# ARTIFICIAL INTELLIGENCE (AI)
# PLANNING


Shreya Ghosh

Assistant Professor, Computer Science and Engineering

*IIT Bhubaneswar*

## Which of these situations are allowed by these actions?

# Partial Order Planning

- **Basic Idea:** **Make choices only that are relevant to solving the current part of the problem**

- **Least Commitment Choices**
  - **Orderings:** **Leave actions unordered, unless they must be sequential**
  - **Bindings:** **Leave variables unbound, unless needed to unify with conditions being achi**
  - **Actions:** **Usually not subject to “least commitment”**

# Terminology

- Totally Ordered Plan
  - There exists sufficient orderings O such that all actions in A are ordered with respect to each other
- Fully Instantiated Plan
  - There exists sufficient constraints in B such that all variables are constrained to be equal to some constant
- Consistent Plan
  - There are no contradictions in O or B
- Complete Plan
  - Every precondition P of every action $A_i$ in A is achieved:
    - There exists an effect of an action $A_j$ that comes before $A_i$ and unifies with P, and no action $A_k$ that deletes P comes between $A_j$ and $A_i$

# PLANNING (LECTURE 2)

SLIDES TAKEN FROM SIMILAR COURSE OFFERED AT PENN STATE AND IIT KGP

# POP Example: Get Tea, Biscuits, Book

## Initial state:

Op( ACTION: Start,
EFFECT: At(Home) $\wedge$ Sells(BS, Book)
$\wedge$ Sells(TS, Tea)
$\wedge$ Sells(TS, Biscuits) )

## Goal state:

Op( ACTION: Finish,
PRECOND: At(Home) $\wedge$ Have(Tea)
$\wedge$ Have(Biscuits)
$\wedge$ Have(Book) )

## Actions:

Op( ACTION: Go(y),
PRECOND: At(x),
EFFECT: At(y) $\wedge$ $\neg$At(x))

Op( ACTION: Buy(x),
PRECOND: At(y) $\wedge$ Sells(y, x),
EFFECT: Have(x))

# The Partial Order Planning Algorithm

```
Function POP( initial, goal, operators )
// Returns plan
    plan ← Make-Minimal-Plan( initial, goal )
    Loop do
        If Solution( plan ) then return plan
        S, c ← Select-Subgoal( plan )
        Choose-Operator( plan, operators, S, c )
        Resolve-Threats( plan )
    end
```

# POP: Selecting Sub-Goals

**Function Select-Subgoal( *plan* )**

**// Returns S, c**

**pick a plan step S from STEPS( *plan* )**

**with a precondition C that has not been achieved**

**Return S, c**

# POP: Choosing operators

**Procedure Choose-Operator**( *plan, operators*, S, c )

Choose a step S' from *operators* or STEPS( *plan* ) that has c as an effect

If there is no such step then fail

Add the causal link S' $\rightarrow$ c: S to LINKS( *plan* )

Add the ordering constraint S' $\prec$ S to ORDERINGS( *plan* )

If S' is a newly added step from *operators* then add S' to STEPS( *plan* ) and add Start $\prec$ S' $\prec$ Finish to ORDERINGS( *plan* )

# POP: Resolving Threats

**Procedure Resolve-Threats( *plan* )**

for each S' that threatens a link $S_i \rightarrow c: S_j$ in LINKS( *plan* ) do

choose either

*Promotion:* Add S'' $\prec S_i$ to ORDERINGS( *plan* )

*Demotion:* Add $S_j \prec$ S'' to ORDERINGS( *plan* )

if not Consistent( *plan* ) then fail

# Partially instantiated operators

- So far we have not mentioned anything about binding constraints
- Should an operator that has the effect, say, $\neg$*At(x)*, be considered a threat to the condition, *At(Home)* ?
    - Indeed it is a *possible threat* because *x* may be bound to *Home*

**Proc Choose-Operator**( *plan, operators*, S, c )

choose a step S' from *operators* or STEPS( *plan* ) that has c' as an effect

such that *u* = UNIFY( c, c', BINDINGS( plan ))

if there is no such step then fail

add *u* to BINDINGS( *plan* )

add the causal link $S' \rightarrow c: S$ to LINKS( *plan* )

add the ordering constraint $S' \prec S$ to ORDERINGS( *plan* )

if S' is a newly added step from *operators* then

add S' to STEPS( *plan* ) and add $Start \prec S' \prec Finish$ to ORDERINGS( *plan* )

**Procedure Resolve-Threats( *plan* )**

for each $S_i \rightarrow c: S_j$ in LINKS( *plan* ) do

for each S" in STEPS( *plan* ) do

for each c' in EFFECTS( S") do

if SUBST( BINDINGS(*plan*), c ) = SUBST( BINDINGS(*plan*), ¬c' )

then choose either

*Promotion:* Add $S'' \prec S_i$ to ORDERINGS( *plan* )

*Demotion:* Add $S_j \prec S''$ to ORDERINGS( *plan* )

if not Consistent( *plan* ) then fail