INTELLIGENT SYSTEMS (CSE-303-F)

Section C

Partial Order Planning

Outline

- Partial Order Planning
  Execution and Conditional Planning

- * AIMA = “Artificial Intelligence: A Modern Approach,” by Russell and Norvig.

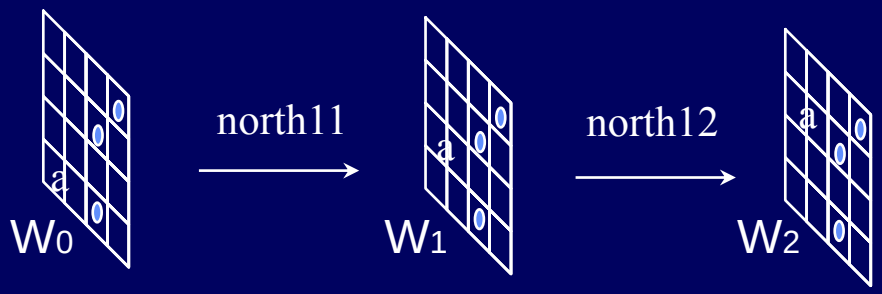
- Optional Reading:
  Weld “Introduction to Least Commitment Planning,” AI Magazine.
  (To be posted on the class web site)

Planning with Atomic Time

- Operator-based planning as search
- Declarative encoding of states and operators
- Partial order planning
  - Planning problem
  - Partial order planning algorithm

Operator-based Planning Problem

- Input
  - Set of world states
  - Action operators
    - Fn: world-state→world-state
  - Initial state of world
  - Goal
    - partial state (set of world states)
- Output
  - Sequence of actions



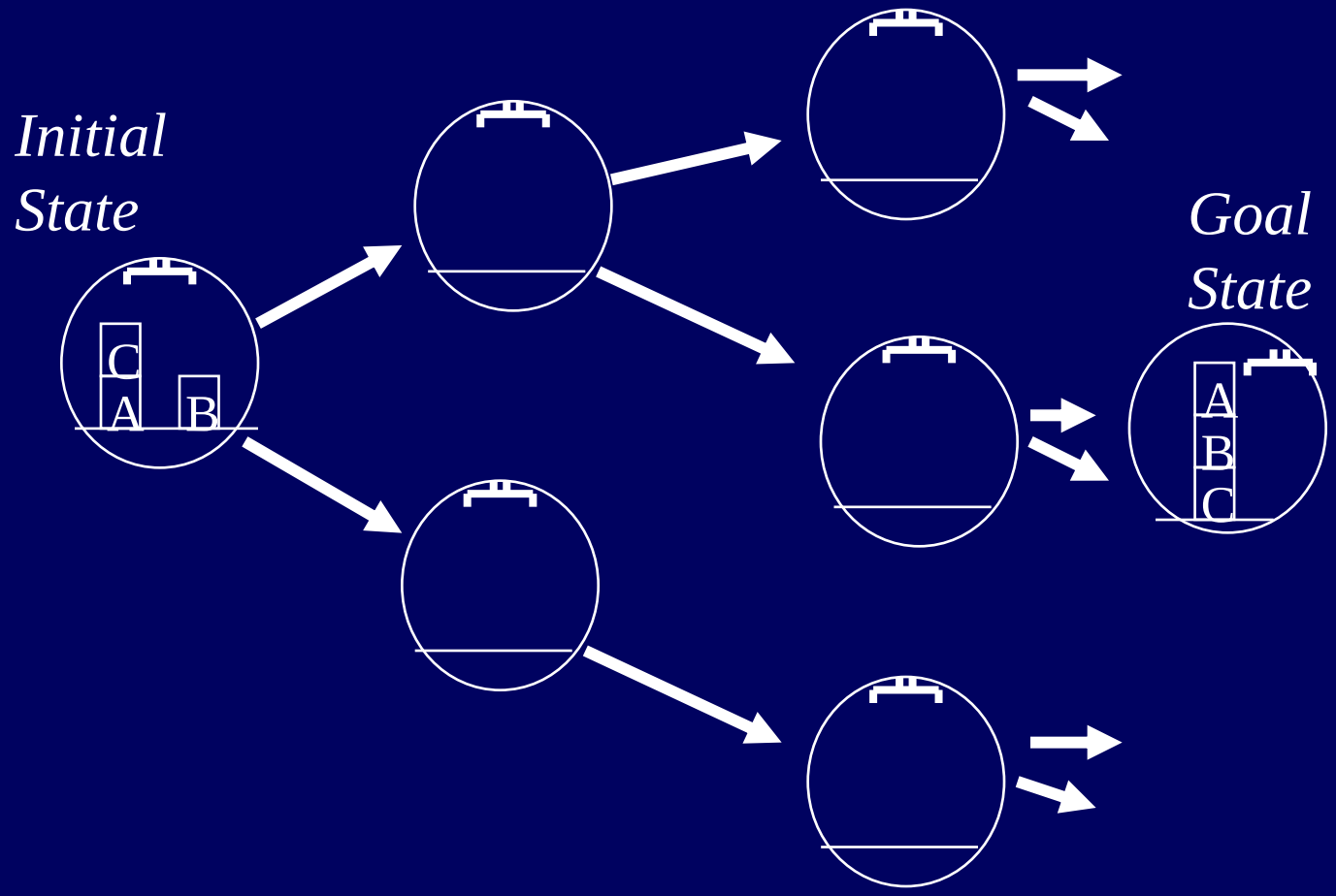
What assumptions are implied?

- Atomic time.
- Agent is omniscient (no sensing necessary).
- Agent is sole cause of change.
- Actions have deterministic effects.
- No indirect effects.

⇨ STRIPS Assumptions

Operator-based Planning as Search

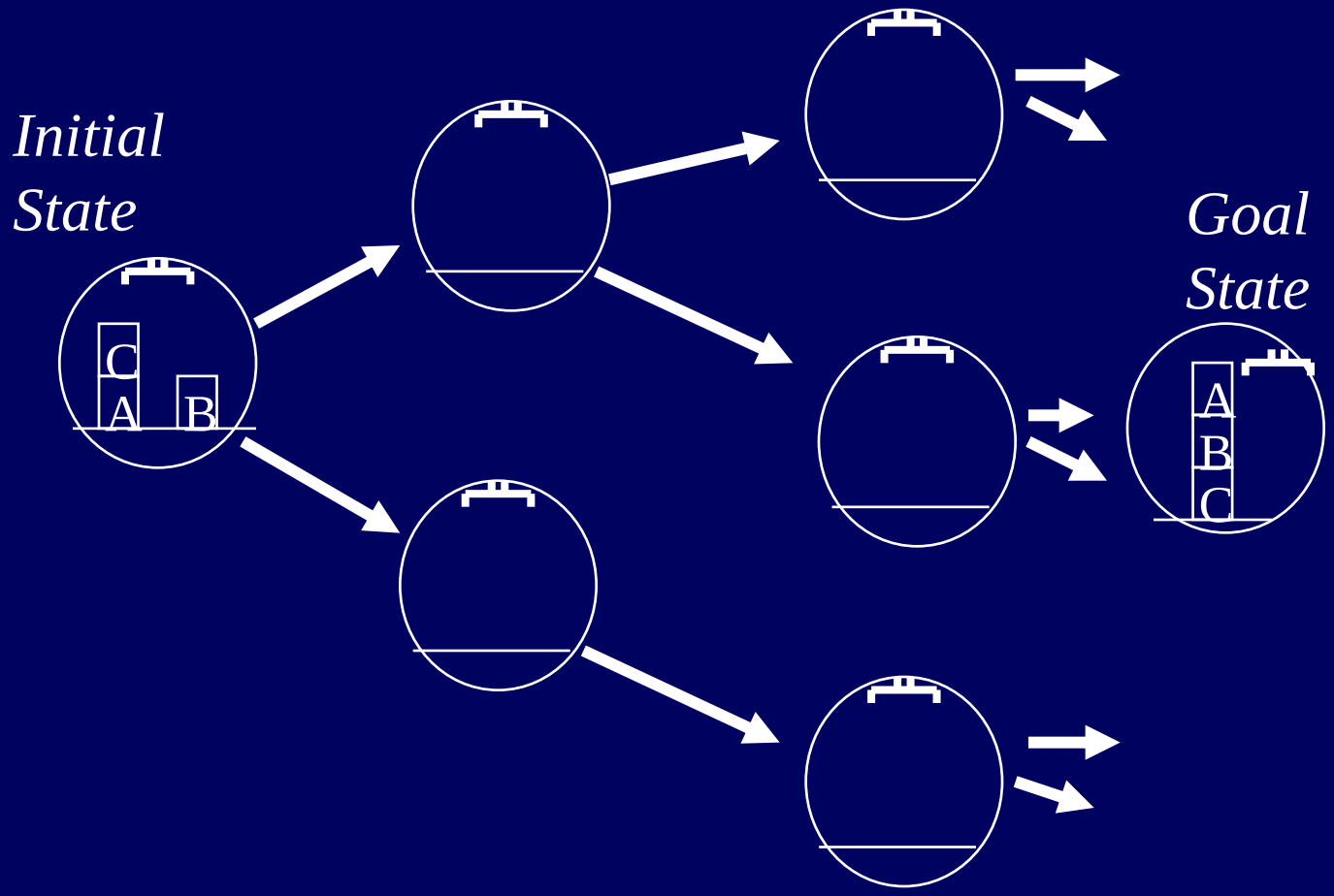
1) Forward-Chaining State-Space Search



What problems arise?

What Are Alternative Strategies?

Are these strategies supported by our current representation?



Need more declarative description of operators and state

Planning as Search

2) Backward-Chaining (goal-directed search)

- Search partial worlds (partial assignment)

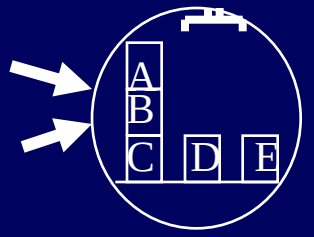
Subgoals weakly interact:

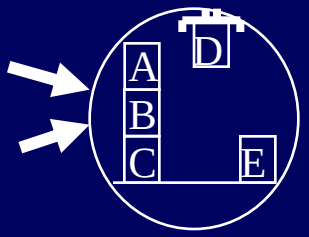
- Maintain goal/subgoal decomposition

- Order action sequences only where needed (partial orders)

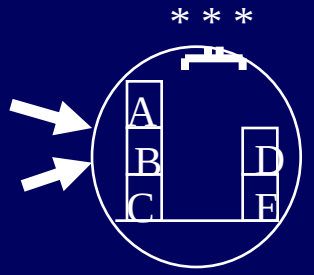
➔How do we represent the search state?

Initial State is completely defined





* * *

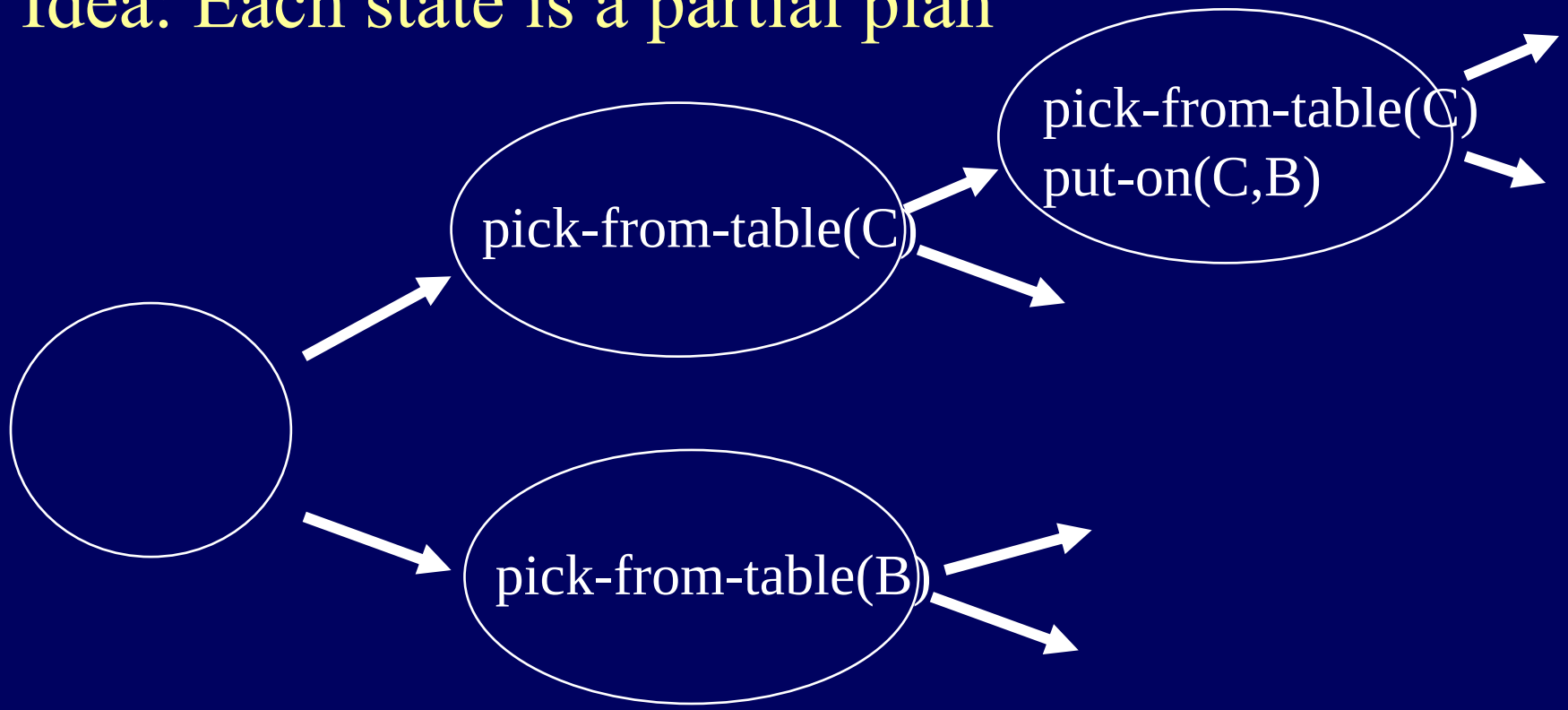


Many Goal States

Where do we start?

Plan-Space Search

Idea: Each state is a partial plan



- How do we represent plans?
- How do we test if a plan is a solution?
- How do we generate a plan?

Partial Order Planning

- Plan from goals, back to initial state
- Search through partial plans

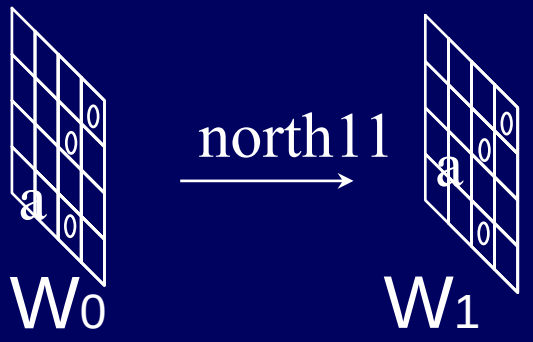
- Representation:
  - Operators given in declarative representation, rather than black box functions.
  - Plans represent only relevant commitments (e.g., relevant ordering of operators, not total ordering)

Planning with Atomic Time

- Operator-based planning as search
- Declarative encoding of state and operators
- Partial order planning
  - Planning problem
  - Partial order planning algorithm

STRIPS Representation: Encode world states as conjunctions of literals

- *Propositions*
  - True/False Statements (block a)
- *Literals*
  - Proposition or its negation (not (block a))
- *Conjunction*
  - And of literals
  - (And (block a) (block b) (block c) (on-table a) …)

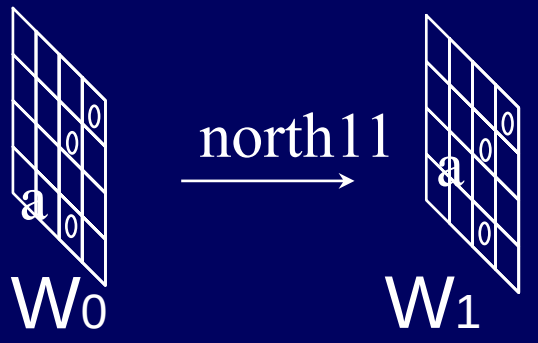


- A *World state* is a conjunction with every proposition appearing exactly once.
- A *Partial state* is a conjunction with every proposition appearing at most once.

What is missing from this logic?

STRIPS Operator Representation

- *Initial state:*
  - ((block a) (block b) (block c) (on-table a) (on-table b) (clear a) (clear b) (clear c) (arm-empty))
- *goal (partial state):*
  - ((on a b) (on b c)))
- *Available actions*
  - Strips operators



precond: (and (agent-at 1 1) (agent-facing north))

North11

effect: (and (agent-at 1 2) (not (agent-at 1 1)))

- Effects specify how to change the set of propositions.

(Parameterized) Operator Schemata

- Instead of defining:

  pickup-A and **pickup-B** and …

- Define a schema:

?var denotes a free variable

(:operator **pick-up**
:***parameters*** **((block ?ob1))**
:*precondition* (and (clear ?ob1)
(on-table ?ob1)
(arm-empty))
:*effect* (and (not (clear ?ob1))
(not (on-table ?ob1))
(not (arm-empty))
(holding ?ob1)))

Note: strips doesn't allow derived effects; you must be complete!

Planning with Atomic Time

- Operator-based planning as search
- Declarative encoding of state and operators
- Partial order planning
  - Planning problem
  - Partial order planning algorithm

Given Initial and Goal State

Start

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

Initial and Goal states are encoded as operators, Why?

Don’t need to introduce (partial) states as separate objects.

Keeps theory minimal.

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

Finish

Given Plan Operators

At(HWS)
Go(SM)
At(SM)

At(Home)
Go(HWS)
At(HWS)

At(SM)
Go(Home)
At(Home)

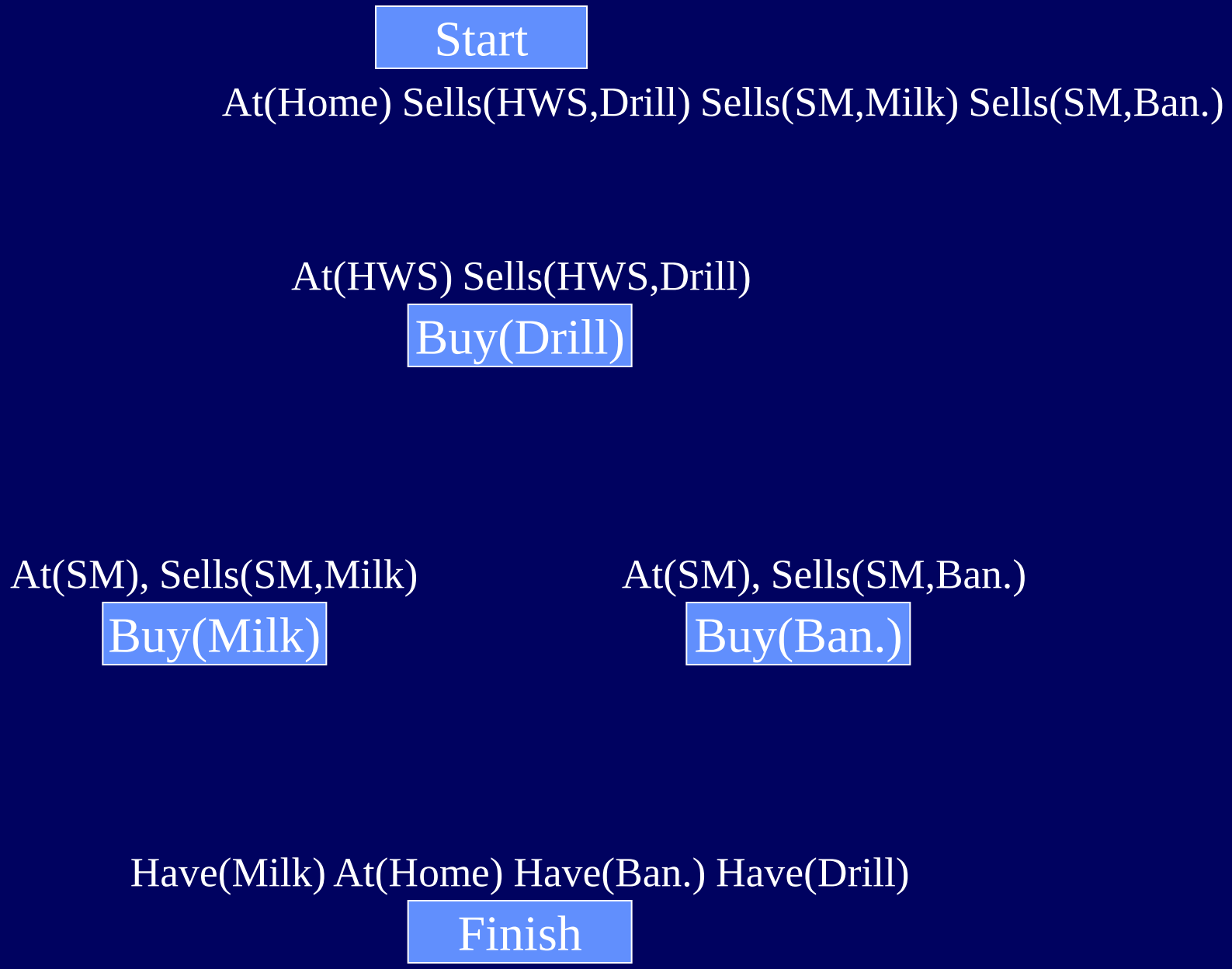
At(SM), Sells(SM,Ban.)
Buy(Ban.)
Have(Ban)

At(HWS) Sells(HWS,Drill)
Buy(Drill)
Have(Drill)

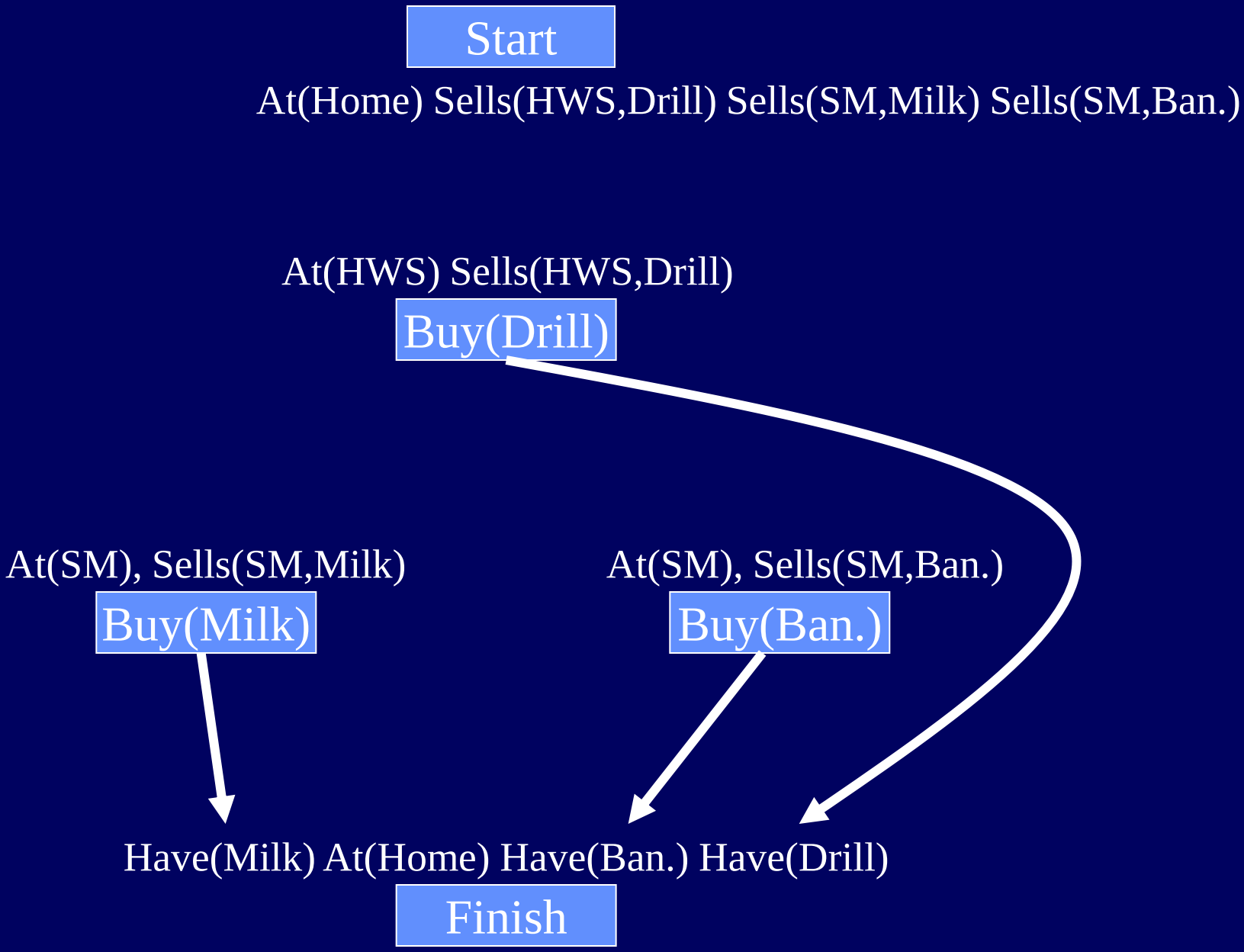
At(SM), Sells(SM,Milk)
Buy(Milk)
Have(Milk)

What is a solution?

Partial Order Plan <Actions,Orderings,Links>



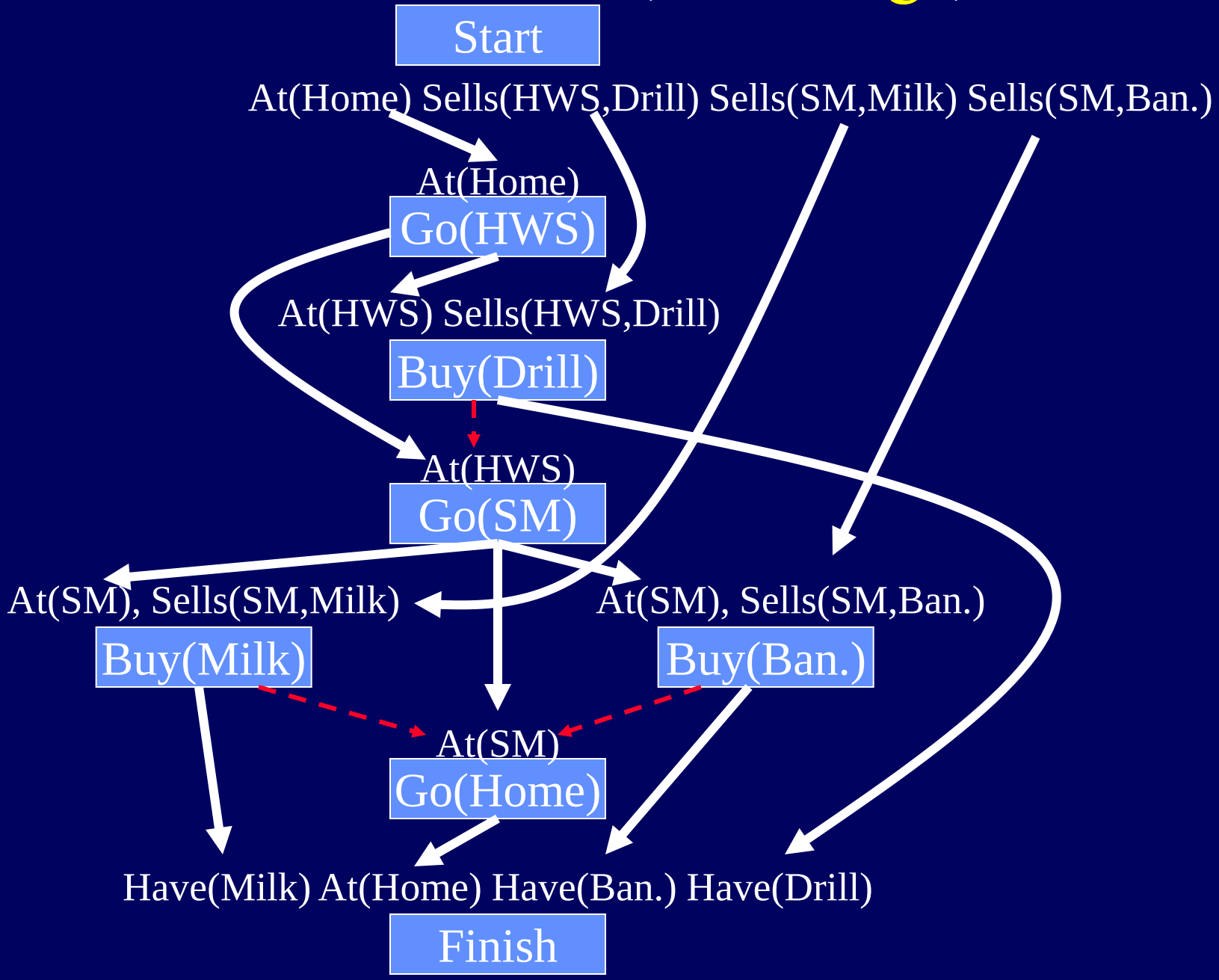
Partial Order Plan <Actions,Orderings,Links>



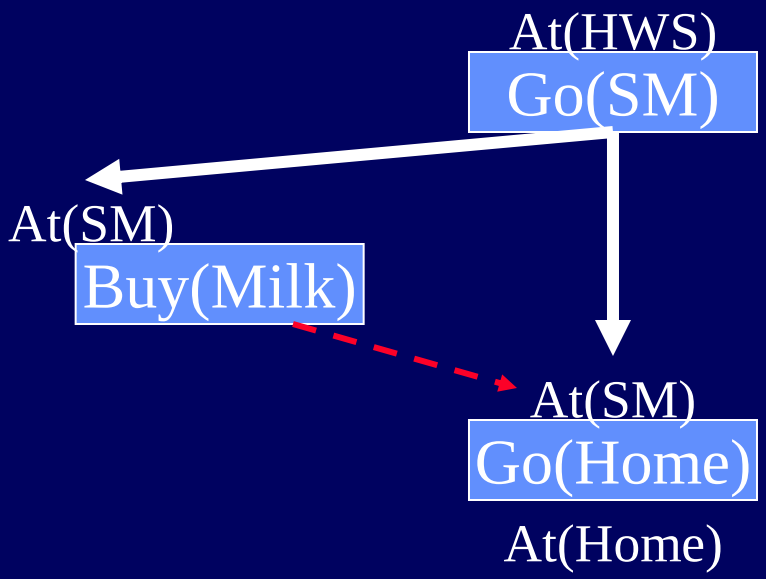
Partial Order Plan <Actions,Orderings,Links>

Partial Order Plan <Actions,Orderings,Links>

Partial Order Plan <Actions,Orderings,Links>

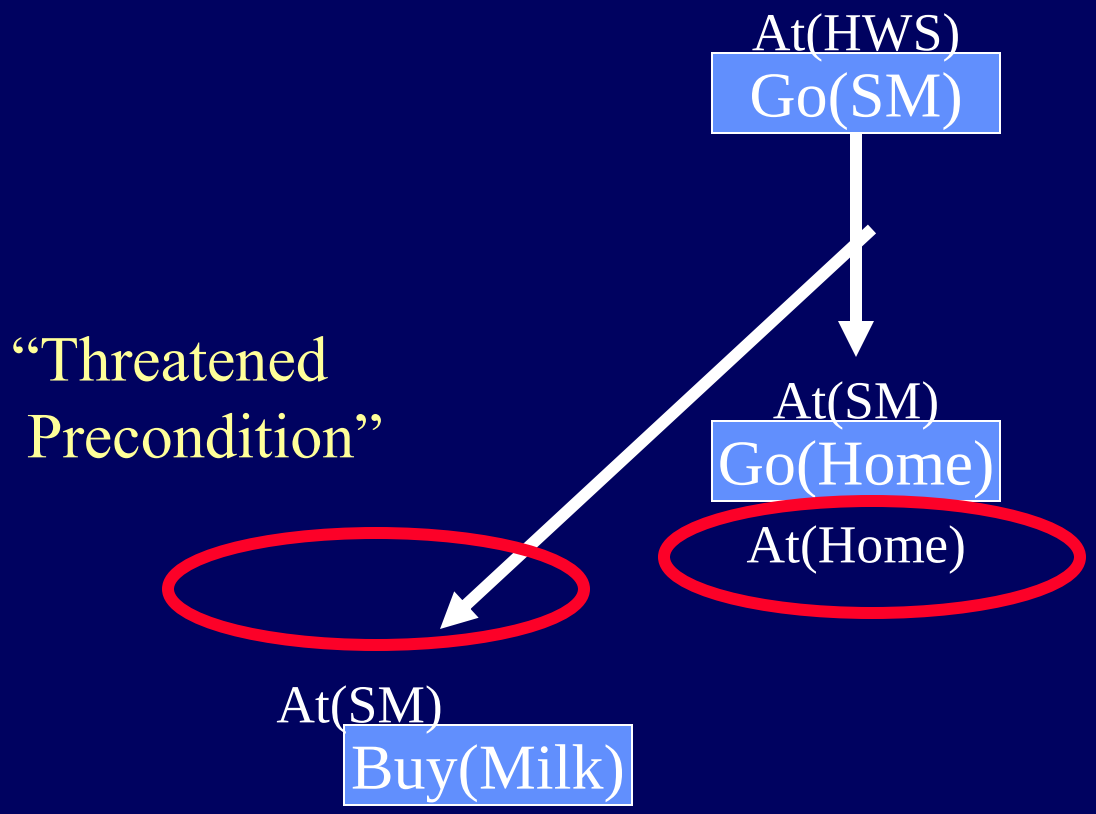


Why is an ordering needed?



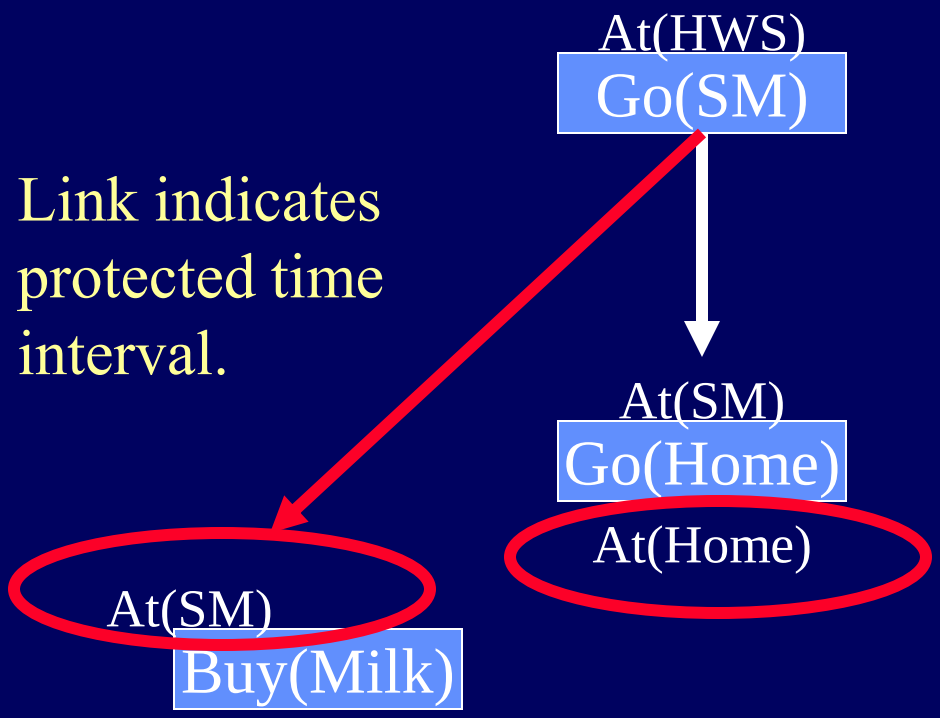
Why is an ordering needed?

Suppose the other order is allowed, what happens?

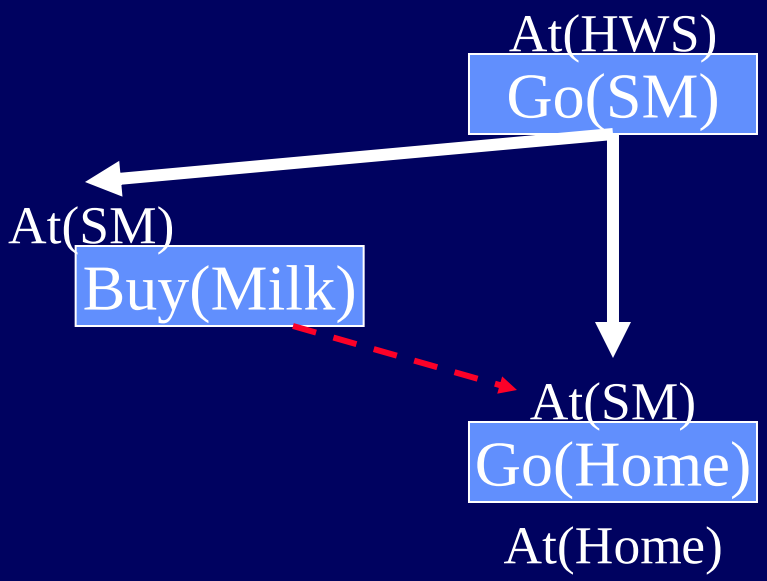


Why is an ordering needed?

Suppose the other order is allowed, what happens?



Ordering Resolves Threat



A Solution: Complete and Consistent Plan

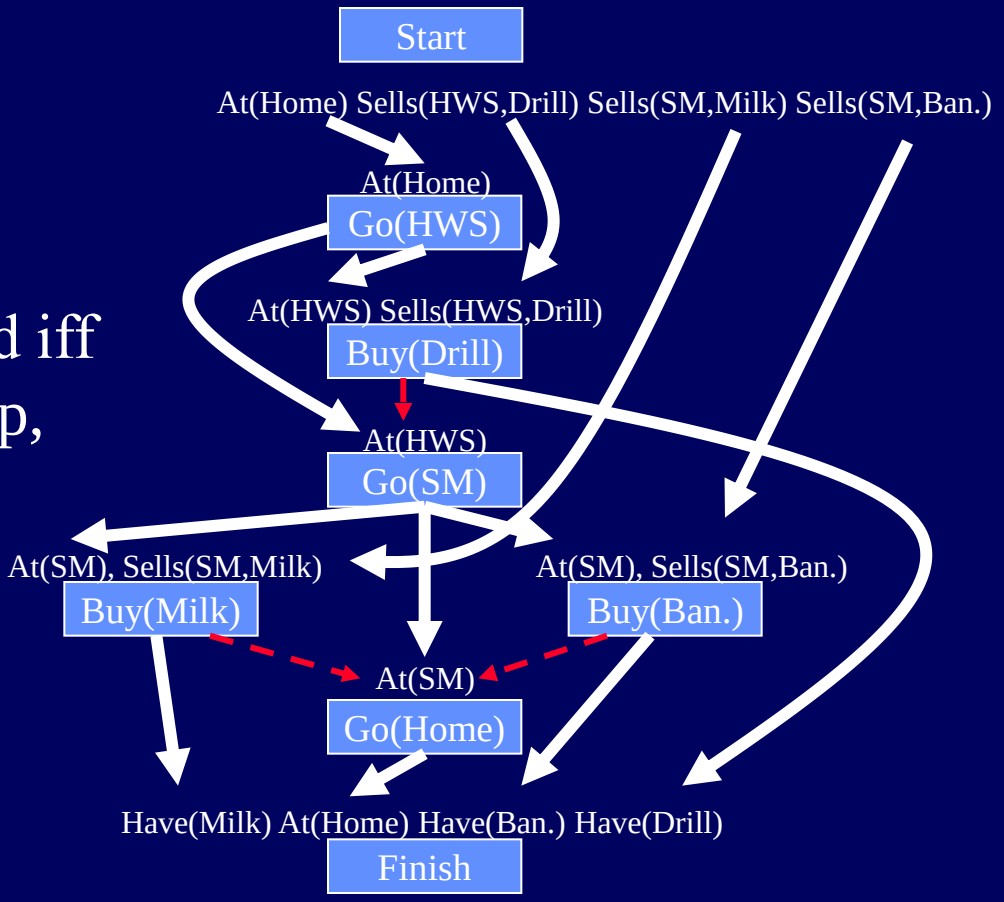
- Complete Plan

  IFF every precondition of every step is achieved

  A step's precondition is achieved iff
  - its the effect of a preceding step,
  - no possibly intervening step undoes it.

- Consistent Plan

  IFF there is no contradiction in the ordering constraints (I.e., never $s_i < s_j$ and $s_j < s_i$.)



Planning with Atomic Time

- Operator-based planning as search
- Declarative encoding of state and operators
- Partial order planning
  - Planning problem
  - Partial order planning algorithm

POP(<A,O,L>, agenda, actions)

- <A,O,L>, A partial plan to expand
- Agenda: A queue of open conditions still to be satisfied: <p, a_{need} >
- Actions: A set of actions that may be introduced to meet needs.
- a_{add}: an action that produces the needed condition p for a_{need}
- A_{threat} : an action that might threaten a causal link from $a_{producer}$ to $a_{consumer}$

POP(<A,O,L>, agenda, actions)

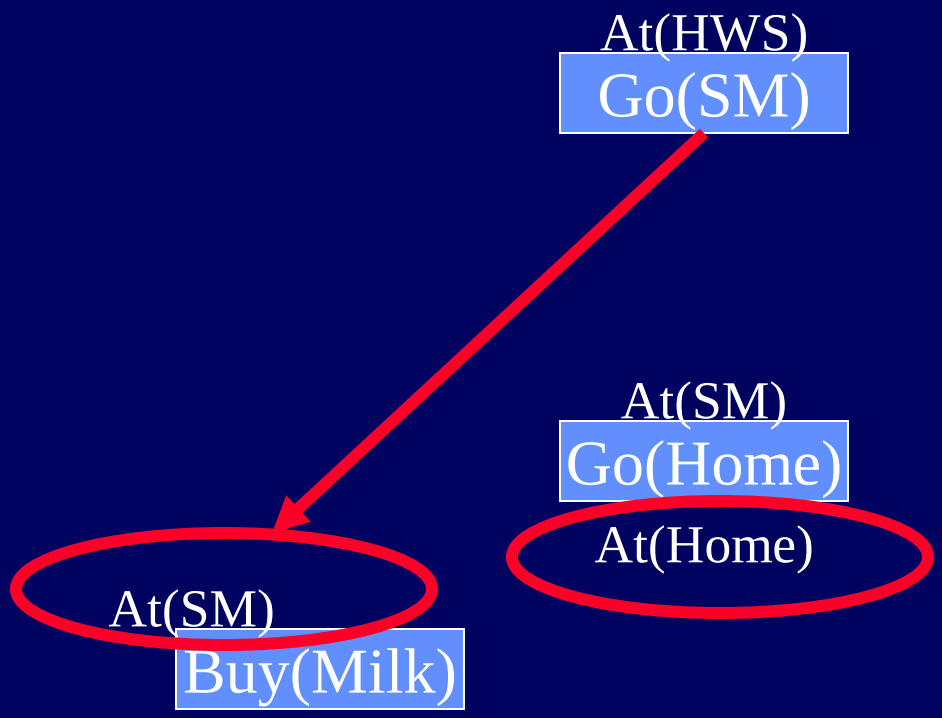
1. **Termination**: If agenda is empty, return plan <A,O,L>.
2. **Goal Selection**: select and remove open condition <p, a_{need}> from agenda.
3. **Action Selection**: Choose new or existing action a_{add} that can precede a_{need} and whose effects include p.
   Link and order actions.
4. **Update Agenda**: If a_{add} is new, add its preconditions to agenda.
5. **Threat Detection**: For every action a_{threat} that might threaten some causal link from $a_{produce}$ to $a_{consume}$, choose a consistent ordering:
   a) Demotion: Add $a_{threat} < a_{produce}$
   b) Promotion: Add $a_{consume} < a_{threat}$
6. **Recurse:** on modified plan and agenda

Choose is nondeterministic

Select is deterministic

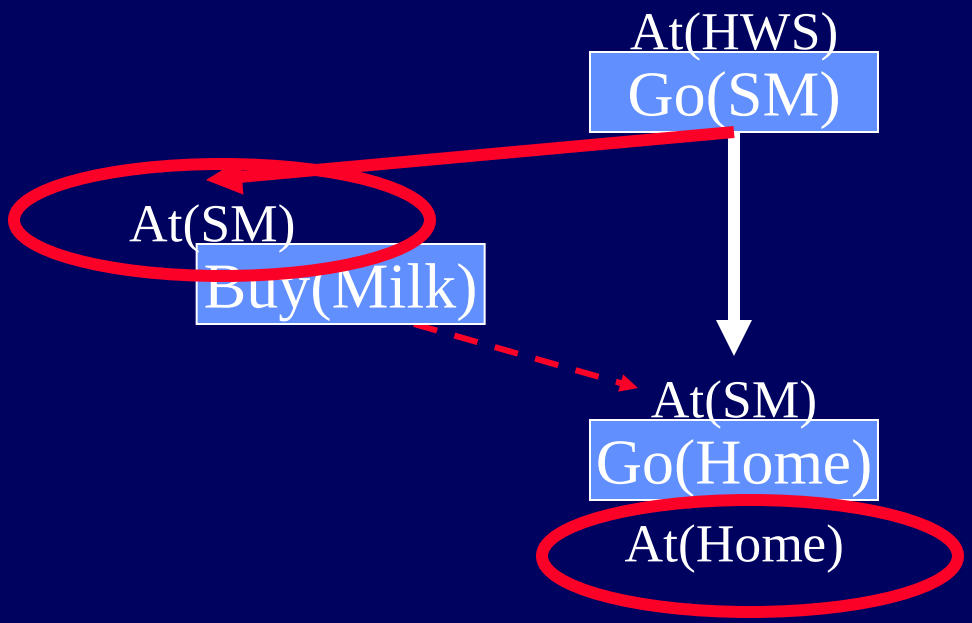
To remove threats…

promote the threat…



To remove threats…

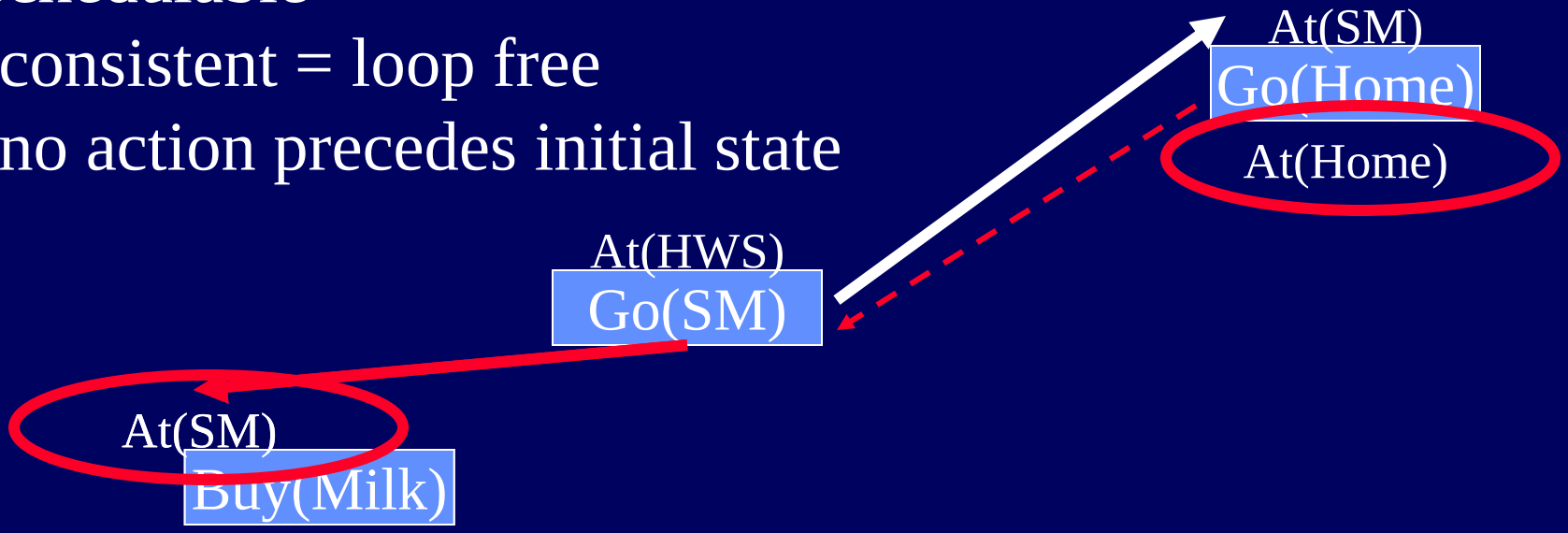
promote the threat… demote the threat…



To remove threats…

promote the threat… demote the threat…

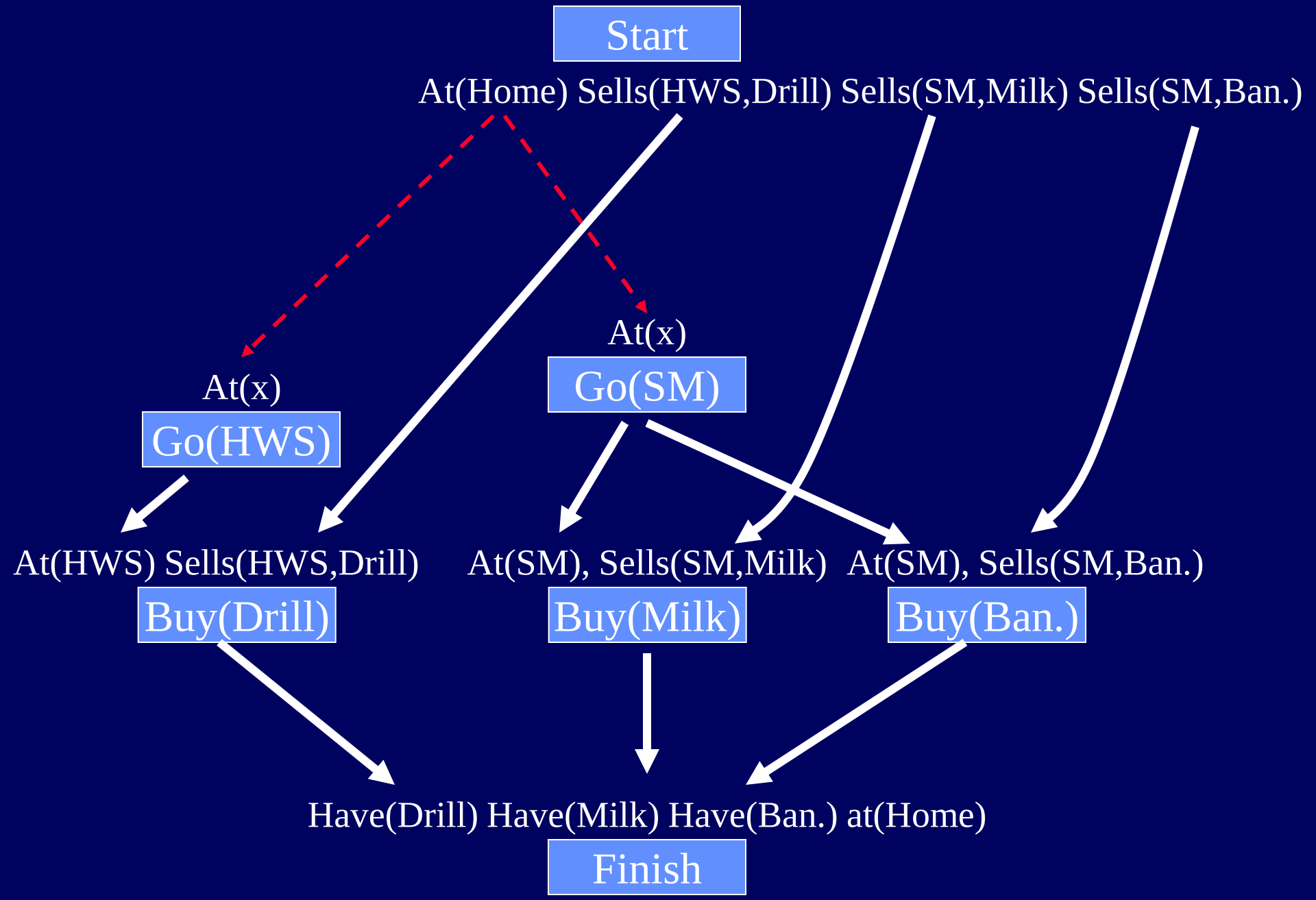
- But only allow demotion/promotion if schedulable
  - consistent = loop free
  - no action precedes initial state

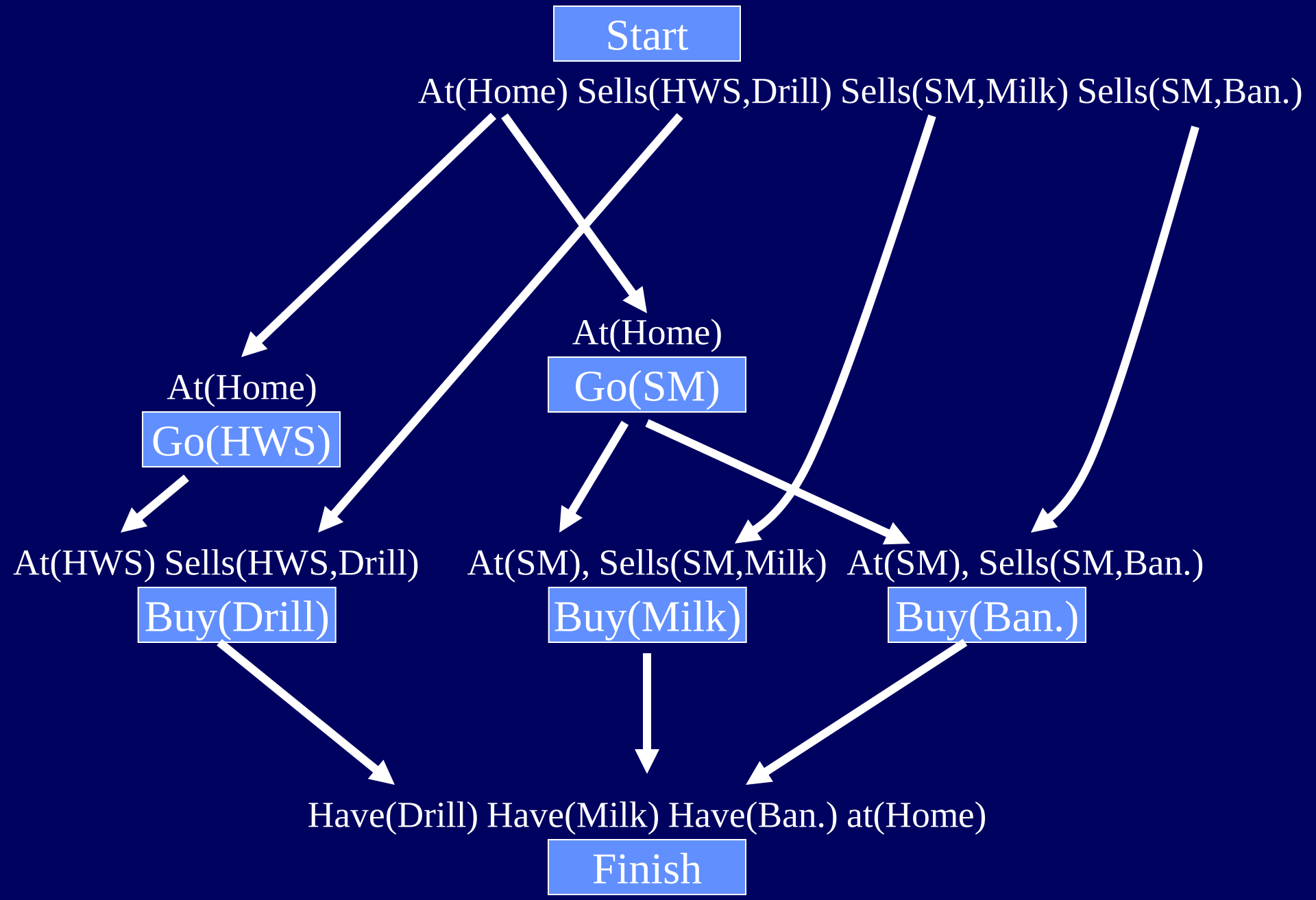


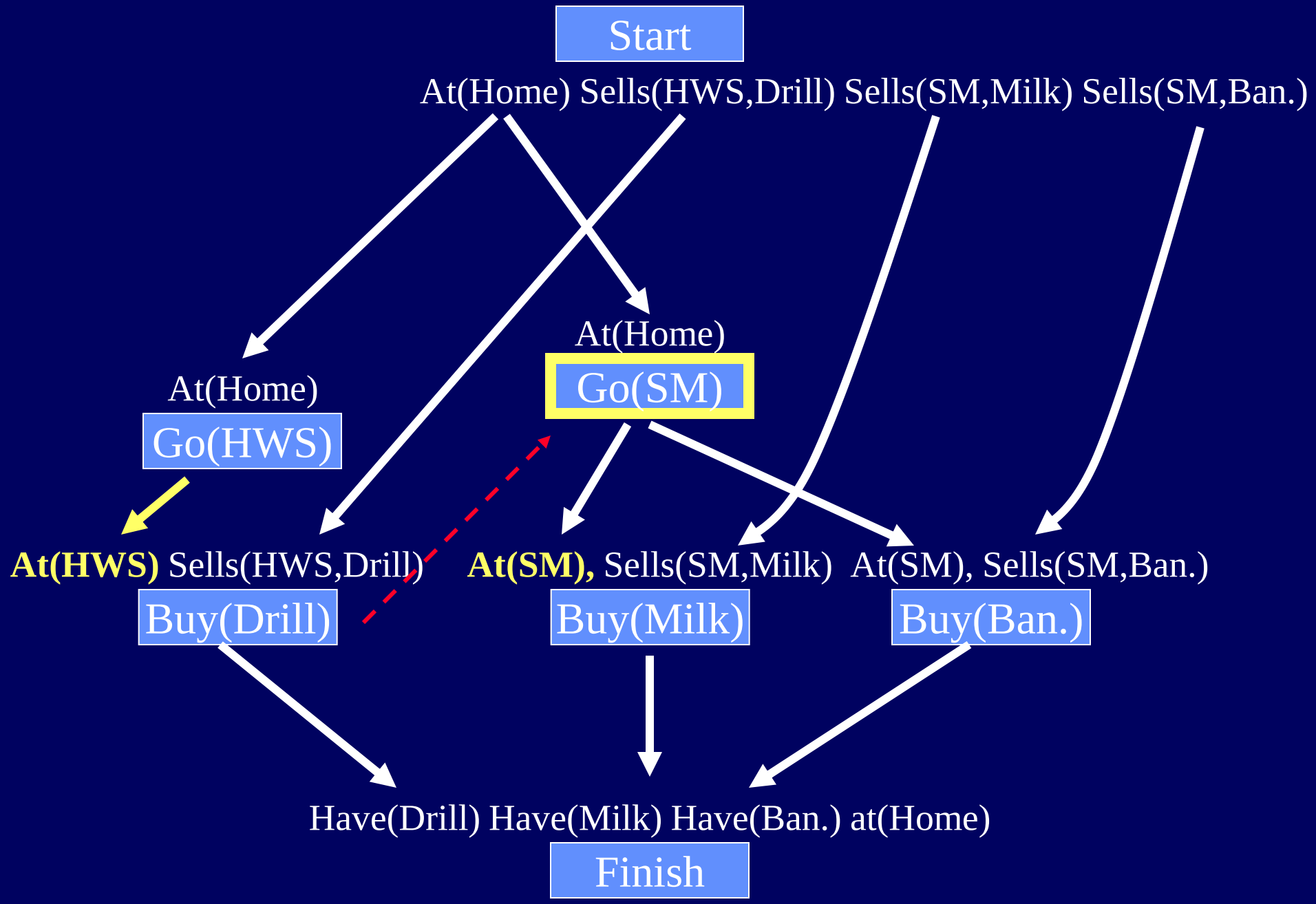

Start
At(Home) Sells(HWS,Drill) Sells(SM,Milk) Sells(SM,Ban.)
Have(Drill) Have(Milk) Have(Ban.) at(Home)
Finish

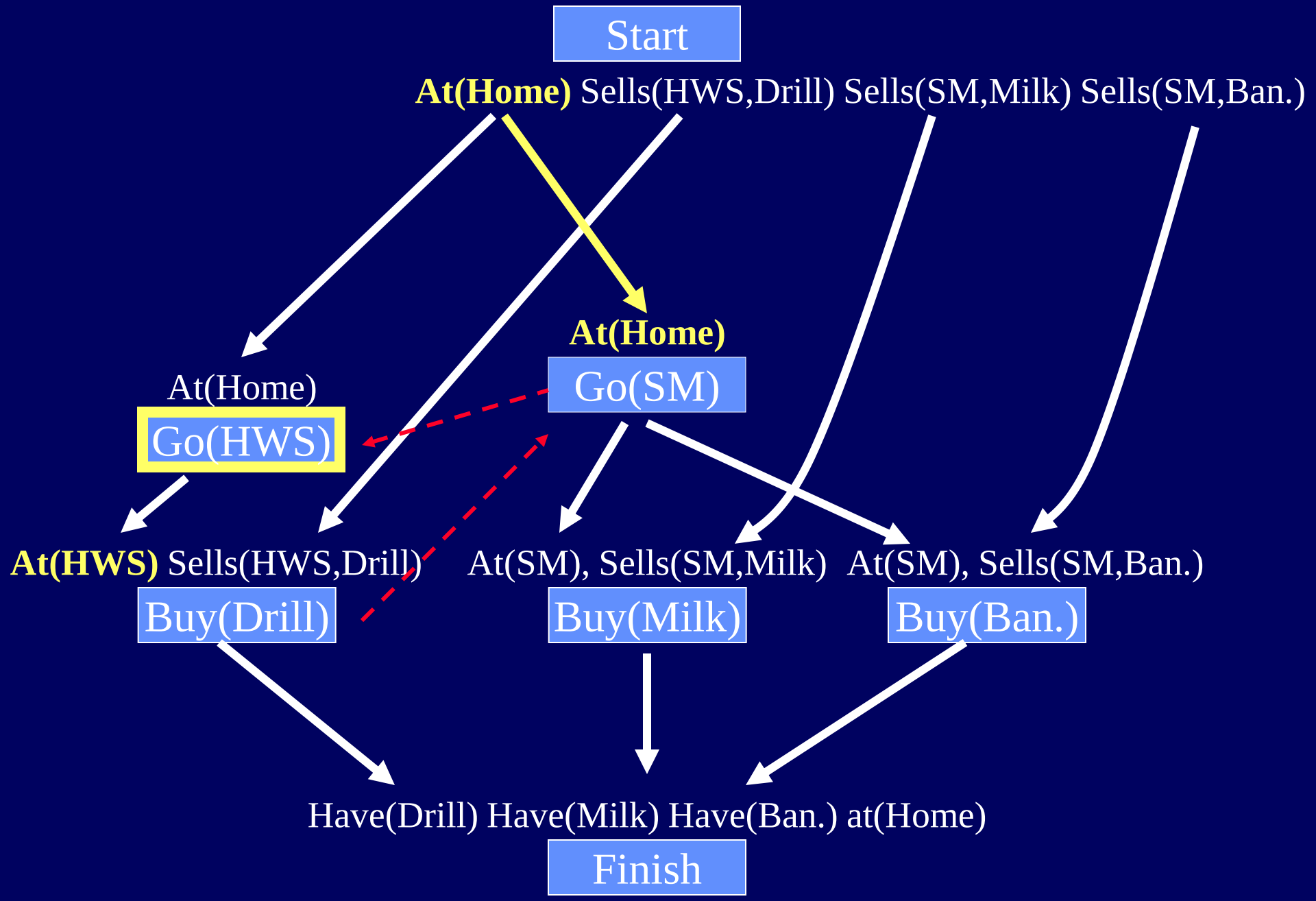

Start
At(Home) Sells(HWS,Drill) Sells(SM,Milk) Sells(SM,Ban.)
At(HWS) Sells(HWS,Drill)
At(SM), Sells(SM,Milk)
At(SM), Sells(SM,Ban.)
Buy(Drill)
Buy(Milk)
Buy(Ban.)
Have(Drill) Have(Milk) Have(Ban.) at(Home)
Finish

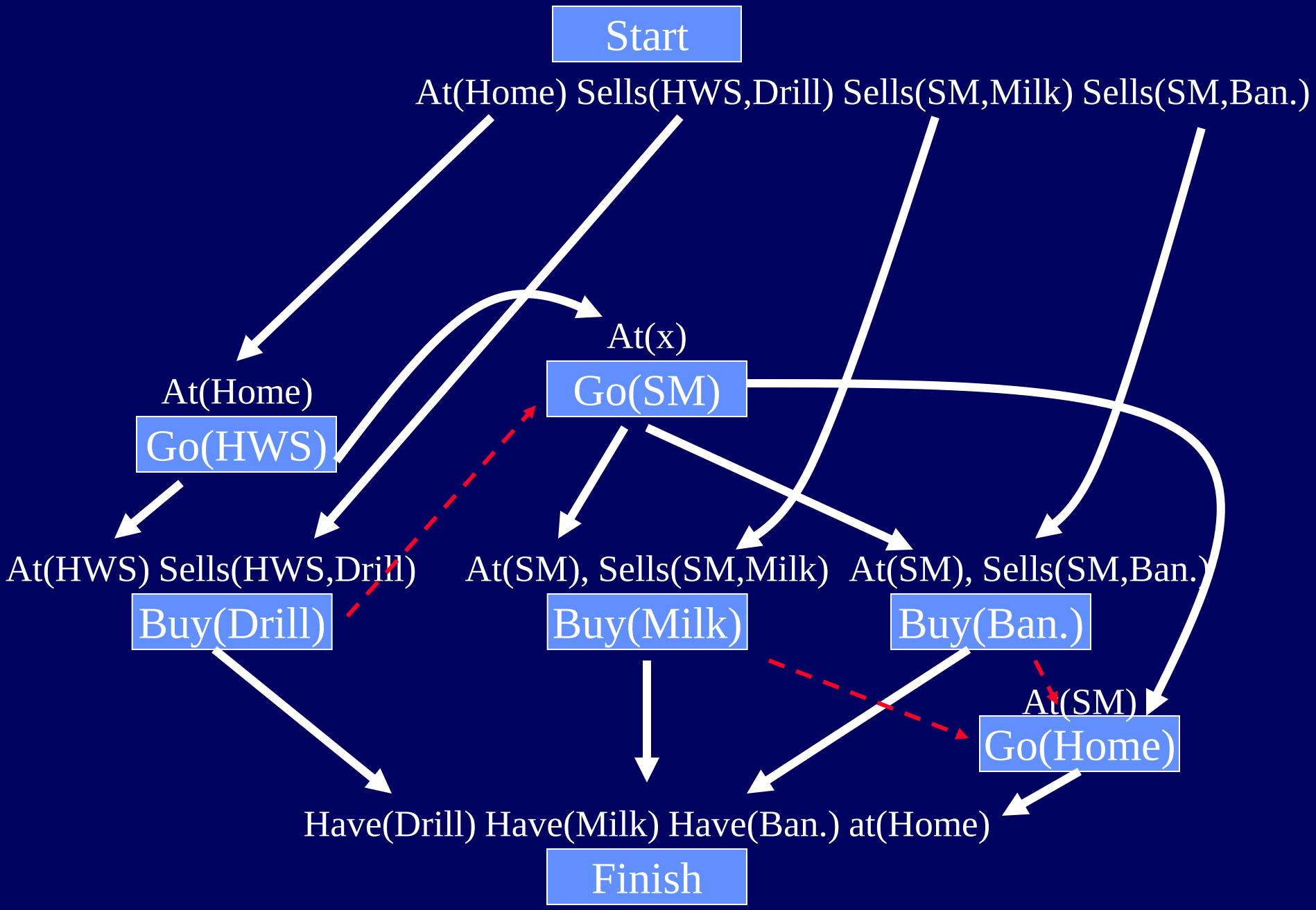

Start
At(Home) Sells(HWS,Drill) Sells(SM,Milk) Sells(SM,Ban.)
At(HWS) Sells(HWS,Drill)
At(SM), Sells(SM,Milk)
At(SM), Sells(SM,Ban.)
Buy(Drill)
Buy(Milk)
Buy(Ban.)
Have(Drill) Have(Milk) Have(Ban.) at(Home)
Finish

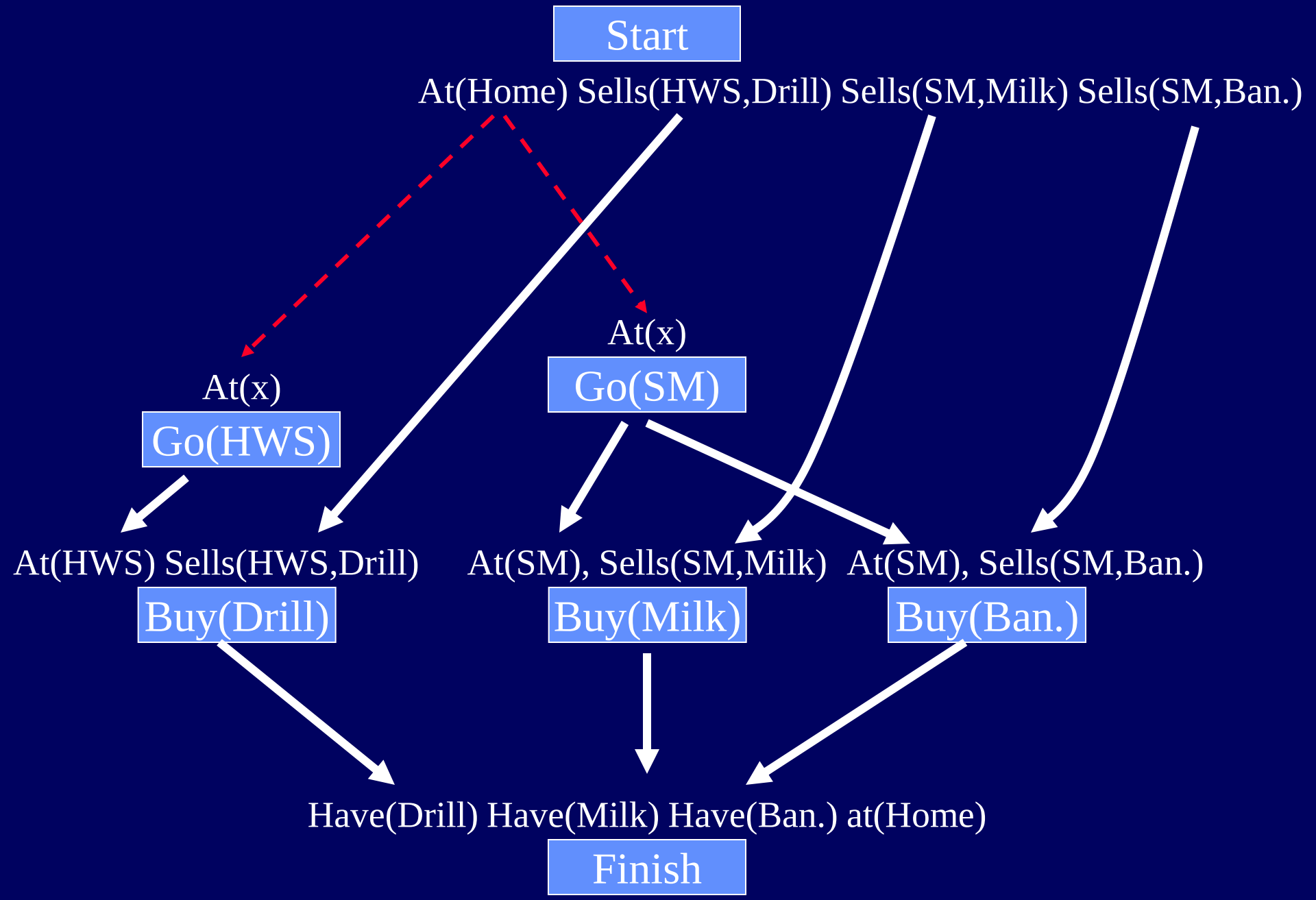

Start
At(Home) Sells(HWS,Drill) Sells(SM,Milk) Sells(SM,Ban.)
At(x)
Go(HWS)
At(x)
Go(SM)
At(HWS) Sells(HWS,Drill)
Buy(Drill)
At(SM), Sells(SM,Milk)
Buy(Milk)
At(SM), Sells(SM,Ban.)
Buy(Ban.)
Have(Drill) Have(Milk) Have(Ban.) at(Home)
Finish

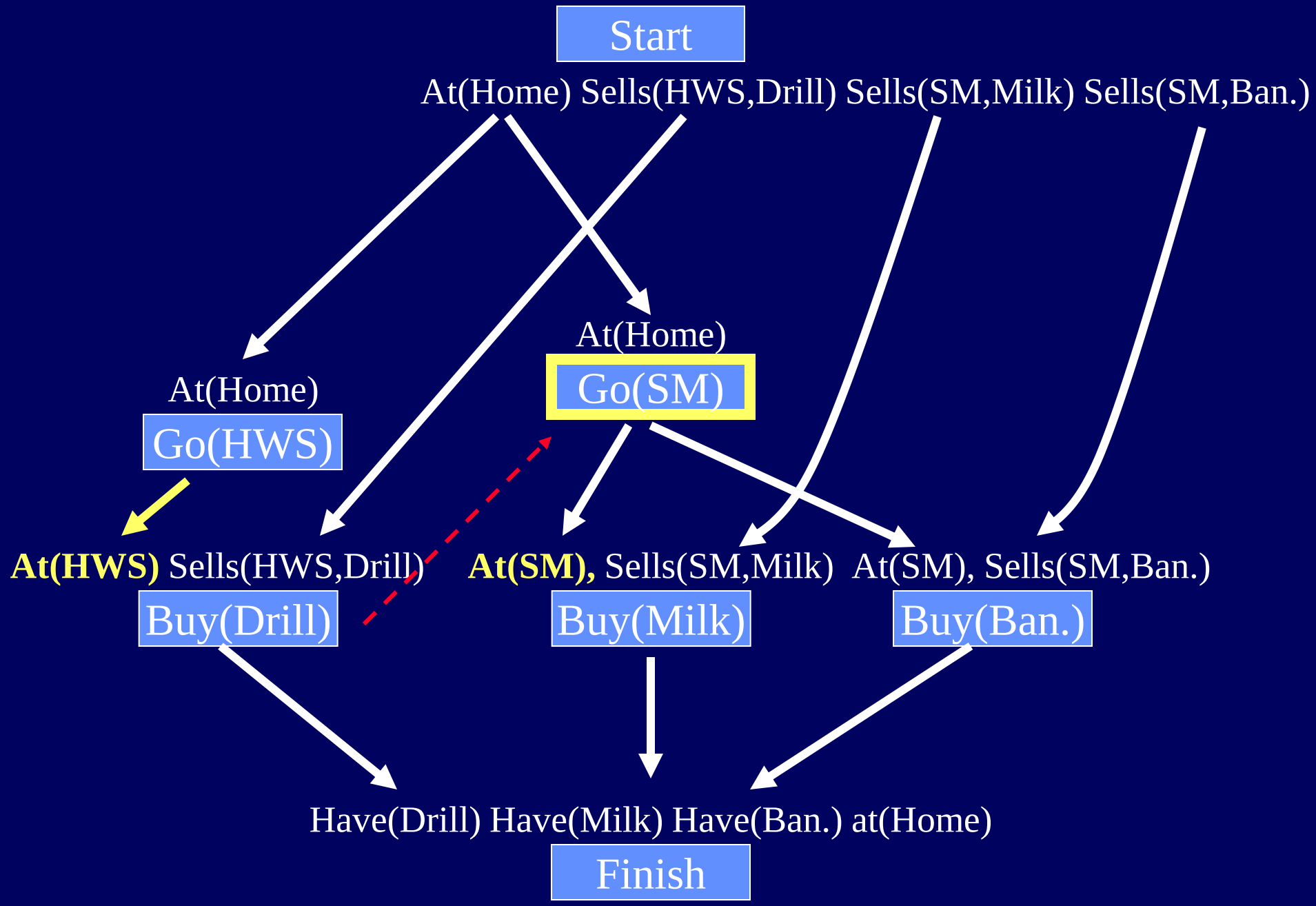

Start
At(Home) Sells(HWS,Drill) Sells(SM,Milk) Sells(SM,Ban.)
At(Home)
Go(HWS)
At(Home)
Go(SM)
At(HWS) Sells(HWS,Drill)
Buy(Drill)
At(SM), Sells(SM,Milk)
Buy(Milk)
At(SM), Sells(SM,Ban.)
Buy(Ban.)
Have(Drill) Have(Milk) Have(Ban.) at(Home)
Finish

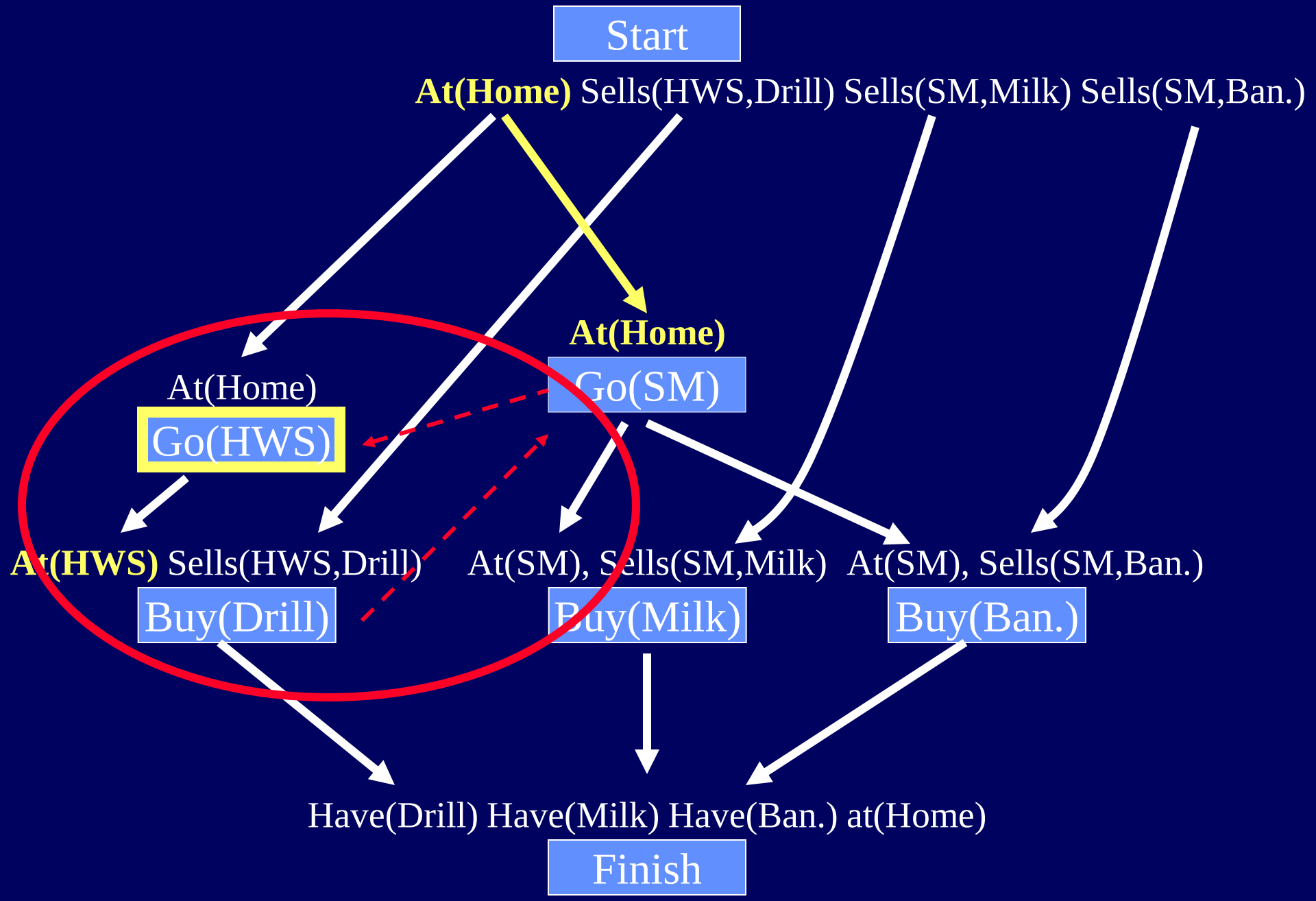

Start
At(Home) Sells(HWS,Drill) Sells(SM,Milk) Sells(SM,Ban.)
At(Home)
Go(HWS)
At(Home)
Go(SM)
At(HWS) Sells(HWS,Drill)
Buy(Drill)
At(SM), Sells(SM,Milk)
Buy(Milk)
At(SM), Sells(SM,Ban.)
Buy(Ban.)
Have(Drill) Have(Milk) Have(Ban.) at(Home)
Finish

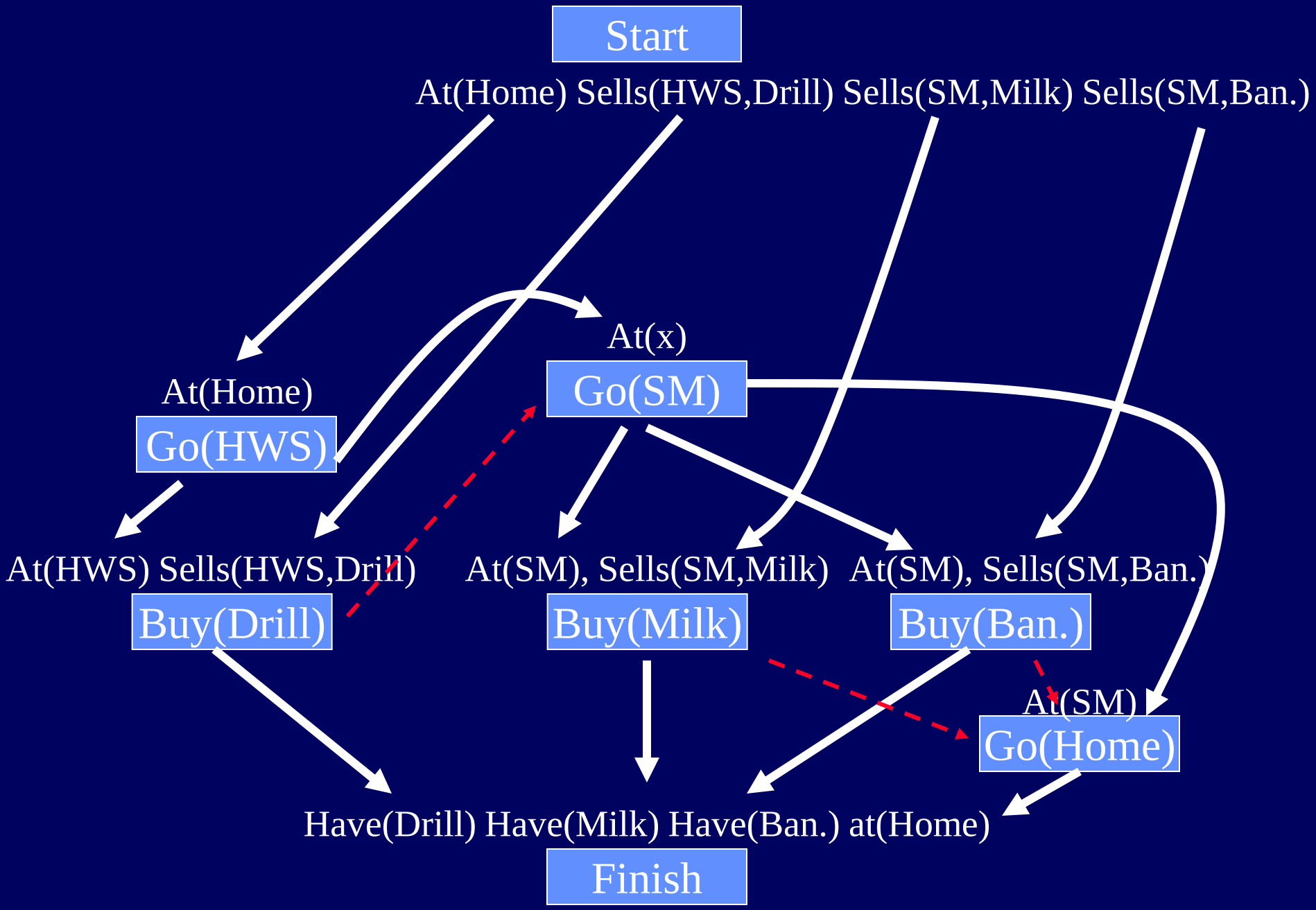

Start
At(Home) Sells(HWS,Drill) Sells(SM,Milk) Sells(SM,Ban.)
At(Home)
Go(HWS)
At(Home)
Go(SM)
At(HWS) Sells(HWS,Drill)
Buy(Drill)
At(SM), Sells(SM,Milk)
Buy(Milk)
At(SM), Sells(SM,Ban.)
Buy(Ban.)
Have(Drill) Have(Milk) Have(Ban.) at(Home)
Finish


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


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


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


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


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

Autonomous Agents: What is missing?

Self-commanding
Self-diagnosing
Self-repairing

Monitors

Command dispatch
Fault protection
Attitude control

Mission Goal Scenario

Many Action Representations: (Many Studied In This Course)