DRONACHARYA College of Engineering

INTELLIGENT SYSTEMS (CSE-303-F)

Section C

Partial Order Planning

Partial Ordering

- Any planning algorithm
  - that can place two actions into a plan without specifying which comes first is called a *partial-order planner***.**
  - actions dependent on each other are ordered in relation to themselves but not necessarily in relation to other independent actions.
- The solution is represented as a *graph* of actions, not a sequence of actions.

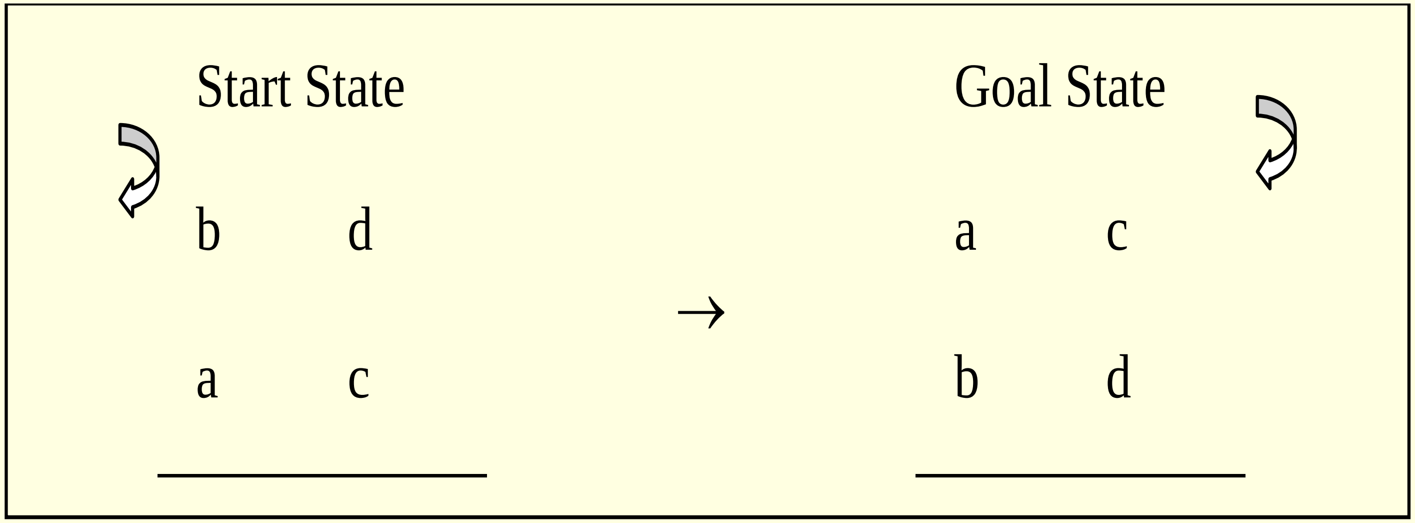
Cont..

- Let us define following two macros for the sake of simplicity for block world example.

Macro Operator	Description	Body
MOT(X)	Move X onto table	US(X, _), PD(X)
MOVE(X, Y)	Move X onto Y	PU(X), ST(X, Y)

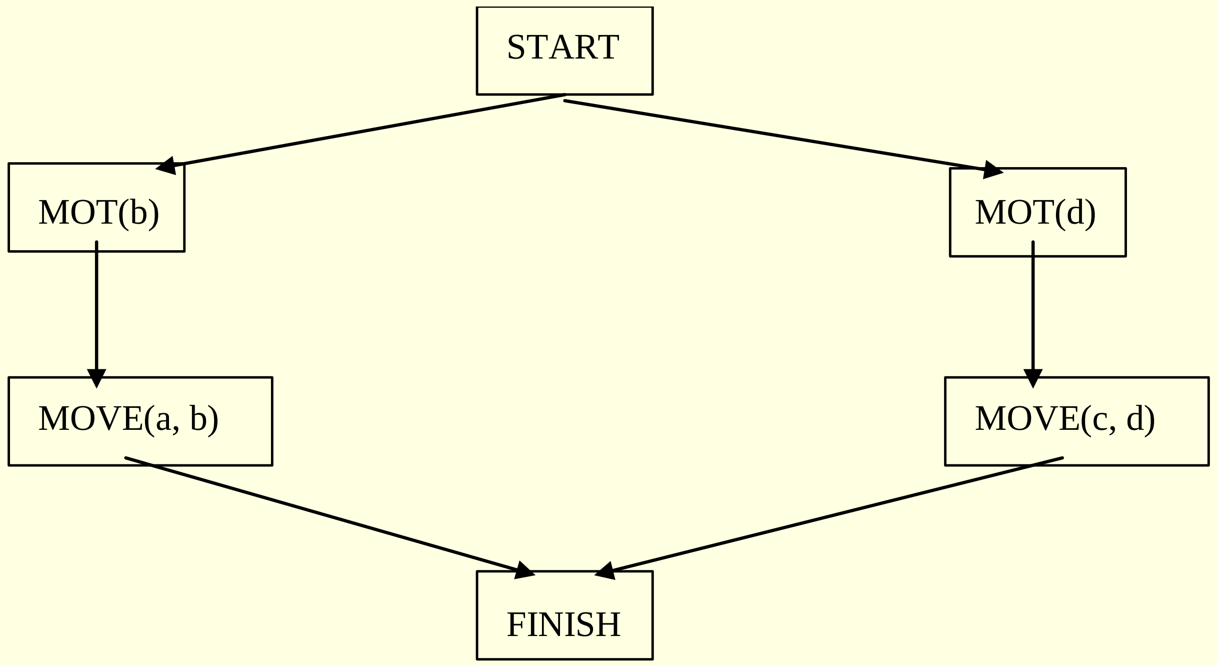
Cont…

- To achieve Goal state ON(a,b),
  - move ‘b’ onto table should occur before move ‘a’ to ‘b’
  - Hence partial ordering MOT(b) $\leftarrow$ MOVE(a, b) holds true
  - MOT(b) should come before MOVE(a, b) in the final plan.
- Similarly to achieve Goal state ON(c,d),
  - partial order MOT(d) $\leftarrow$ MOVE(c, d) is established.



Partial Graph

- Partial graph contains the dummy actions START and FINISH to mark the beginning and end of the plan in the graph.
- The planner can generate total plans from the graph.



Total Plans Generation

- From this representation total six plans are generated.
- Each of these is called a *linearization* of the partial-order plan.

Different Total Plans					
Plan1	**Plan2**	**Plan3**	**Plan4**	**Plan5**	**Plan6**

MOT(b)
MOVE(a, b)
MOT(d)
MOVE(c, d)
	MOT(b)
	MOT(d)
	MOVE(a, b)
	MOVE(c, d)
		MOT(b)
		MOT(d)
		MOVE(c, d)
		MOVE(a, b)
			MOT(d)
			MOVE(c, d)
			MOT(b)
			MOVE(a, b)
				MOT(d)
				MOT(b)
				MOVE(c, d)
				MOVE(a, b)
					MOT(d)
					MOT(b)
					MOVE(a, b)
					MOVE(c, d)

Nonlinear Planning - Constraint Posting

- Idea of constraint posting is to build up a plan by incrementally
  - hypothesizing operators,
  - partial ordering between operators and
  - binding of variables within operators
- At any given time in planning process, a solution is a partially ordered.
- To generate actual plan, convert the partial order into total orders.

Steps in Non Linear Plan Generation

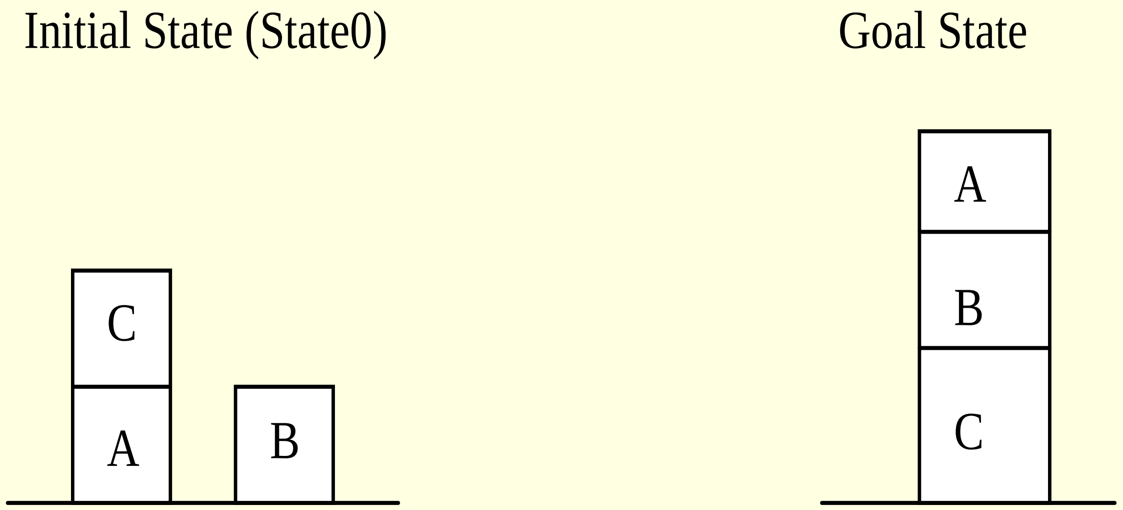
- Step addition
  - Creating new operator (step) for a plan
- Promotion
  - Constraining one operator to come before another in final plan
- Declobbering
  - Placing operator Op2 between two operators Op1 and Op3 such that Op2 reasserts some pre conditions of Op3 that was negated by Op1
- Simple Establishment
  - Assigning a value to a variable, in order to ensure the pre conditions of some step.

Algorithm

1. Initialize S to be set of propositions in the goal state.
2. Remove some unachieved proposition P from S.
3. Achieve P by using step addition, promotion, declobbering, simple establishment.
4. Review all the steps in the plan, including any new steps introduced by step addition to see if any of their preconditions are unachieved.
5. Add to S the new set of unachieved preconditions.
6. If $S = \phi$, complete the plan by converting the partial order of steps into a total order and instantiate any variables as necessary and exit.
7. Otherwise go to step 2.

Example: Sussman anomaly problem

- Begin with null plan (no operators).
- Look at the goal state and find the operators that can achieve them.
- There are two operators (steps) ST(A, B) and ST(B,C) which have post conditions as ON(A,B) and ON(B, C).



Initial State: ON(C, A) Λ ONT(A) Λ ONT(B) Λ AE Λ CL(C) Λ CL(B)

Goal State: **ON(A, B) Λ ON(B, C)**

Cont…

Pre Cond	CL(B)
	*HOLD(A)
		CL(C)
		*HOLD(B)

Operator	ST(A, B)	ST(B,C)
Post Cond	ON(A, B) AE ~ CL(B) ~ HOLD(A)	ON(B,C) AE ~ CL(C) ~ HOLD(B)

- Here unachieved conditions are marked with *.
- HOLD in both the cases is not true as AE is true initially.
- Introduce new operator (step) to achieve these goals.
- This is called operator (step) addition.
- Add PU operator on both the goals.

Cont...

Pre Con	*CL(A)	*CL(B)
	ONT(A)	ONT(B)
	*AE	*AE
Operator	**PU(A)**	**PU(B)**
Post Cond	HOLD(A)	HOLD(B)
	~ ONT(A)	~ ONT(B)
	~ AE	~ AE
	~ CL(A)	~ CL(B)
Pre Con	CL(B)	CL(C)
	*HOLD(A)	*HOLD(B)
Operator	**ST(A, B)**	**ST(B,C)**
	ON(A, B)	ON(B,C)
Post Cond	AE	AE
	~ CL(B)	~ CL(C)
	~ HOLD(A)	~ HOLD(B)

Cont..

- It is clear that in a final plan, PU must precede STACK operator.
- Introduce the ordering as follows:
  - Whenever we employ operator, we need to introduce ordering constraints called promotion.

__________________________Plan 1____________________________

PU(A) $\leftarrow$ ST(A, B)

PU(B) $\leftarrow$ ST(B, C)

____

- Here we partially ordered operators and four unachieved pre conditions:- CL(A), CL(B), AE on both the paths
  - CL(A) is unachieved as C is on A in initial state.
  - Also CL(B) is unachieved even though top of B is clear in initial state but there exist a operator ST(A,B) with post condition as ~CL(B).

  Initial State: ***ON(C, A)****Λ ONT(A) Λ ONT(B) Λ AE Λ CL(C) Λ CL(B)*

Cont..

- If we make sure that PU(B) precede ST(A, B) then **CL(B)** is achieved. So post the following constraints.

Plan 1

PU(A) $\leftarrow$ ST(A, B)

PU(B) $\leftarrow$ ST(B, C)

Plan2

PU(B) $\leftarrow$ ST(A, B)

- Note that pre cond CL(A) of PU(A) still is unachieved.
  - Let us achieve AE preconditions of each Pick up operators before CL(A).
  - Initial state has AE. So one PU can achieve its pre cond but other PU operator could be prevented from being executed.
  - Assume AE is achieved as pre condition of PU(B) as its other preconditions have been achieved. So put constraint.

Cont..

- Similarly, following plans are generated

Plan3

PU(B) $\leftarrow$ PU(A) (pre conds of PU(A) are not still achieved.)

- Since PU(B) makes ~AE and ST(B,C) will make AE which is precondition of PU(A), we can put the following constraint.

Plan4

PU(B) $\leftarrow$ **ST(B, C)** $\leftarrow$ PU(A)

  - Here PU(B) is said to clobber pre condition of PU(A) and ST(B, C) is said to declobber it. (removing deadlock)

Cont..

Plan 5

US(C, A) $\leftarrow$ ST(B, C)

US(C, A) $\leftarrow$ PU(A)

US(C, A) $\leftarrow$ PU(B)

- Declobbering:

Plan 6

US(C, A) $\leftarrow$ PD(C) $\leftarrow$ PU(B)

Cont..

- Combine the following partial plans to generate final plan.

PU(A) ← ST(A, B)
PU(B) ← ST(B, C)

PU(B) ← ST(A, B)

PU(B) ← PU(A)
(pre conds of PU(A) are not still achieved.)

PU(B) ← ST(B, C) ← PU(A)

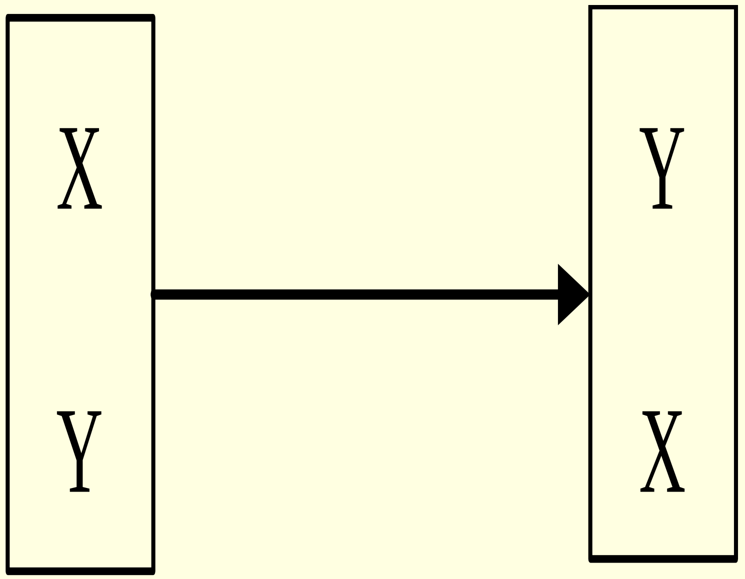
US(C, A) ← ST(B, C)
US(C, A) ← PU(A)
US(C, A) ← PU(B)

US(C, A) ← PD(C) ← PU(B)

Final plan: **US(C,A) ← PD(C) ← PU(B) ← ST(B,C) ← PU(A) ← ST(A,B)**

Learning Plans

- In many problems, plans may share a large number of common sequence of actions.
- So planner requires the ability to recall and modify or reuse the existing plans.
- **Macro operators** can be defined as sequence of operators for performing some task and saved for future use.
- Example:
  - **Reverse_blocks(X, Y)** can be macro operator with plan – US(X, Y), PD(X), PU(Y), ST(Y, X), where X, Y are variables.
- The generalized plan schema is called MACROP and is stored in a data structure called **Triangle table**.
- It helps planner to build new plans efficiently by using existing plans.



Reversing the blocks

Triangle Table

- A useful graphical mechanism to
  - show the plan evolution as well as link the succession of operators in a triangle table.
- The structure of the table is staircase type which gives compact summary of the plan.
- Let us use the following acronyms.
  - A(OP) → add-list of OP
  - CC → copy content of above cell
  - D(OP) → del-list of OP

Triangle Table – cont…

	j = 0	j = 1 OP_1	j = 2 OP_2		j = k OP_k
i = 0	Start state				
i = 1	CC – D(OP_1)	A(OP_1)			
i = 2	CC – D(OP_2)	CC – D(OP_2)	A(OP_2)		
	⋮				
i = k	CC – D(OP_k)	CC – D(OP_k)	CC – D(OP_k)		A(OP_k)

Cont..

- Each column of the table is headed by
  - one of the operators in the plan in the order of their occurrence.
- The table displays the preconditions of each operator.
- The cells below each operator OP contains
  - predicates added by the operator.
- The cells left to the cell mentioned below OP contains
  - predicates that are preconditions of OP.

Rules for forming such tables

- Given a plan requiring the successive use of k operators, Op1, Op2, …Opk, the table is constructed as follows:
  - The triangle table is constructed that consists of (k+1) rows and columns.
  - The columns are indexed by ‘j’ from left to right and rows indexed by ‘i’ from top to bottom with values 0 to k.
  - The construction of triangle table starts with Cell(0, 0) containing the start state predicates.

Cont…

- Traverse the columns from top to bottom
  - having reduction in the system state entries due to the succession of operator del-list applications.
- Finally when the table is complete,
  - the union of the facts in the bottom row (n = k) represents the goal state.
- Let us consider macro-operator “reverse” with the sequence of operators as:

  US(X,Y), PD(X), PU(Y), S(Y,X)

  - Here X and Y are variables and can be bound with actual objects.

Algorithm for Triangle Table

- Entries in the cells are made as follows:
  - Cell(0,0) contains the start state
  - In Cell(k, k), for k > 0, contains add-list of operator OP_k.
- For each cell Cell(i, j), i > j, copy the contents of Cell(i-1, j) with del-list of operator OP_j removed.

Cont..

	m =0	m=1	m=2	m=3	m=4
n = 0	~~O(X, Y)~~, C(X), T(Y), ~~AE~~	US(X, Y)			
n = 1	C(X), T(Y)	~~H(X)~~, C(Y)	PD(X)		
n = 2	C(X), ~~T(Y)~~	~~C(Y)~~	T(X), ~~AE~~	PU(Y)	
n = 3	~~C(X)~~		T(X)	~~H(Y)~~	ST(Y, X)
n = 4			T(X)		O(Y, X), AE, C(Y)

REVERSE (X,Y)