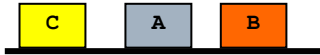


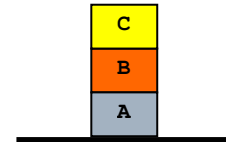
Progression Algorithm

Planning Problem's Start and Goal State

Initial State



Goal State



Initial State of the algorithm:

$\text{On}(A, \text{Table}) \wedge \text{On}(B, \text{Table}) \wedge \text{On}(C, \text{Table}) \wedge$
 $\text{Block}(A) \wedge \text{Block}(B) \wedge \text{Block}(C) \wedge \text{Clear}(A) \wedge$
 $\text{Clear}(B) \wedge \text{Clear}(C)$

Actions Available:

1. (Action :**Move**(p, x, y)

Precond: $\text{On}(p, x) \wedge \text{clear}(y) \wedge \text{clear}(p)$

Effect : $\text{On}(p, y) \wedge \text{clear}(x) \wedge \neg \text{clear}(y)$
 $\wedge \neg \text{On}(p, x)$

Moves Block p which is on top of x to top of y.

2. (Action :**MovetoTable**(p, x)

Precond: $\text{On}(p, x) \wedge \text{clear}(p)$

Effect : $\text{On}(p, \text{Table}) \wedge \text{clear}(x)$
 $\wedge \neg \text{On}(p, x)$

Moves Block p which is on top of x to top of Table.

Why have a separate action **MovetoTable**?

- Action **Move(p, x, y)** does not maintain Clear property when x or y is the table.

E.g.

When x=Table,

this action has the effect `Clear(Table)`, but the table should not become clear.

When y =Table,

This action has the precondition `Clear(Table)`, but the table does not have to be clear to move a block onto it.

Interpretation of `Clear(b)`: "there is a clear space on b to hold a block". So `Clear(Table)` is always true.

Actions that can be applied to this initial State are:

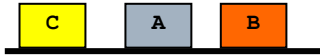
1. `Move(A, Table, B)`
2. `Move(A, Table, C)`
3. `Move(B, Table, A)`
4. `Move(B, Table, C)`
5. `Move(C, Table, A)`
6. `Move(B, Table, A)`
7. `MovetoTable(A, Table)`
8. `MovetoTable(B, Table)`
9. `MovetoTable(B, Table)`

So we see that this algorithm
-considers a lot of **irrelevant actions**.
-High Branching Factor.

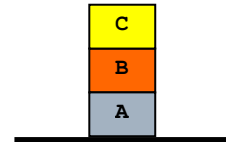
Regression Algorithm

Planning Problem's Start and Goal State

Initial State



Goal State



Initial State of the Algorithm:

$\text{On}(A, \text{Table}) \wedge \text{On}(B, A) \wedge \text{On}(C, B) \wedge \text{Clear}(C)$

Actions Available:

3. (Action :**Move**(p, x, y)

Precond: $\text{On}(p, x) \wedge \text{clear}(y) \wedge \text{clear}(p)$

Effect : $\text{On}(p, y) \wedge \text{clear}(x) \wedge \neg \text{clear}(y) \wedge \neg \text{On}(p, x)$

Moves Block p which is on top of x to top of y.

4. (Action :**MovetoTable**(p, x)

Precond: $\text{On}(p, x) \wedge \text{clear}(p)$

Effect : $\text{On}(p, \text{Table}) \wedge \text{clear}(x) \wedge \neg \text{On}(p, x)$

Moves Block p which is on top of x to top of Table.

Possible Predecessor States:

Let start with the first part of conjunction from initial state of the algorithm.

On (B, A)

To get this as an effect of one of the available actions.

We see only action $\text{Move}(B, x, A)$ is applicable
And its preconditions are :

$\text{On}(B, x) \wedge \text{clear}(B) \wedge \text{clear}(A)$

So the predecessor state would become:

$\text{On}(B, x) \wedge \text{clear}(B) \wedge \text{clear}(A) \wedge \text{On}(C, B)$

So this action cannot be applied as it is not consistent.

So Let us consider next part of conjunction

On (C, B)

Again to get this as an effect of one of the available actions, we see only action $\text{Move}(C, x, B)$ is applicable

Its preconditions are:

$\text{On}(C, x) \wedge \text{clear}(B) \wedge \text{clear}(C)$

So the predecessor state would become:

$\text{On}(C, x) \wedge \text{clear}(B) \wedge \text{clear}(C) \wedge \text{On}(B, A) \wedge \text{On}(A, \text{Table})$.

So we see that this algorithm

-does not consider irrelevant and inconsistent actions.

-Low Branching Factor.