Planning in the block world

Triangle table

Precondition list

Delete list

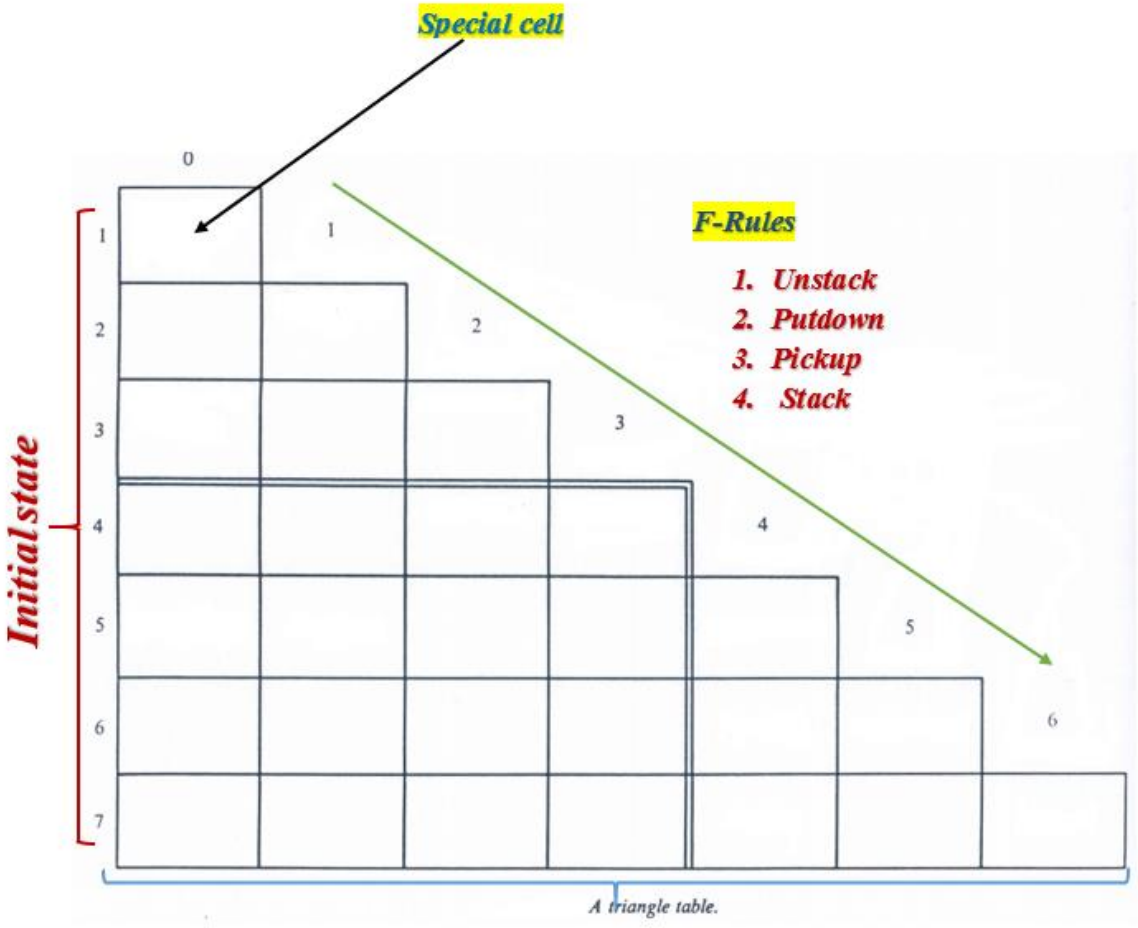
1) **pickup**(x)
   P & D: $ONTABLE(x)$, $CLEAR(x)$, $HANDEMPTY$
   A: $HOLDING(x)$

Add list

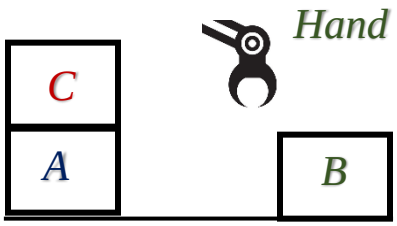
2) **putdown**(x)
   P & D: $HOLDING(x)$
   A: $ONTABLE(x)$, $CLEAR(x)$, $HANDEMPTY$

3) **stack**(x,y)
   P & D: $HOLDING(x)$, $CLEAR(y)$
   A: $HANDEMPTY$, $ON(x,y)$, $CLEAR(x)$

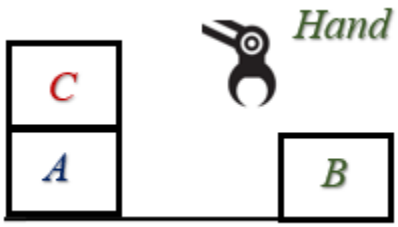
4) **unstack**(x,y)
   P & D: $HANDEMPTY$, $CLEAR(x)$, $ON(x,y)$
   A: $HOLDING(x)$, $CLEAR(y)$


Special cell
0
1
2
3
4
5
6
7
1
2
3
4
5
6
F-Rules
1. Unstack
2. Putdown
3. Pickup
4. Stack
Initial state
A triangle table.
Goal State

Example



the initial state



the goal state

[On (B, C) ^ On (A, B)]

States predicates

- *Clear (B) , On (C, A), On-table (A)*
- *Clear (C), Hand empty (HE), On-table(B)*

Using triangle table to solve

Generate plan

1. *Unstack (C , A)*
2. *Putdown(C)*
3. *Pickup(B)*
4. *Stack(B,C)*
5. *Pickup(A)*
6. *Stack(A,B)*

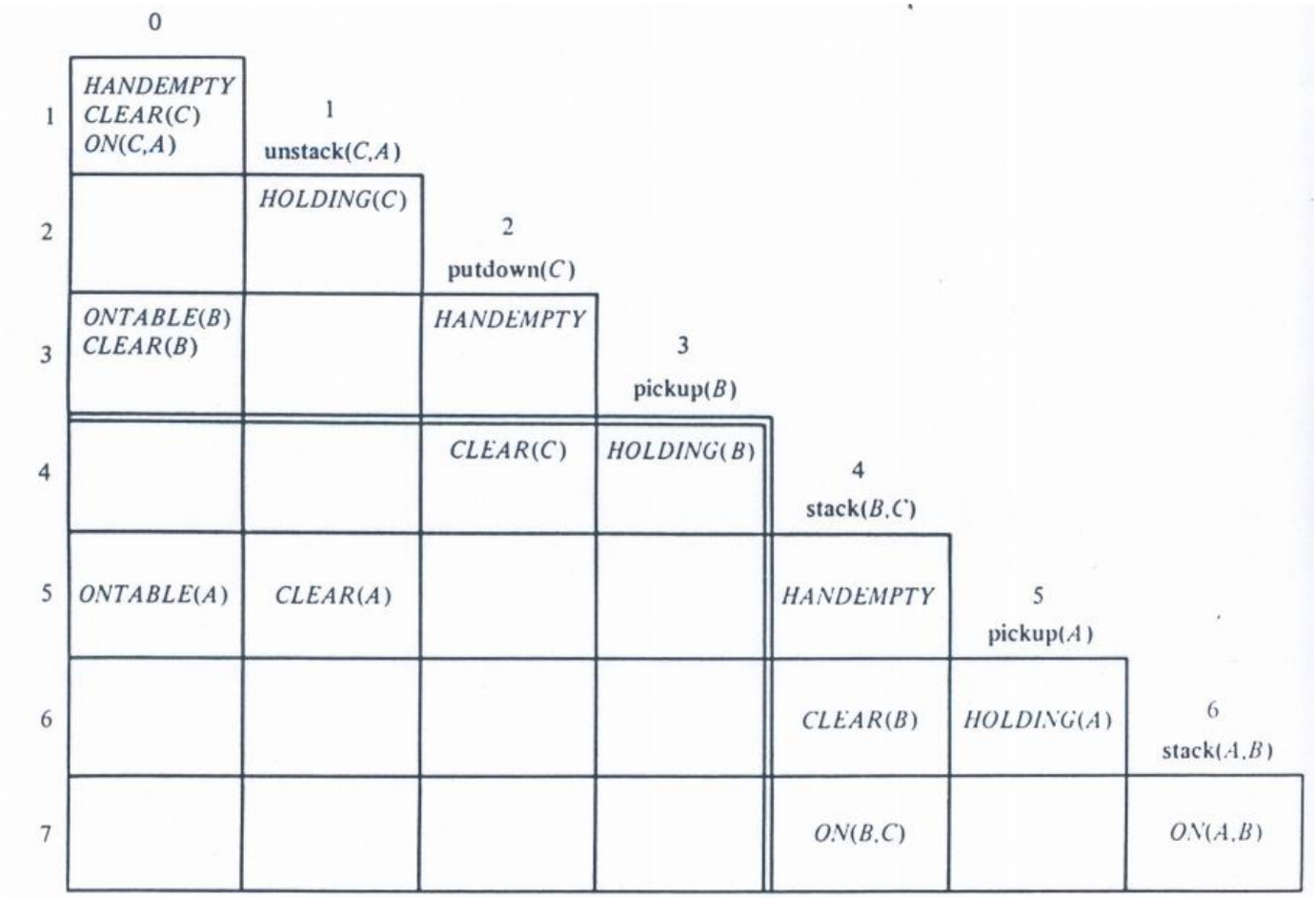
Applying F- Rules

1) **pickup**(x)
   P & D: $ONTABLE(x)$, $CLEAR(x)$, $HANDEMPTY$
   A: $HOLDING(x)$

2) **putdown**(x)
   P & D: $HOLDING(x)$
   A: $ONTABLE(x)$, $CLEAR(x)$, $HANDEMPTY$

3) **stack**(x,y)
   P & D: $HOLDING(x)$, $CLEAR(y)$
   A: $HANDEMPTY$, $ON(x,y)$, $CLEAR(x)$

4) **unstack**(x,y)
   P & D: $HANDEMPTY$, $CLEAR(x)$, $ON(x,y)$
   A: $HOLDING(x)$, $CLEAR(y)$


0
1
HANDEMPTY
CLEAR(C)
ON(C,A)
1
unstack(C,A)
2
HOLDING(C)
2
putdown(C)
3
ONTABLE(B)
CLEAR(B)
HANDEMPTY
3
pickup(B)
4
CLEAR(C)
HOLDING(B)
4
stack(B,C)
5
ONTABLE(A)
CLEAR(A)
HANDEMPTY
5
pickup(A)
6
CLEAR(B)
HOLDING(A)
6
stack(A,B)
7
ON(B,C)
ON(A,B)

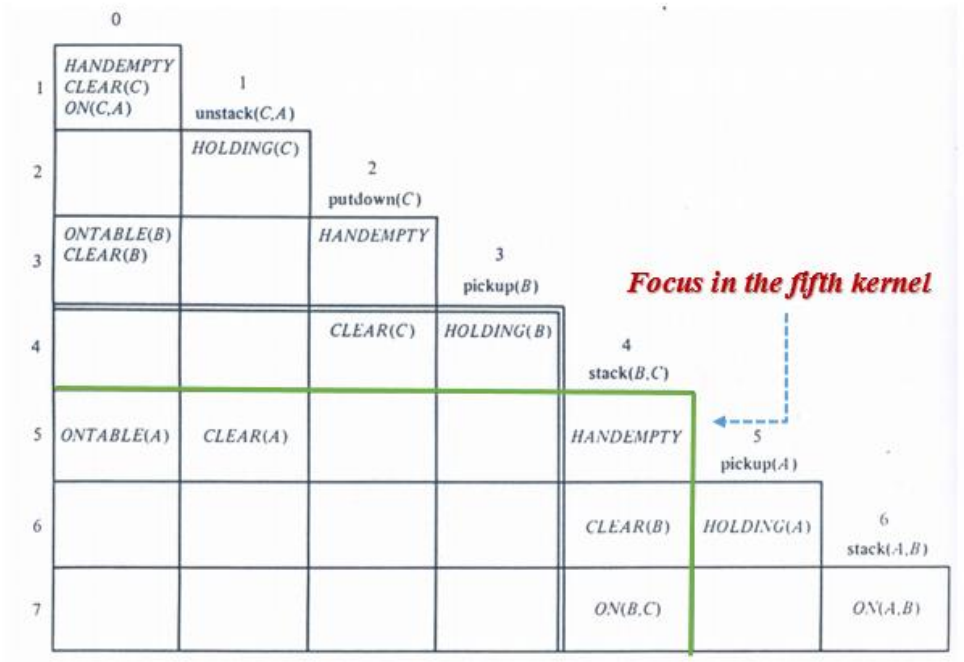
Triangle Table

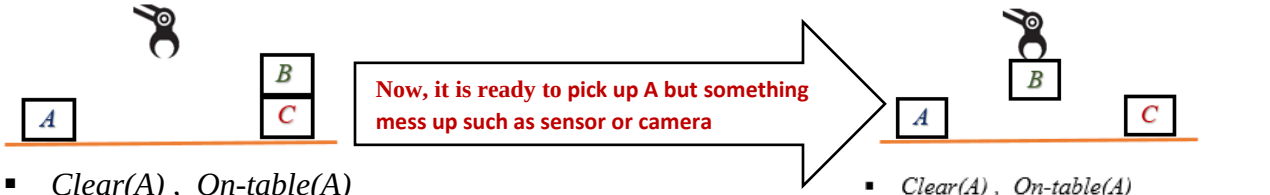
Triangle table represents set of snapshots that correspond to state description that result from executing plan

Concept of kernel

What is the i-th kernel?

the i-th kernel is the part of the triangle table that is below i-th row (include this row)and to the left the i-th column



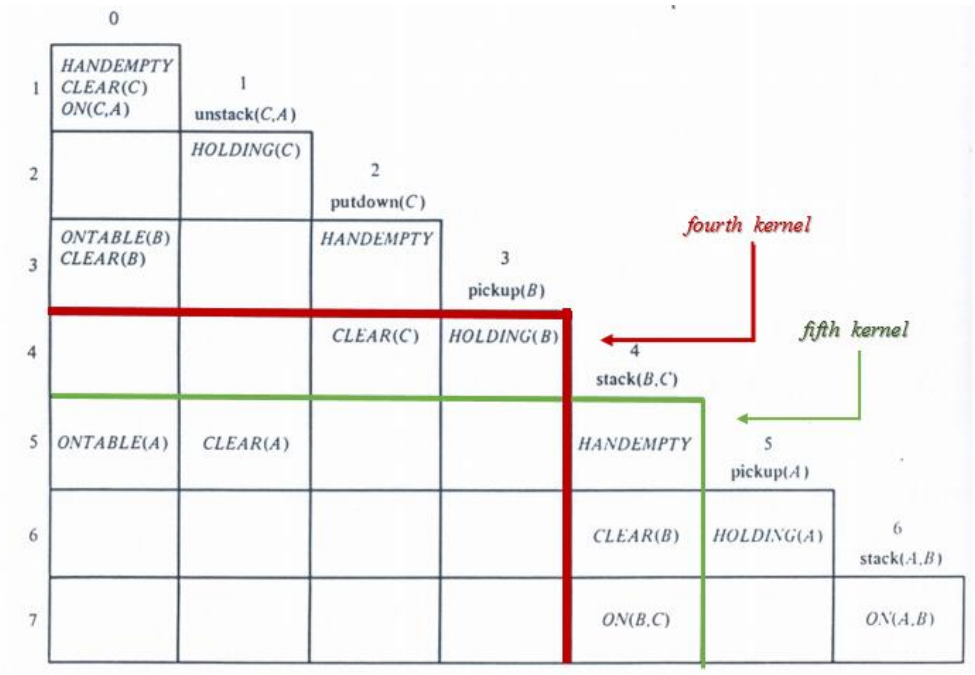


- *Clear(A) , On-table(A)*
- *On(B,C) , On-table(C)*
- *Hand empty (HE)*

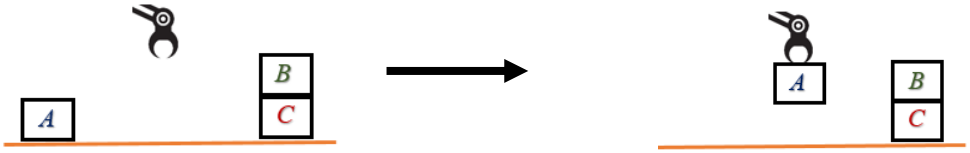
- *Clear(A) , On-table(A)*
- *Clear(C) , On-table(C) , Holding(B)*

Question

How to re-plan? Back track through sets of kernel staring where you are and keep checking backward (i-th -1)

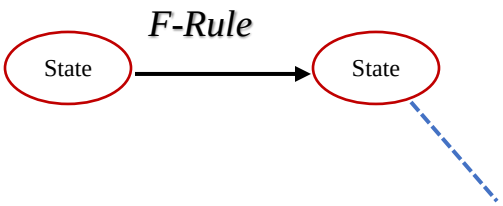


Now, restack the (B, C) then try to repeat



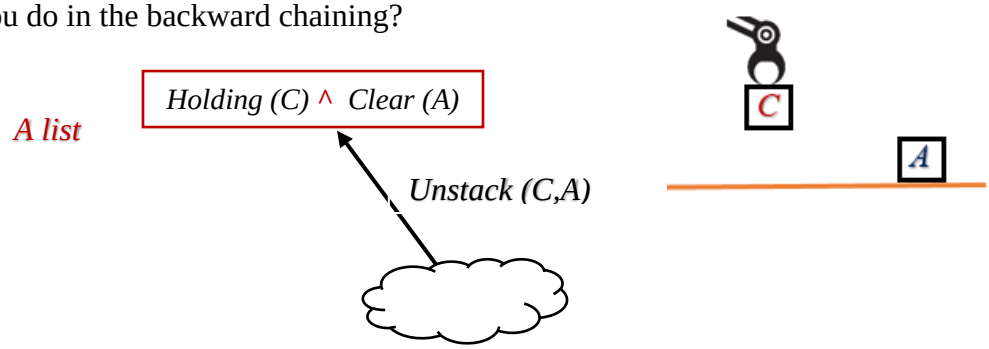
Plan generation

Using forward chaining mechanism



- The search process of plan would be starting with initial state and figuring out the sequences of these operation.

- Can you do in the backward chaining?



- What is state from which could arrive at (this particular description) though use one of rules
- Can regress the state though one of the rule!
- Using backward chaining more efficient finding plan

The F-rule (Robot action) can be used in backward chaining way as following

- The goal expression must contain a predicate that unifies with one of the predicates in the add list of the rule.
- The sub-goal expression is created by regressing the unmatched predicates in the goal expression though the match instance of the precondition list of the rule

The regression procedure

Let R(Q , Fu) be the regression if Q na Fu

Fu has the P, D , A lists

F is Rule

U is unifiction instance

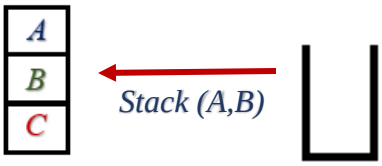
If *Q,.u* ∈ *Au* ***then*** *R(Q, Fu) = T (true)*

else ***If*** *Q.u* ∈ *Du* ***then*** *R(Q,Fu) = F (false)*

else R (Q , Fu)= Qu

Example

Regress



goal state [on(A,B) ^ on(B,C)]

stack (x,y) P, D : holding (x), clear (y)

A : HE , On (x,y) Clear(x)

Applying the rule (stack) to reach the goal state

- *First, knowing if it is possible to use this rule or not*
- *Second, which state could arrive to the target state by using this rule*

There are two cases

1) ***Case 1 : {< x , A > , < y , B >}***
2) ***Case 2: {< x , B > , < y , C >}***

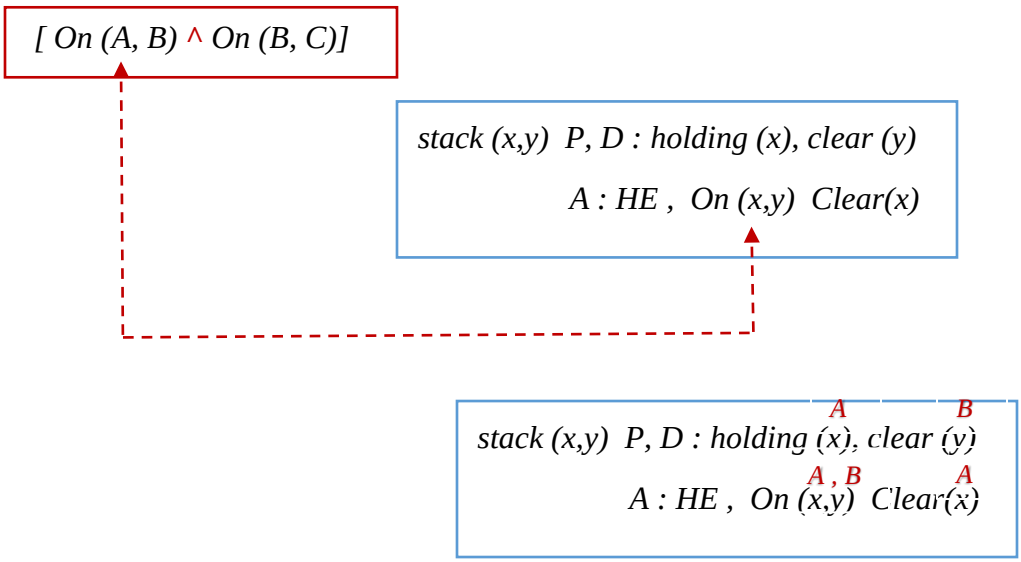
Case 1:

{< x , A > , < y , B >}

- *Regress: On (B, C) via case1 , rule stack(A, B)*
- *Procedure: On (B, C) is it in the Add list? → NO*

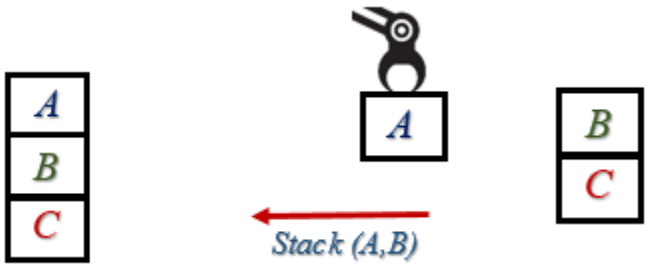
is it in the Delete list ? → NO

else *On (B, C)*



In sub-goal

- On (B,C) ^ Holding (A) ^ Clear (A)

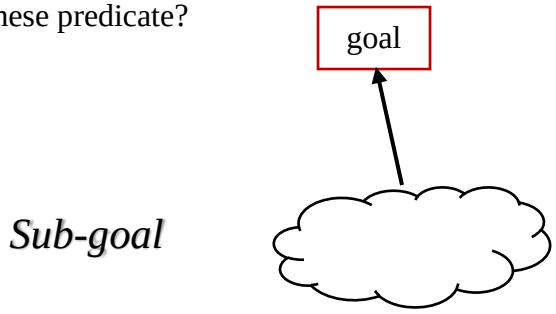


Regression

What is the processes?

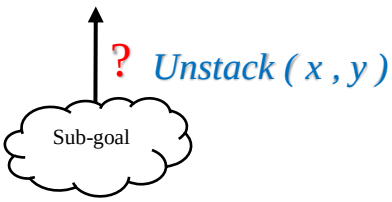
There is goal state and ask question "what is the sub-goal?" which

What is the list of these predicate?



Example 2

goal clear(A)



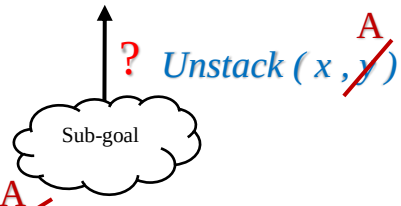
Using < y , A >

Unstack (x , y) P&D : HE , Clear (x) , On (x , y)

A : holding (x) , Clear (y)

Substitute each y by A

goal clear(A)



Unstack (x , ~~y~~) P&D : HE , Clear (x) , On (x , ~~y~~)

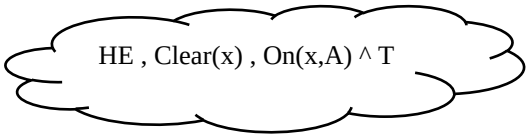
A A

A : holding (x) , Clear (~~y~~)

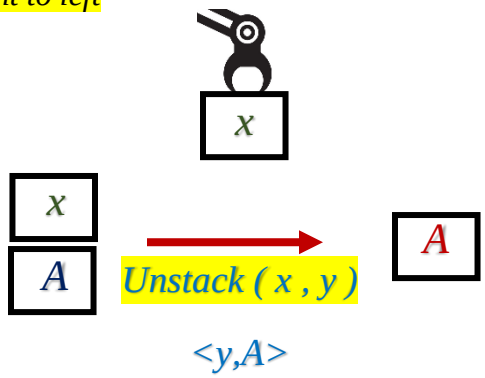
A

What should the list of predicate which arrive at goal state or " Clear (A) "

Regression Procedure: holding (x) ∈ Au " Add list " ? → Yes



Think from right to left



Example 3

Sub-goal clear (A) ^ HE

? *Unstack (x , y)* *<y,A>*

Unstack (x , y) P&D : HE , Clear (x) , On (x , y)

A : holding (x) , Clear (y)

HE , Clear (x), On(x,A) ^ F

Substituting < y , A >

Unstack (x , ~~y~~ A) P&D : HE , Clear (x) , On (x , ~~y~~ A)

A : holding (x) , Clear (~~y~~ A)