

---

# Planning

Philipp Koehn

26 March 2020



# Outline

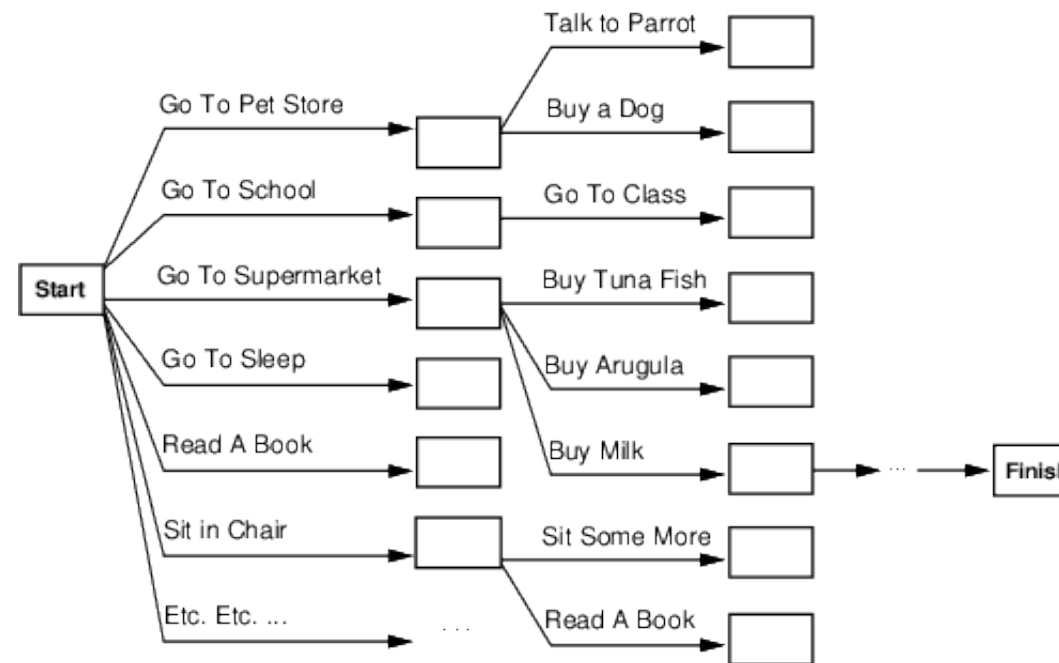


- Search vs. planning
- STRIPS operators
- Partial-order planning
- The real world
- Conditional planning
- Monitoring and replanning

# search vs. planning

# Search vs. Planning

- Consider the task *get milk, bananas, and a cordless drill*
- Standard search algorithms seem to fail miserably:



- After-the-fact heuristic/goal test inadequate

# Search vs. Planning



4

- Planning systems do the following
  1. improve action and goal representation to allow selection
  2. divide-and-conquer by **subgoal**ing
  3. relax requirement for sequential construction of solutions
- Differences

	Search	Planning
<b>States</b>	Data structures	Logical sentences
<b>Actions</b>	Program code	Preconditions/outcomes
<b>Goal</b>	Program code	Logical sentence (conjunction)
<b>Plan</b>	Sequence from $S_0$	Constraints on actions

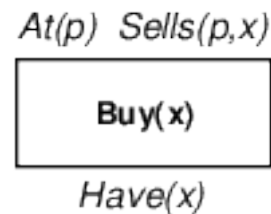
# strips operators

# STRIPS Operators



6

- Tidily arranged actions descriptions, restricted language



- ACTION: *Buy(x)*  
PRECONDITION: *At(p), Sells(p, x)*  
EFFECT: *Have(x)*
- Note: this abstracts away many important details!
- Restricted language  $\implies$  efficient algorithm  
Precondition: conjunction of positive literals  
Effect: conjunction of literals

# partial-order planning



# Partially Ordered Plans



8

- *Partially ordered* collection of steps with
  - *Start step* has the initial state description as its effect
  - *Finish step* has the goal description as its precondition
  - *causal links* from outcome of one step to precondition of another
  - *temporal ordering* between pairs of steps■
- *Open condition* = precondition of a step not yet causally linked
- A plan is *complete* iff every precondition is achieved
- A precondition is *achieved* iff it is the effect of an earlier step and no *possibly intervening* step undoes it

# Example



**Start**

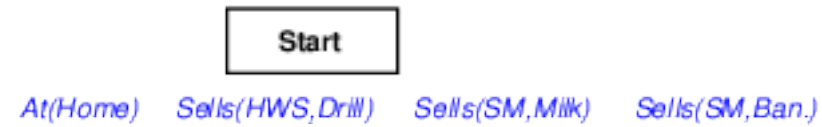
*At(Home)   Sells(HWS,Drill)   Sells(SM,Milk)   Sells(SM,Ban.)*

*Have(Milk)   At(Home)   Have(Ban.)   Have(Drill)*

**Finish**

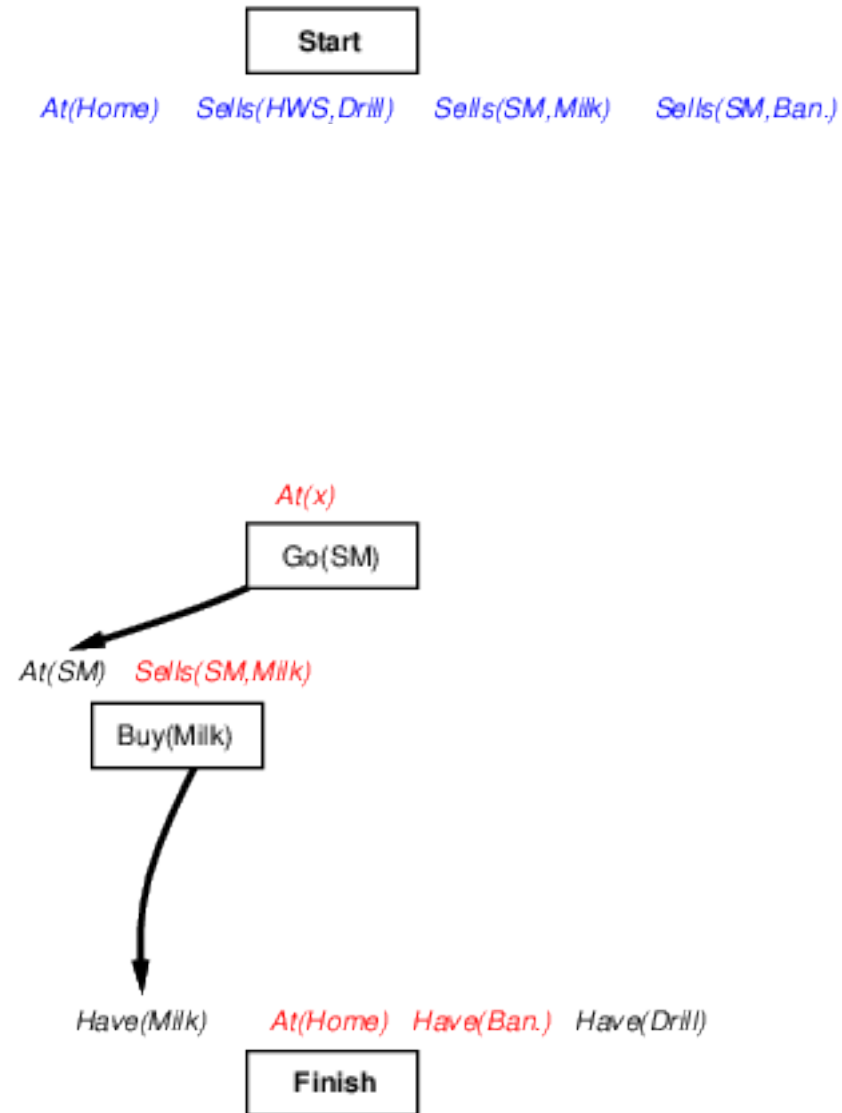
# Example

10

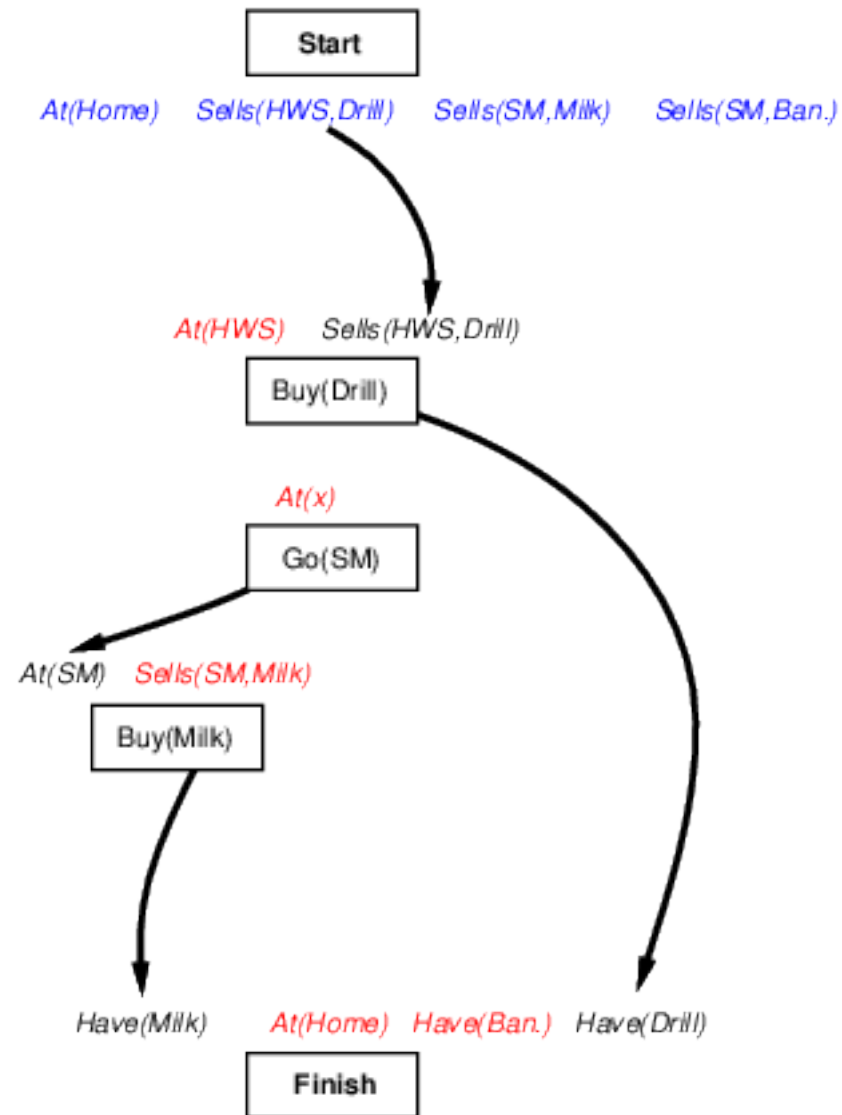


# Example

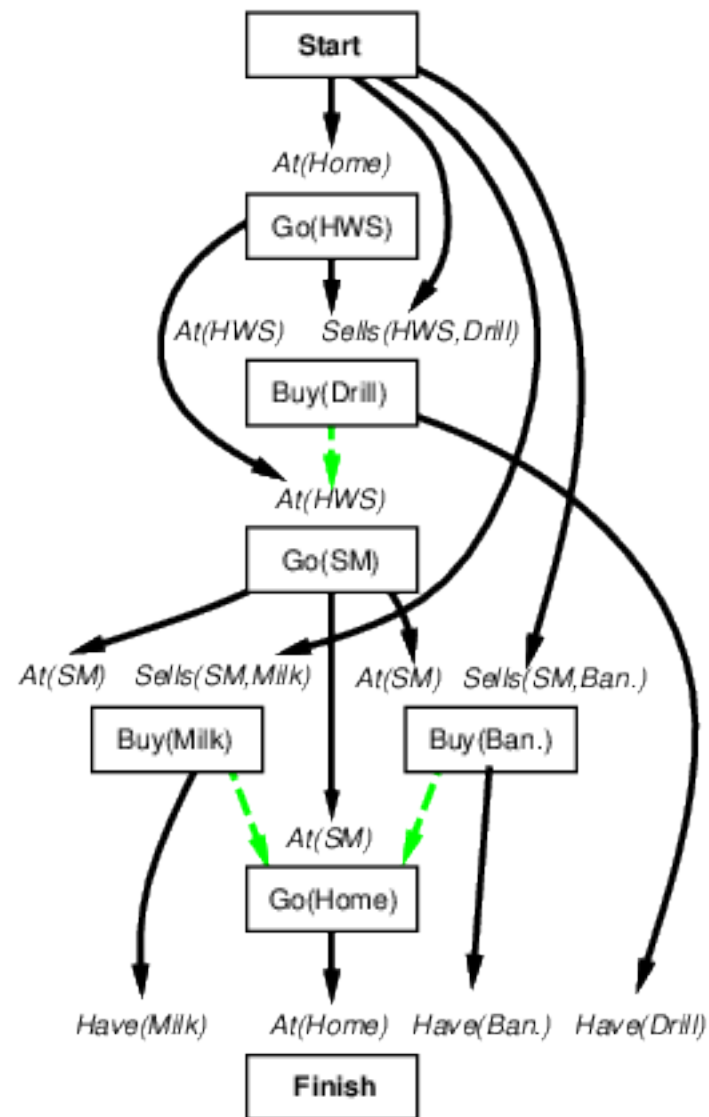
11



# Example



# Example



# Planning Process

- Operators on partial plans
  - **add a link** from an existing action to an open condition
  - **add a step** to fulfill an open condition
  - **order** one step wrt another to remove possible conflicts
- Gradually move from incomplete/vague plans to complete, correct plans
- Backtrack if an open condition is unachievable or if a conflict is unresolvable

# Partially Ordered Plans Algorithm

**function** POP(*initial*, *goal*, *operators*) **returns** *plan*

*plan*  $\leftarrow$  MAKE-MINIMAL-PLAN(*initial*, *goal*)

**loop do**

**if** SOLUTION?(*plan*) **then return** *plan*

$S_{need}, c \leftarrow$  SELECT-SUBGOAL(*plan*)

    CHOOSE-OPERATOR(*plan*, *operators*,  $S_{need}$ ,  $c$ )

    RESOLVE-THREATS(*plan*)

**end**

---

**function** SELECT-SUBGOAL(*plan*) **returns**  $S_{need}, c$

    pick a plan step  $S_{need}$  from STEPS(*plan*)

        with a precondition  $c$  that has not been achieved

**return**  $S_{need}, c$



# Partially Ordered Plans Algorithm

**procedure** CHOOSE-OPERATOR(*plan*, *operators*,  $S_{need}$ ,  $c$ )

**choose** a step  $S_{add}$  from *operators* or STEPS(*plan*) that has  $c$  as an effect

**if** there is no such step **then fail**

add the causal link  $S_{add} \xrightarrow{c} S_{need}$  to LINKS(*plan*)

add the ordering constraint  $S_{add} < S_{need}$  to ORDERINGS(*plan*)

**if**  $S_{add}$  is a newly added step from *operators* **then**

add  $S_{add}$  to STEPS(*plan*)

add  $Start < S_{add} < Finish$  to ORDERINGS(*plan*)

---

**procedure** RESOLVE-THREATS(*plan*)

**for each**  $S_{threat}$  that threatens a link  $S_i \xrightarrow{c} S_j$  in LINKS(*plan*) **do**

**choose** either

*Demotion*: Add  $S_{threat} < S_i$  to ORDERINGS(*plan*)

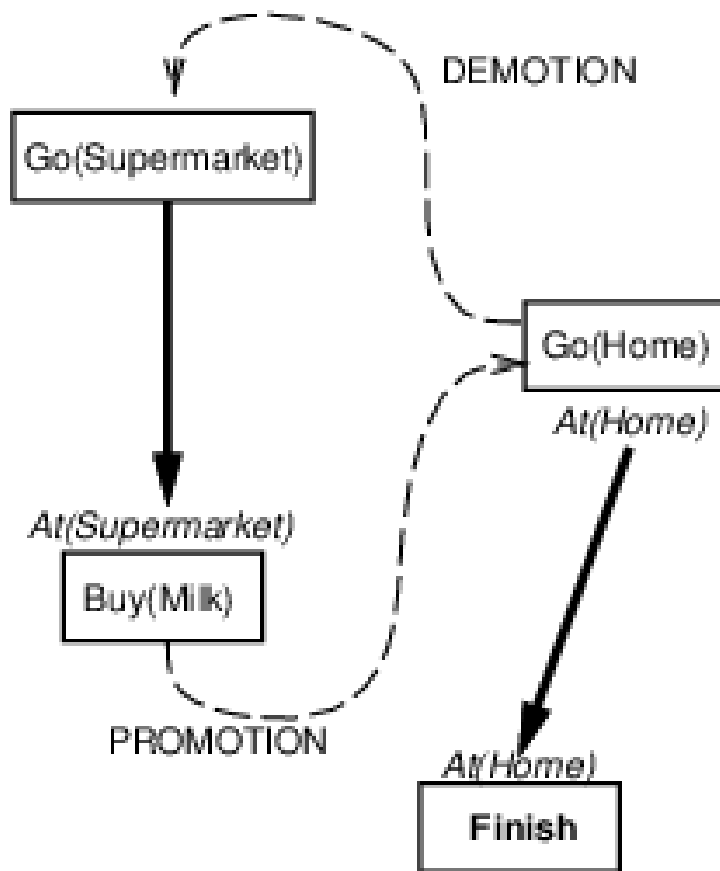
*Promotion*: Add  $S_j < S_{threat}$  to ORDERINGS(*plan*)

**if not** CONSISTENT(*plan*) **then fail**

**end**

# Clobbering and Promotion/Demotion

- A **clobberer** is a potentially intervening step that destroys the condition achieved by a causal link. E.g.,  $Go(Home)$  clobbers  $At(Supermarket)$ :



**Demotion:** put before  $Go(Supermarket)$

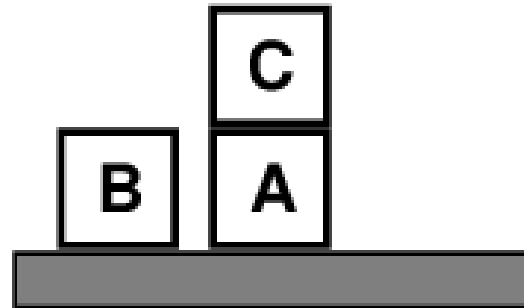
**Promotion:** put after  $Buy(Milk)$

# Properties of Partially Ordered Plans

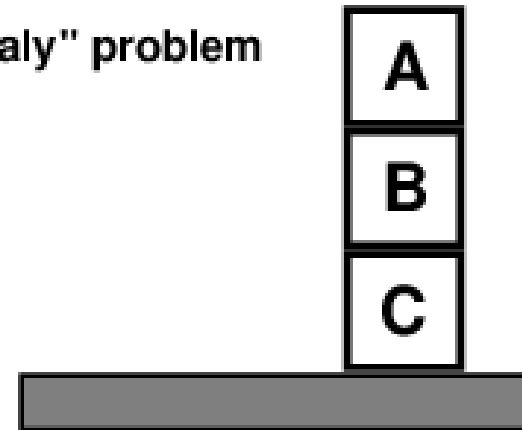
- Nondeterministic algorithm: backtracks at **choice** points on failure
  - choice of  $S_{add}$  to achieve  $S_{need}$
  - choice of demotion or promotion for clobberer
  - selection of  $S_{need}$  is irrevocable■
- Partially Ordered Plans is sound, complete, and **systematic** (no repetition)
- Extensions for disjunction, universals, negation, conditionals
- Can be made efficient with good heuristics derived from problem description
- Particularly good for problems with many loosely related subgoals

# Example: Blocks World

"Sussman anomaly" problem



Start State



Goal State

$Clear(x) \ On(x,z) \ Clear(y)$

$PutOn(x,y)$

$\sim On(x,z) \ \sim Clear(y)$   
 $Clear(z) \ On(x,y)$

$Clear(x) \ On(x,z)$

$PutOnTable(x)$

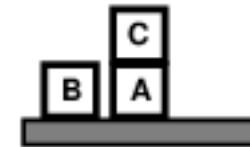
$\sim On(x,z) \ Clear(z) \ On(x, Table)$

+ several inequality constraints

# Example

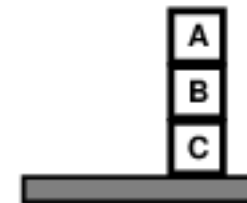
START

*On(C,A) On(A,Table) Cl(B) On(B,Table) Cl(C)*



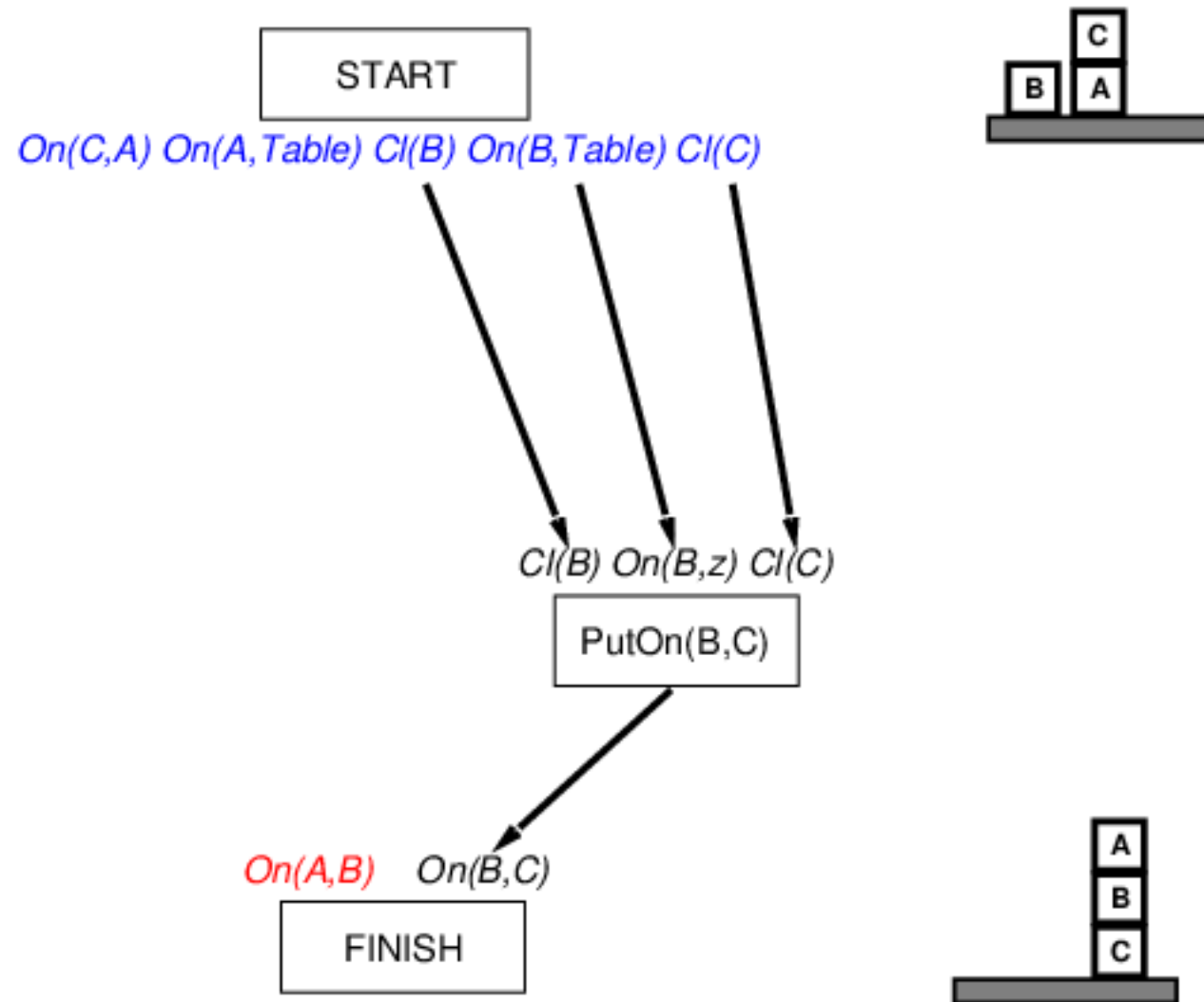
*On(A,B) On(B,C)*

FINISH

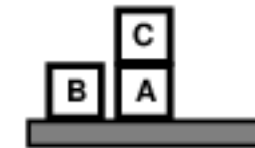
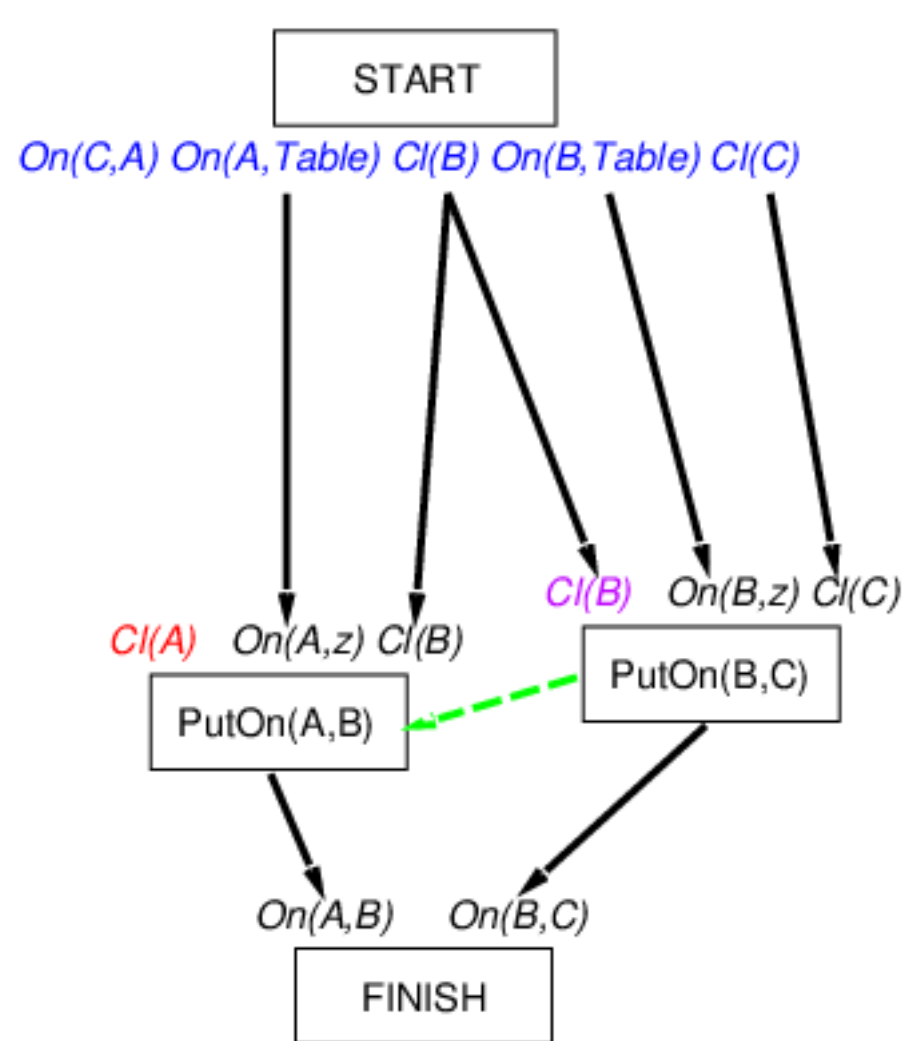


# Example

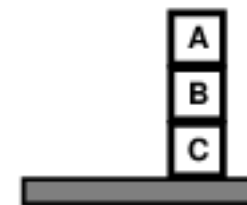
21



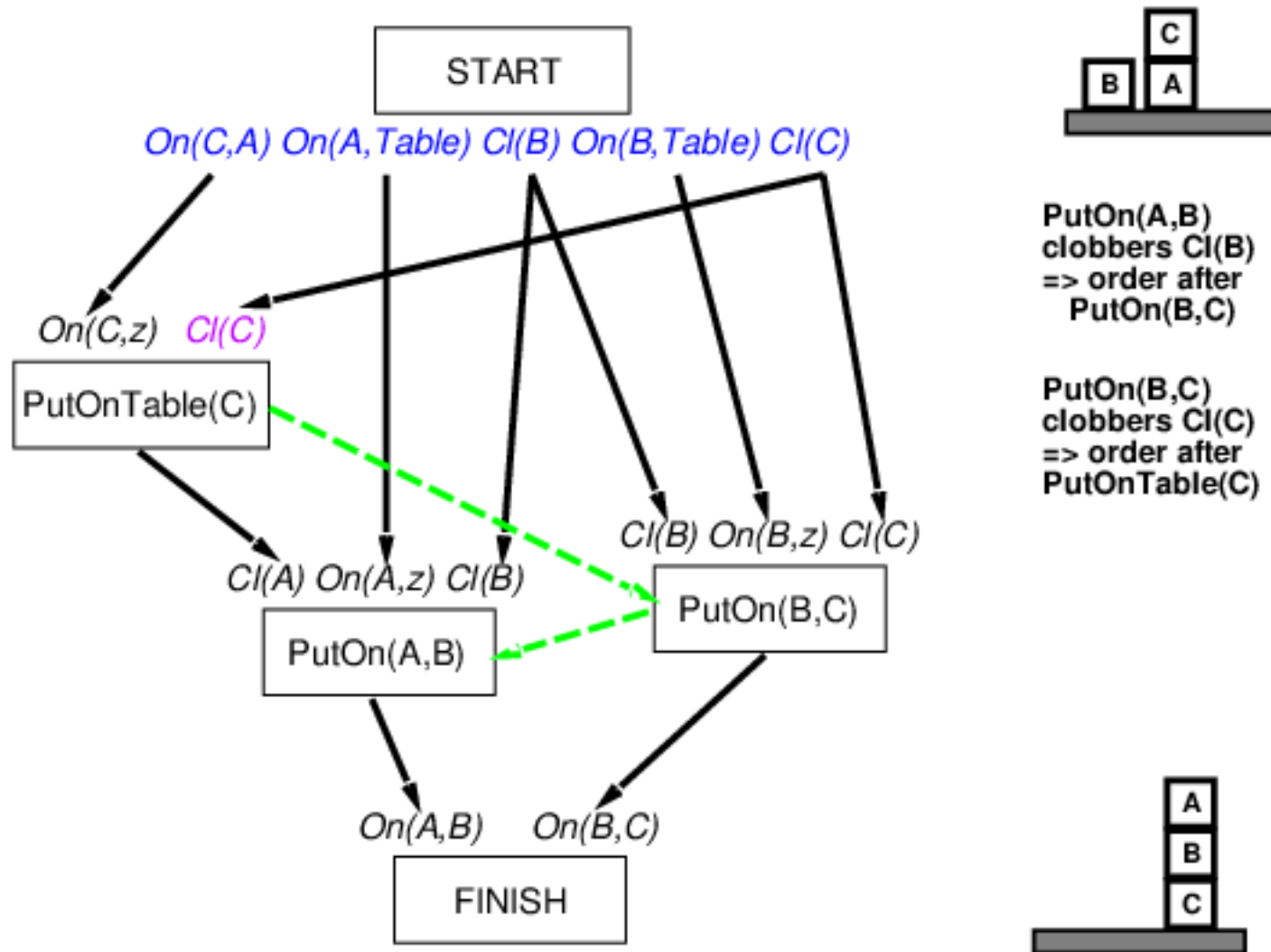
# Example



PutOn(A,B)  
clobbers Cl(B)  
=> order after  
PutOn(B,C)



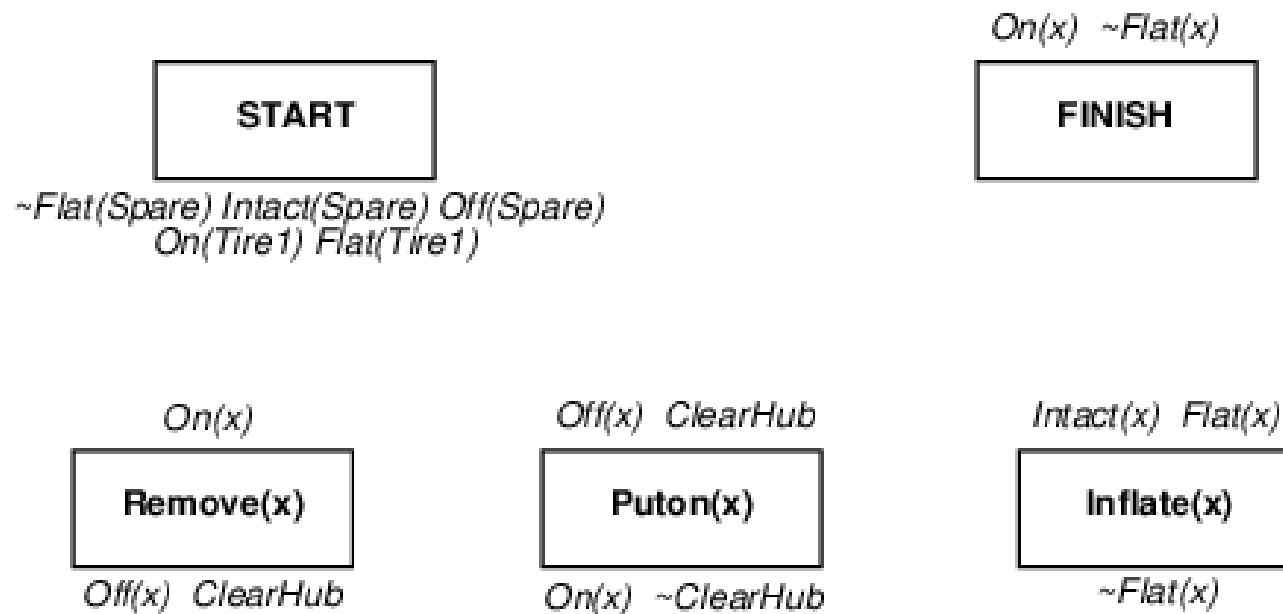
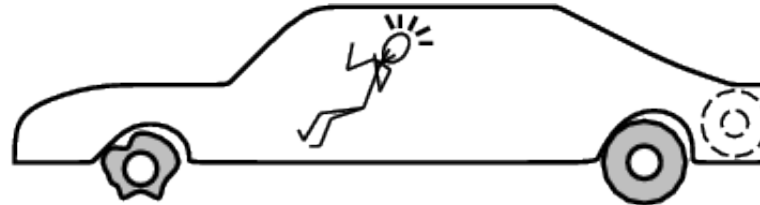
# Example





# the real world

# The Real World



# Things Go Wrong

- *Incomplete information*
  - Unknown preconditions, e.g.,  $Intact(Spare)?$
  - Disjunctive effects, e.g.,  $Inflate(x)$  causes  $Inflated(x) \vee SlowHiss(x) \vee Burst(x) \vee BrokenPump \vee \dots$  ■
- *Incorrect information*
  - Current state incorrect, e.g., spare NOT intact
  - Missing/incorrect postconditions in operators ■
- *Qualification problem* can never finish listing all
  - required preconditions of actions
  - possible conditional outcomes of actions

- Conformant or sensorless planning

Devise a plan that works regardless of state or outcome

*Such plans may not exist*

- Conditional planning

Plan to obtain information (**observation actions**)

Subplan for each contingency, e.g.,

[*Check(Tire1)*, **if** *Intact(Tire1)* **then** *Inflate(Tire1)* **else** *CallAAA*]

*Expensive because it plans for many unlikely cases*

- Monitoring/Replanning

Assume normal states, outcomes

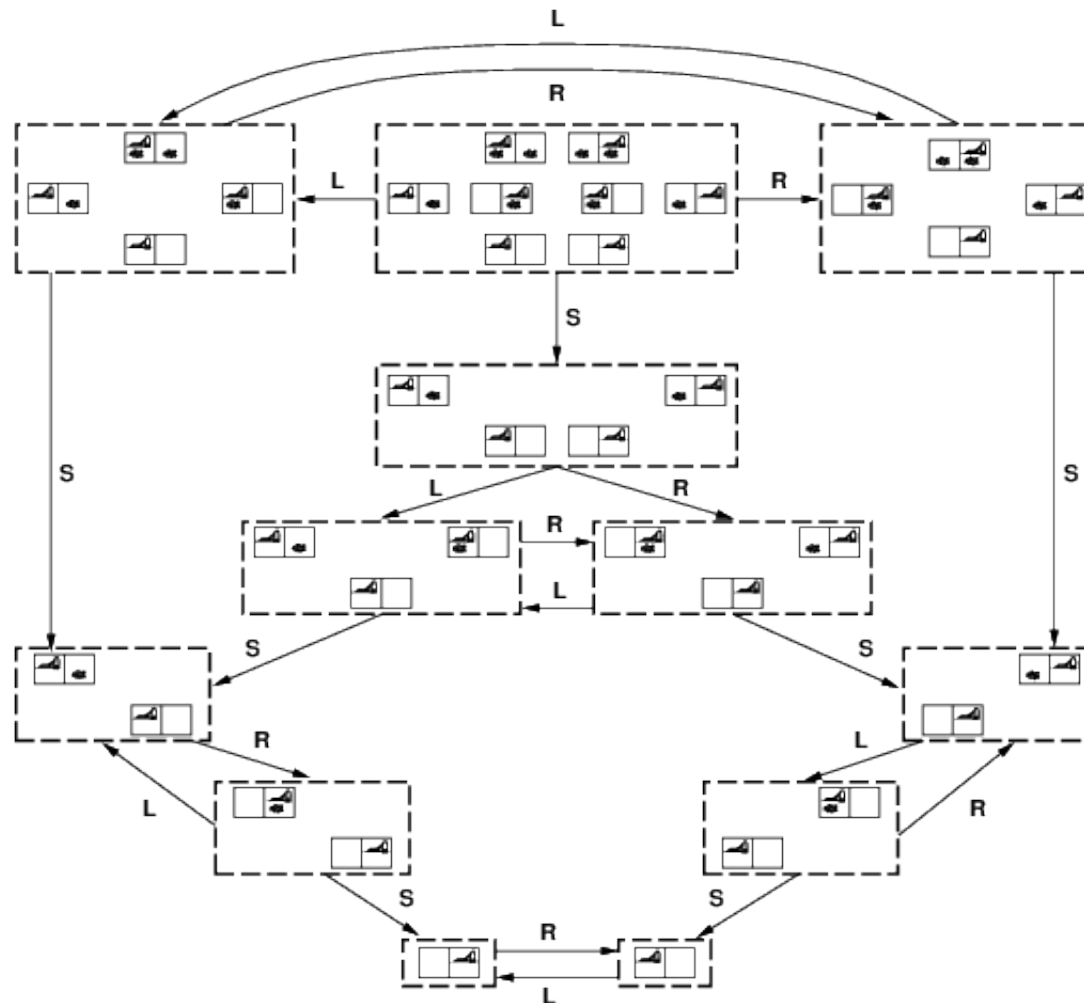
Check progress *during execution*, replan if necessary

*Unanticipated outcomes may lead to failure (e.g., no AAA card)*

⇒ Really need a combination; plan for likely/serious eventualities,  
deal with others when they arise, as they must eventually.

# Conformant Planning

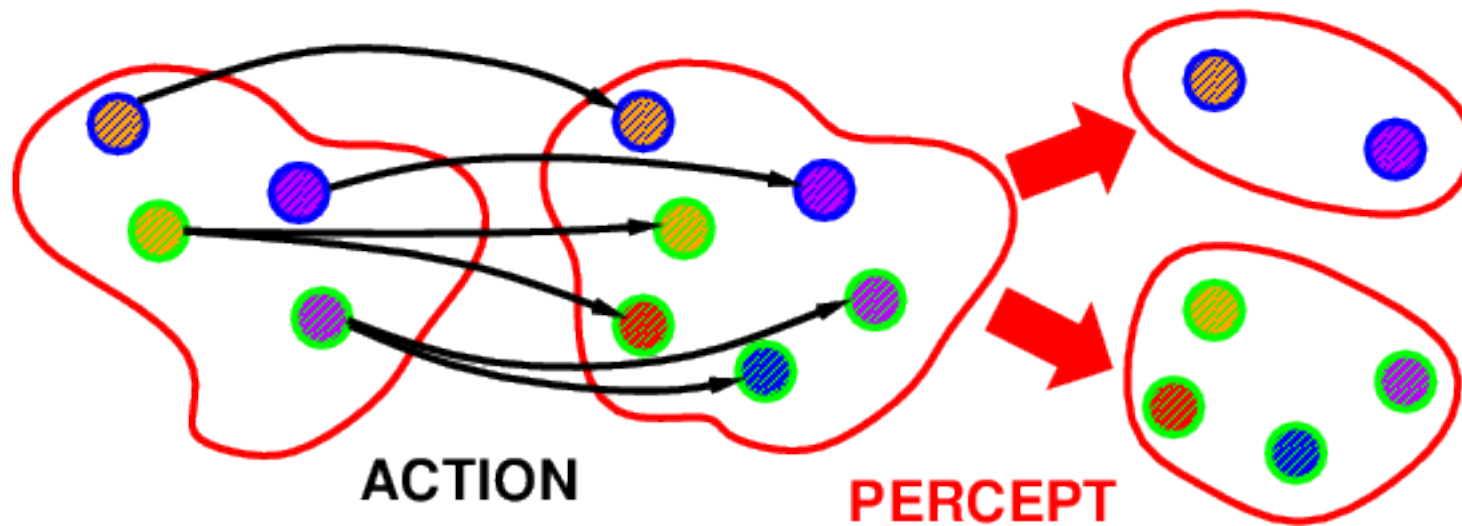
- Search in space of **belief states** (sets of possible actual states)



# conditional planning

# Conditional Planning

- If the world is nondeterministic or partially observable then percepts usually *provide information*, i.e., *split up* the belief state



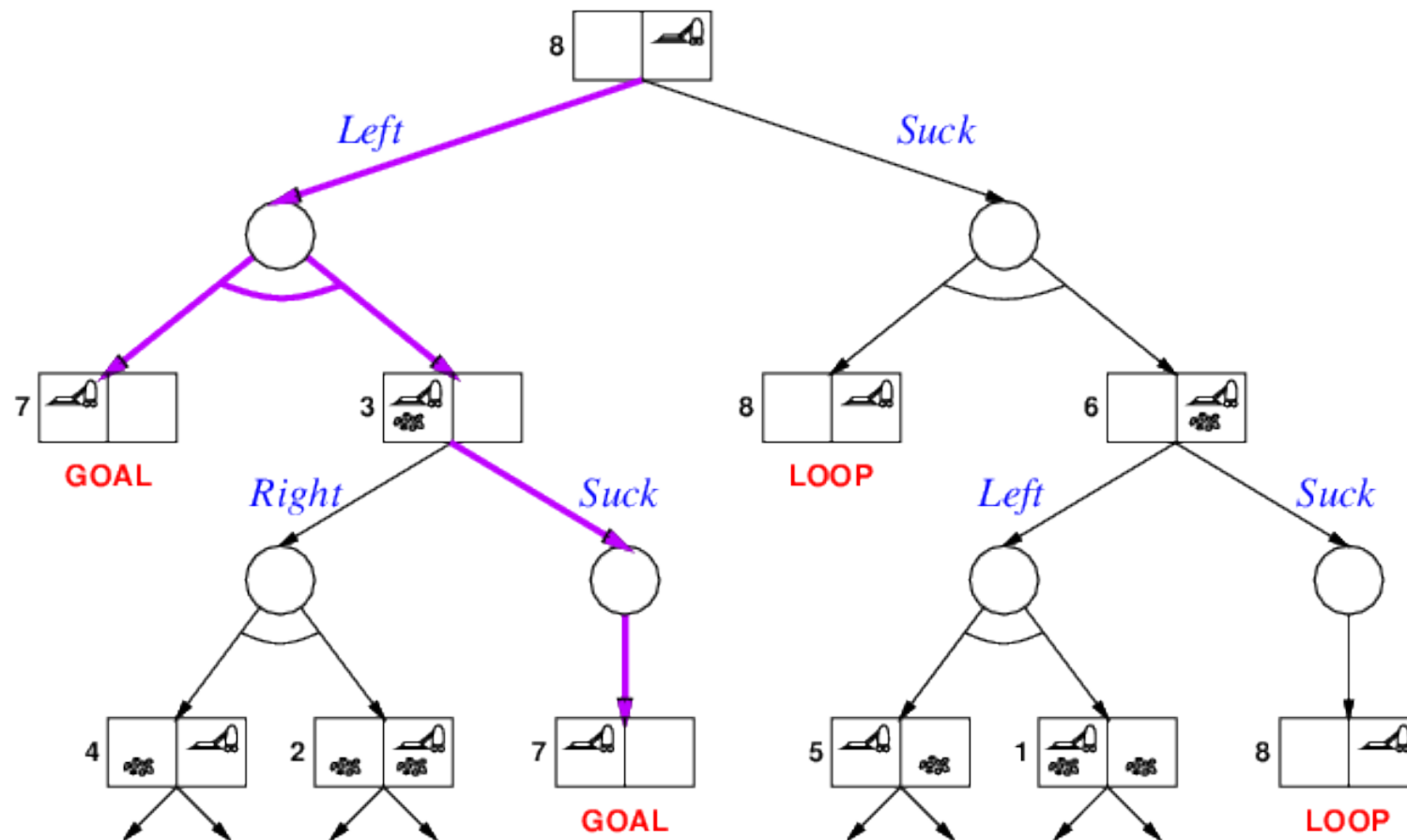
# Conditional Planning

- Conditional plans check (any consequence of KB +) percept
- [..., **if**  $C$  **then**  $Plan_A$  **else**  $Plan_B$ , ...]
- Execution: check  $C$  against current KB, execute “then” or “else”
- Need *some* plan for *every* possible percept
  - game playing: *some* response for *every* opponent move
  - backward chaining: *some* rule such that *every* premise satisfied
- AND–OR tree search (very similar to backward chaining algorithm)



# Example

- Double Murphy: sucking or arriving may dirty a clean square



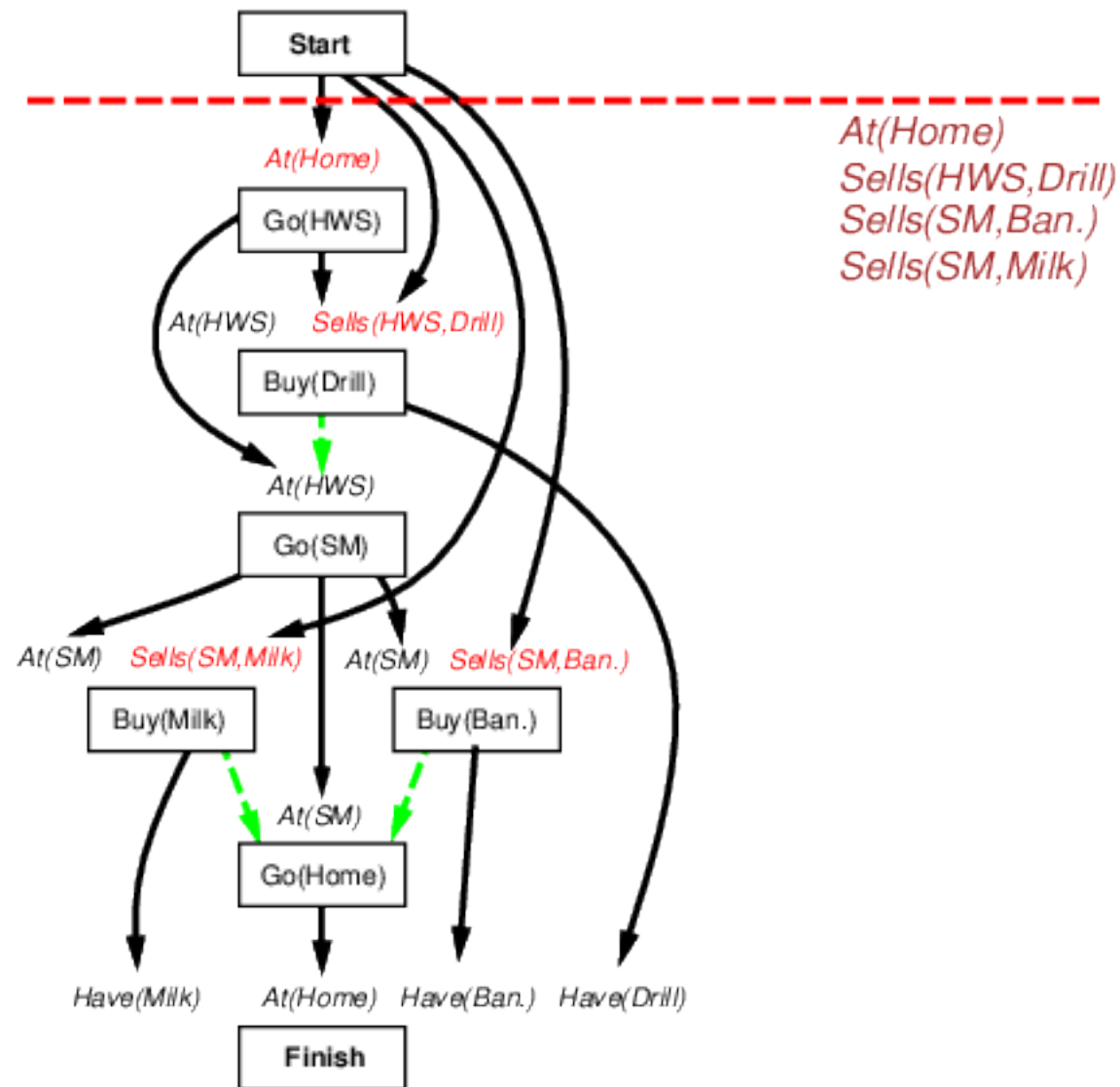
# monitoring and replanning

- Plan with Partially Ordered Plans algorithms
- Process plan, one step at a time
- Validate planned conditions against perceived reality
- “Failure” = preconditions of *remaining plan* not met
- Preconditions of remaining plan
  - = all preconditions of remaining steps not achieved by remaining steps
  - = all causal links *crossing* current time point

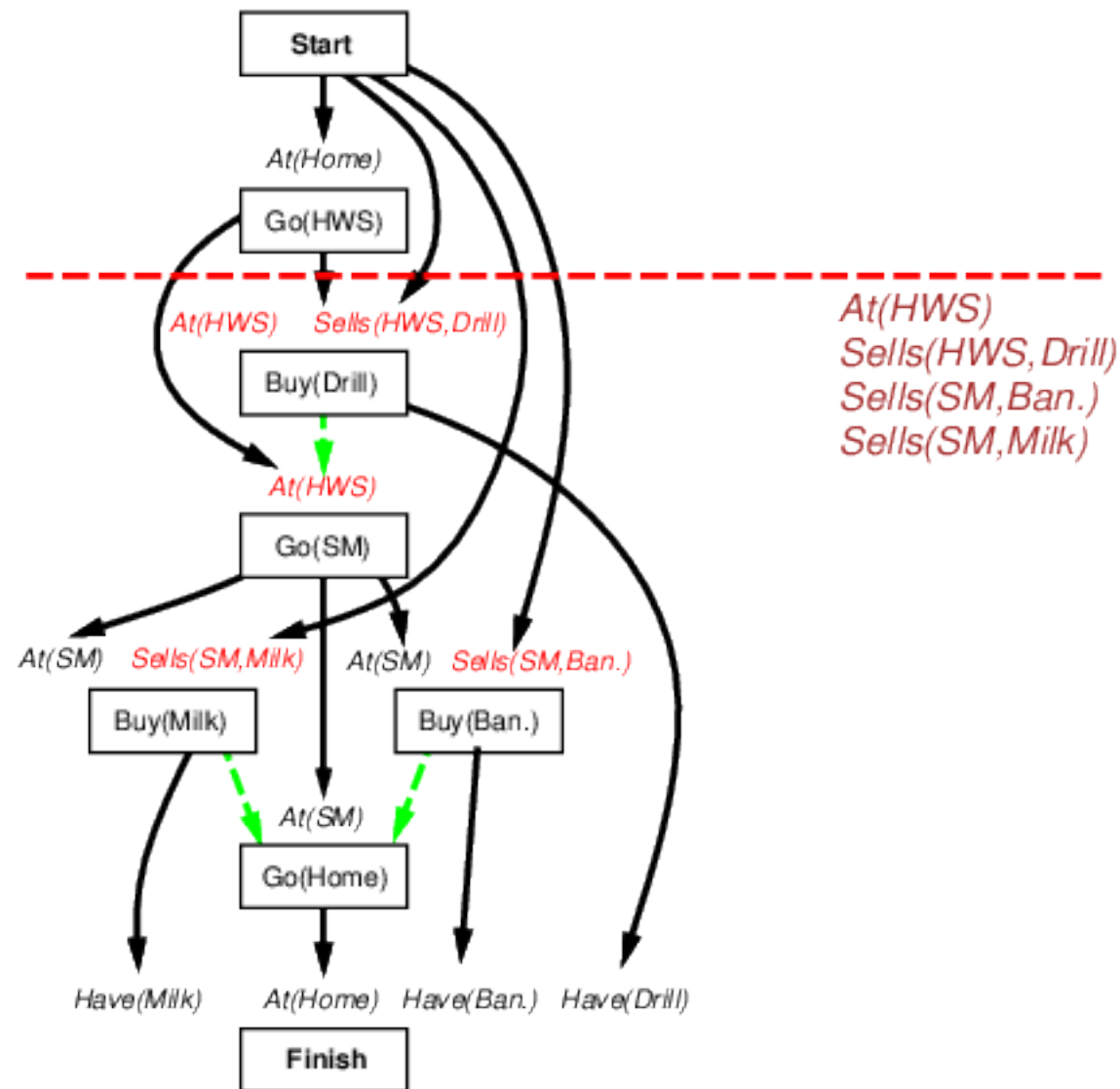
# Responding to Failure

- Run Partially Ordered Plans algorithms again
- Resume Partially Ordered Plans to achieve open conditions from current state
- IPEM (Integrated Planning, Execution, and Monitoring)
  - keep updating *Start* to match current state
  - links from actions replaced by links from *Start* when done

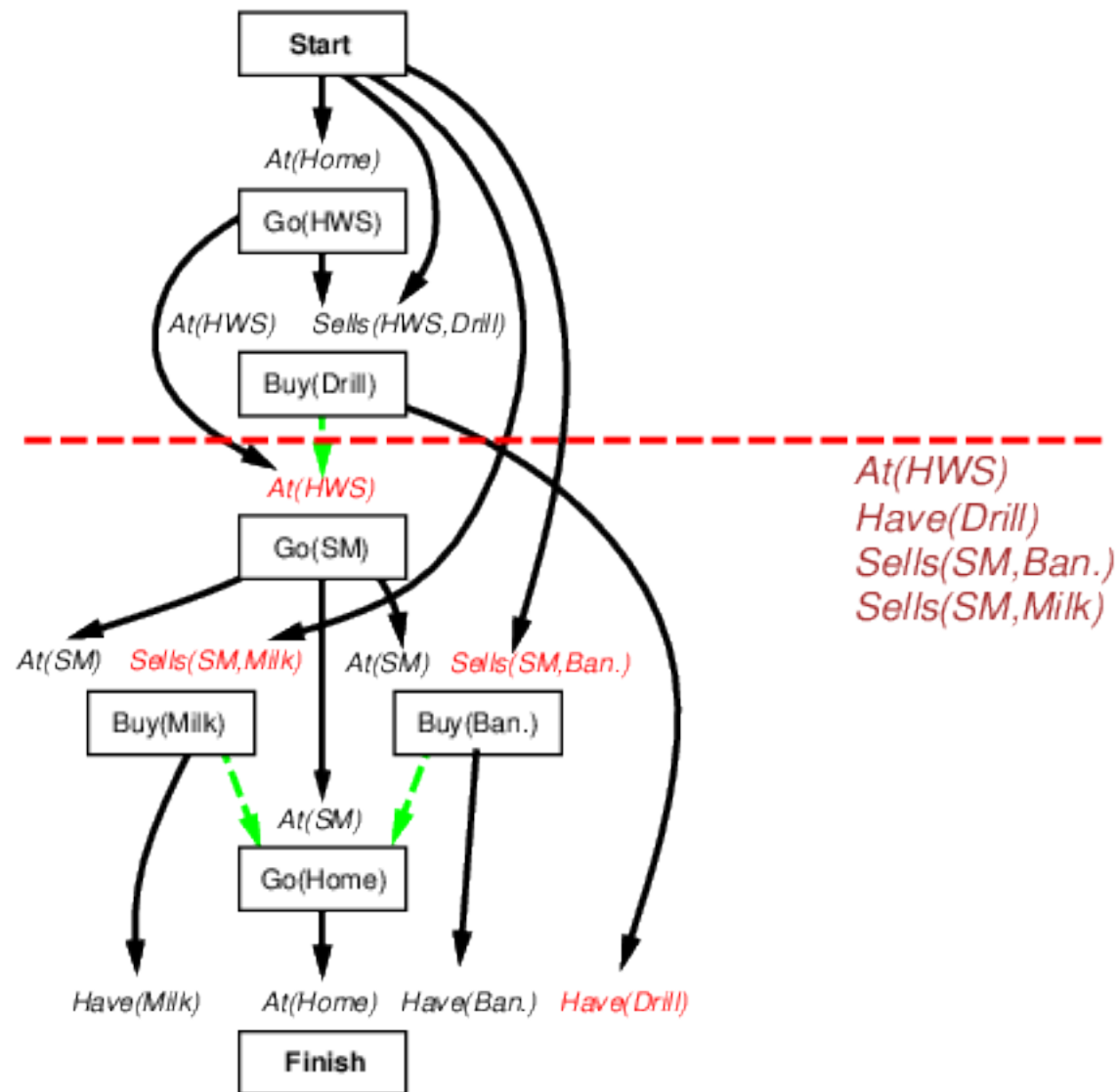
# Example



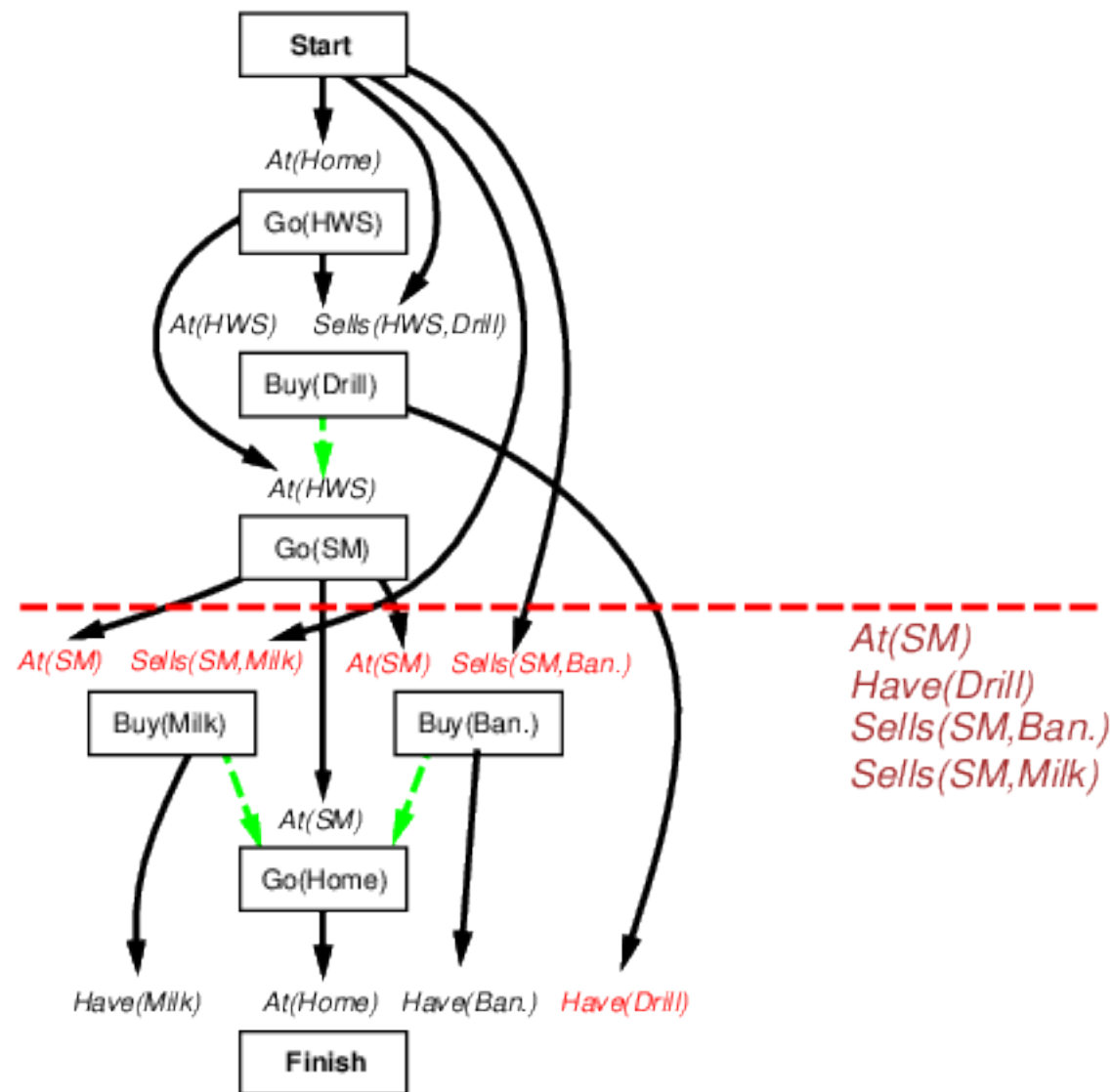
# Example



# Example

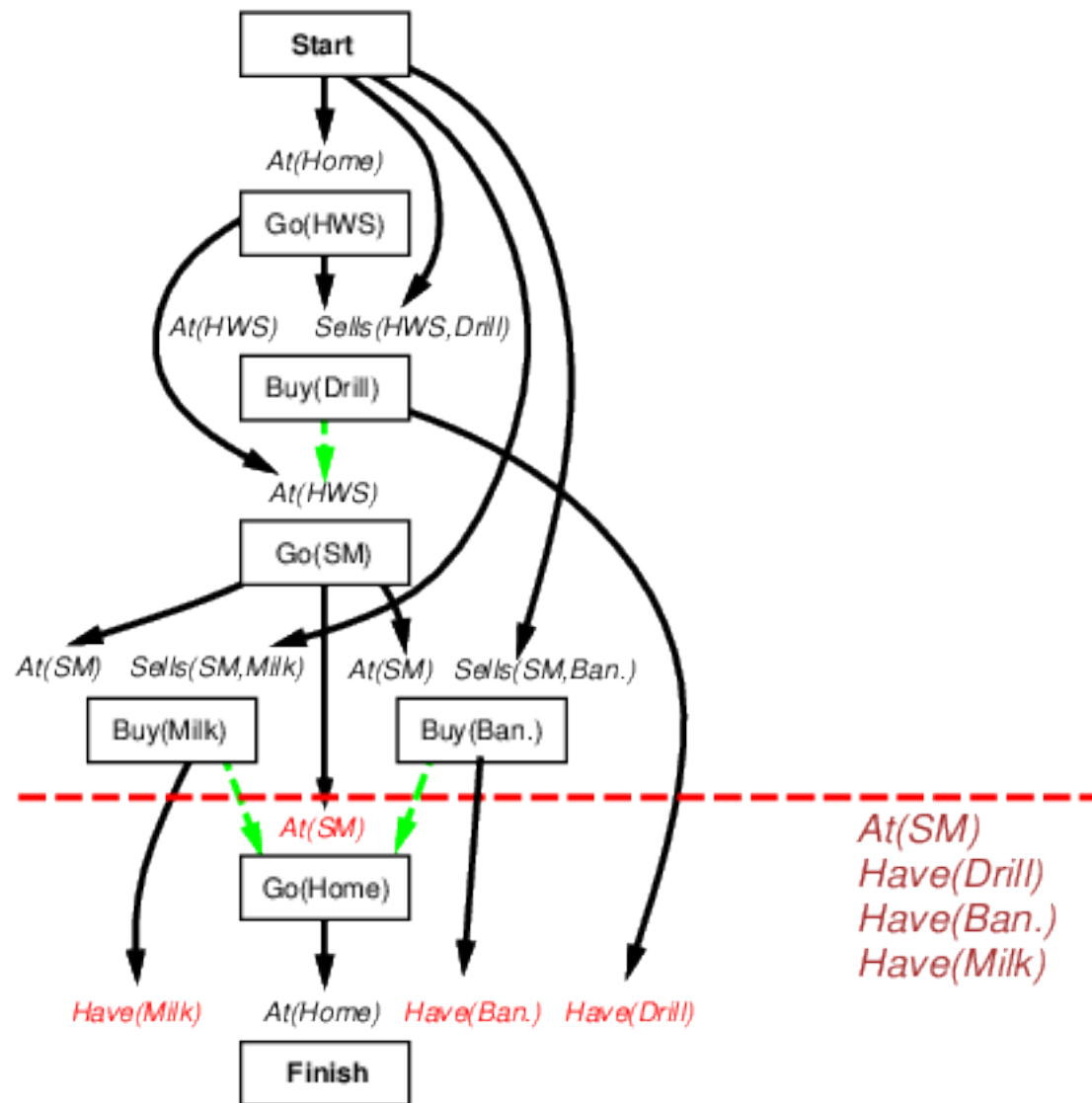


# Example



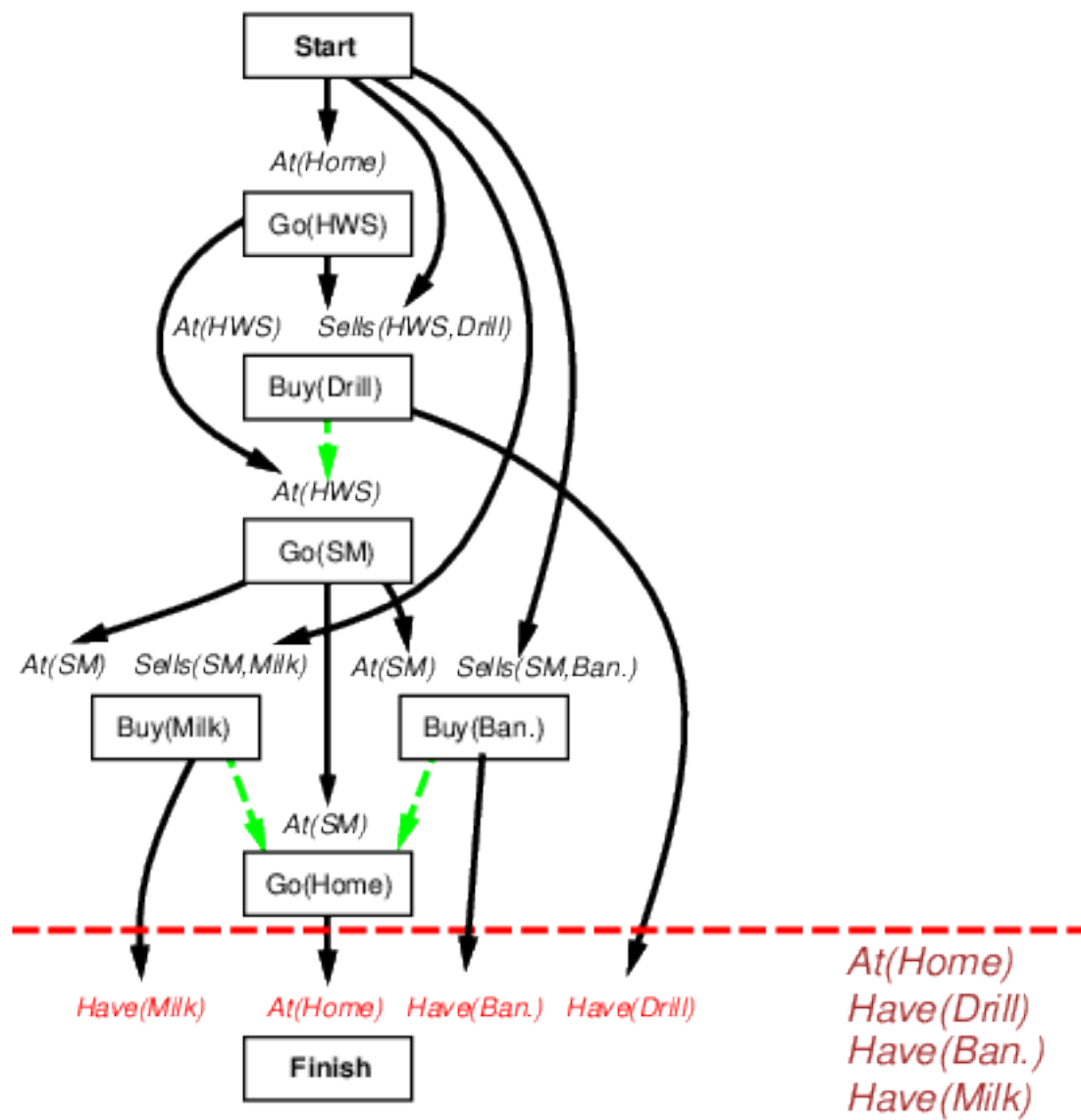


# Example



# Example

41



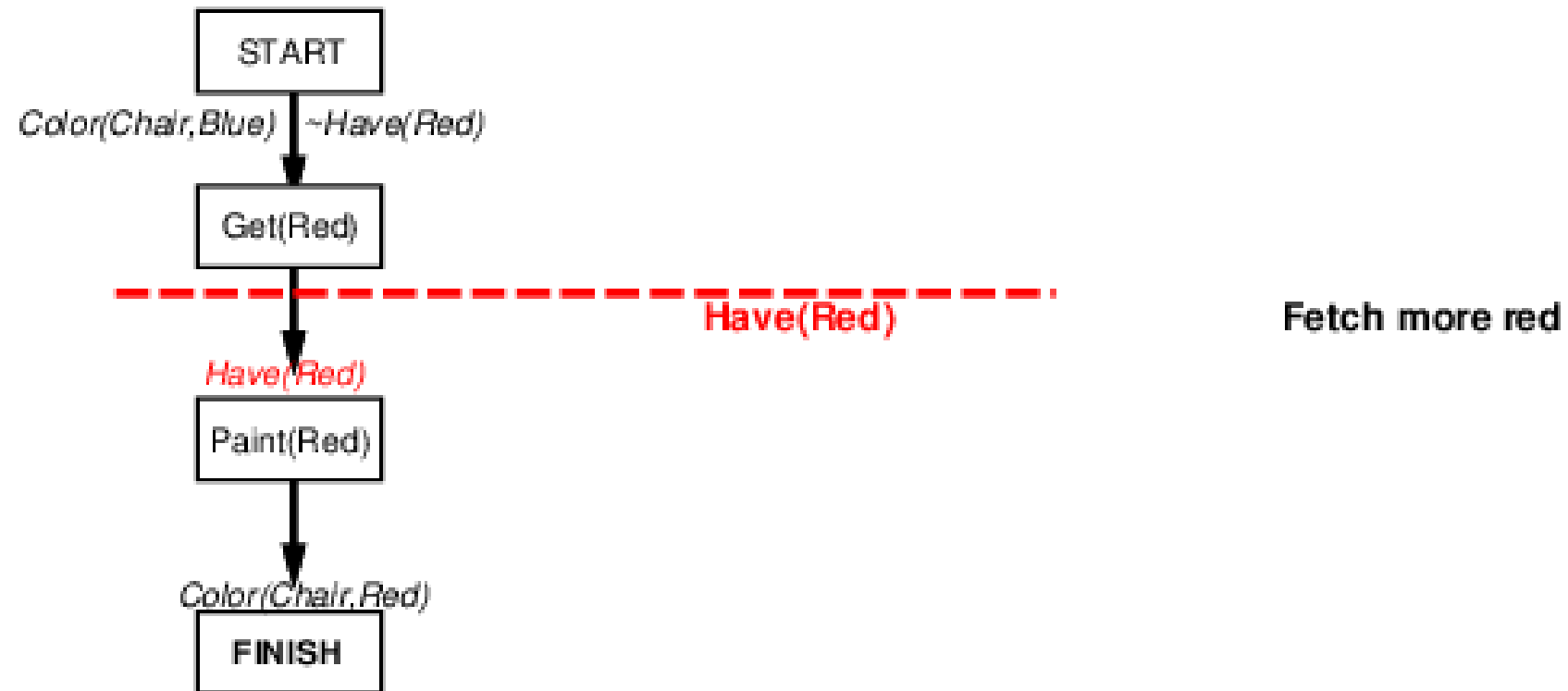
# Emergent Behavior

42

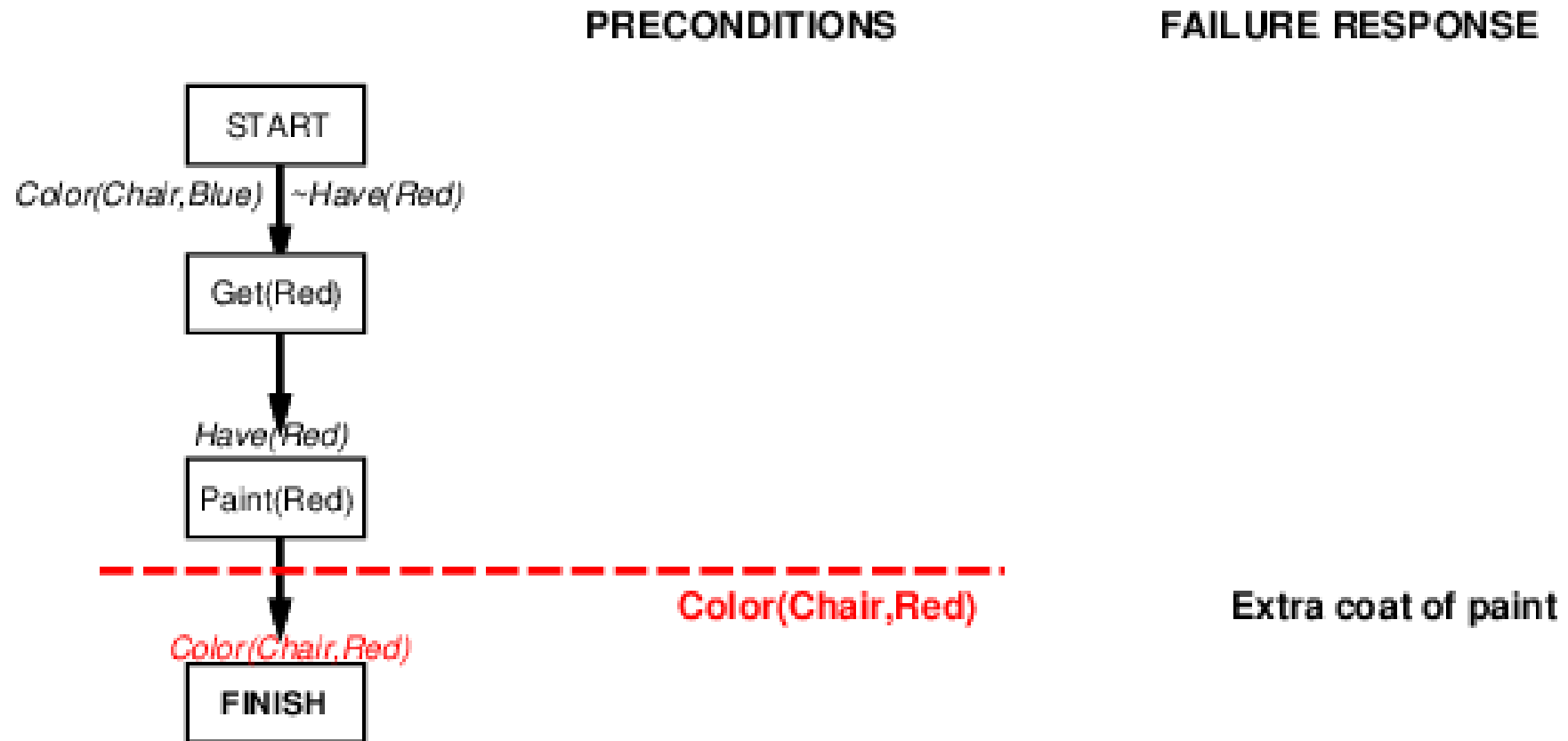


## PRECONDITIONS

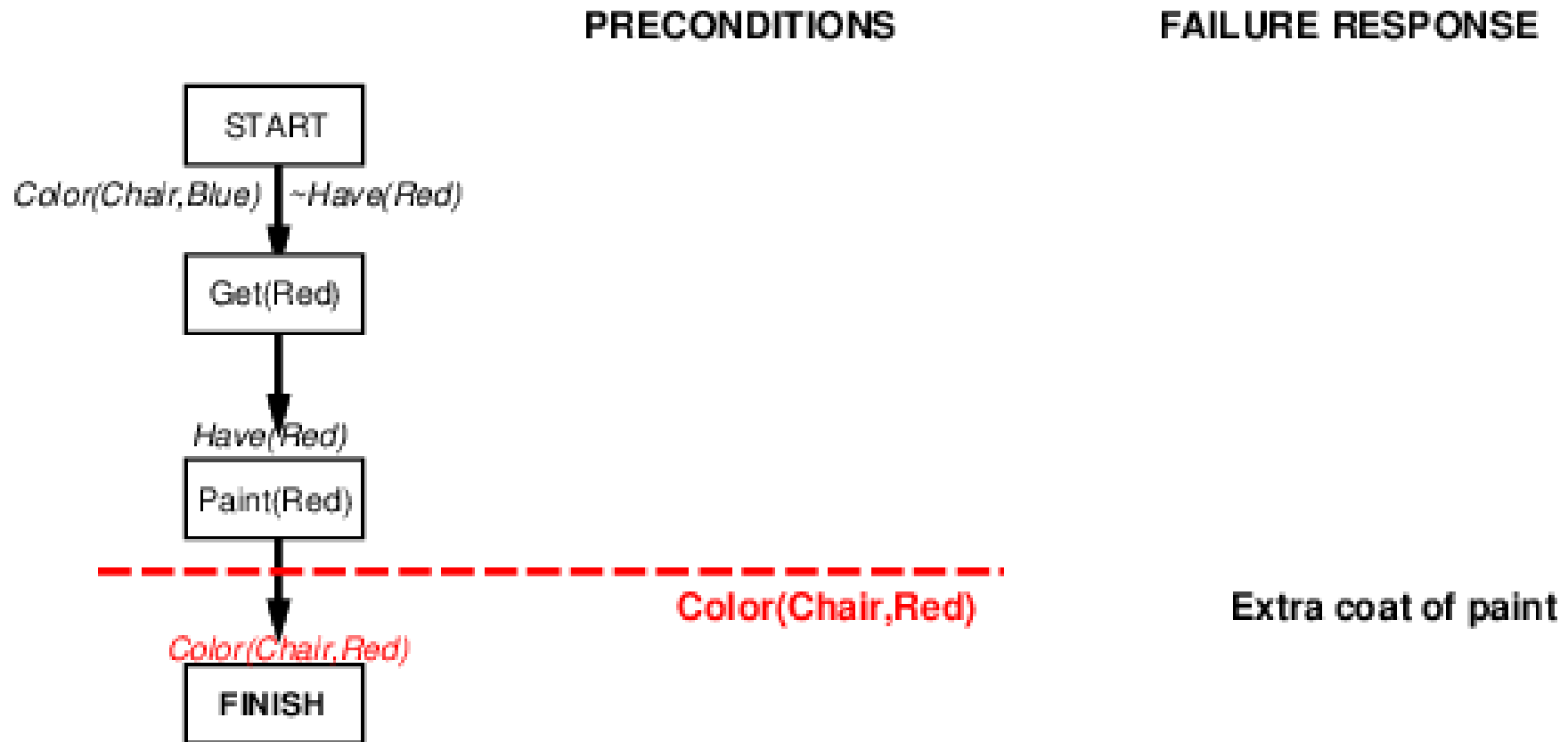
## FAILURE RESPONSE



# Emergent Behavior



# Emergent Behavior



- “Loop until success” behavior *emerges* from interaction between monitor/replan agent design and uncooperative environment

- Planning
    - break down problem into subgoals
    - search for plans for subgoals
    - merge sub-plans
  - Defined actions in terms of preconditions and effects
  - Partially Ordered Plans algorithm
  - Clobbering: need to deal with steps that destroy clausal link in plan
  - Real world: incomplete and incorrect information
- ⇒ conformant or conditional planning, monitoring and replanning