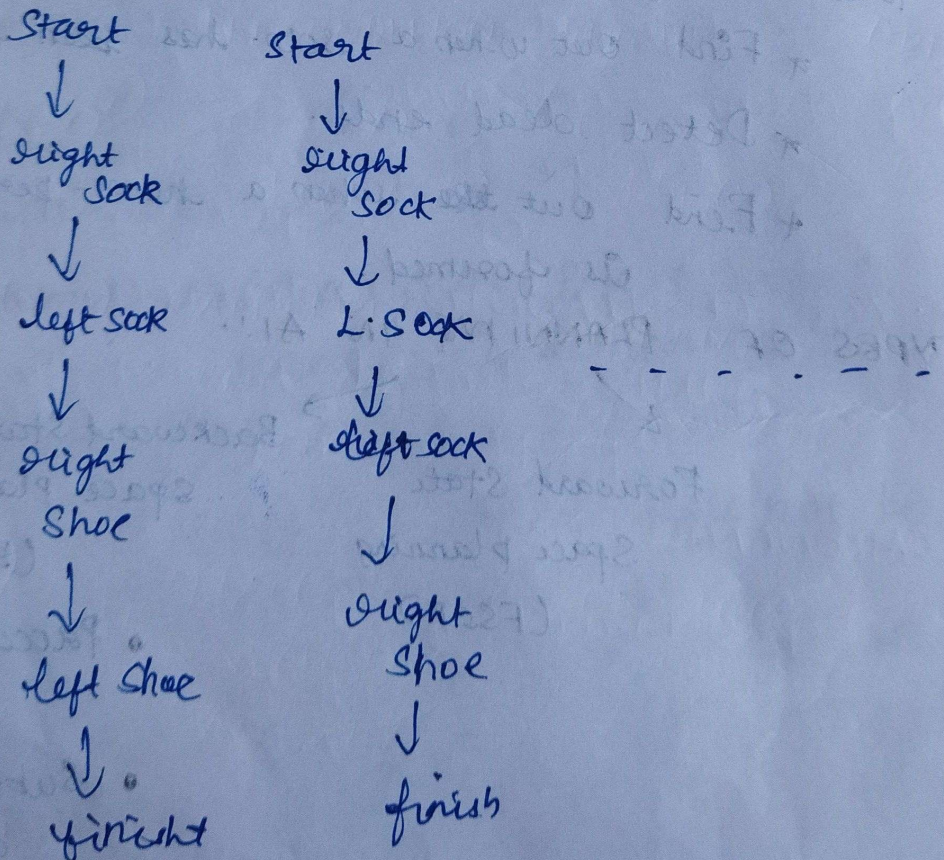
Actions
Orderings
links
Open preconditions.

↳ Not any particular sequence.

PARTIAL ORDER:



TOTAL ORDER PLANS:



PARTIAL ORDER PLANNING ALGORITHM: POP:-

① right sock
effect :- right sock ON

② right shoe
Preconditions :. {right sock on}
effect :. right shoe on.

③ FINISH
Preconditions :. {right shoe on,
left shoe on}

SET OF ORDERINGS :.

A ∠ B

RIGHT SOCK ∠ right shoe

left shoe ∧
right shoe ∠ finish

"A achieves p for B"

$A \xrightarrow{P} B$

Actions :. {rightSock, right shoe, leftSock, left shoe,
start, finish}

Orderings :. {right sock ∠ rightshoe, leftsock ∠ left shoe}

~~Right shoe → right shoe or Fi~~

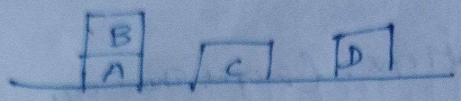
links :. { Right sock $\xrightarrow{\text{right sock on}}$ right shoe, leftsock $\xrightarrow{\text{left sock on}}$ leftshoe,

right Shoe $\xrightarrow{\text{right shoe on}}$ Finish,

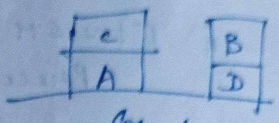
left shoe $\xrightarrow{\text{left shoe on}}$ Finish

GOAL STACK PLANNING - BLOCK WORLD PROBLEM

Initial State:- ON(B, A) ∧ ONTABLE(A) ∧ ONTABLE(C) ∧ ONTABLE(D) ∧ CLEAR(B) ∧ clear(C) ∧ clear(D) ∧ Armempty



Initial



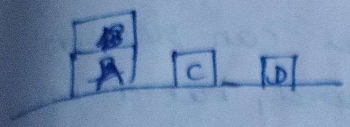
Goal

on(C, A) ∧ ON(B, D) ∧ ONTABLE(A) ∧ ONTABLE(D) ∧ Clear(B) ∧ clear(C) ∧ ARMEMPTY

Actions:-

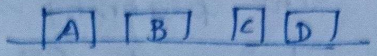
- Stack
- unstack
- pickup
- putdown

OPERATORS	PRECONDITION	DELETE	ADD
Stack(x,Y)	clear(Y) ∧ HOLDING(x)	clear(Y) holding(x)	ARMEMPTY ON(x,Y)
unstack(X,Y)	ARMEMPTY ∧ ON(X,Y) ∧ CLEAR(X)	ARMEMPTY ∧ ON(X,Y)	HOLDING(X) ∧ CLEAR(Y)
pickup(X)	CLEAR(X) ∧ ONTABLE(X) ∧ ARMEMPTY	ONTABLE(X) ∧ ARMEMPTY	HOLDING(X
putdown(X)	HOLDING(X)	HOLDING(X)	ONTABLE(X) ∧ ARMEMPTY



① UNSTACK(B,A):

Effect:

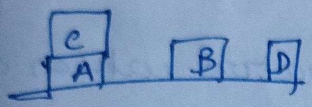


Pre-condition:

On(B,A) clear B

---

② STACK(C,A)



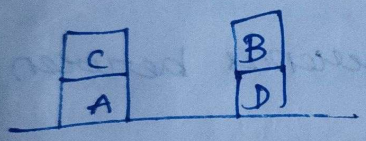
Pre-conditions:

clear(A) holding(C)

effect:

ARMEMPTY, ON(C,A)

③ STACK(B,D)



Preconditions:

Clear(D) holding(B)

effect:

Armempty, on(B,D)

MOD-V

MEANS ENDS ANALYSIS

The planning strategies can reason either in forward or backward, but a mixture of the two directions is appropriate for solving a complex large problem.

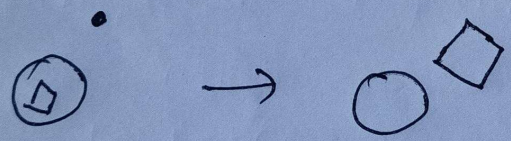
First evaluate the difference between initial state and final state.

Select the various operators which can be applied for each difference

Apply the operator at each difference,

OPERATOR SUBGOALING:

we detect the differences between current and goal state.



Initial goal

STEP-1: remove the dot Action: delete

STEP-2: Move the diamond outside

Action: Move

STE-3: Enlarge the size of the diamond

Action: Expand.

STEPS:

STEP-1: Evaluate the initial state comparing to the goal state and list the differences.

STEP-2: The dot is outside the circle but in the goal state dot is not there.

STEP-3: The diamond is inside the circle but in the goal state diamond is outside and its size is bigger.

STEP-4: Apply the delete operator dot is deleted.

STEP-5: Apply the move operator

STEP-6: Apply the expand operator.

GRAPH PLANNING:

Planning graph can be used to give better heuristic estimates.

ALGORITHM:- GRAPH PLAN

level-0 represents initial state

Each level consists of a set of literals and a set of actions.

HAVE CAKE AND EAT TOO PROBLEM:

Init (Have (Cake))

Goal (Have (Cake) ∧ Eaten (Cake))

Action (Eat (Cake))

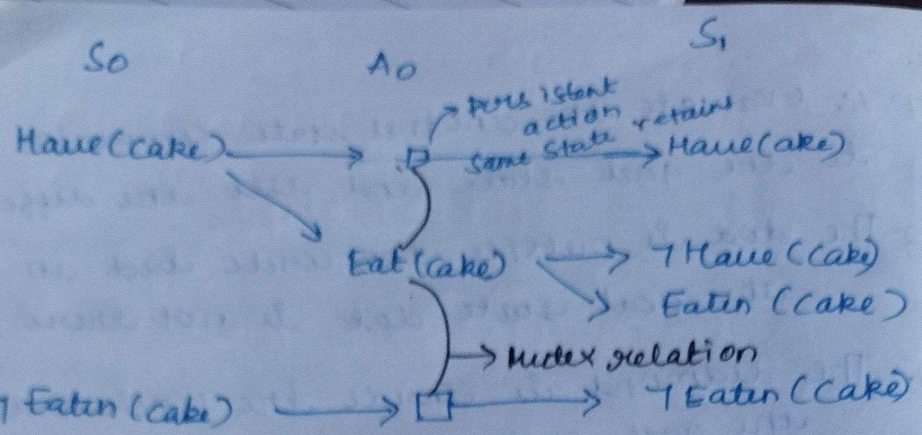
Precond: Have (Cake))

effect: ¬Have (Cake) ∧ Eaten (Cake))

Action (Bake (Cake))

Precond: ¬Have (Cake)

effect: Have (Cake)



CONDITIONS FOR MUTEX RELATION:.

→ Inconsistent effect – one action negates an effect of the other.

Eg:. Eat(cake) and the persistence of have(cake) have inconsistent effects because they disagree on the effect have(cake)

→ INTERFERENCE:. One of the effects of one action is the negation of a precondition of the other.

Eg:. Eat(cake) interferes with the persistence of Have(cake) by negating its pre-condition.

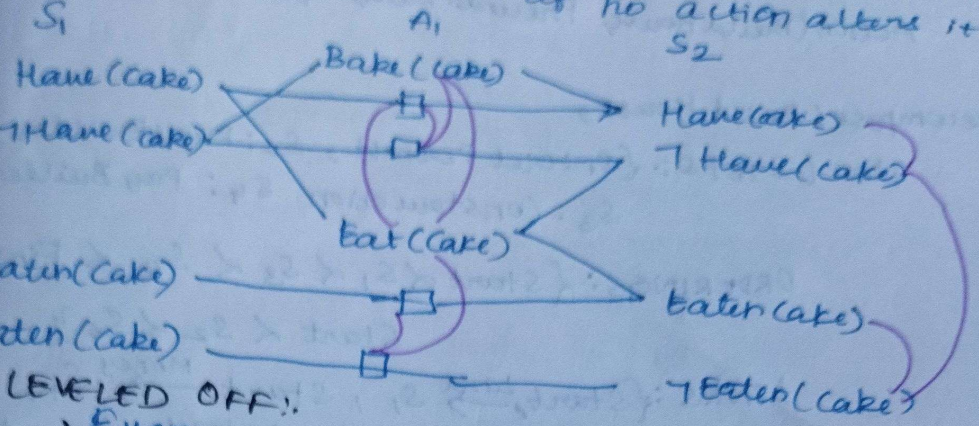
→ COMPETING NEEDS:.

One of the preconditions of one action is mutually exclusive with a precondition of the other.

Eg: Bake(Cake) and Eat(cake) are mutex because they compete on the value of the have(cake) precondition

PERSISTENCE ACTIONS:

Allows a literal to remain true from one situation to the next if no action alters it.



LEVELED OFF:

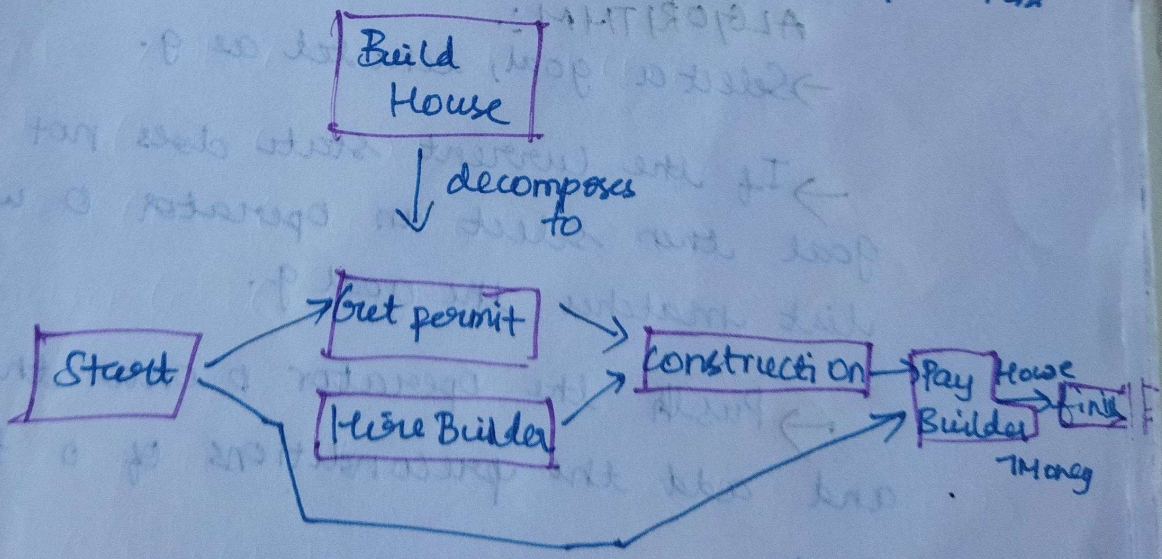
→ Every subsequent level will be identical, so further expansion is unnecessary.

HIERARCHICAL TASKS:

→ Planning method is based on hierarchical task networks OR HTN's

→ This approach we take combines ideas from both partial-order and the area are known as "HTN Planning"

→ Refined by applying action decomposition



Action (Buy Land, precond: Money, effect: Land ∧ ¬Money)
Action (Get Loan, Precond: Good Credit, Effect: Money ∧ Mortage)
ACTION (Build House, Precond: Land, effect: House)

Decompose (Build House,
Plan (STEPS: {S_1: Get Permit, S_2: Hire Builder,
S_3: Construction S_4: Pay Builder}
ORDERINGS: {Start ≺ S_1 ≺ S_3 ≺ S_4 ≺ Finish,
Start ≺ S_2 ≺ S_3}
LINKS: {Start $\xrightarrow{Land}$ S_1, Start $\xrightarrow{Money}$ S_4,
$S_1 \xrightarrow{Permit} S_3$, $S_2 \xrightarrow{Contract} S_3$, $S_3 \xrightarrow{House\ Built} S_4$,
$S_4 \xrightarrow{House}$ Finish, $S_4 \xrightarrow{\neg Money}$ Finish}))

NON-LINEAR PLANNING:
→ This planning technique involves creating a goal stack and incorporating it into the search space of all potential sub-goal goal sequences.

→ Essential problem-solving approach that deals with complex situations, where the solution is not easily predicted.

ALGORITHM:
→ Select a goal, denoted as g.

→ If the current state does not match the goal then select an operator O whose add-list matches the goal g.

→ Push the operator O onto the Openstack and add the preconditions of O to the set of goals.

→ Continue this process while all preconditions of the operator on the top of the openstack are satisfied in the current state.

→ pop the operator o from the top of the OpenStack apply it to the current state, and add it to the plan.

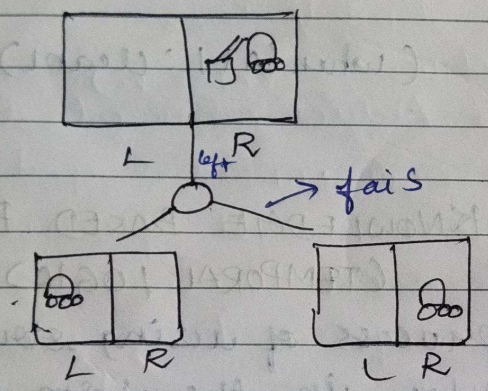
PAGE NO.
DATE / /

CONDITIONAL PLANNING:
used only in uncertain environment

UNCERTAINITY:
* Initial state
* Goal state
* Plans may have branches which may not be linear; using conditional steps.

CONDITIONS:
if <test> then plan A else plan B

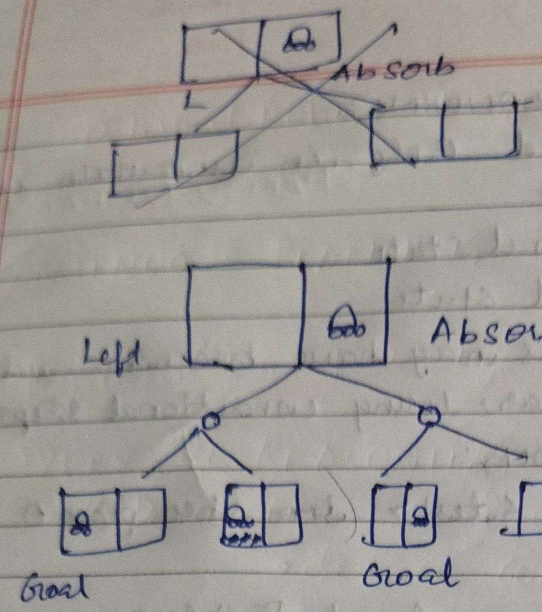
Vaccum cleaner Agent Problem:
① Sometimes action fails.
Action (left, precond: At R, Effect: At L ∨ At R)



② conditional effects:
Action (Absorb, precond: At L ∨ At R, effects:
(when At L: clean L ∧
when At R: clean R)

③ Moving may dump dirt in destination room only when room is clean.

Move Action (left, precond: At R,
effects: At L ∨ (when clean L:
¬clean L)



Action (Absorb, precond: , effect:
((when (At R: clean R) V when (clean R:
¬clean R)) ∧
(when (At L: clean L) V when (clean L:
¬clean L)))

KNOWLEDGE BASED PLANNING (TEMPORAL LOGIC)

The process of using structured knowledge often in the form of rules, facts and logical relationships to guide decision making and achieve specific goal.

Branch of logic that extends a classical logical by introducing time based elements.

Allows for reasoning, how things change over time (schedules), automated systems.

TEMPORAL OPERATORS:

Before, After, during, until.

Consider a robo delivery agent who is at location C, and needs to deliver a package from location A to location B and deliver before 12Noon.

Initial state :- At(Robo, location C) ∧
At(package, location A) ∧
Time : 9AM

Goal state :- Delivered(package, location B) ∧
Time : 12.00 PM.

Plan:- ① Action
Move(Robo, location C, location A)
Precond: At(Robo, location C)
effect :- ¬At(Robo, location C) ∧
At(Robo, location A)
temporal Constraint: This action must complete before the package arrives.

② Action(Pick up(Robo, package location A),
Precond:- At(Robo, location A) At(package, location A)
effects:- holding(Robo, package location A)
temporal constraint:- Before Robo leaves the location A, this action must Complete.

③ Action(Move(Robo, location A, location B

PAGE No.
DATE : / /

(4) Deliver (Robo, Package, Location B)

2M CONTINUOUS PLANNING:
Replanning:
Multi Agent Planning

TYPES OF REASONING:
(1) Deductive reasoning.
- deduce conclusion from the given fact.

(2) Inductive Reasoning:
- Generalise something based on the limited info.

(3) Abductive: (observe)
- Abduct several things from the environment & and searches for the most plausible explanation.

(4) Common sense:
- informal reason
- prior we know some data.
- learned through personal experience.

(5) Monotonic:
- Even though when the new data is added the conclusion will not change.

(6) NON-Monotonic:
- The conclusion changes on adding a new data.