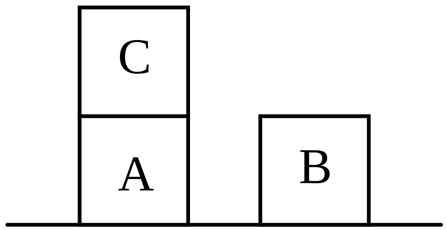
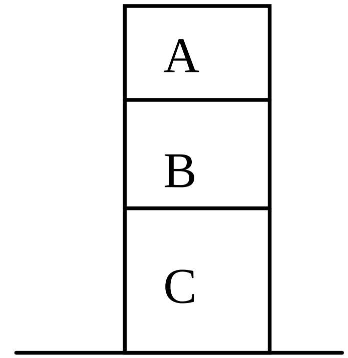
Nonlinear Planning using Constraint Posting

- 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, partially instantiated set of operators.
- To generate actual plan, convert the partial order into any of a number of total orders.
- Let us incrementally generate a nonlinear plan to solve Sussman anomaly problem.

Initial State (State0)



Goal State



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)**

- Begin with null plan (no operators).
- Look at the goal state and find the operators that can achieve them.
- MEA tells to choose two operators (steps) ST(A, B) and ST(B,C) with respect to post conditions as ON(A,B) and ON(B, C)

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

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

Pre Con	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)

- 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.

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

- Here we have four (partially ordered) operators and four unachieved pre conditions:- CL(A), CL(B), AE on both the paths
  - CL(A) is unachieved as top of A is not clear 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)

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

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.
- Promotion:

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

- Now all preconditions of PU(B) are achieved.
- Here apply another heuristic called declobbering.
- Decobbering:
  - *Placing operator Op2 between two operators Op1 and Op3 such that Op2 reasserts some pre conditions of Op3 that was negated by Op1*.
- Since PU(B) makes ~AE and ST(B,C) will make AE which is precondition of PU(A), we can put the following constraint.

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

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

- Now try to achieve CL(A). This can be done by US(C, A)

Pre Con	ON(C, A)
	* CL(C)
	*AE
Operator	**US(C, A)**
Post Cond	~ AE
	CL(A)
	HOLD(C)
	~ ON(C, A)

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)

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

- ON(C, A) can easily be seen to be true in initial state.
- Even though *CL(C) is also true in initial state but may be denied by operator ST(B,C) already used earlier.
- Similarly *AE may be denied by operators PU(A) and PU(B). So put constraints
- Promotion:

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

- Now adding new operator requires checking, if the new step clobber some pre conditions of later.

- We notice that PU(B) requires AE but denied by new operator US(C,A). One way is to add a new declobbering operator that makes AE to the plan . This can be done by PD(C).

Pre Con	HOLD(C)
Operator	PD(C)
Post Cond	~HOLD(C) ONT(C) AE ~ CL(A)

- Declobbering:

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

Combine the following partial plans to generate final plan.

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

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

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

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

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

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

Final plan:

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

- Main steps involved in non linear plan generation are:

1. Step addition -

   Creating new operator (step) for a plan

2. Promotion -

   Constraining one operator to come before another in final plan

3. Declobbering -

   Placing operator Op2 between two operators Op1 and Op3 such that Op2 reasserts some pre conditions of Op3 that was negated by Op1

4. Simple Establishment-

   Assigning a value to a variable, in order to ensure the pre conditions of some step.

Summary

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)

ON(A, B)

ON(B, C)

CL(B)

*HOLD(A)

ST(A, B)

ST(B,C)

CL(C)

*HOLD(B)

ON(A, B)
AE
~ CL(B)
~ HOLD(A)

ON(B,C)
AE
~ CL(C)
~ HOLD(B)

***CL(A)**

ONT(A)
*AE

PU(A)

PU(B)

*CL(B)

ONT(B)
*AE

HOLD(A)
~ ONT(A)
~ AE
~ CL(A)

HOLD(B)
~ ONT(B)
~ AE
~ CL(B)

PU(A) ← ST(A, B) and PU(B) ← ST(B, C) – Ordering of operators

---

- CL(A) is not unachieved as top of A is not clear 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).
- So if we make sure that CL(B) is achieved, we post a constraints that PU(B) must precede ST(A,B)

Promotion: PU(B) ← ST(A, B)

- Here either PU operators could prevent the other from executing.
- Assume AE be achieved as pre condition of PU(B), then all pre conditions of PU(B) are satisfied.

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

- PU(B) makes ~AE and ST(B,C) will make AE which is precondition of PU(A).

Declobbering: PU(B) ← ST(B, C) ← PU(A)

- Now try to achieve CL(A). This can be done by US(C, A)

CL(A)

ON(C, A)

* CL(C)

*AE

US(C, A)

~ AE

CL(A)

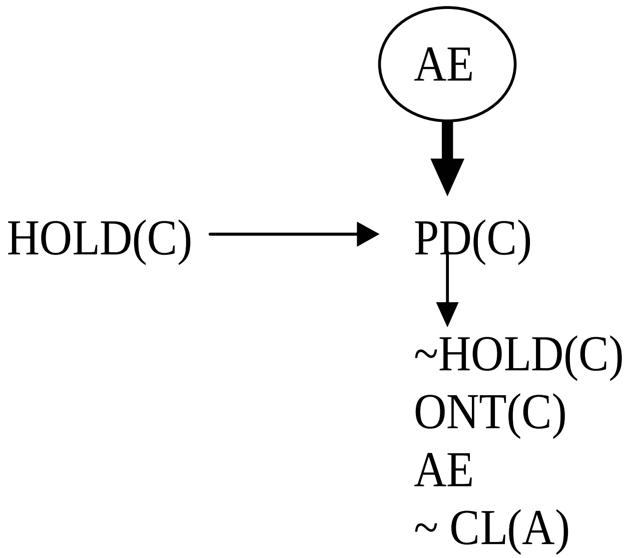
HOLD(C)

~ ON(C, A)

- ON(C, A) can be easily be seen to be true in initial state.
- But CL(C) may be denied by operator ST(B, C) already used earlier and AE may be denied by operators PU(A) and PU(B).

Promotion: US(C, A) ← ST(B, C); US(C, A) ← PU(A); US(C, A) ← PU(B)

- Now adding new operator requires checking, if the new step clobber some pre conditions of later.
- Here we see that PU(B) requires AE but denied by new operator US(C,A). One way is to add a new declobbering operator to the plan.



Declobbering: US(C, A) ← PD(C) ← PU(B)

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

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 = ϕ, 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.

Triangle Table

- This is another planning technique.
- It provides a way of removing the goals that each operator is expected to satisfy as well as goals that must be true for it to execute correctly.
- 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.

AL(Op)	-	Add-list of op
ACC	-	Above cell contents
DL(Op)	-	del-list of op

	m = 0	m = 1 Op1	m = 2 Op2		m = k Opk
n = 0	Initial state				
n = 1	ACC – DL(Op1)	AL(Op1)			
	ACC – DL(Op2)	ACC – DL(Op2)	AL(Op2)		
	⋮				
n = k	ACC – DL(Opk)	ACC – DL(Opk)	ACC – DL(Opk)		AL(Opk)

Rules for forming such tables

- Given a resulting plan requiring the successive use of k operators, Op1, Op2, …Opk, the table consists of
  - k+1 columns indexed by m from left to right with values 0 to k.
  - Similarly k+1 rows indexed by n from top to bottom with values 0 to k.
- Each cell may be empty or composed of a subset of the system state.
- Cell(0,0) contains initial state.

- Entries in the cells are made as follows:
  1. In Cell(m, n), for m > 0, add list of operator.

2. In Cell(m, n) in column m, for n > m, apply the following sub steps recursively.
    - Cell(m, n), n > m contains the contents of Cell(m, n-1) with delete list of operator m removed.

- This process starts with Cell(0, 0).

- Traversing the columns from top to bottom have reduction in the system state entries due to the succession of operator delete list applications.

- Finally when the table is complete, the union of the facts in the bottom row (n = k) represents the goal state.