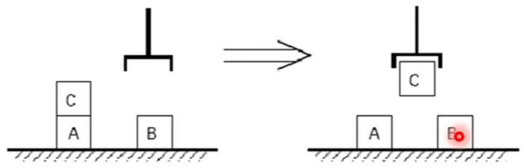
○

What is Planning in AI?

- The Planning in Artificial Intelligence is about the **decision making tasks performed by the robots or computer programs to achieve a specific goal.**
- The execution of planning is about **choosing a sequence of actions.**
- Planning refers to the **process of computing several steps** of a problem-solving procedure before executing any of them.
- **Planning is arranging a sequence of actions to achieve a goal.**

EXAMPLE-BLOCKS WORLD (BW) PLANNING

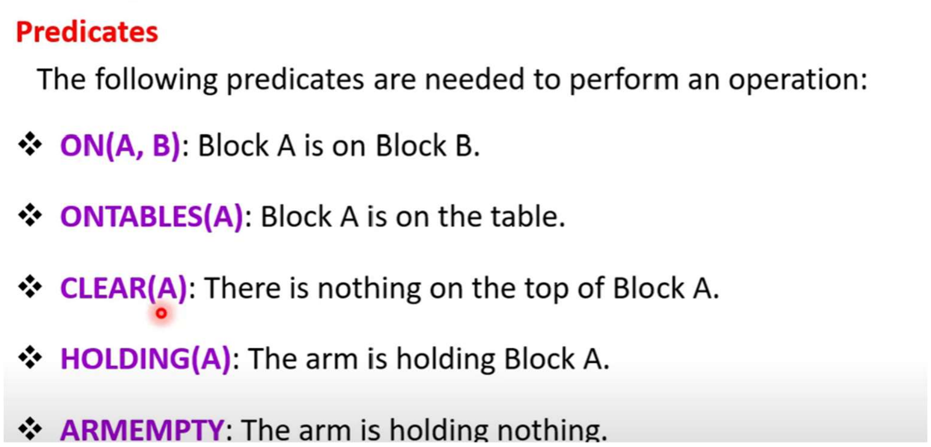
- The Blocks world consists of:



→A flat surface such as a table top
→An adequate set of identical blocks which are identified by letters.
→The blocks can be stacked one upon another.
→There is a robot arm that can manipulate the blocks.
→The robot can hold one block at a time and only one block can be moved at a time.
→Any number of blocks can be on the table.

The **actions** it can perform include

- **UNSTACK(A,B)**:
  - remove block A from block B.
  - The arm must be empty
  - block A must have no blocks on top of it.
- **STACK(A,B)**:
  - put block A on block B.
  - The arm must already be holding A
  - the surface of B must be clear.
- **PICKUP(A)**:
  - pickup block A from the table.
  - The arm must be empty and there must be nothing on top of A.
- **PUTDOWN(A)**:
  - put block A on the table.
  - The arm must have been holding block A.



Predicates

The following predicates are needed to perform an operation:

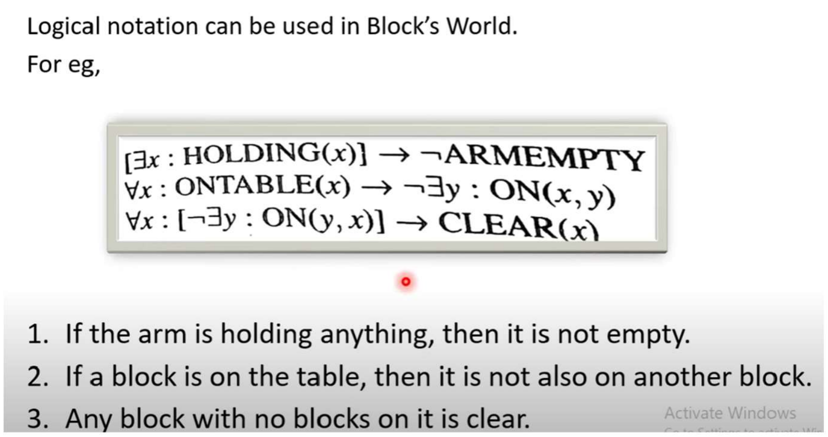
❖ **ON(A, B)**: Block A is on Block B.

❖ **ONTABLES(A)**: Block A is on the table.

❖ **CLEAR(A)**: There is nothing on the top of Block A.

❖ **HOLDING(A)**: The arm is holding Block A.

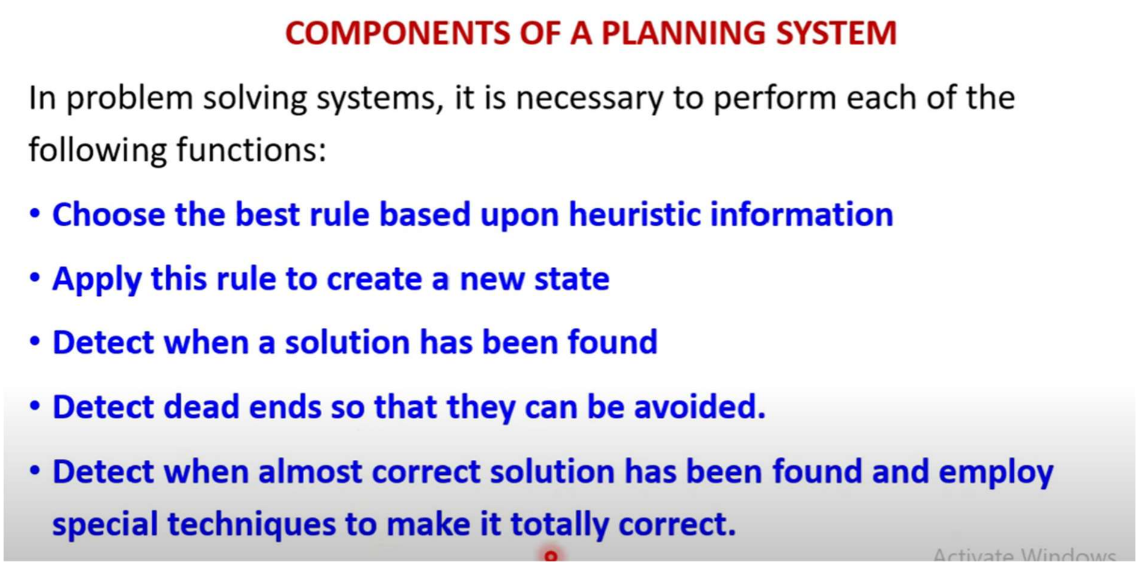
❖ **ARMEMPTY**: The arm is holding nothing.



Logical notation can be used in Block's World.
For eg,

$$[\exists x : \text{HOLDING}(x)] \rightarrow \neg\text{ARMEMPTY}$$
$$\forall x : \text{ONTABLE}(x) \rightarrow \neg\exists y : \text{ON}(x, y)$$
$$\forall x : [\neg\exists y : \text{ON}(y, x)] \rightarrow \text{CLEAR}(x)$$

1. If the arm is holding anything, then it is not empty.
2. If a block is on the table, then it is not also on another block.
3. Any block with no blocks on it is clear.



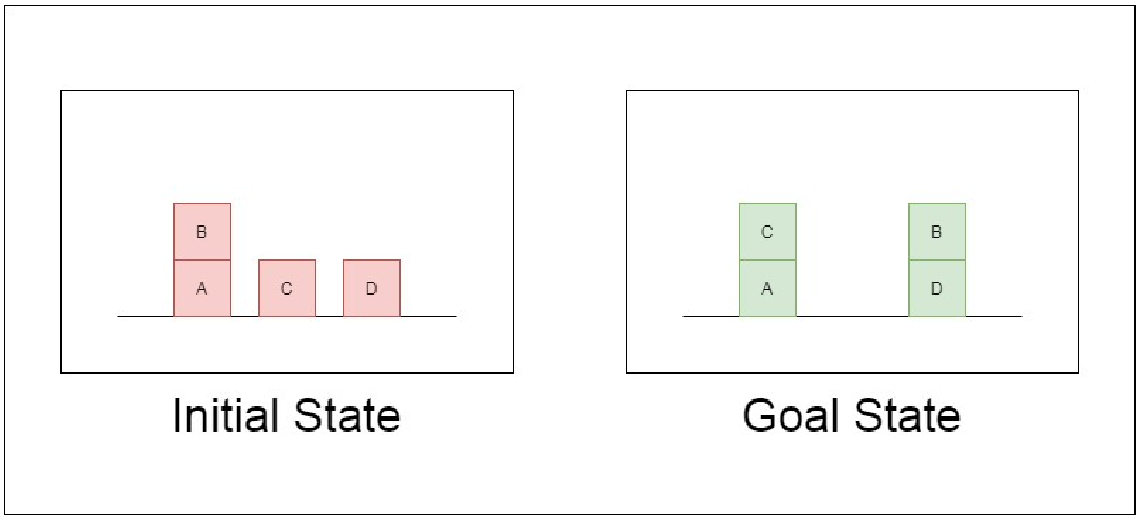
COMPONENTS OF A PLANNING SYSTEM

In problem solving systems, it is necessary to perform each of the following functions:

- **Choose the best rule based upon heuristic information**
- **Apply this rule to create a new state**
- **Detect when a solution has been found**
- **Detect dead ends so that they can be avoided.**
- **Detect when almost correct solution has been found and employ special techniques to make it totally correct.**

Goal Stack Planning for Blocks World Problem

Implementing Goal Stack Planning for the given configuration of Blocks World Problem



Blocks World Problem — Initial State and Goal State for this article

In this medium article, we will take look at the Blocks World Problem and implement Goal Stack Planning to solve the same.

What is Blocks World Problem?

This is how the problem goes — There is a table on which some blocks are placed. Some blocks may or may not be stacked on other blocks. We have a robot arm to pick up or put down the blocks. The robot arm can move only one block at a time, and no other block should be stacked on top of the block which is to be moved by the robot arm.

Our aim is to change the configuration of the blocks from the Initial State to the Goal State, both of which have been specified in the diagram above.

What is Goal Stack Planning?

Goal Stack Planning is one of the earliest methods in artificial intelligence in which we work **backwards from the goal state to the initial state.**

We start at the goal state and we try fulfilling the preconditions required to achieve the initial state. These preconditions in turn have their own set of preconditions, which are required to be satisfied first. We keep solving these “goals” and “sub-goals” until we finally arrive at the Initial State. **We make use of a stack to hold these goals that need to be fulfilled as well the actions that we need to perform for the same**.

Apart from the “Initial State” and the “Goal State”, we maintain a **“World State”** configuration as well. Goal Stack uses this world state to work its way from Goal State to Initial State. World State on the other hand starts off as the Initial State and ends up being transformed into the Goal state.

At the end of this algorithm we are left with an empty stack and a set of actions which helps us navigate from the Initial State to the World State.

Representing the configurations as a list of “predicates”

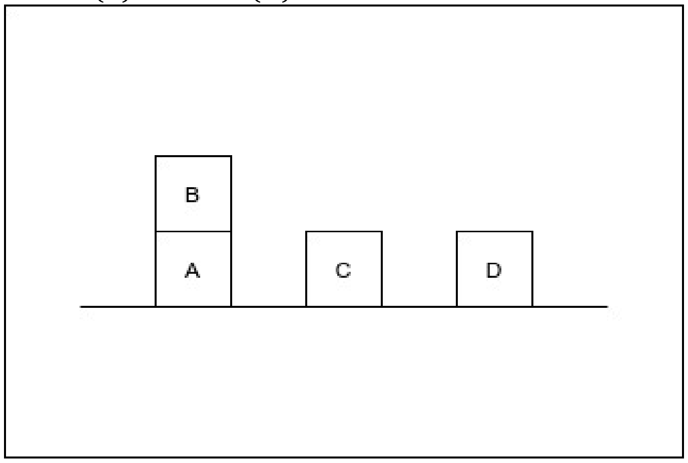
Predicates can be thought of as a statement which helps us convey the information about a configuration in Blocks World.

Given below are the list of predicates as well as their intended meaning

1. ON(A,B) : Block A is on B
2. ONTABLE(A) : A is on table
3. CLEAR(A) : Nothing is on top of A
4. HOLDING(A) : Arm is holding A.
5. ARMEMPTY : Arm is holding nothing

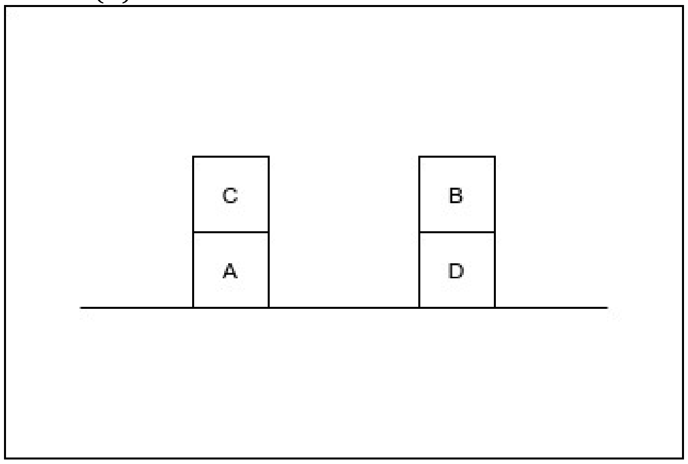
Using these predicates, we represent the Initial State and the Goal State in our example like this:

Initial State — ON(B,A) ∧ ONTABLE(A) ∧ ONTABLE(C) ∧ ONTABLE(D) ∧ CLEAR(B) ∧ CLEAR(C) ∧ CLEAR(D) ∧ ARMEMPTY



Initial State

Goal State — ON(C,A) ∧ ON(B,D) ∧ ONTABLE(A) ∧ ONTABLE(D) ∧ CLEAR(B) ∧ CLEAR(C) ∧ ARMEMPTY



Goal State

Thus a configuration can be thought of as a list of predicates describing the current scenario.

"Operations" performed by the robot arm

The Robot Arm can perform 4 operations:

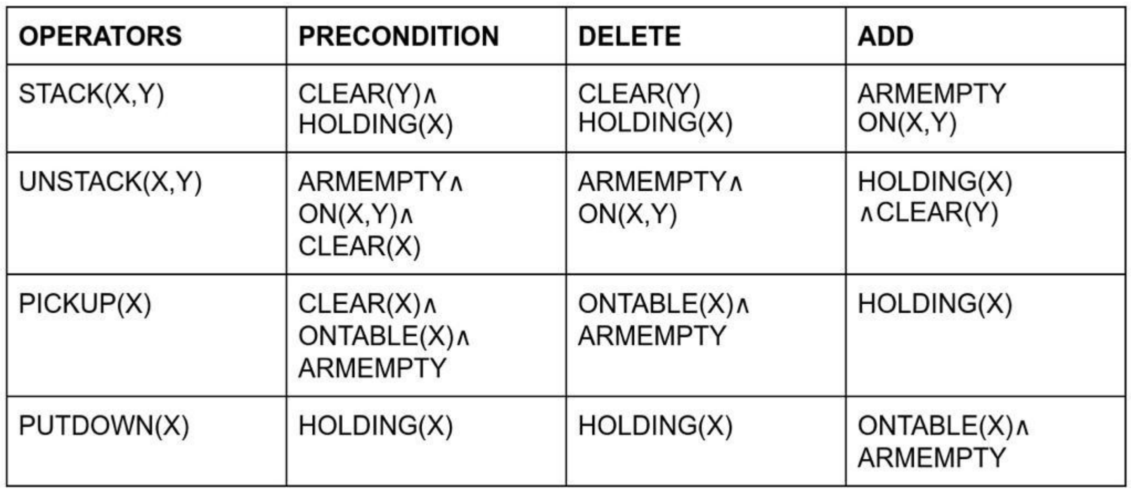
1. STACK(X,Y) : Stacking Block X on Block Y

2. UNSTACK(X,Y) : Picking up Block X which is on top of Block Y
3. PICKUP(X) : Picking up Block X which is on top of the table
4. PUTDOWN(X) : Put Block X on the table

All the four operations have certain preconditions which need to be satisfied to perform the same. These preconditions are represented in the form of predicates.

The effect of these operations is represented using two lists **ADD** and **DELETE**. DELETE List contains the predicates which will cease to be true once the operation is performed. ADD List on the other hand contains the predicates which will become true once the operation is performed.

The Precondition, Add and Delete List for each operation is rather intuitive and have been listed below.



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

Operations performed by the Robot Arm

For example, to perform the **STACK(X,Y)** operation i.e. to Stack Block X on top of Block Y, No other block should be on top of Y **(CLEAR(Y))** and the Robot Arm should be holding the Block X (**HOLDING(X)**).

Once the operation is performed, these predicates will cease to be true, thus they are included in **DELETE List** as well. (Note : It is not necessary for the Precondition and DELETE List to be the exact same).

On the other hand, once the operation is performed, The robot arm will be free (**ARMEMPTY**) and the block X will be on top of Y (**ON(X,Y)**).

The other 3 Operators follow similar logic, and this part is the cornerstone of Goal Stack Planning.