Primitive goal based ideas

- Once you have the gold, your goal is to get back home

$$\forall s \;\; Holding(Gold, s) \Rightarrow GoalLocation([1,1], s)$$

- How to work out actions to achieve the goal?
  - Inference: Lots more axioms. Explodes.
  - Search: Best-first (or other) search. Need to convert KB to operators
  - Planning: Special purpose reasoning systems (next class)

Planning

Initial condition in KB:

$At(Agent, [1,1], S_0)$

$At(Gold, [1,2], S_0)$

Query: $\text{ASK}(KB, \exists\, s \;\; Holding(Gold, s))$

i.e., in what situation will I be holding the gold?

Answer: $\{s/Result(Grab, Result(Forward, S_0))\}$

i.e., go forward and then grab the gold

This assumes that the agent is interested in plans starting at S_0 and that S_0 is the only situation described in the KB

Generating action sequences

Represent plans as action sequences $[a_1, a_2, \ldots, a_n]$

$PlanResult(p, s)$ is the result of executing p in s

Then the query $\text{Ask}(KB, \exists p \;\; Holding(Gold, PlanResult(p, S_0)))$
has the solution $\{p/[Forward, Grab]\}$

Definition of $PlanResult$ in terms of $Result$:

$\forall s \;\; PlanResult([], s) = s$ [ ] = empty plan

$\forall a, p, s \;\; PlanResult([a|p], s) = PlanResult(p, Result(a, s))$

Recursively continue until it gets to empty plan [ ]

Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner

Summary

First-order logic:

- objects and relations are semantic primitives
- syntax: constants, functions, predicates, equality, quantifiers

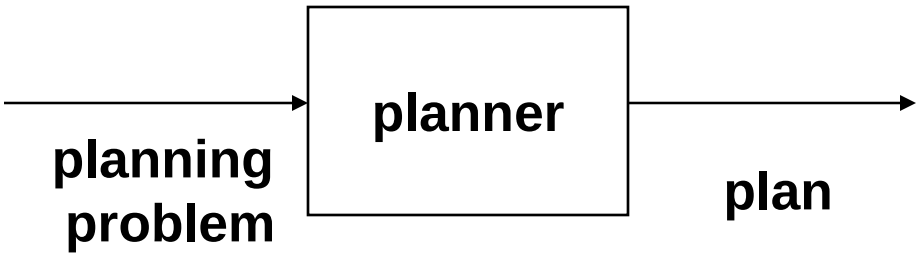
Increased expressive power: sufficient to define wumpus world

Situation calculus:

- conventions for describing actions and change in FOL
- can formulate planning as inference on a situation calculus KB

What is planning?

- A *planner* is a system that finds a sequence of actions to accomplish a specific task
- A planner synthesizes a plan



What is planning? (cont'd)

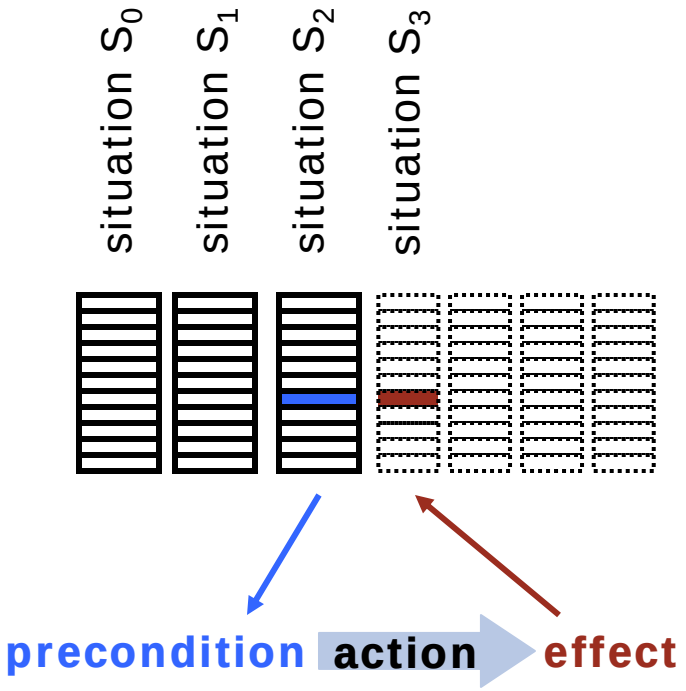
- The main components of a *planning problem* are:
  - a description of the starting situation (*the initial state*), (the world now)
  - a description of the desired situation (*the goal state*), (how should the world be)
  - the actions available to the executing agent (*operator library*, a.k.a. *domain theory*) (possible actions to change the world)
- Formally, a (classical) planning problem is a triple: <I, G, D>

where, I is the initial state,
G is the goal state, and
D is the domain theory.

Situation calculus

universe is defined as sequence of 'situations'

- 'actions' are like inferences:
  - preconditions – required facts in current situation
  - effects – facts that are true in subsequent situation if **action** is applied

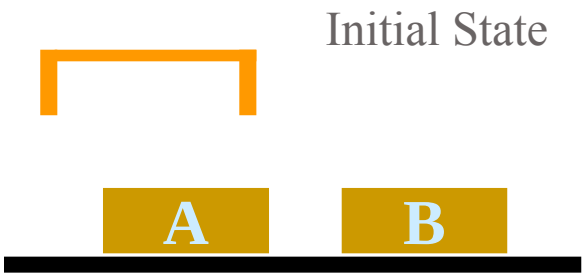


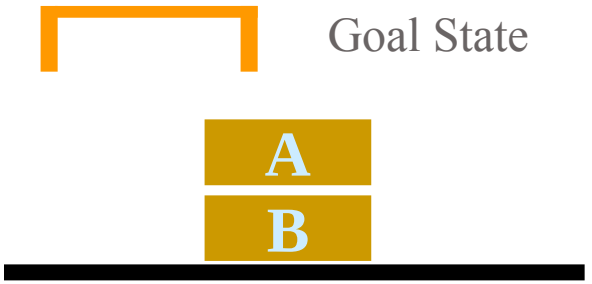
Introduction to Planning

- Plan: a **sequence of steps** to achieve a goal.
- Problem solving agent knows: **Actions**, **states**, **goals** and **plans**.
- Planning is a **special case** of problem solving: reach a **state** satisfying the **requirements** from the current state using **available actions**.

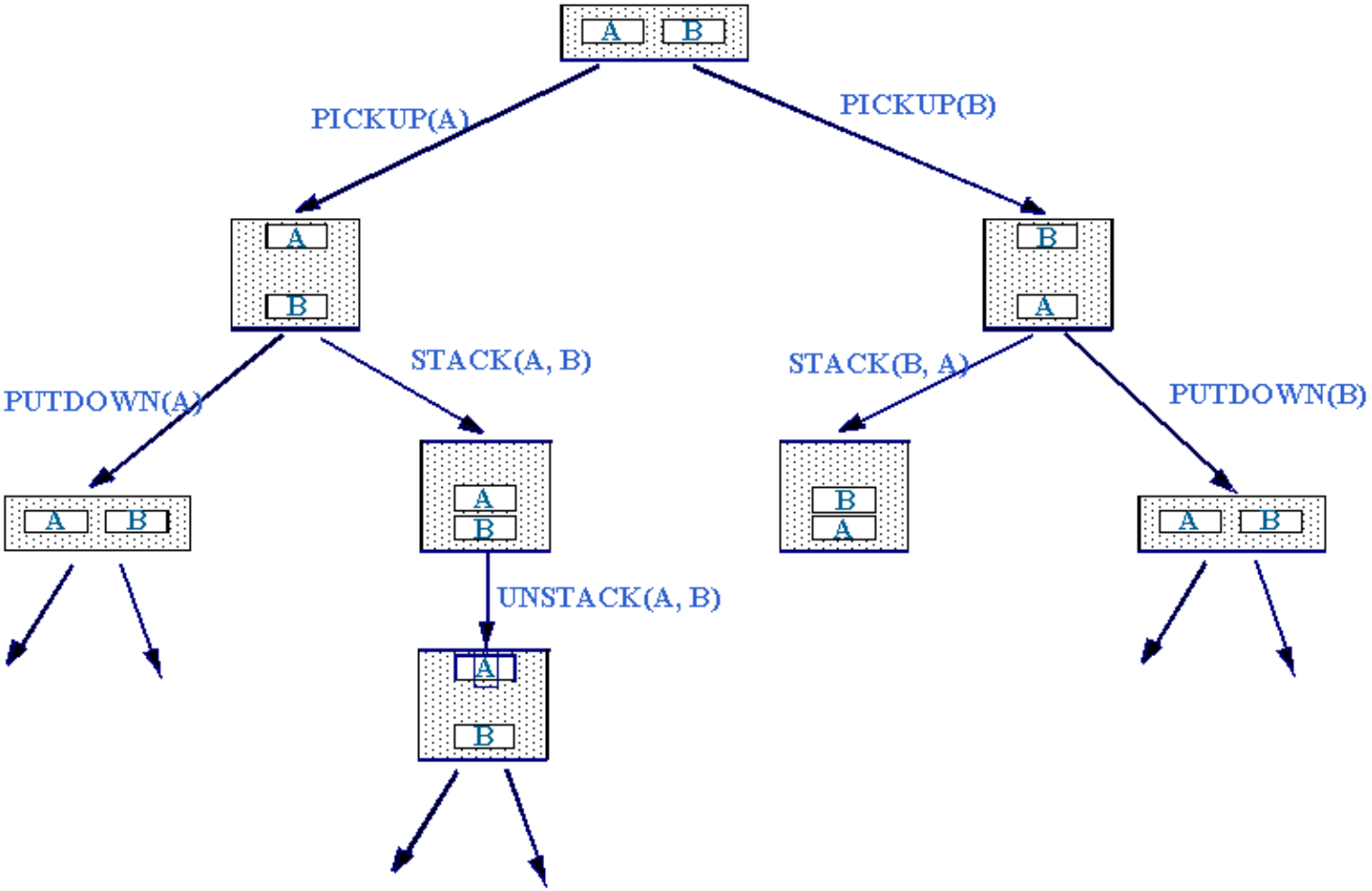
Blocks World

- Action: PickUp(X)
  - Preconditions:
  - X on table, hand empty, X free
- Action: PutDown(X)
  - Precondition:
  - X in hand
- Action: Stack(X,Y)
  - Preconditions:
  - X in hand, y free
- Action:Unstack(X,Y)
  - Preconditions:
  - X free, X on Y, hand free





Blocks World (State Space)



Assumptions of the "Standard" AI Planning Paradigm

- There is a **single causal agent** and this agent is the planner.
- The planner is given a **well-defined goal** which remains **fixed** over the course of planning.
- The planner is assumed to have **functionally complete** and **accurate knowledge** of the starting situation.
- The planner is assumed to **possess** the **knowledge** required to accurately model the world.
- The planner is assumed to **possess** the **resources** (time and memory) required to use this model to reason about the possible worlds associated with different courses of action that might be pursued.

STRIPS - Linear Planner

- First planner developed by SRI, stands for STanford Research Institute Problem Solver.
- In STRIPS notation, a model of the world is just a **list of variables free atomic propositions** that hold in the world.
- **Operators involving variables** are called **operator schemas**.
- The following expresses an initial state in the block world:

  <on(a,t), on(b,a),clear(b),on(c,t),clear(c))>

- It is assumed that **anything not mentioned** in the description of the initial state of the world is **false**.

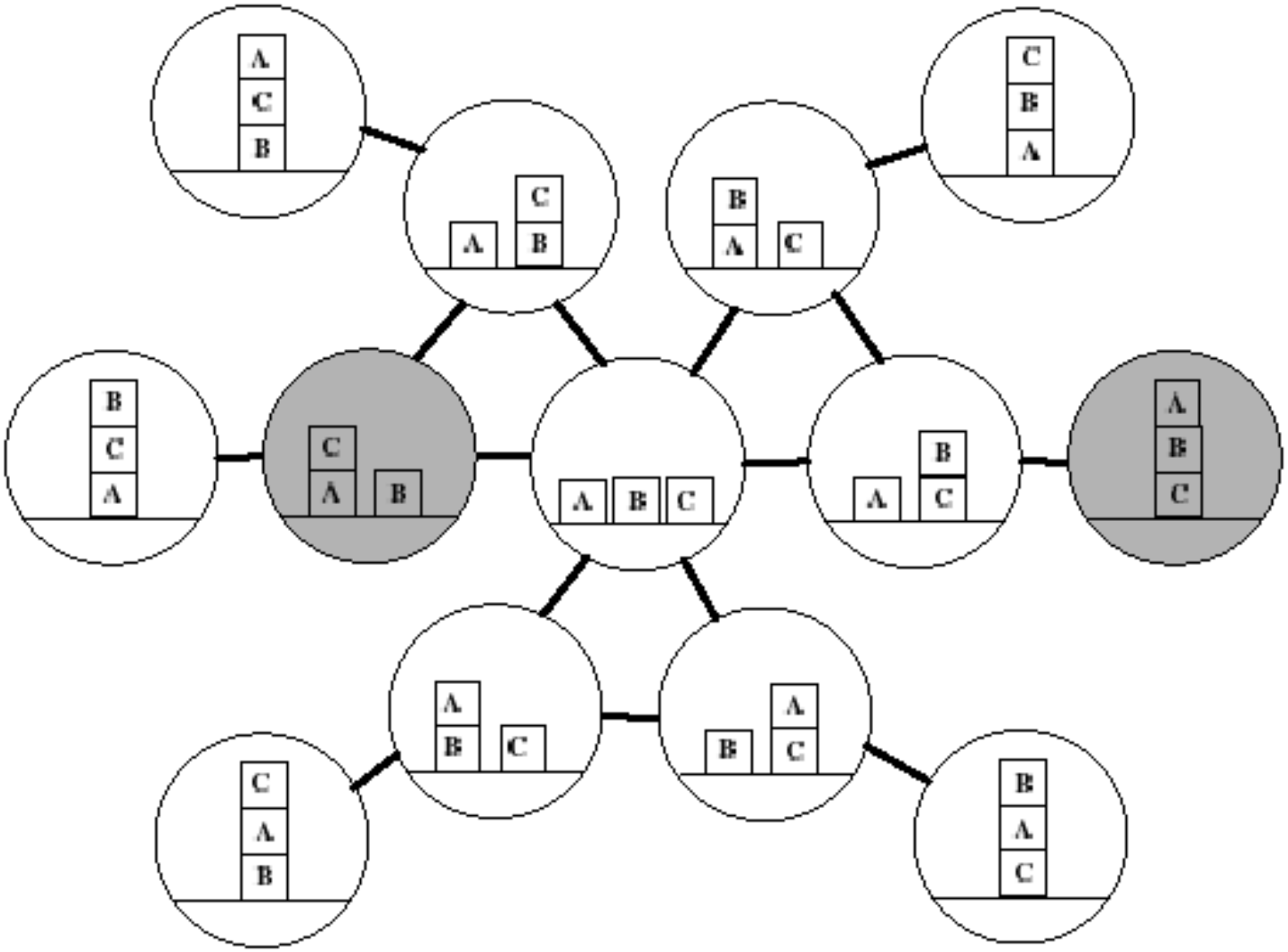
Characteristics of *classical planners*

- They operate on basic STRIPS actions
- Important assumptions:
  - the agent is the only source of change in the world, otherwise the environment is static
  - all the actions are deterministic
  - the agent is omniscient: knows everything it needs to know about start state and effects of actions
  - the goals are categorical, the plan is considered successful iff all the goals are achieved

A state space algorithm for STRIPS operators

- Search the space of situations (or states). This means each node in the search tree is a state.
- The root of the tree is the start state.
- Operators are the means of transition from each node to its children.
- The goal test involves seeing if the set of goals is a subset of the current situation.

State Space: Blocks World



STRIPS

- The **description** of the goal state is again a **list of atomic propositions** where all variables are interpreted existentially.

- The goal state of plan, for example, will be given by such a description. An example goal state in the block world is:

<on(X,c), on(c,t)>

This means that some block should be on c, which is itself directly on the table

STRIPS (cont)

- The main element of the language is the operator description, which has **three parts**:

  (1) the **action** name, which may be parametrized

  (2) the **precondition**, which is a conjunction of positive literals

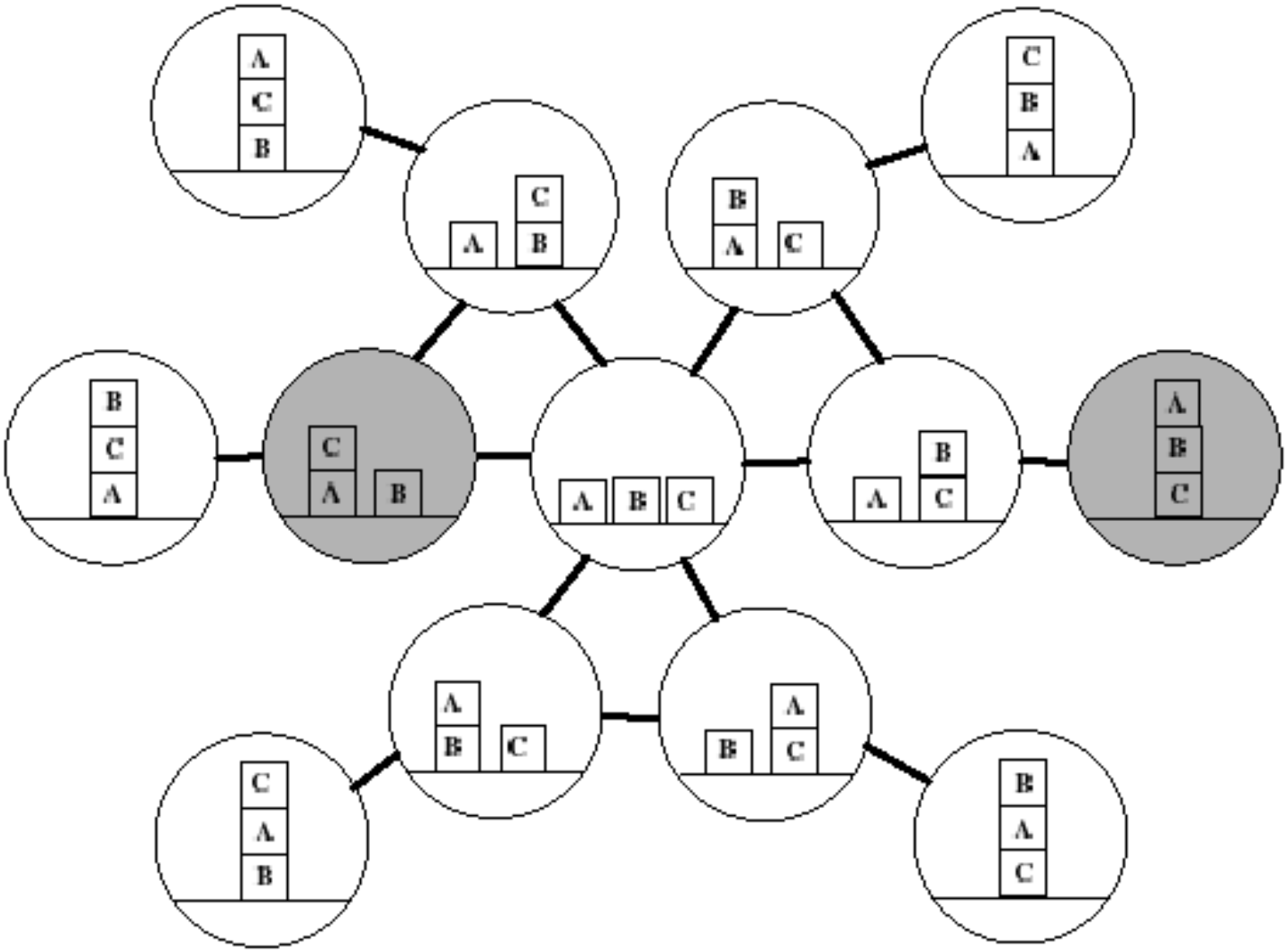
  (3) the **effect**, which is a conjunction of positive and/or negative literals

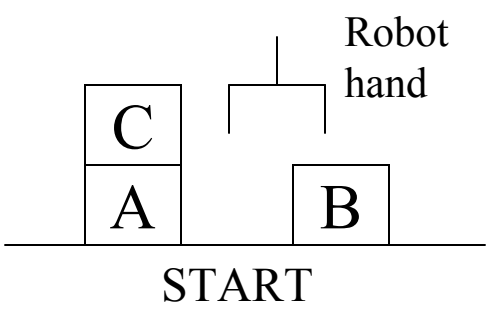
- The Preconditions consist of a **conjunctive logical expression** which is intended to describe the conditions that must be true in order to apply the operator.

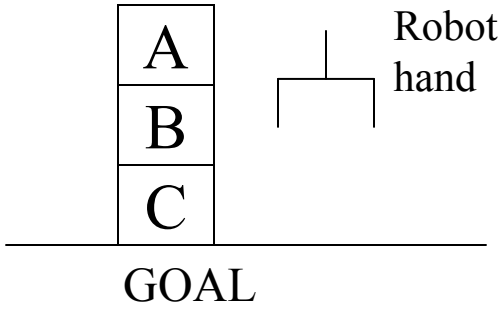
- The **positive or additions** consist of a set of expressions that must be **added** to a model of the situation if the operator is applied.

- The **negative or deletions** consist of a set of expressions that must be **deleted** from a model of a situation if the operator is applied.

State Space: Blocks World





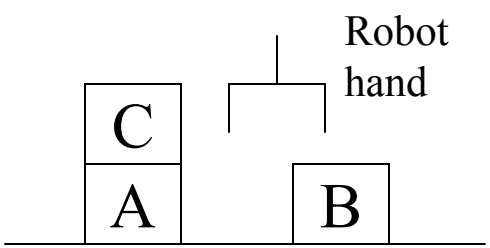


Sequence of actions :

1. Grab C
2. Pickup C
3. Place on table C
4. Grab B
5. Pickup B
6. Stack B on C
7. Grab A
8. Pickup A
9. Stack A on B

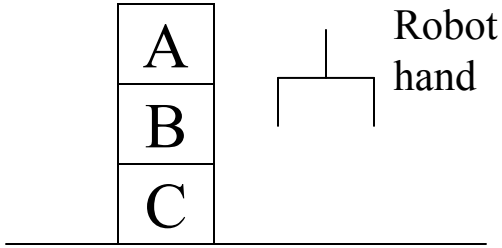
Stanford research Institute Problem Solver

•STRIPS : A planning system – Has rules with precondition deletion list and addition list



START

on(B, table)
on(A, table)
on(C, A)
hand empty
clear(C)
clear(B)



GOAL

on(C, table)
on(B, C)
on(A, B)
hand empty
clear(A)

Rules

- *R1 :* ***pickup(x)***

  Precondition & Deletion List : handempty, on(x,table), clear(x)

  Add List (Postcondition) : holding(x)

- *R2 :* ***putdown(x)***

  Precondition & Deletion List : holding(x)

  Add List (Postconditions) : handempty, on(x,table), clear(x)

Rules

- *R3 :* ***stack(x,y)***

  Precondition & Deletion List :holding(x), clear(y)

  Add List (Postconditions) : on(x,y), clear(x), handempty

- *R4 :* ***unstack(x,y)***

  Precondition & Deletion List : on(x,y), clear(x),handempty

  Add List (Postconditions) : holding(x), clear(y)

Plan for the block world problem

- For the given problem, Start → Goal can be achieved by the following sequence :
  1. Unstack(C,A)
  2. Putdown(C)
  3. Pickup(B)
  4. Stack(B,C)
  5. Pickup(A)
  6. Stack(A,B)
- Execution of a plan: achieved through a data structure called Triangular Table.

Triangular Table

1	on(C,A) clear(C) hand empty	unstack(C,A)				
2		holding(C)	putdown(C)			
3	on(B,table)		hand empty	pickup(B)		
4			clear(C)	holding(B)	stack(B,C)	
5	on(A,table)	clear(A)			hand empty	pickup(A)
6					clear(B)	holding(A)	stack(A,B)
7			on(C,table)		on(B,C)		on(A,B) clear(A)
	0	1	2	3	4	5	6

Triangular Table

- For n operations in the plan, there are :
  - (n+1) rows : 1 → n+1
  - (n+1) columns : 0 → n
- At the end of the i^{th} row, place the i^{th} component of the plan.
- The row entries for the i^{th} step contain the pre-conditions for the i^{th} operation.
- The column entries for the j^{th} column contain the add list for the rule on the top.
- The $<i,j>^{th}$ cell (where $1 \leq i \leq n+1$ and $0 \leq j \leq n$) contain the pre-conditions for the i^{th} operation that are added by the j^{th} operation.
- The first column indicates the starting state and the last row indicates the goal state.