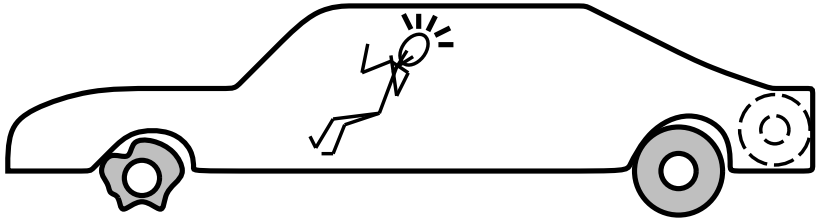
# Planning and Acting

## Chapter 12

# Outline

- ♢ The real world
- ♢ Conditional planning
- ♢ Monitoring and replanning

# 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

# 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

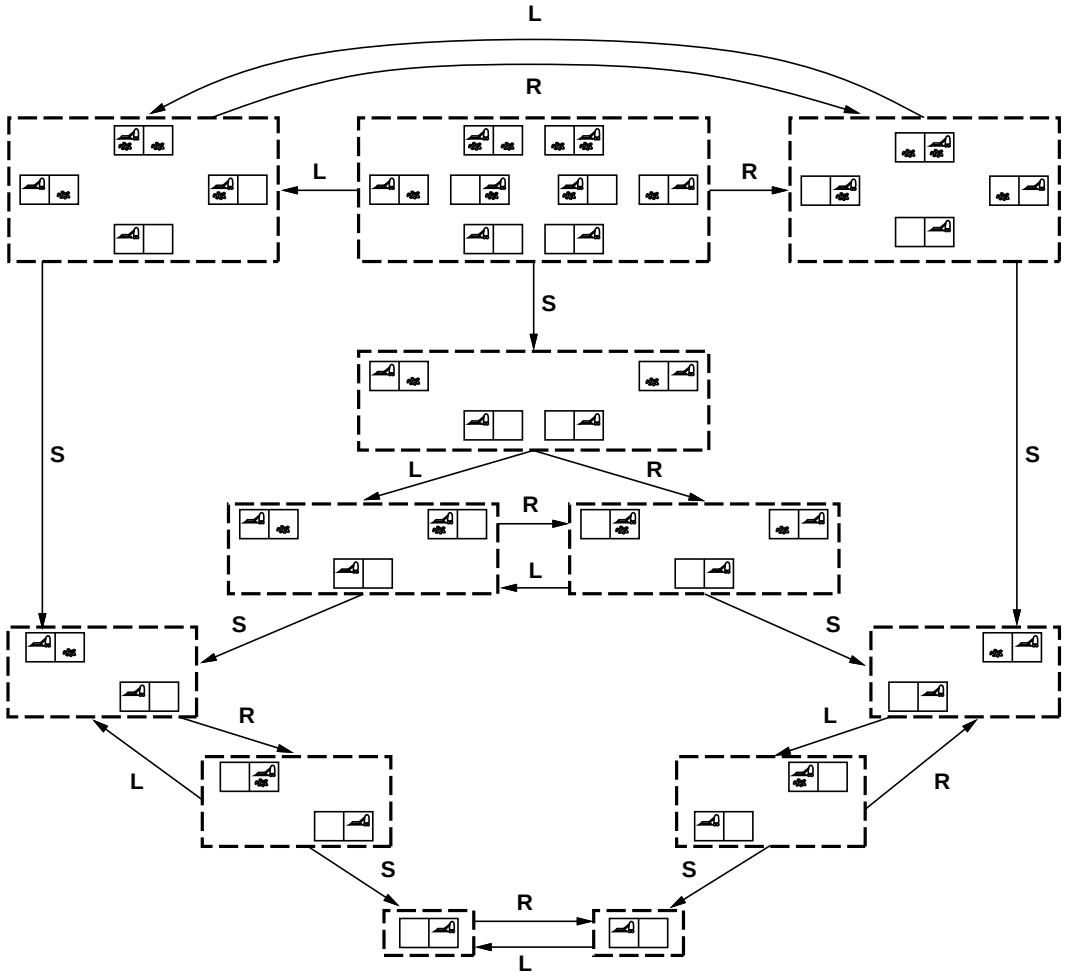
Check progress *during execution*, replan if necessary

*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)

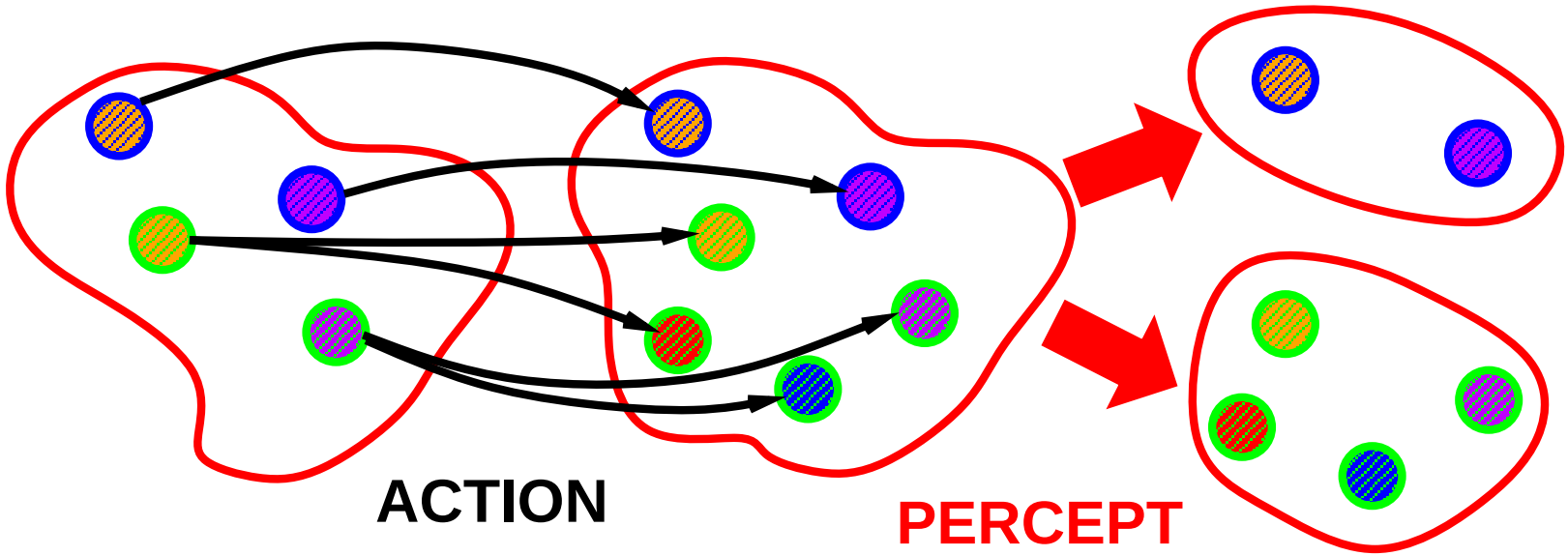
# Conformant planning

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 contd.

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"

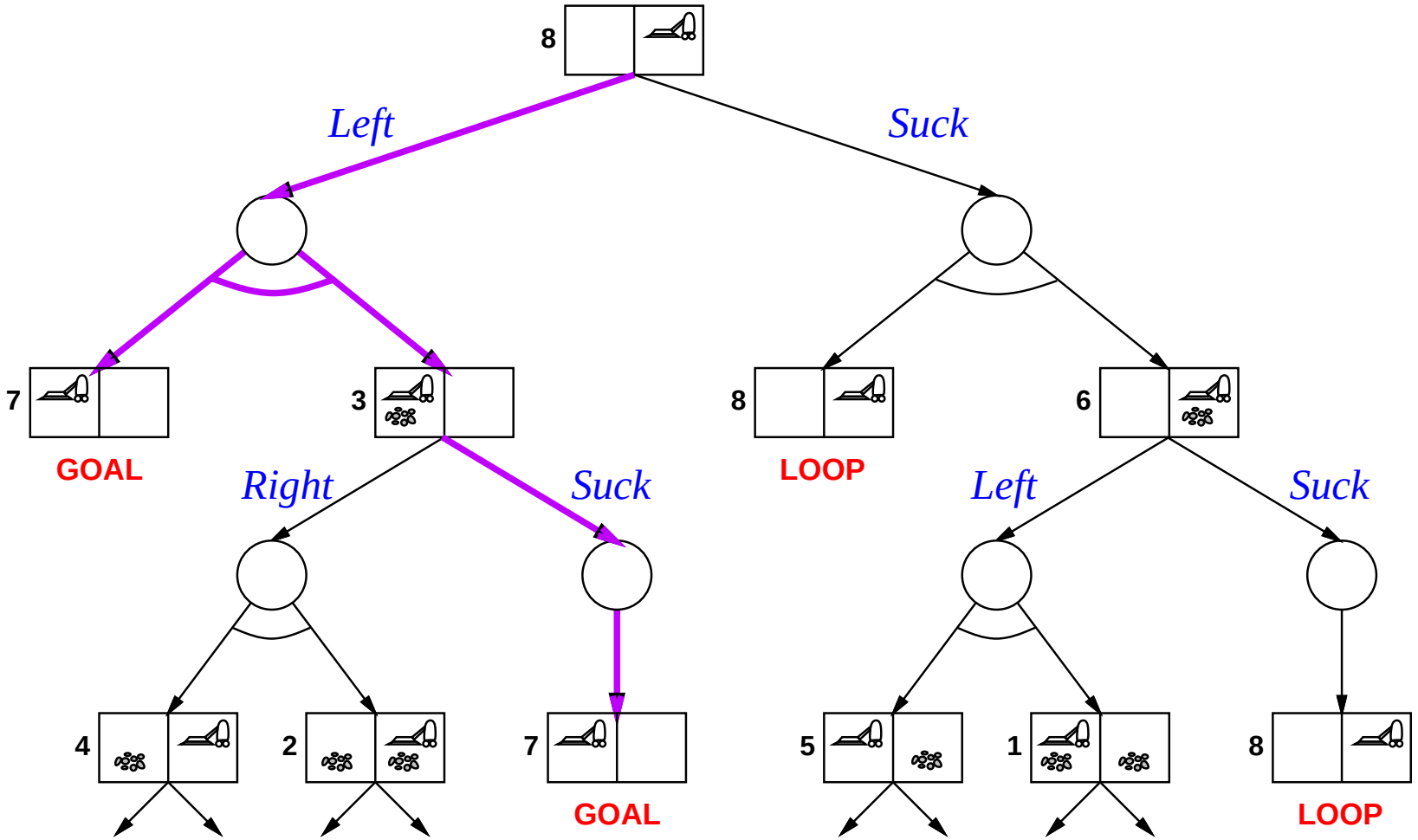
Need *some* plan for *every* possible percept

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

AND–OR tree search (very similar to backward chaining algorithm)

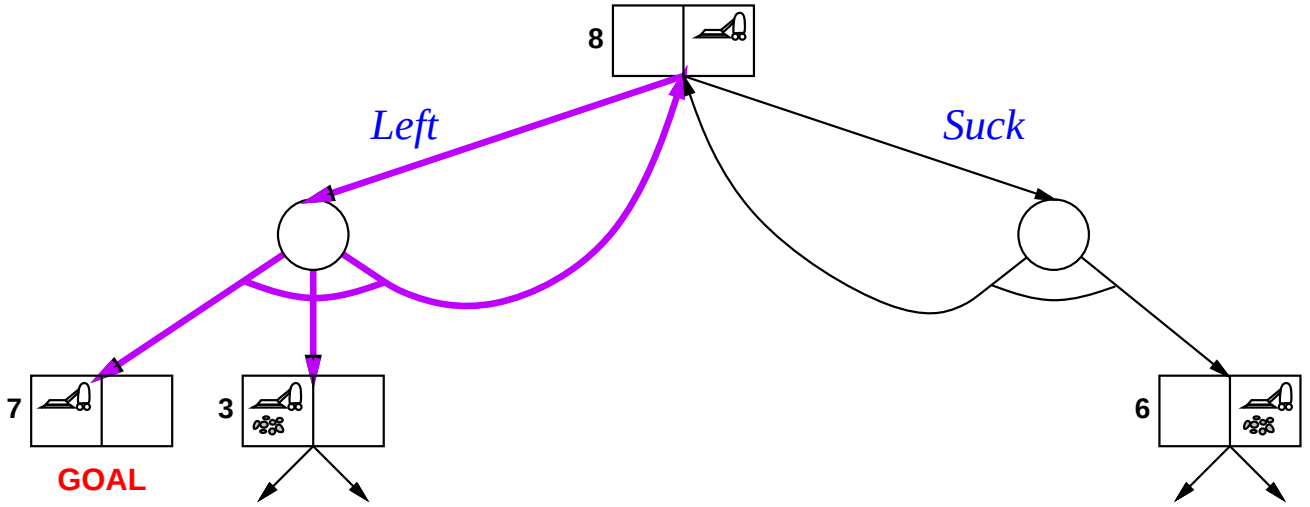
# Example

Double Murphy: sucking or arriving may dirty a clean square



# 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

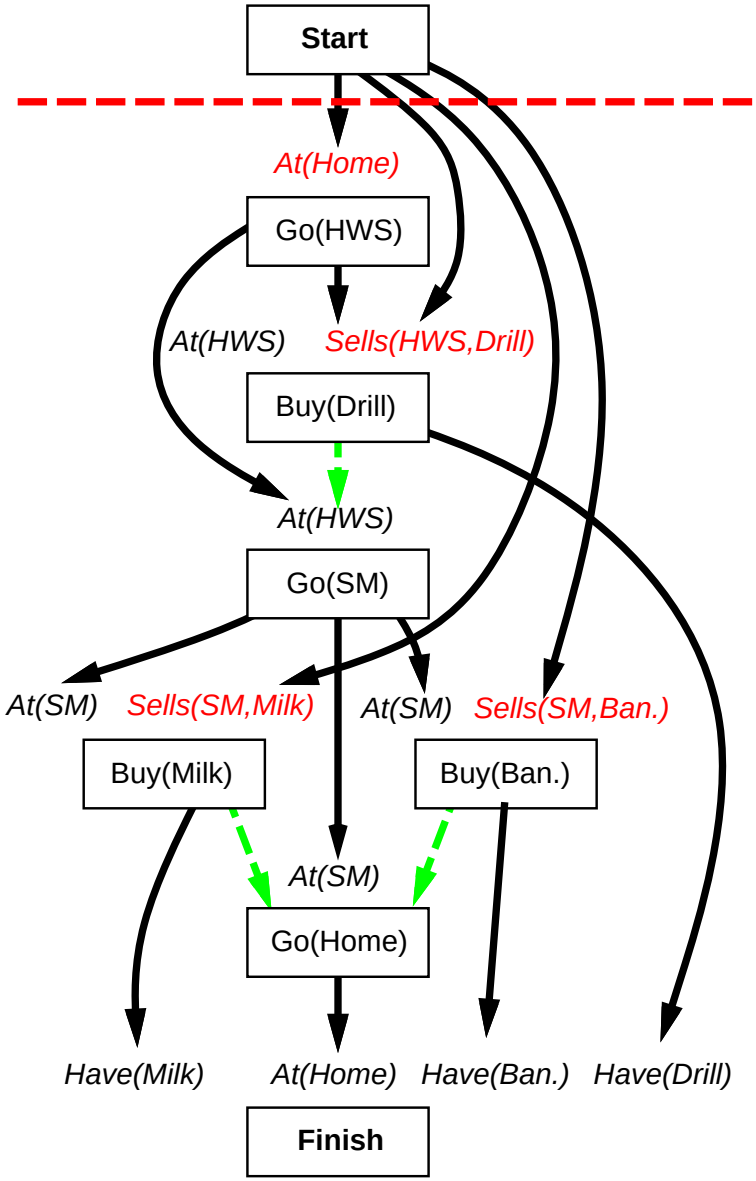
"Failure" = preconditions of *remaining plan* not met

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

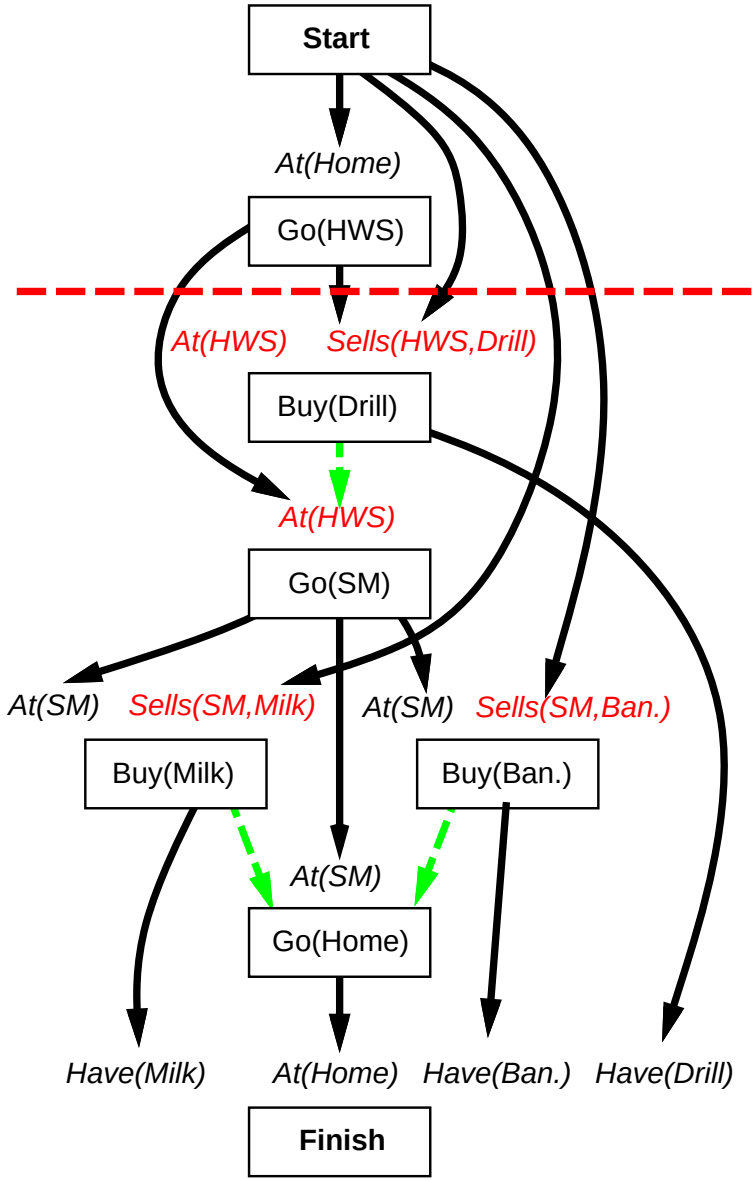
IPEM (Integrated Planning, Execution, and Monitoring):
keep updating $Start$ to match current state
links from actions replaced by links from $Start$ when done

# Example



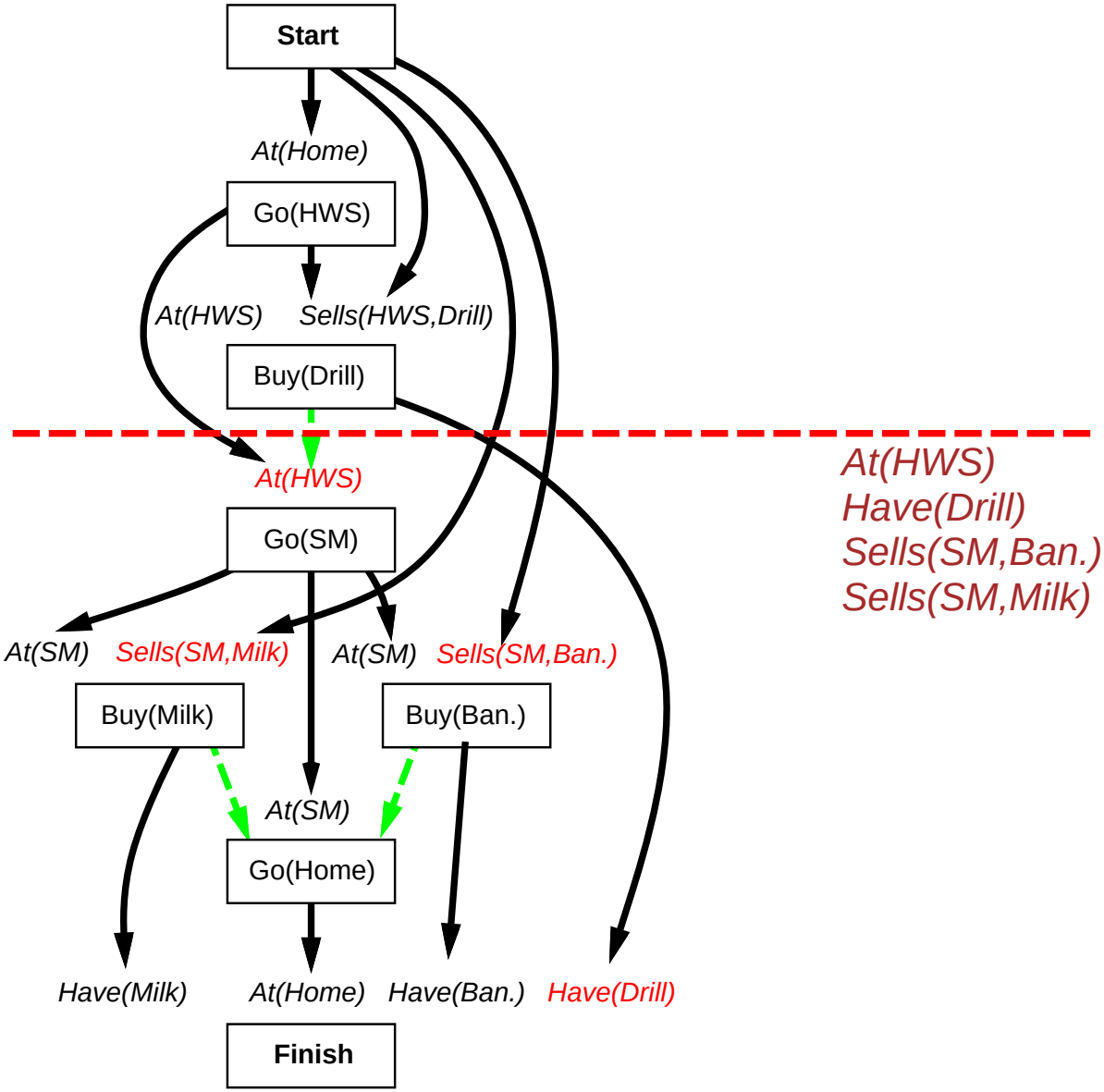

*At(Home)*
*Sells(HWS,Drill)*
*Sells(SM,Ban.)*
*Sells(SM,Milk)*

# Example

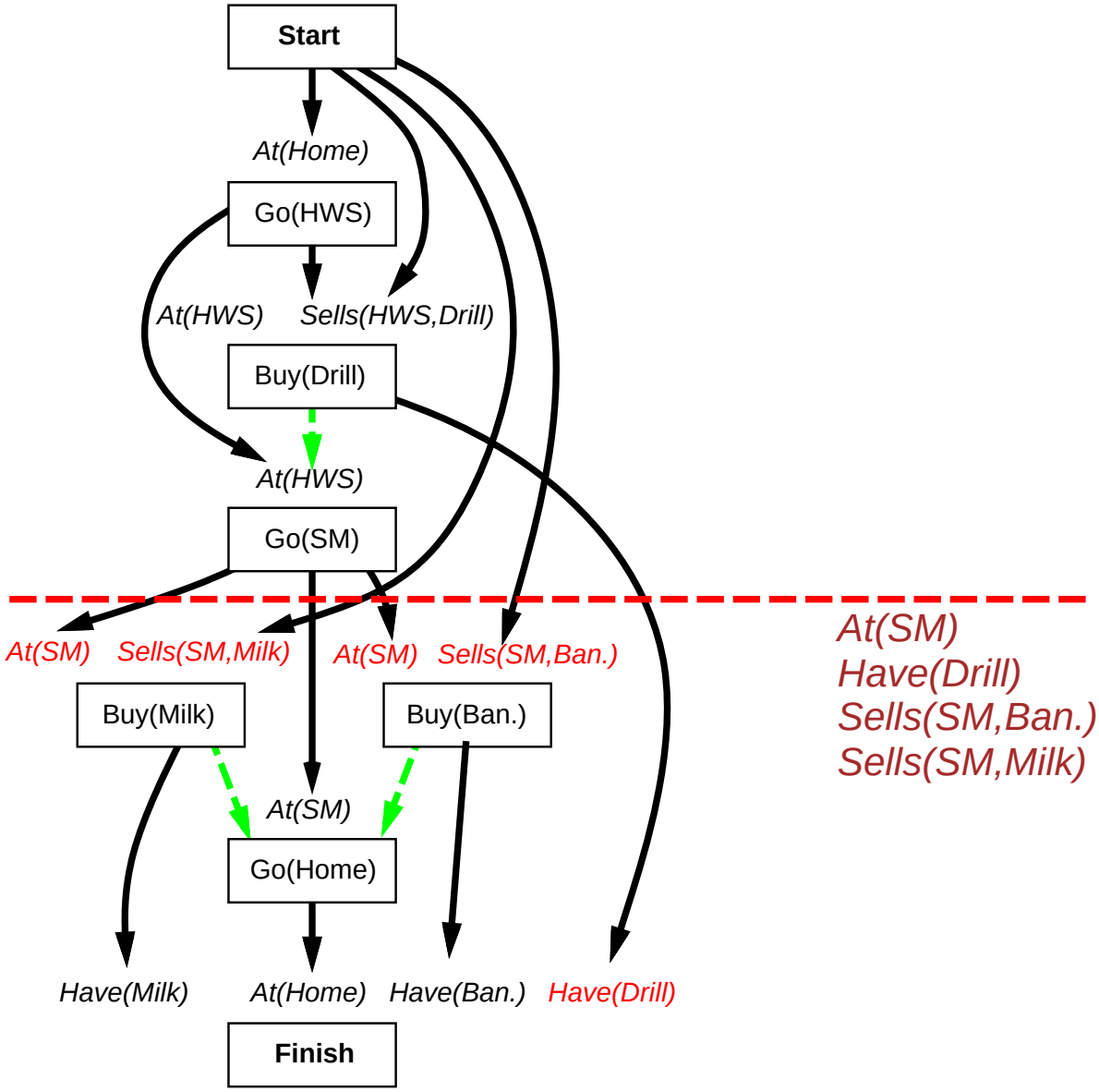



*At(HWS)*
*Sells(HWS,Drill)*
*Sells(SM,Ban.)*
*Sells(SM,Milk)*

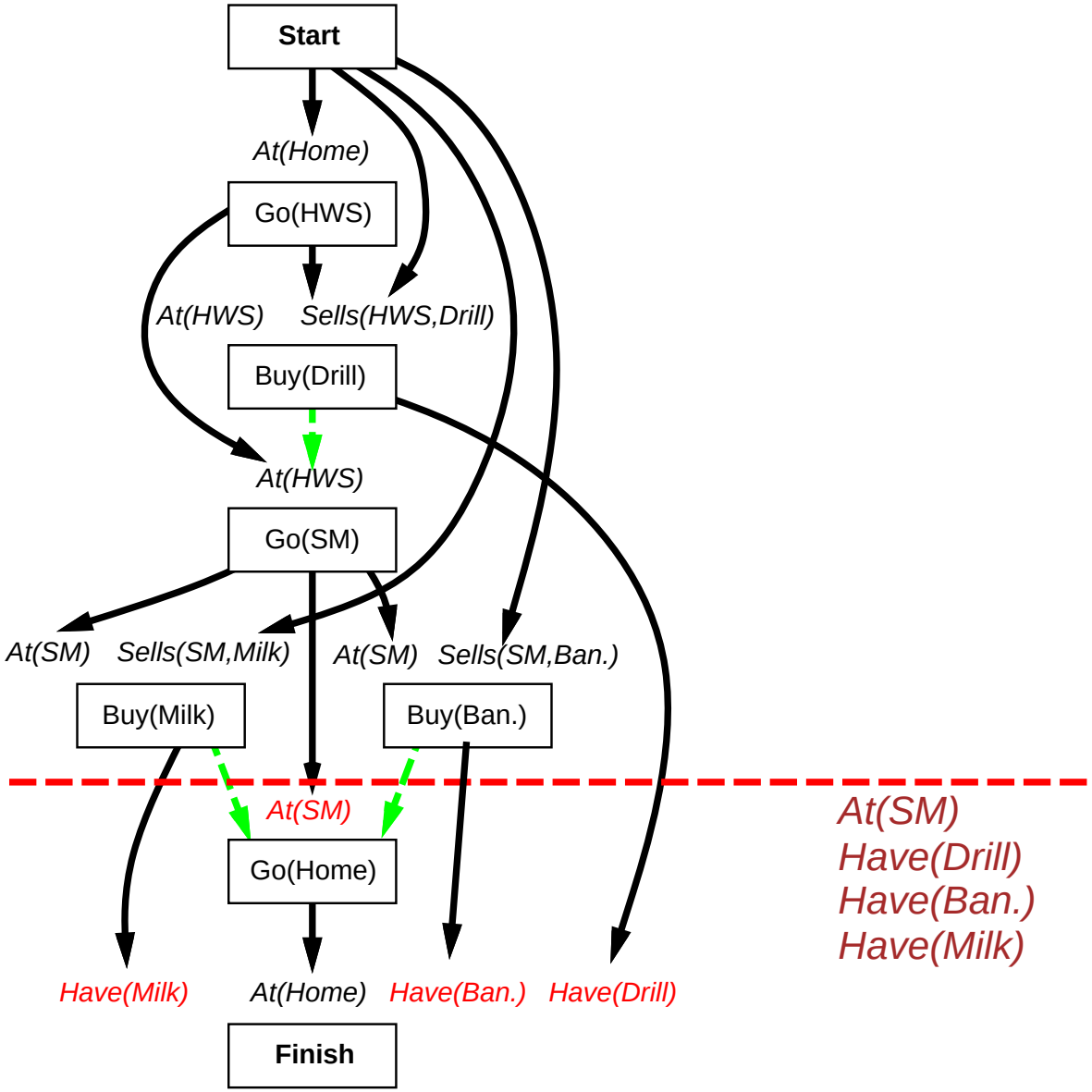
# Example


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)

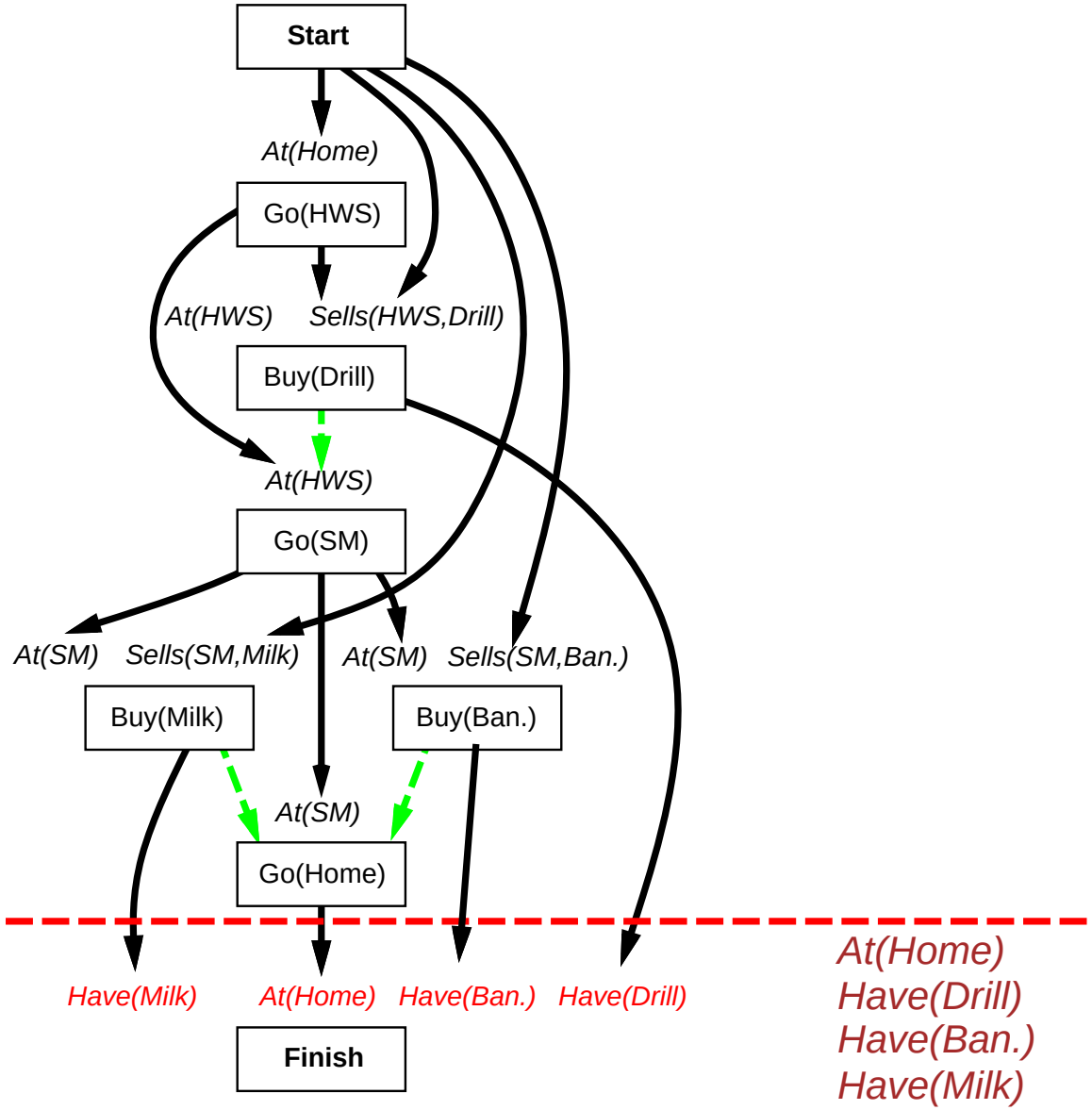
# Example


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)

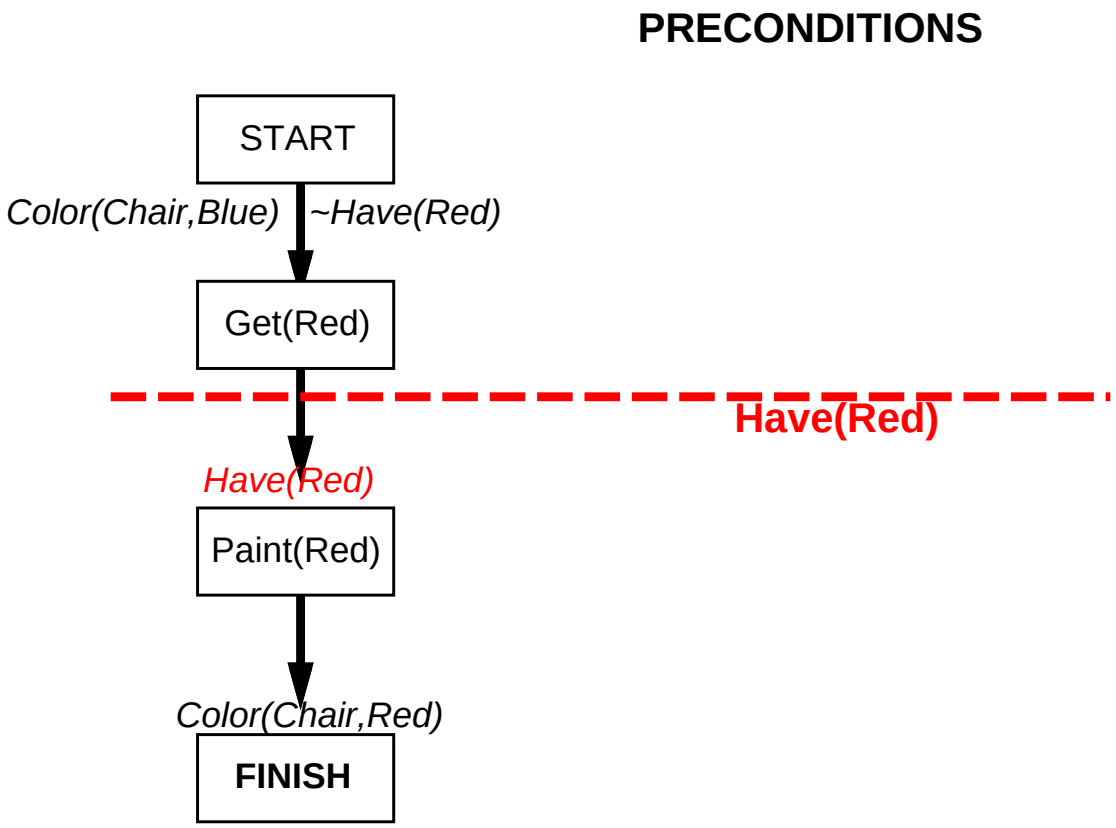
# Example


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)

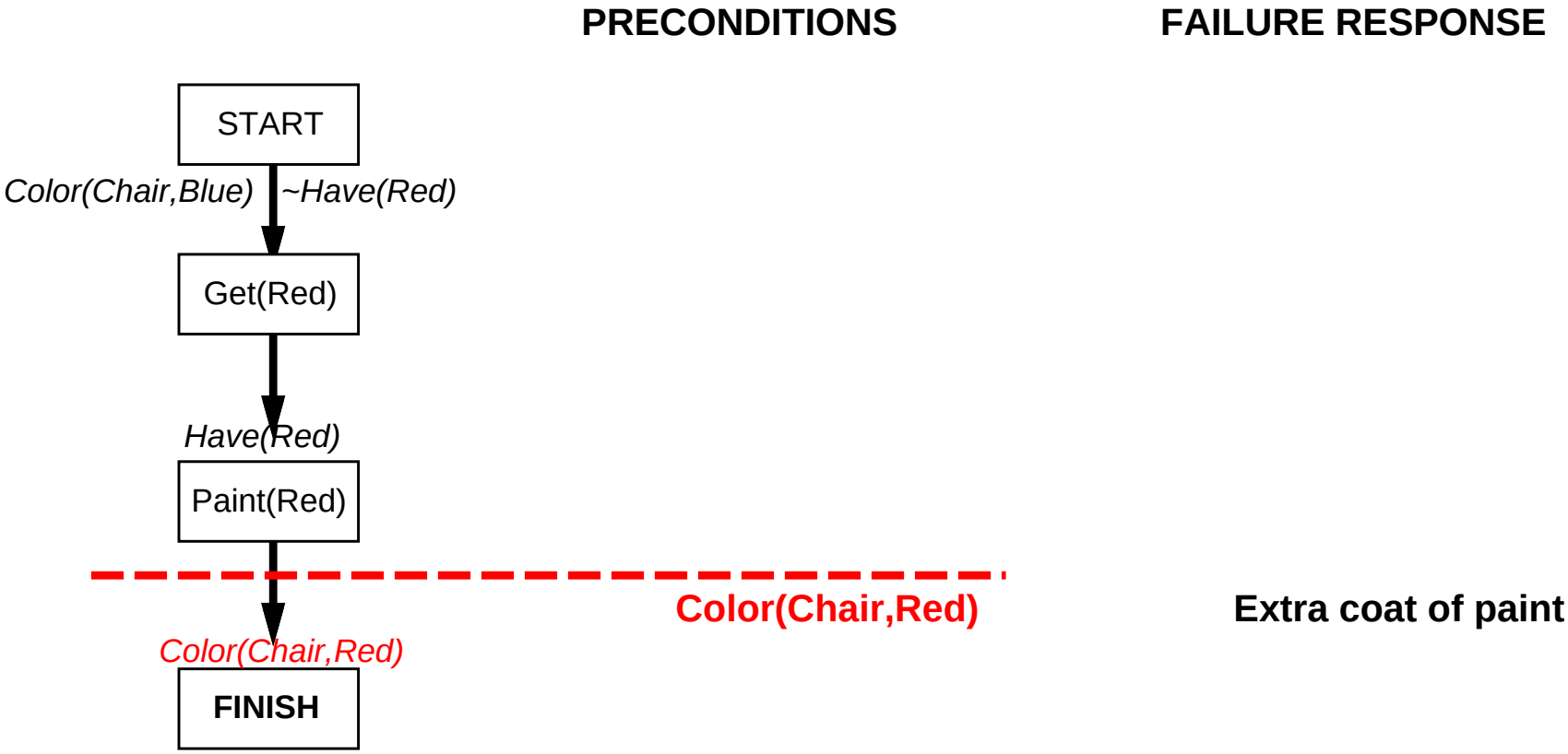
# Example


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)

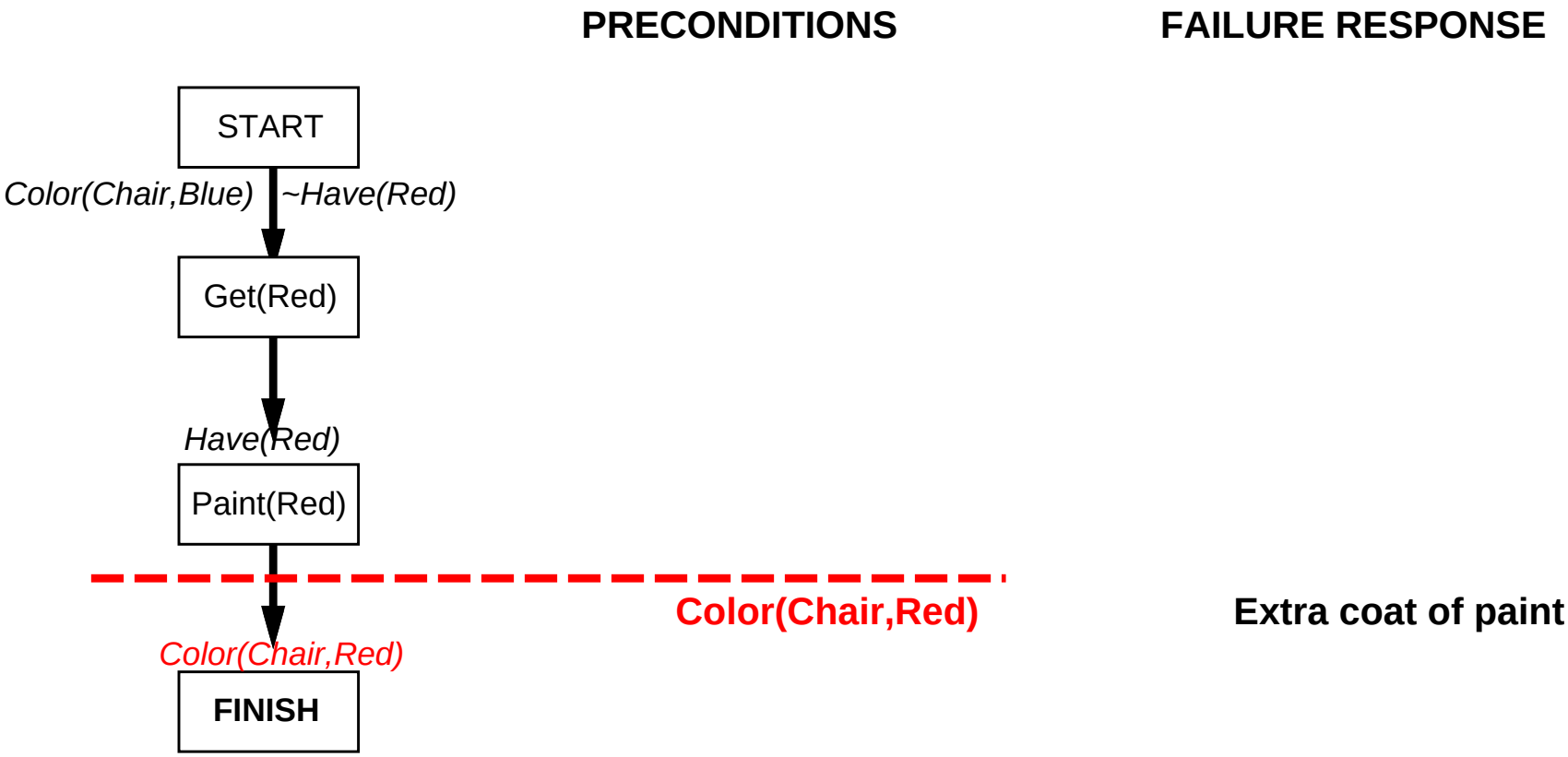
# Emergent behavior



# Emergent behavior



# Emergent behavior




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