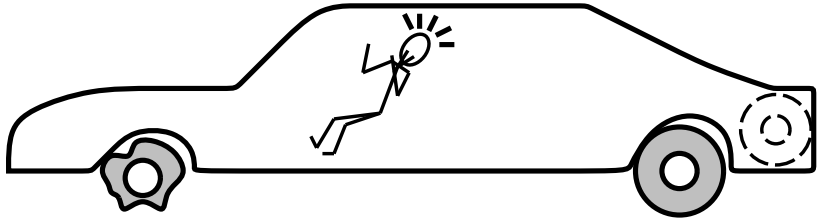
Planning/Acting in Real World

Chapter 11

Topics

- The real world
- Time, schedules, resources
- Hierarchical planning
- Planning in nondeterministic domains
- Multi-agent planning

The real world



On(x) ~Flat(x)

START

FINISH

~Flat(Spare) Intact(Spare) Off(Spare)
On(Tire1) Flat(Tire1)

On(x)

Remove(x)

Off(x) ClearHub

Off(x) ClearHub

Puton(x)

On(x) ~ClearHub

Intact(x) Flat(x)

Inflate(x)

~Flat(x)

Things go wrong

Incomplete information

Unknown preconditions, e.g., $Intact(Spare)$?

Disjunctive effects, e.g., $Inflate(x)$ causes

$Inflated(x) \lor SlowHiss(x) \lor Burst(x) \lor BrokenPump \lor \ldots$

Incorrect information

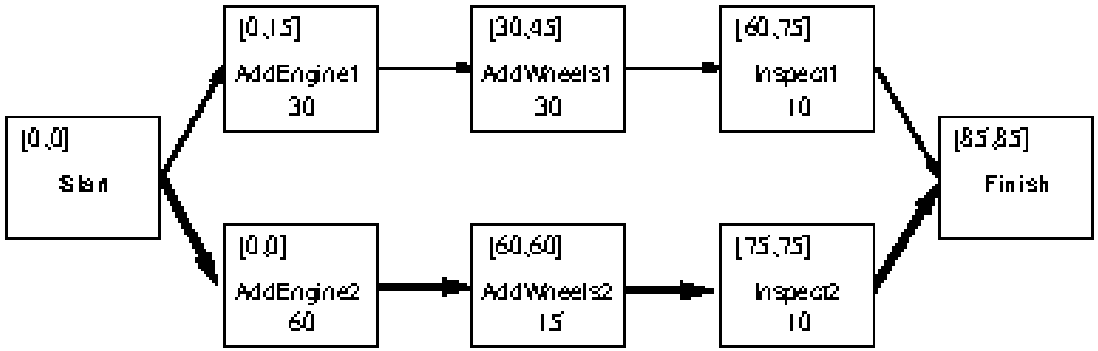
Current state incorrect, e.g., spare NOT intact

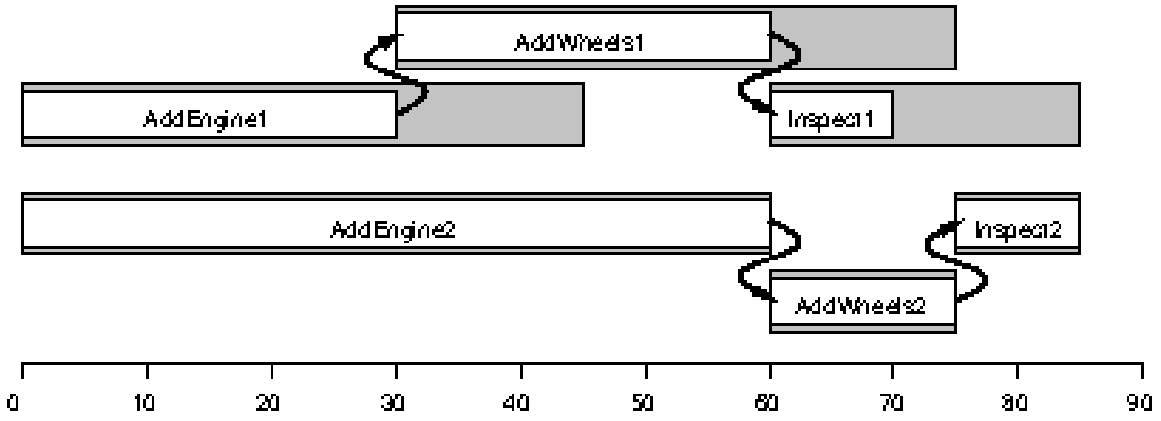
Missing/incorrect postconditions in operators

Qualification problem:

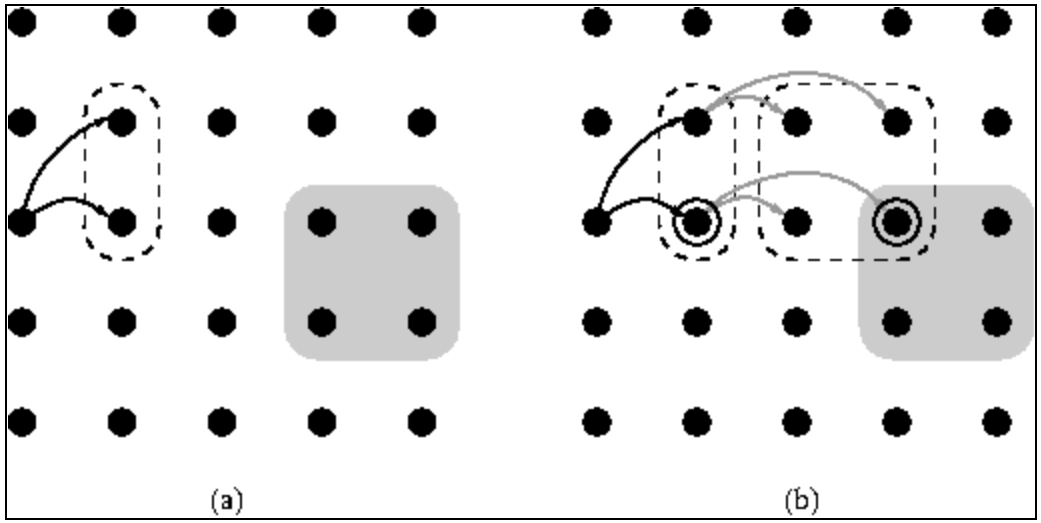
can never finish listing all the required preconditions and possible conditional outcomes of actions

Time, Schedule, Resources



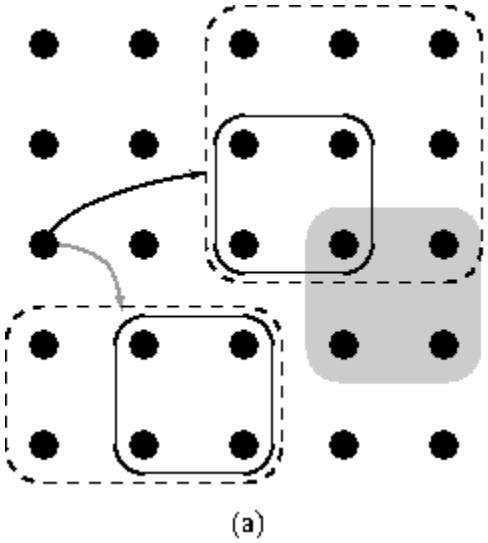


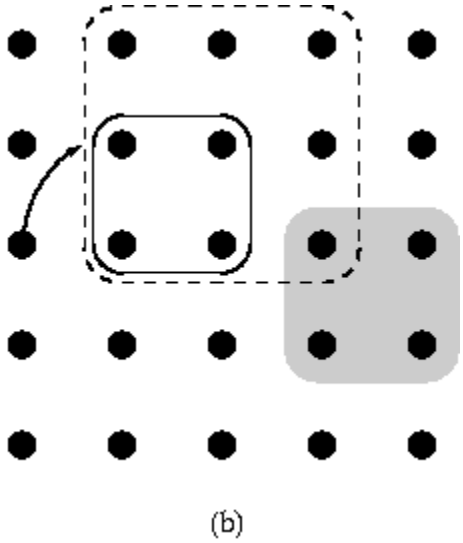
Hierarchical Planning – High Level Actions



Reachable states

Goal achievement





Planning and Acting with Nondeterminism

- Conformant planning (w/o observations)
- Contingency planning (for partially observable/nondeterministic environments)
- Online planning/replanning (for unknown environments)

Indeterminacy in the World

Bounded indeterminacy: actions can have unpredictable effects, but the possible effects can be listed in the action description axioms

Unbounded indeterminacy: set of possible preconditions or effects either is unknown or is too large to be completely enumerated

Closely related to qualification problem

Solutions

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.,

$[Check(Tire1), \textbf{if}\ Intact(Tire1)\ \textbf{then}\ Inflate(Tire1)\ \textbf{else}\ CallAAA$

Expensive because it plans for many unlikely cases

Monitoring/Replanning

Assume normal states, outcomes

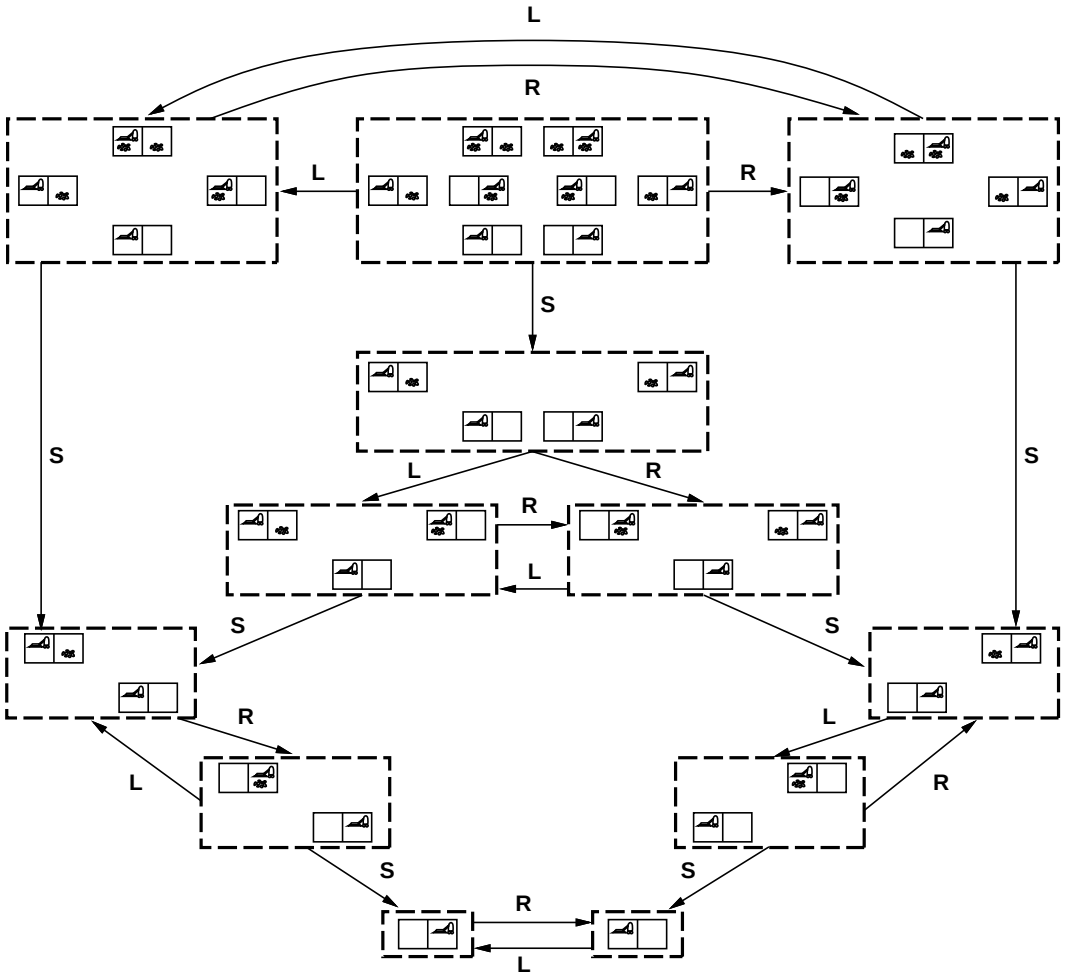
Check progress *during execution*, replan if necessary

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)

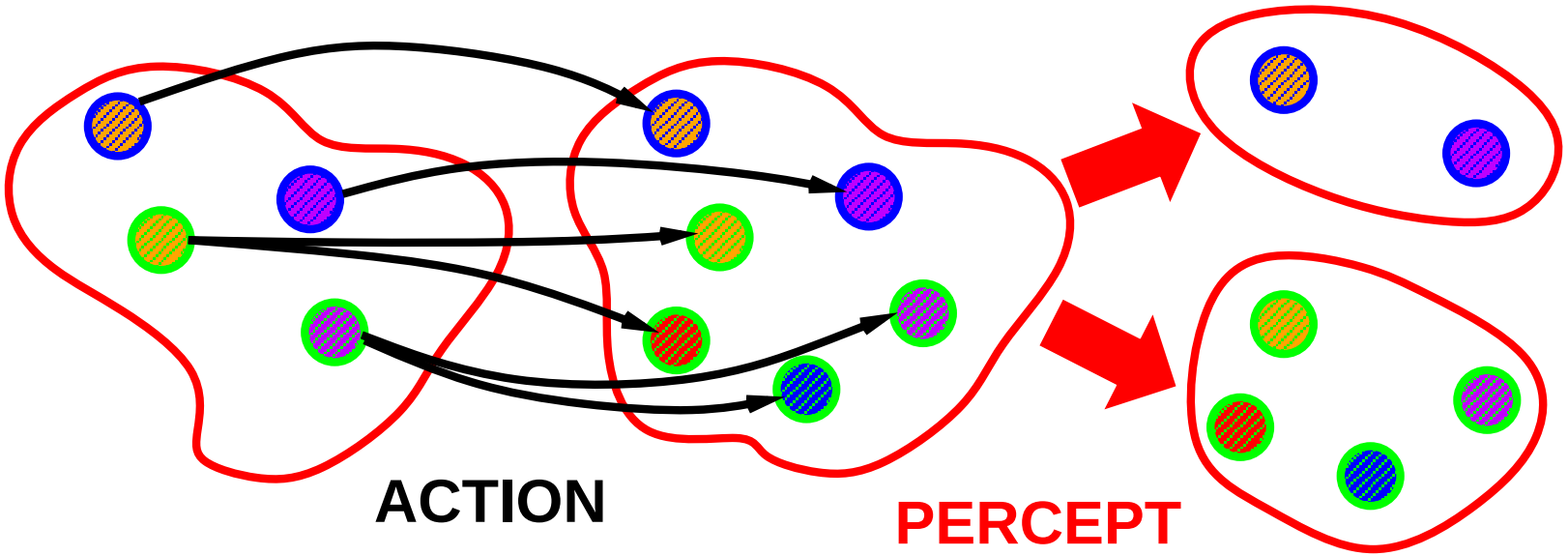


Conditional planning

If the world is nondeterministic or partially observable

then percepts usually *provide information*,

i.e., *split up* the belief state



Conditional planning (con't.)

Conditional plans check (any consequence of KB +) percept

$[\ldots, \textbf{if } C \textbf{ then } Plan_A \textbf{ else } Plan_B, \ldots]$

Execution: check C against current KB, execute "then" or "else"

Conditional planning (con't.)

Need to handle nondeterminism by building into the plan conditional steps that check the state of the environment at run time, and then decide what to do.

Augment STRIPS to allow for nondeterminism:

Add Disjunctive effects (e.g., to model when action sometimes fails):

$Action(Left,$ PRECOND:$AtR,$ EFFECT: $AtL \vee AtR)$

Add Conditional effects (i.e., depends on state in which it's executed):

Form: **when** $<condition>$: $<effect>$

$Action(Suck,$ PRECOND:,

EFFECT: (**when** AtL: $CleanL) \wedge$ (**when** AtR: $CleanR))$

Create Conditional steps:

if $<test>$ **then** $plan\text{-}A$ **else** $plan\text{-}B$

Conditional planning (con't.)

Need *some* plan for *every* possible percept and action outcome

(Cf. game playing: *some* response for *every* opponent move)
(Cf. backward chaining: *some* rule such that *every* premise satisfied

Use: AND–OR tree search (very similar to backward chaining algorithm)
Similar to game tree in minimax search
Differences: MAX and MIN nodes become OR and AND nodes

Robot takes action in "state" nodes.

Nature decides outcome at "chance" nodes.

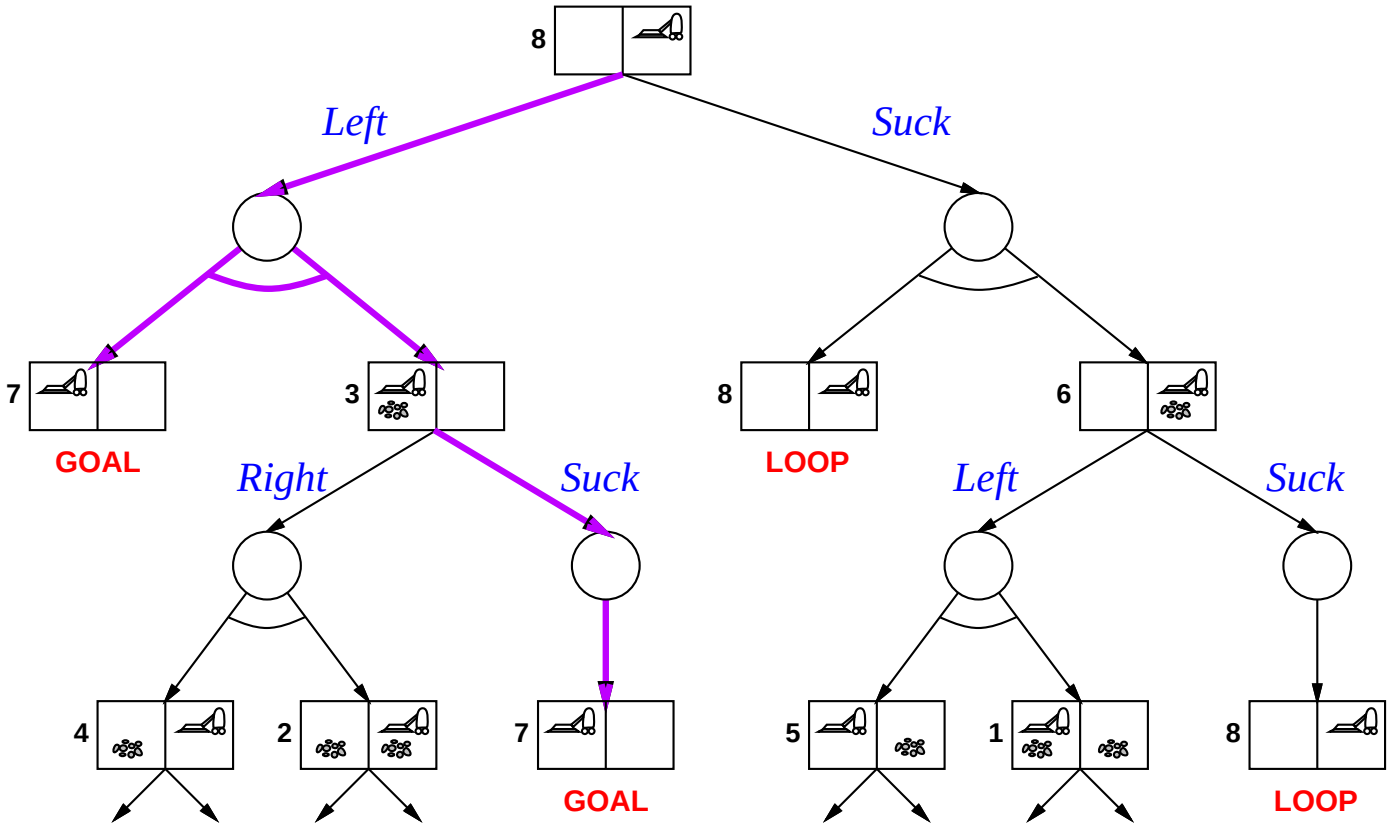
Plan needs to take *some* action at every state it reaches (i.e., OR nodes)

Plan must handle *every* outcome for the action it takes (i.e., AND nodes)

Solution is a subtree with (1) goal node at every leaf, (2) one action specified at each state node, and (3) includes every outcome branch at chance nodes.

Example: "Game Tree", Fully Observable World

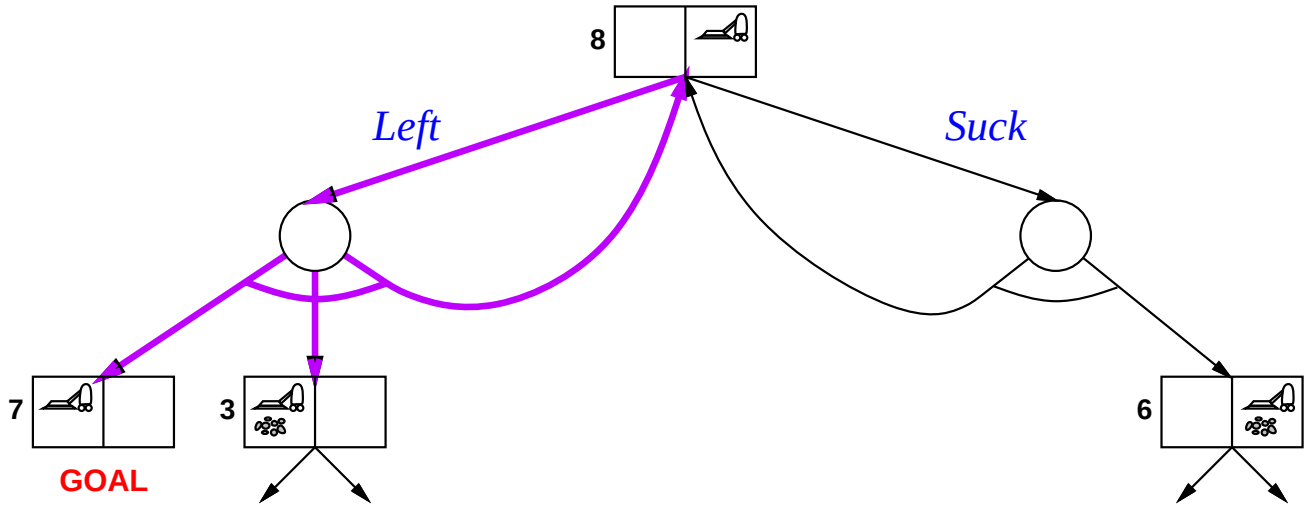
Double Murphy: sucking or arriving may dirty a clean square



Plan: [*Left*, **if** $AtL \wedge CleanL \wedge CleanR$ **then** [] **else** *Suck*]

Example

Triple Murphy: also sometimes stays put instead of moving



$[L_1 : \ Left, \textbf{if } AtR \textbf{ then } L_1 \textbf{ else } [\textbf{if } CleanL \textbf{ then } [\,] \textbf{ else } Suck]]$

or $[\textbf{while } AtR \textbf{ do } [Left], \textbf{if } CleanL \textbf{ then } [\,] \textbf{ else } Suck]$

"Infinite loop" but will eventually work unless action always fails

Execution Monitoring

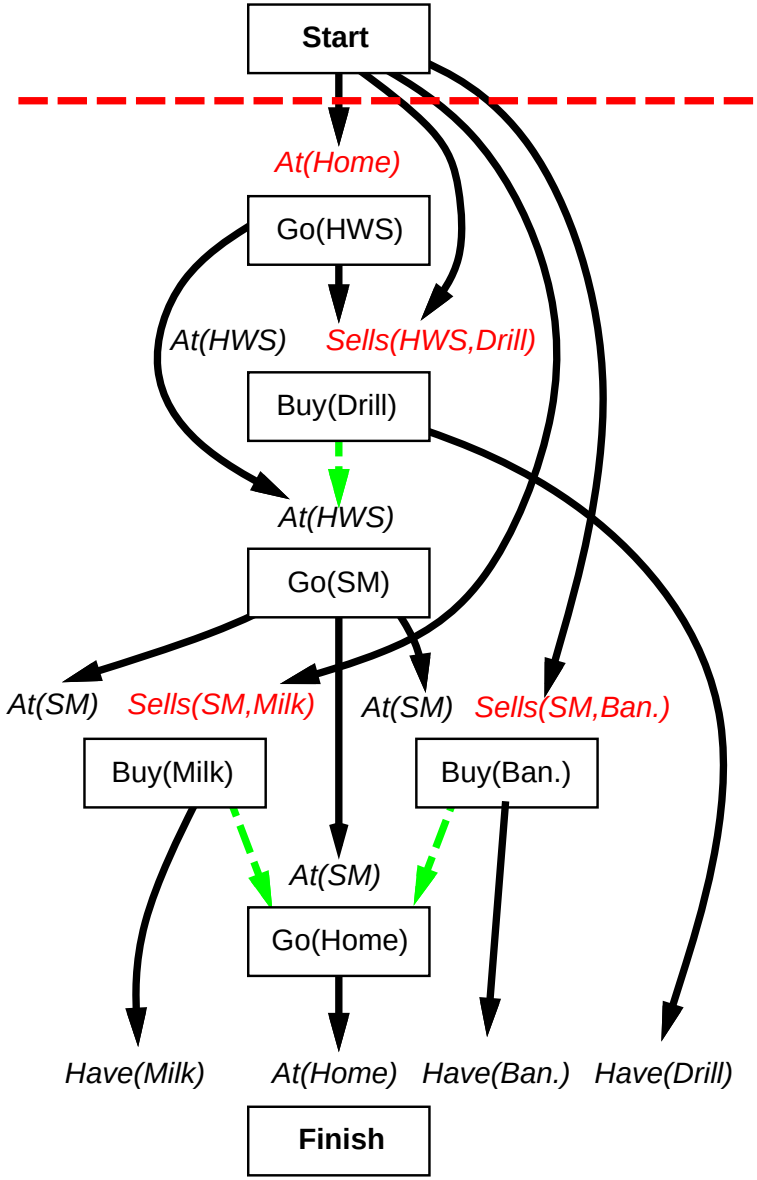
"Failure" = preconditions of *remaining plan* not met

Preconditions of remaining plan
= all preconditions of remaining steps not achieved by remaining steps
= all causal links *crossing* current time point

On failure, resume POP to achieve open conditions from current state

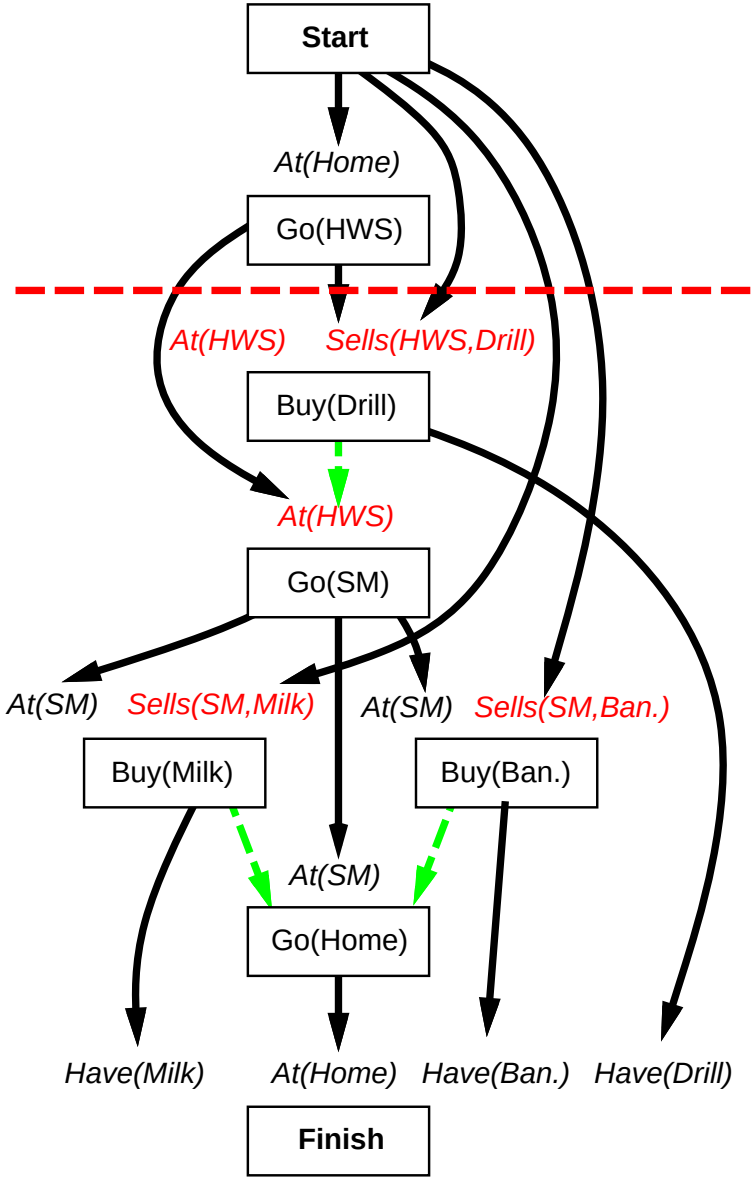
IPEM (Integrated Planning, Execution, and Monitoring):
keep updating $Start$ to match current state
links from actions replaced by links from $Start$ when done

Example



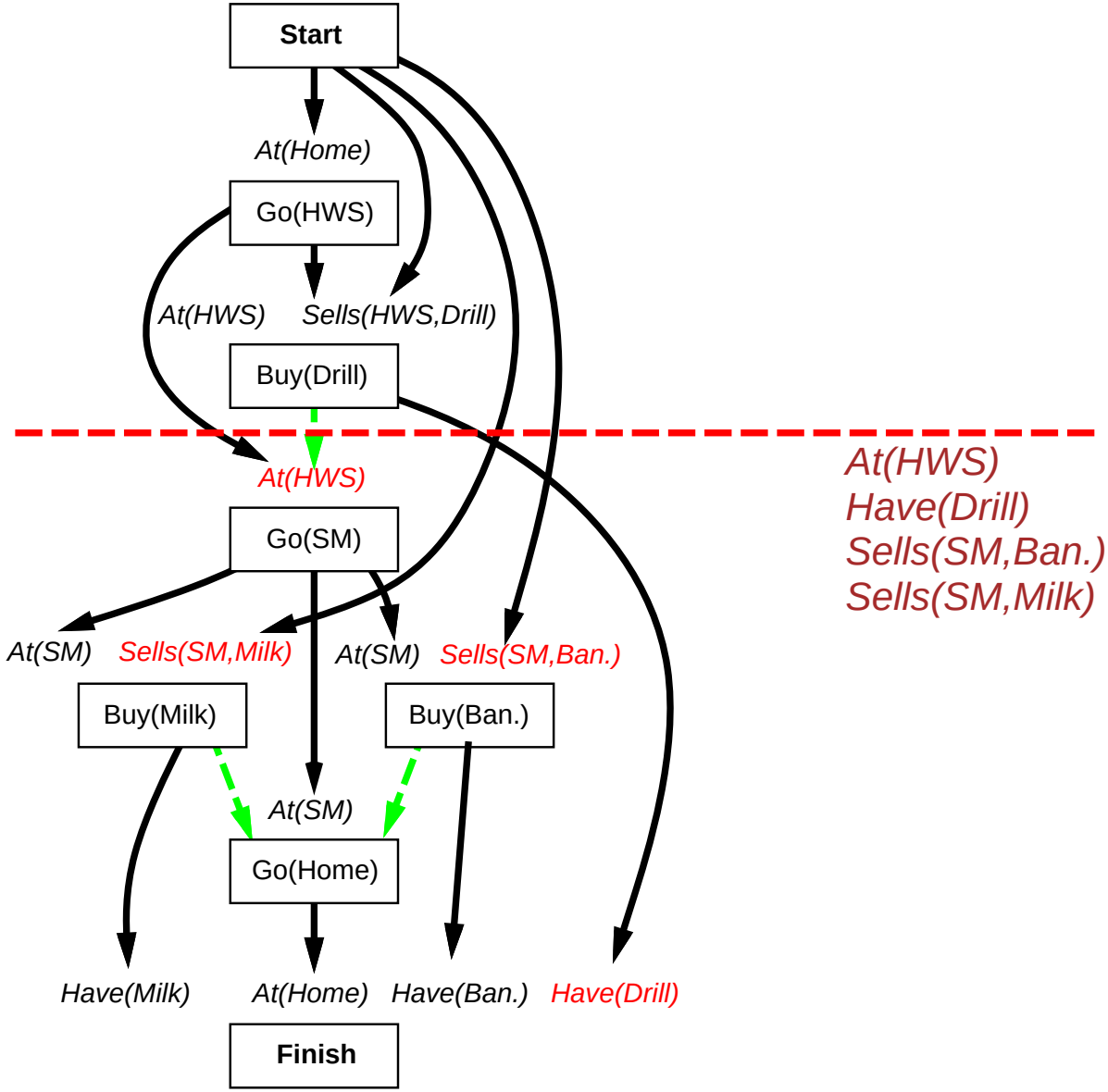
At(Home)
Sells(HWS,Drill)
Sells(SM,Ban.)
Sells(SM,Milk)

Example

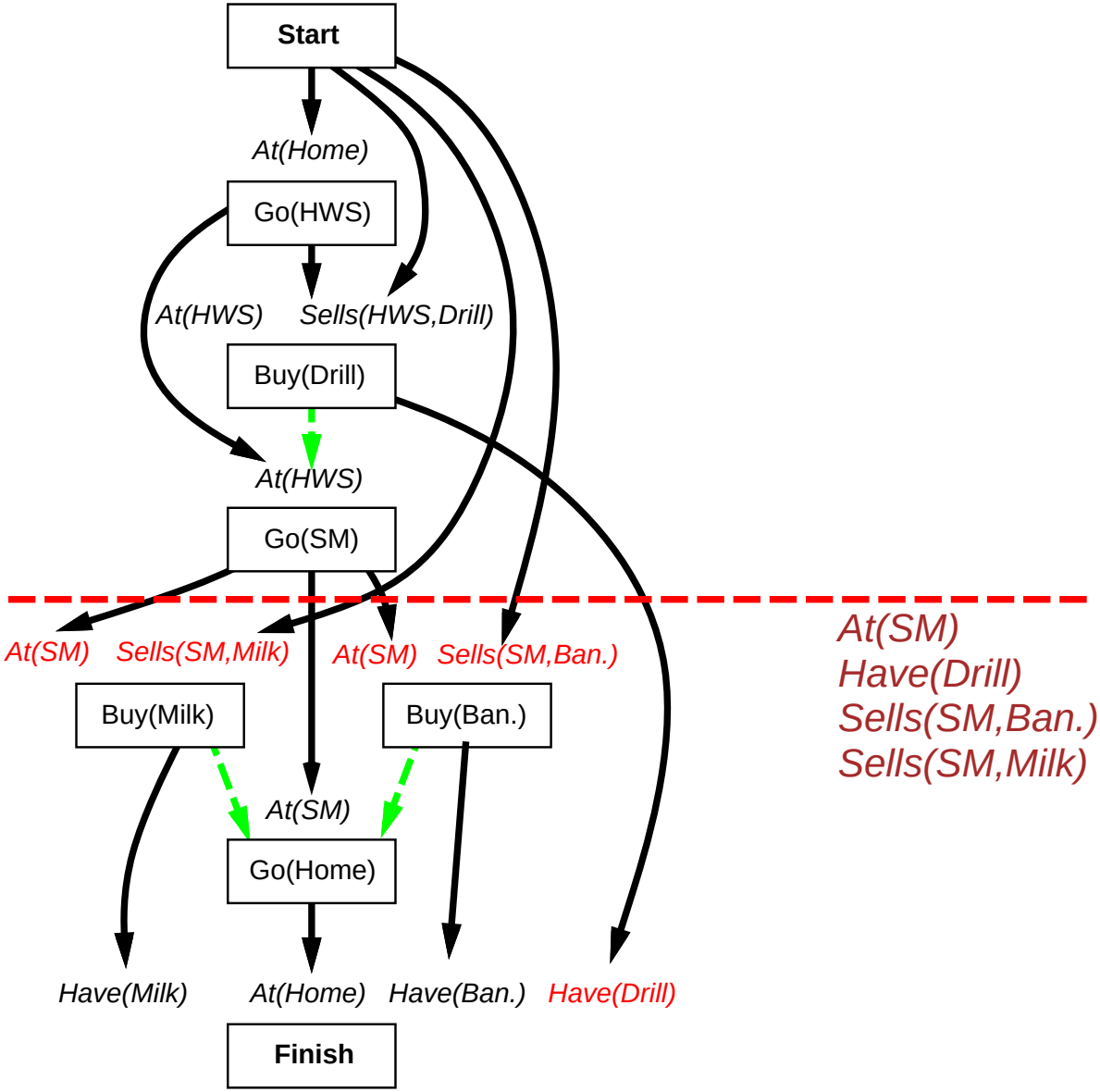


At(HWS)
Sells(HWS,Drill)
Sells(SM,Ban.)
Sells(SM,Milk)

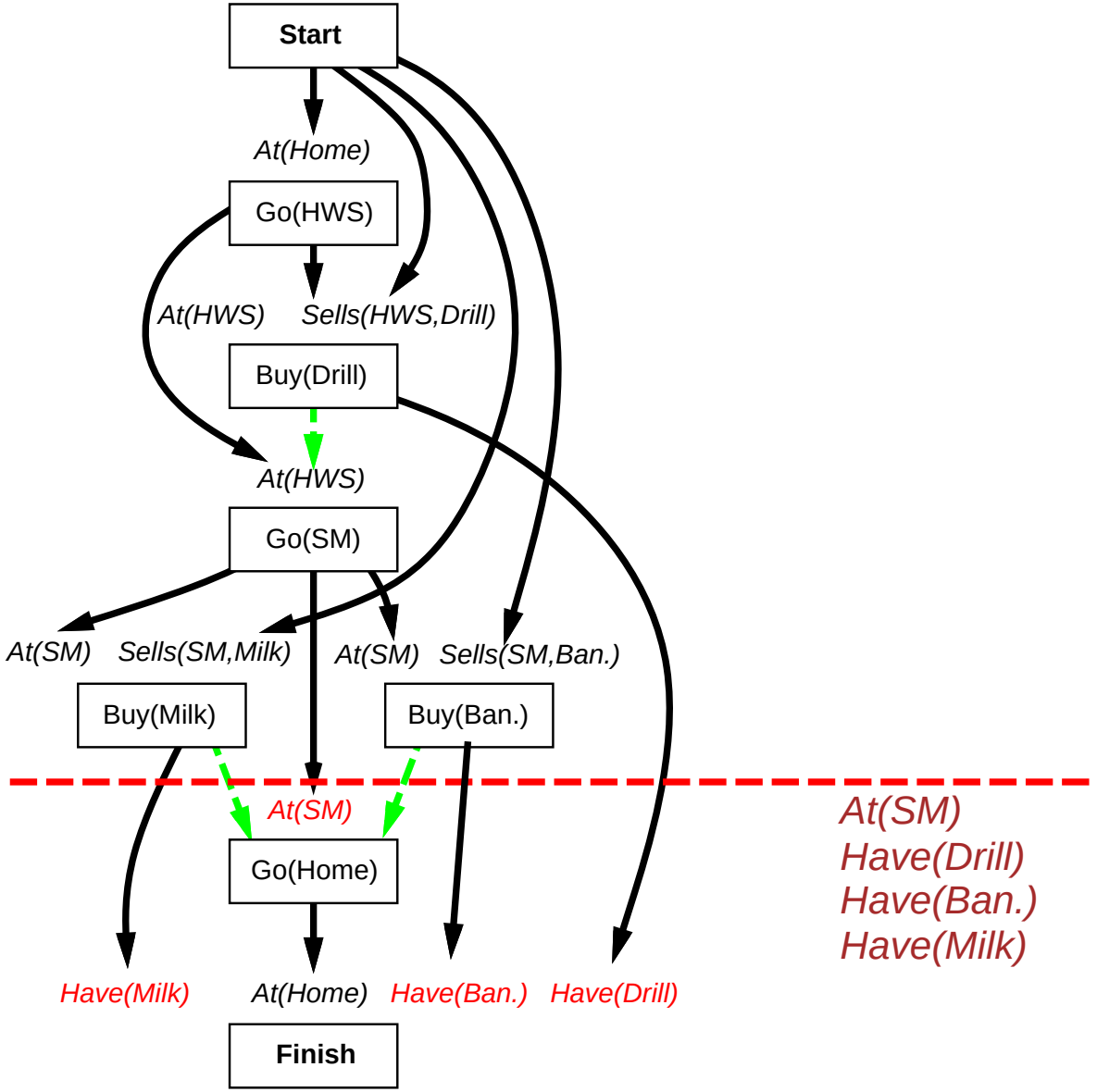
Example


Start
At(Home)
Go(HWS)
At(HWS)
Sells(HWS,Drill)
Buy(Drill)
At(HWS)
Go(SM)
At(SM)
Sells(SM,Milk)
At(SM)
Sells(SM,Ban.)
Buy(Milk)
Buy(Ban.)
At(SM)
Go(Home)
Have(Milk)
At(Home)
Have(Ban.)
Have(Drill)
Finish
At(HWS)
Have(Drill)
Sells(SM,Ban.)
Sells(SM,Milk)

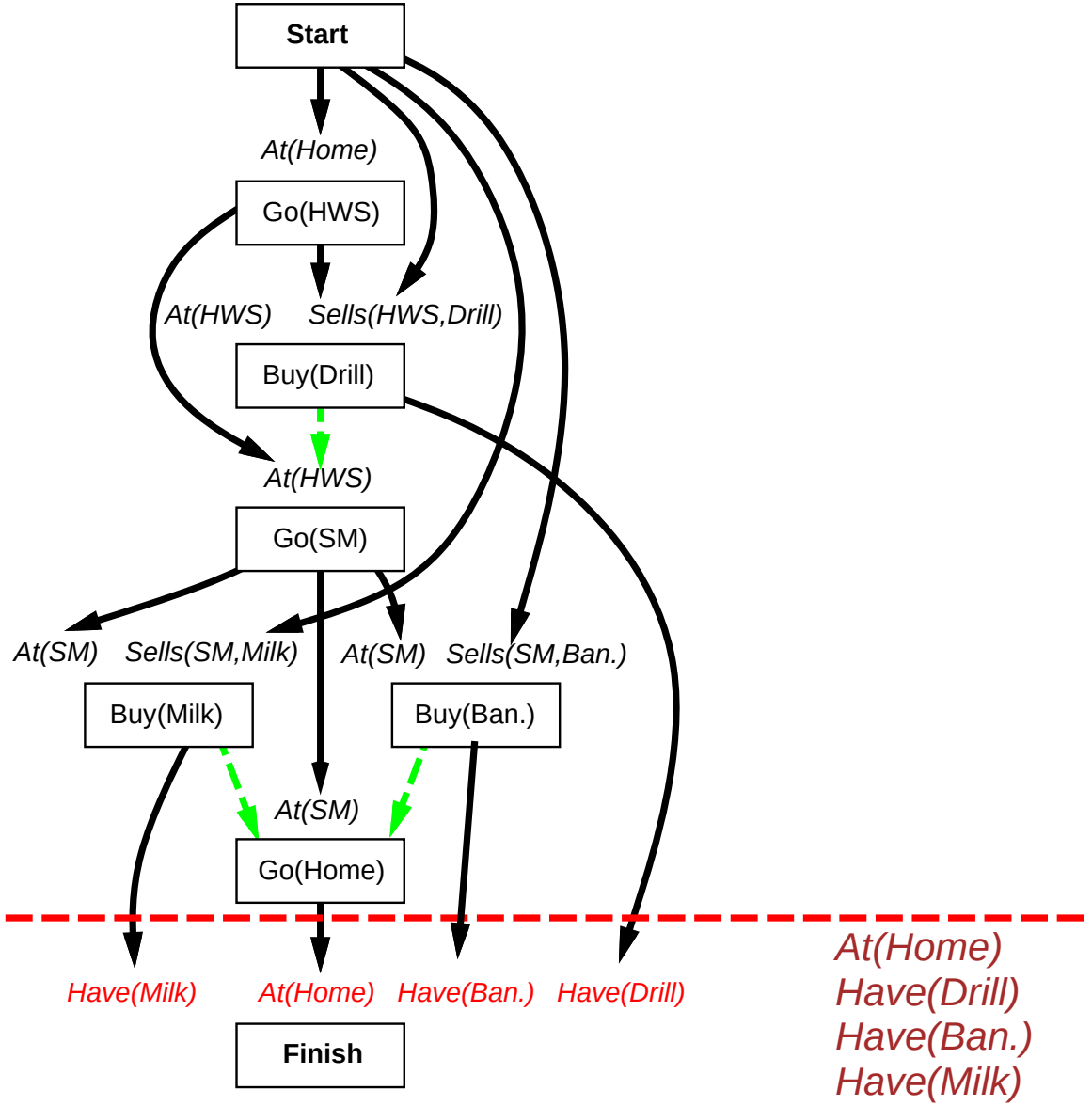
Example


Start
At(Home)
Go(HWS)
At(HWS)
Sells(HWS,Drill)
Buy(Drill)
At(HWS)
Go(SM)
At(SM)
Sells(SM,Milk)
At(SM)
Sells(SM,Ban.)
Buy(Milk)
Buy(Ban.)
At(SM)
Go(Home)
Have(Milk)
At(Home)
Have(Ban.)
Have(Drill)
Finish
At(SM)
Have(Drill)
Sells(SM,Ban.)
Sells(SM,Milk)

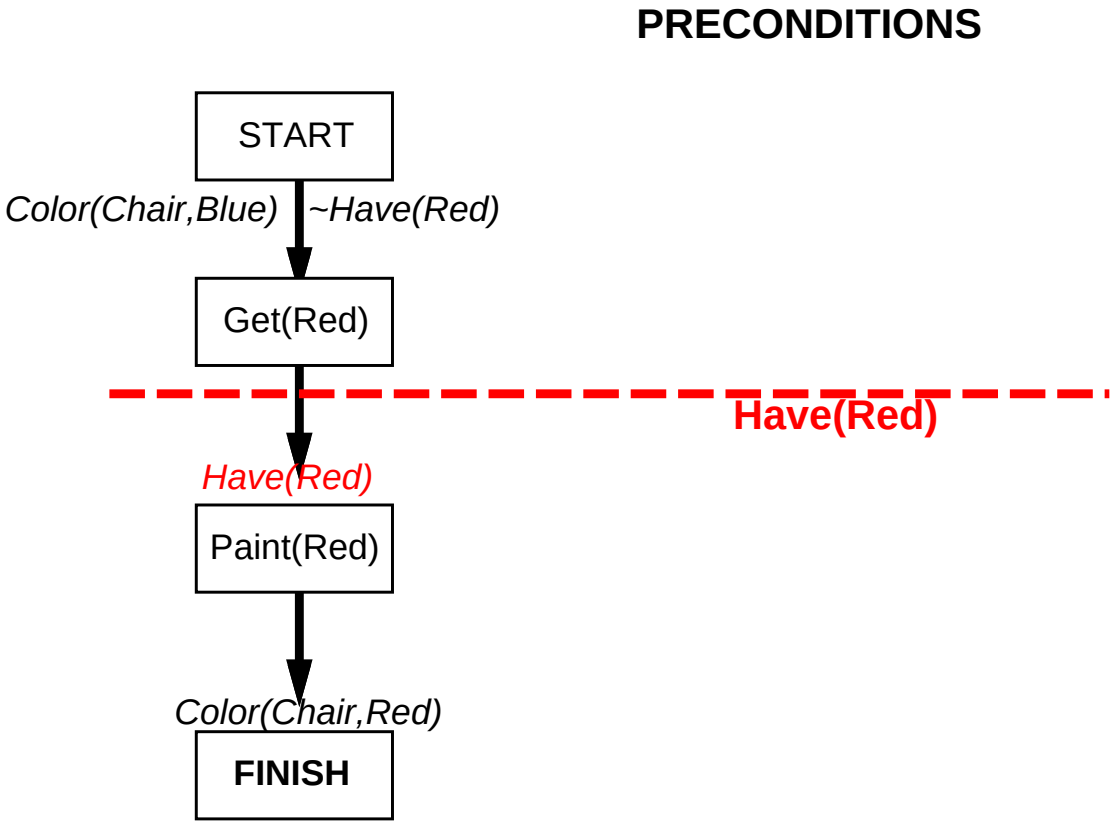
Example


Start
At(Home)
Go(HWS)
At(HWS)
Sells(HWS,Drill)
Buy(Drill)
At(HWS)
Go(SM)
At(SM)
Sells(SM,Milk)
At(SM)
Sells(SM,Ban.)
Buy(Milk)
Buy(Ban.)
At(SM)
Go(Home)
Have(Milk)
At(Home)
Have(Ban.)
Have(Drill)
Finish
At(SM)
Have(Drill)
Have(Ban.)
Have(Milk)

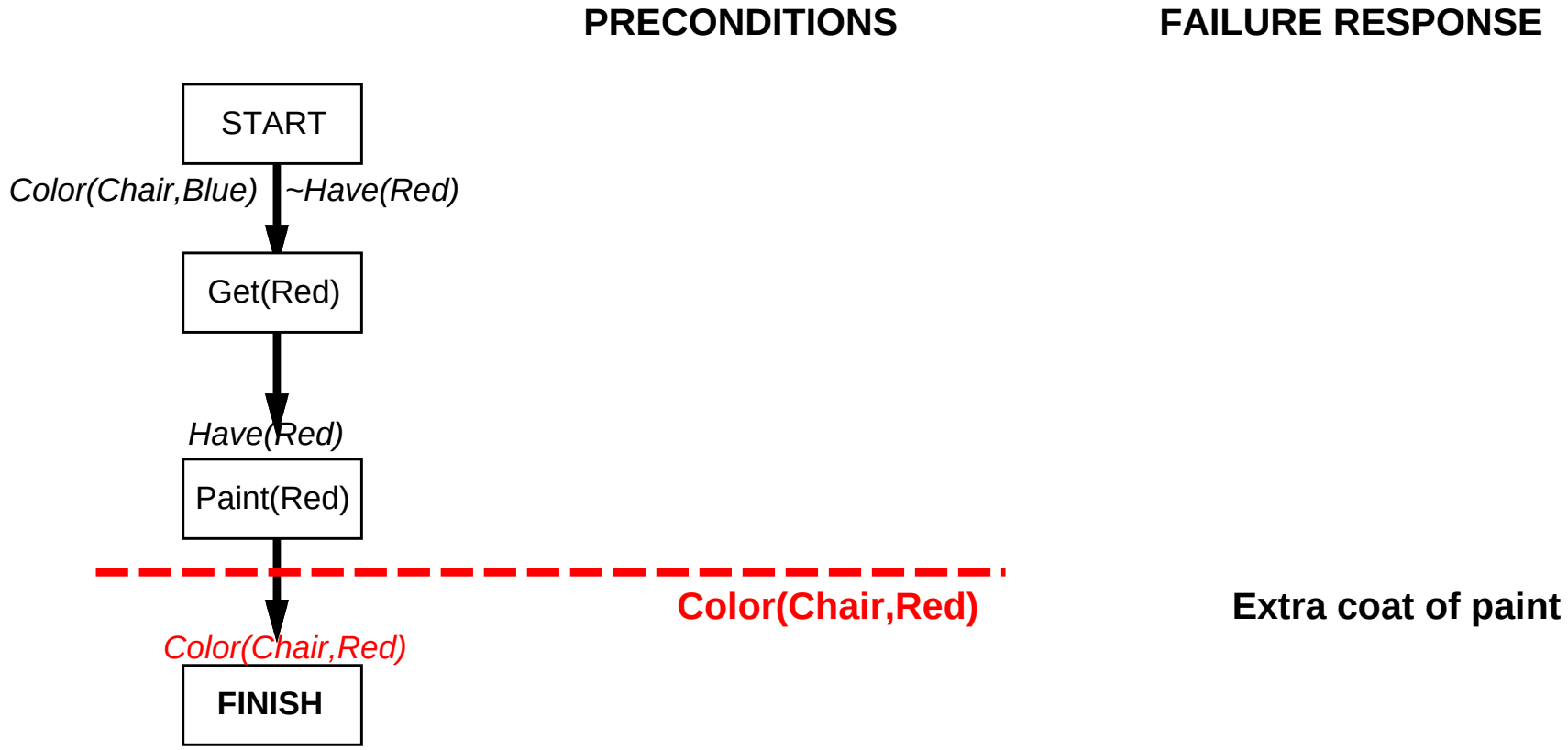
Example


Start
At(Home)
Go(HWS)
At(HWS)
Sells(HWS,Drill)
Buy(Drill)
At(HWS)
Go(SM)
At(SM)
Sells(SM,Milk)
At(SM)
Sells(SM,Ban.)
Buy(Milk)
Buy(Ban.)
At(SM)
Go(Home)
Have(Milk)
At(Home)
Have(Ban.)
Have(Drill)
Finish
At(Home)
Have(Drill)
Have(Ban.)
Have(Milk)

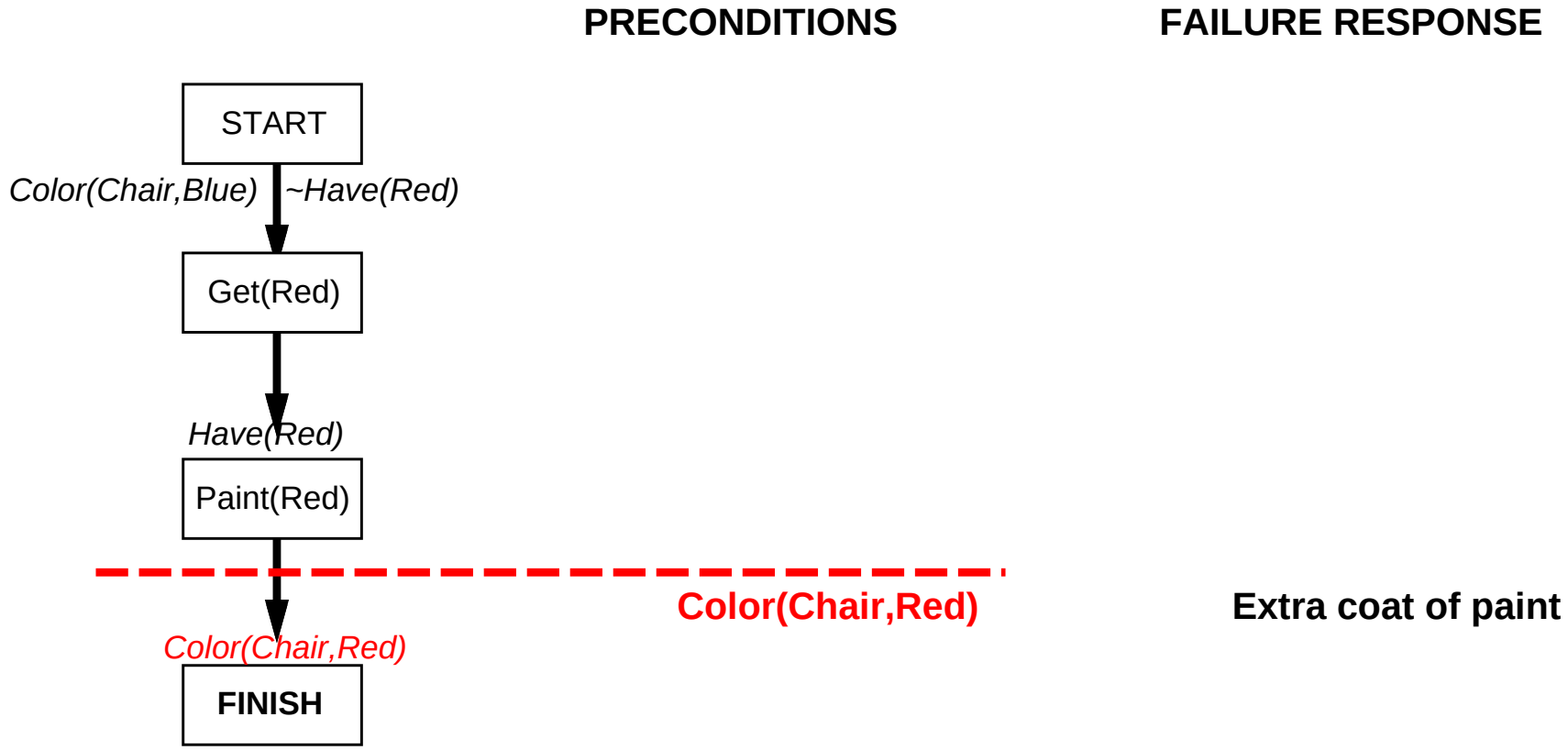
Emergent behavior



Emergent behavior



Emergent behavior



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

Summarizing Example

Assume: You have a chair, a table, and some cans of paint; all colors are unknown. Goal: chair and table have same color.

How would each of the following handle this problem?

Classical planning:

Summarizing Example

Assume: You have a chair, a table, and some cans of paint; all colors are unknown. Goal: chair and table have same color.

How would each of the following handle this problem?

Classical planning: Can't handle it, because initial state isn't fully specified.

Summarizing Example

Assume: You have a chair, a table, and some cans of paint; all colors are unknown. Goal: chair and table have same color.

How would each of the following handle this problem?

Classical planning: Can't handle it, because initial state isn't fully specified.

Sensorless/Conformant planning:

Summarizing Example

Assume: You have a chair, a table, and some cans of paint; all colors are unknown. Goal: chair and table have same color.

How would each of the following handle this problem?

Classical planning: Can't handle it, because initial state isn't fully specified.

Sensorless/Conformant planning: Open can of paint and apply it to both chair and table.

Summarizing Example

Assume: You have a chair, a table, and some cans of paint; all colors are unknown. Goal: chair and table have same color.

How would each of the following handle this problem?

Classical planning: Can't handle it, because initial state isn't fully specified.

Sensorless/Conformant planning: Open can of paint and apply it to both chair and table.

Conditional planning:

Summarizing Example

Assume: You have a chair, a table, and some cans of paint; all colors are unknown. Goal: chair and table have same color.

How would each of the following handle this problem?

Classical planning: Can't handle it, because initial state isn't fully specified.

Sensorless/Conformant planning: Open can of paint and apply it to both chair and table.

Conditional planning: Sense the color of the table and chair. If same, then we're done. If not, sense labels on the paint cans; if there is a can that is the same color as one piece of furniture, then apply the paint to the other piece. Otherwise, paint both pieces with any color.

Summarizing Example

Assume: You have a chair, a table, and some cans of paint; all colors are unknown. Goal: chair and table have same color.

How would each of the following handle this problem?

Classical planning: Can't handle it, because initial state isn't fully specified.

Sensorless/Conformant planning: Open can of paint and apply it to both chair and table.

Conditional planning: Sense the color of the table and chair. If same, then we're done. If not, sense labels on the paint cans; if there is a can that is the same color as one piece of furniture, then apply the paint to the other piece. Otherwise, paint both pieces with any color.

Monintoring/replanning:

Summarizing Example

Assume: You have a chair, a table, and some cans of paint; all colors are unknown. Goal: chair and table have same color.

How would each of the following handle this problem?

Classical planning: Can't handle it, because initial state isn't fully specified.

Sensorless/Conformant planning: Open can of paint and apply it to both chair and table.

Conditional planning: Sense the color of the table and chair. If same, then we're done. If not, sense labels on the paint cans; if there is a can that is the same color as one piece of furniture, then apply the paint to the other piece. Otherwise, paint both pieces with any color.

Monintoring/replanning: Similar to conditional planner, but perhaps with fewer branches at first, which are filled in as needed at runtime. Also, would check for unexpected outcomes (e.g., missed a spot in painting, so repaint)

Summary

♢ Incomplete info: use conditional plans, conformant planning (can use belief states)

♢ Incorrect info: use execution monitoring and replanning

Multiagent Planning

- Multieffector planning
- Multibody planning
- Decentralized planning
- Coordination, cooperation
- Multiactor settings
- Joint actions/joint plans