Plan-Space Planning

Chapter 10

Outline

◇ Motivation

◇ Partial plans // plans that are not complete

◇ Flaws

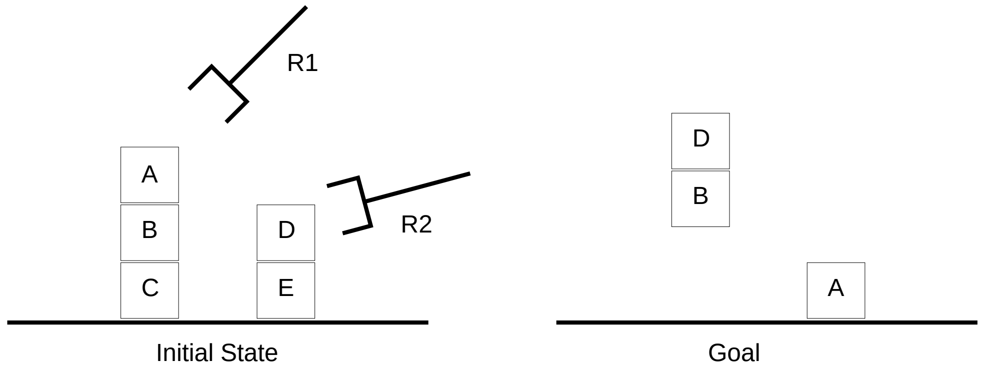
◇ Plan-space planning algorithm

◇ Example

Motivation

State-space search produces inflexible plans.

Part of the ordering in an action sequence is not related to causality:



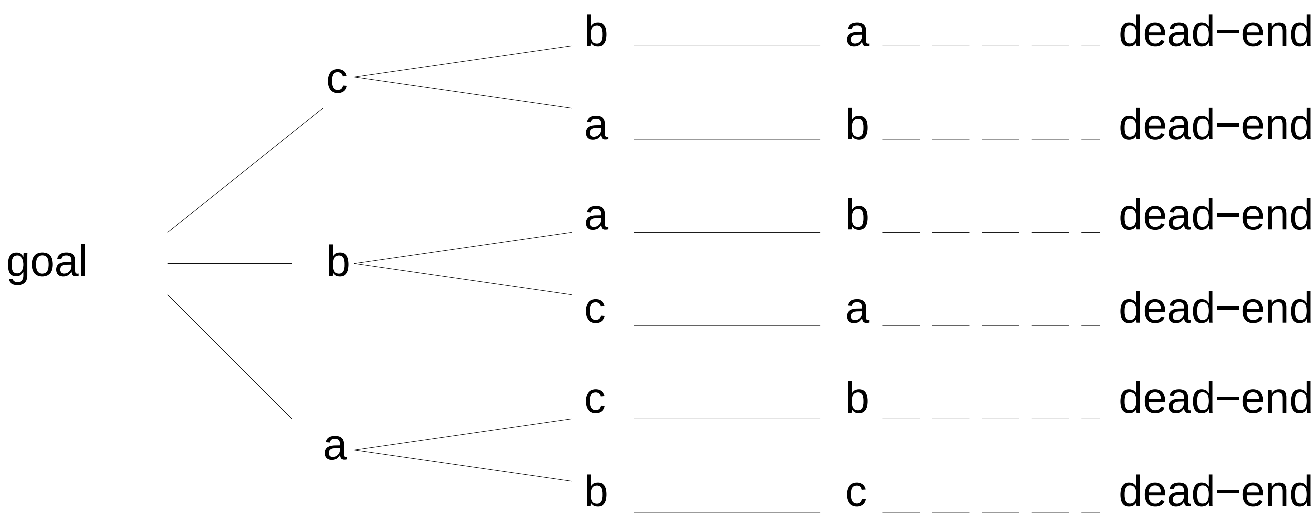
sequence:
$\langle \text{unstack}(R1, A, B), \text{unstack}(R2, D, E), \text{putdown}(R1, A), \text{stack}(R2, D, B) \rangle$

partially ordered plan only needs: $\text{unstack}(R1, A, B) < \text{putdown}(R1, A)$, $\text{unstack}(R2, D, E) < \text{stack}(R2, D, B)$, and $\text{unstack}(R1, A, B) < \text{stack}(R2, D, B)$

Seq. of action
not set

Motivation

State-space search wastes time examining many different orderings of the same set of actions:



Not ordering actions unecessarily can speed up planning

Motivation

why $A \to B$:

① A would be necessary for B

② B may delete precon. of A

Plan space search:

- no notion of states, just partial plans
- adopts a least-commitment strategy: don't commit to orderings, instantiations, etc, unless necessary
- produces a partially ordered plan: represents all sequences of actions compatible with the partial ordering
- benefits: speed-ups (in principle), flexible execution, easier replanning

Plan space search: basic idea

Plan-space search builds a **partial plan**: basic notion (nodes)

if $A \to B$, $B \to C$ then $A \to C$

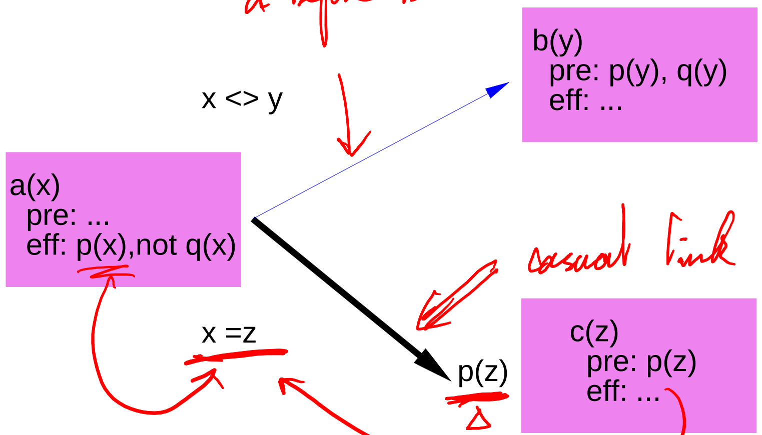
- multiset O of operators $\{o_1, \ldots, o_n\}$
- set $<$ of ordering constraints $o_i < o_j$ (with transitivity built in)
- set B of binding constraints $x = y$, $x \neq y$, $x \in D$, $x \notin D$, substitutions
- set L of causal links $o_i \xrightarrow{p} o_j$ stating that (effect p) of o_i establishes precondition p of o_j, with $o_i < o_j$ and binding constraints in B for parameters of o_i and o_j appearing in p

triples (o_i, o_j, p)

"o_i produces precondition p for o_j"

a before b

causal link



binding the eff. & precon. for different operators

Plan-space search: basic idea

Nodes are partial plans

"empty partial plan"

- initial node is $(O : \{\text{start}, \text{end}\}, <: \{\text{start} < \text{end}\}, B : \{\,\}, L : \{\,\})$ with $\text{EFF}(\text{start}) = s_0$ and $\text{PRE}(\text{end}) = g$.

only ordering constraint.

Successors are determined by plan refinement operations

①②③ Order inconsistency

- each operation add elements to O, $<$, B, L to resolve a flaw in the plan

Search through the plan space until a partial plan is found which has no flaw:

- no open precondition: all preconditions of all operators in O are established by causal links in L
- no threat (each linearisation is safe): for every causal link $o_i \xrightarrow{p} o_j$, every o_k with $\text{EFF}^-(o_k)$ unifable with p is such that $o_k < o_i$ or $o_j < o_k$
- $<$ and B are consistent

How to resolve an open precondition flaw?

Flaw: an operator o in the plan has a precondition p which is not established

p

o

holding(r,A)

putdown(r,A)
pre: holding(r,A)
eff: ...

How to resolve an open precondition flaw?

Flaw: an operator o in the plan has a precondition p which is not established

Resolving the flaw:

1. find an operator o' (either already in the plan or insert it) which can be used to establish p, i.e. o' can be ordered before o and one of its effects can unify with p

o’

unstack(r’,x’,B)
pre: ...
eff: holding(r’,x’), ...

p

holding(r,A)

o

putdown(r,A)
pre: holding(r,A)
eff: ...

How to resolve an open precondition flaw?

Flaw: an operator o in the plan has a precondition p which is not established

Resolving the flaw:

1. find an operator o' (either already in the plan or insert it) which can be used to establish p, i.e. o' can be ordered before o and one of its effects can unify with p
2. add to B binding constraints to unify the effect of o' with p

o'

r = r'

unstack(r',A,B)
pre: ...
eff: holding(r',A), ...

p

holding(r,A)

o

putdown(r,A)
pre: holding(r,A)
eff: ...

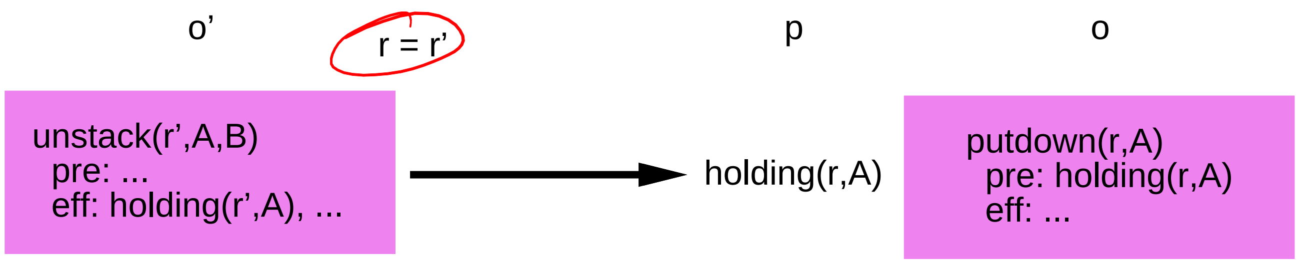
How to resolve an open precondition flaw?

Flaw: an operator o in the plan has a precondition p which is not established

Resolving the flaw:

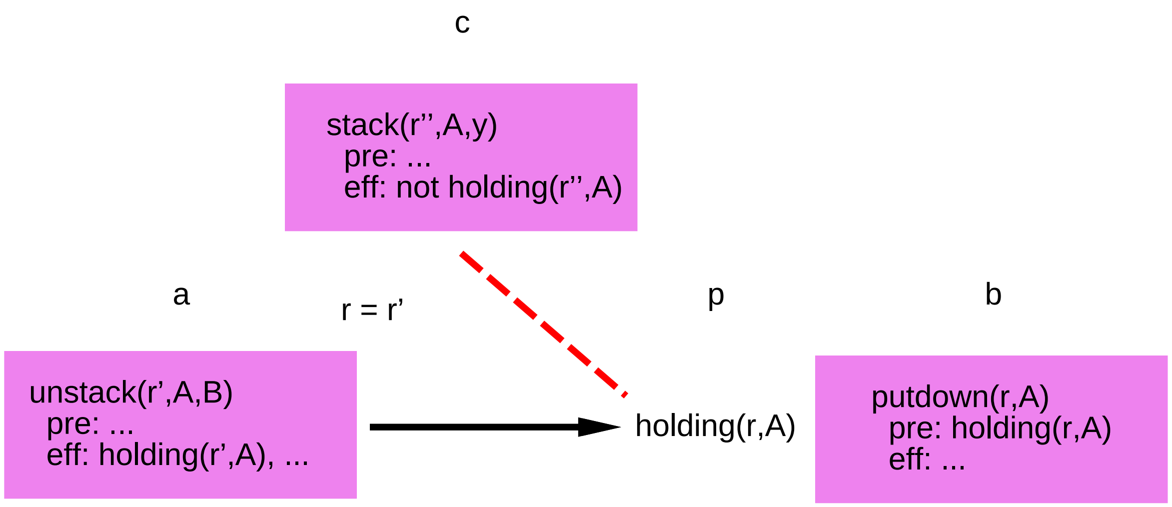
1. find an operator o' (either already in the plan or insert it) which can be used to establish p, i.e. o' can be ordered before o and one of its effects can unify with p
2. add to B binding constraints to unify the effect of o' with p
3. add to L the causal link $o' \xrightarrow{p} o$ (and the ordering constraint $o' < o$).

underlying



How to resolve a threat flaw?

Flaw: An operator a establishes a condition p for operator b, but another operator c is capable of deleting p before b gets to use it

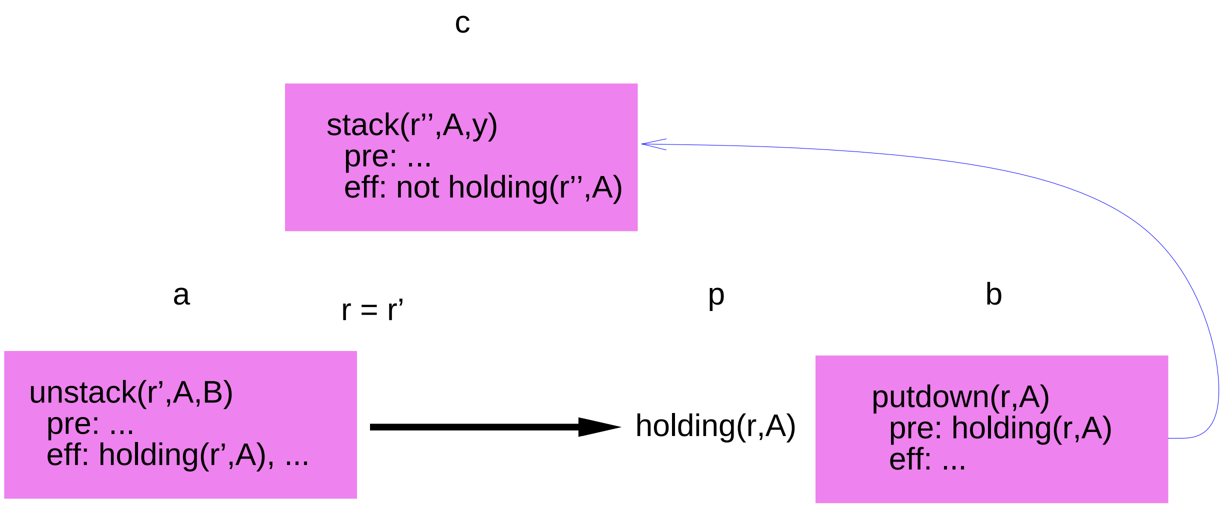


How to resolve a threat flaw?

Flaw: An operator a establishes a condition p for operator b, but another operator c is capable of deleting p before b gets to use it

Resolving the flaw - 3 possibilities:

1. order c after b

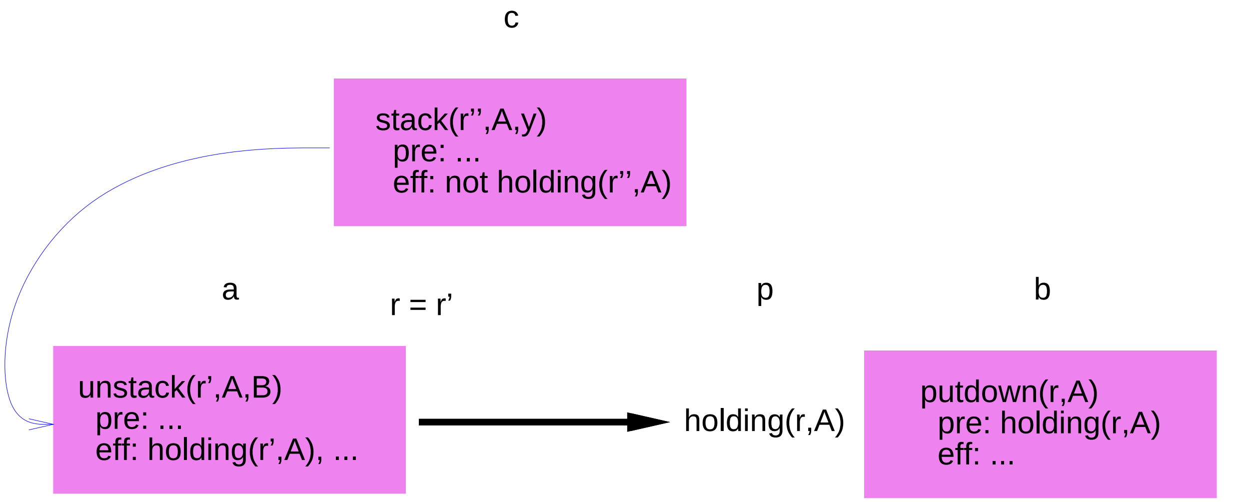


How to resolve a threat flaw?

Flaw: An operator a establishes a condition p for operator b, but another operator c is capable of deleting p before b gets to use it

Resolving the flaw - 3 possibilities:

1. order c after b
2. order c before a

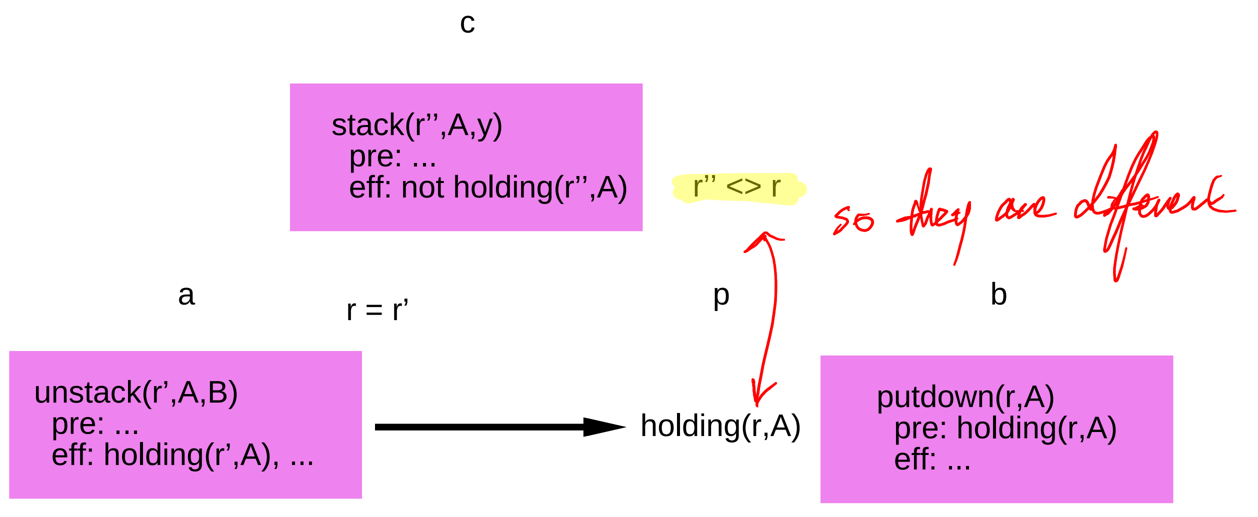


How to resolve a threat flaw?

Flaw: An operator a establishes a condition p for operator b, but another operator c is capable of deleting p before b gets to use it

Resolving the flaw - 3 possibilities:

1. order c after b
2. order c before a
3. add a binding constraint preventing c to delete p



Plan-space planning algorithm

function PLAN-SPACE-PLANNING(π) **returns** a plan, or failure

$F \leftarrow$ OPEN-PRECONDITIONS(π) $\cup$ THREATS(π) // flaws to be resolved
if $F = \{\}$ **then return** π
select a flaw $f \in F$ just choose any [though there could be efficient selections]
$R \leftarrow$ RESOLVE(f, π)
if $R = \{\}$ **then return** failure / expensive bit.
choose a resolver $r \in R$
$\pi' \leftarrow$ REFINE(r, π)
return PLAN-SPACE-PLANNING(π')

PLAN-SPACE-PLANNING is sound and complete

Grounded variant: no binding constraints needed

Example

- operator Start
  - precondition: $\{ \}$
  - effect: $\{$At(Home), Sells(MM,Drill), Sells(WH,Milk), Sells(WH,Bananas)$\}$
- operator End
  - precondition: $\{$At(Home), Have(Drill), Have(Milk), Have(Bananas)$\}$
  - effect: $\{ \}$
- operator Go(l, l')
  - precondition: $\{$At$(l)\}$
  - effect: $\{$At$(l'), \neg$At$(l)\}$
- operator Buy(i, s)
  - precondition: $\{$At(s), Sells$(s, i)\}$
  - effect: $\{$Have$(i)\}$

Example

Initial Plan

Start

Top/Left

At(Home), Sells(MM,Drill), Sells(WH,Milk), Sells(WH, Bananas)

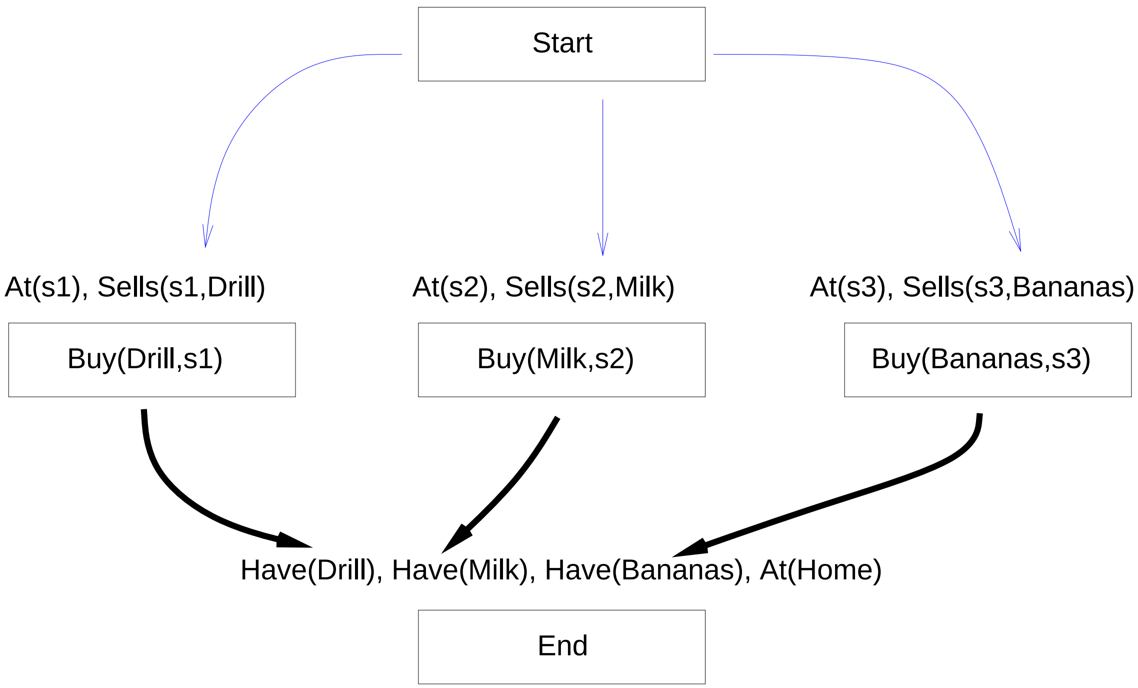
Have(Drill), Have(Milk), Have(Bananas), At(Home)

End

Bottom/Right

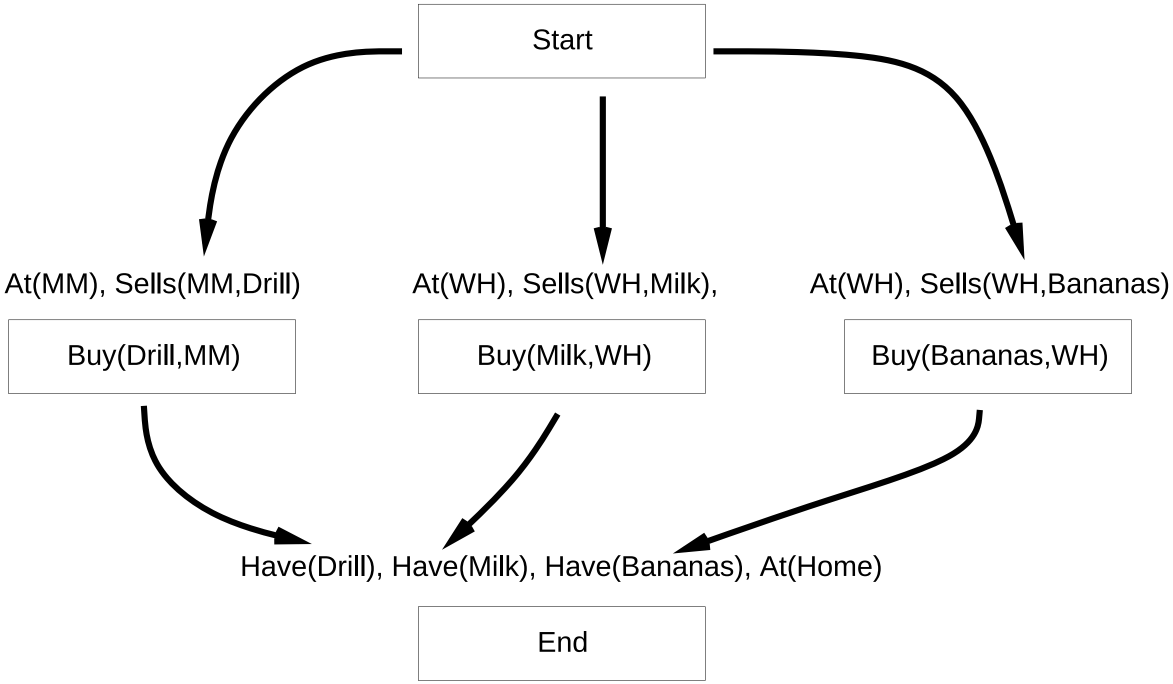
Example

The only possible ways to establish the "Have" preconditions



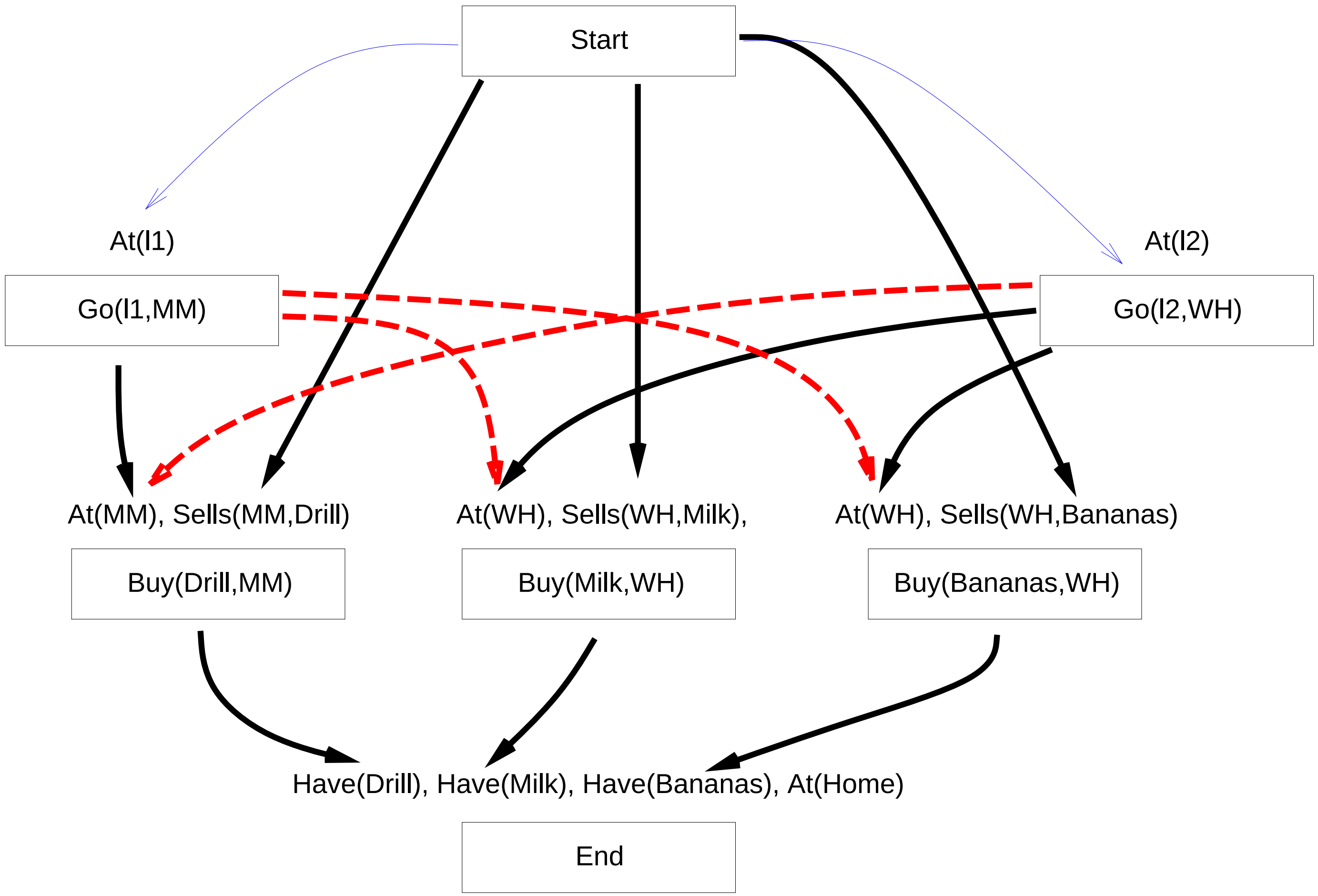
Example

The only possible ways to establish the “Sells” preconditions



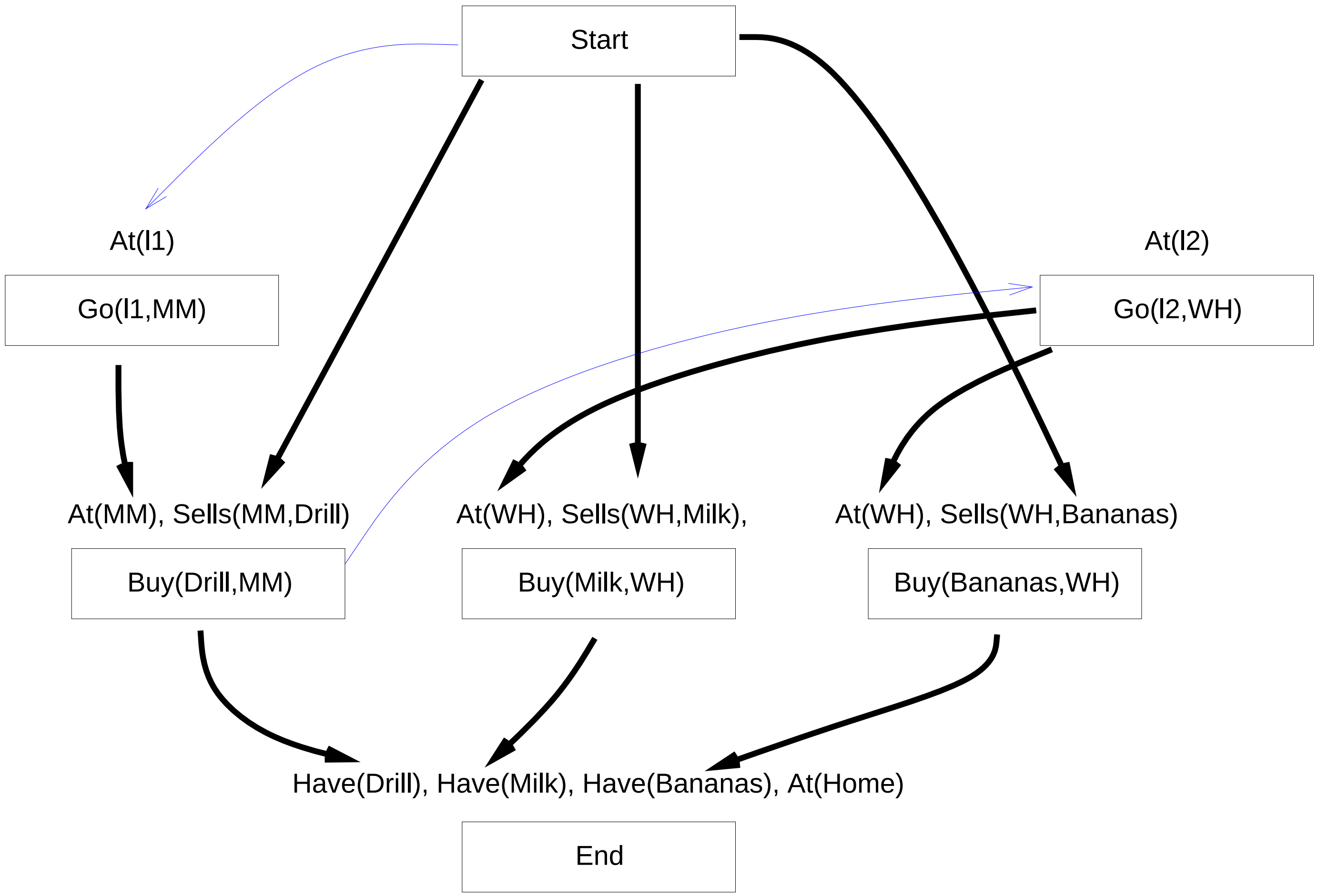
Example

The only ways to establish At(MM) and At(WH).
Note the threats



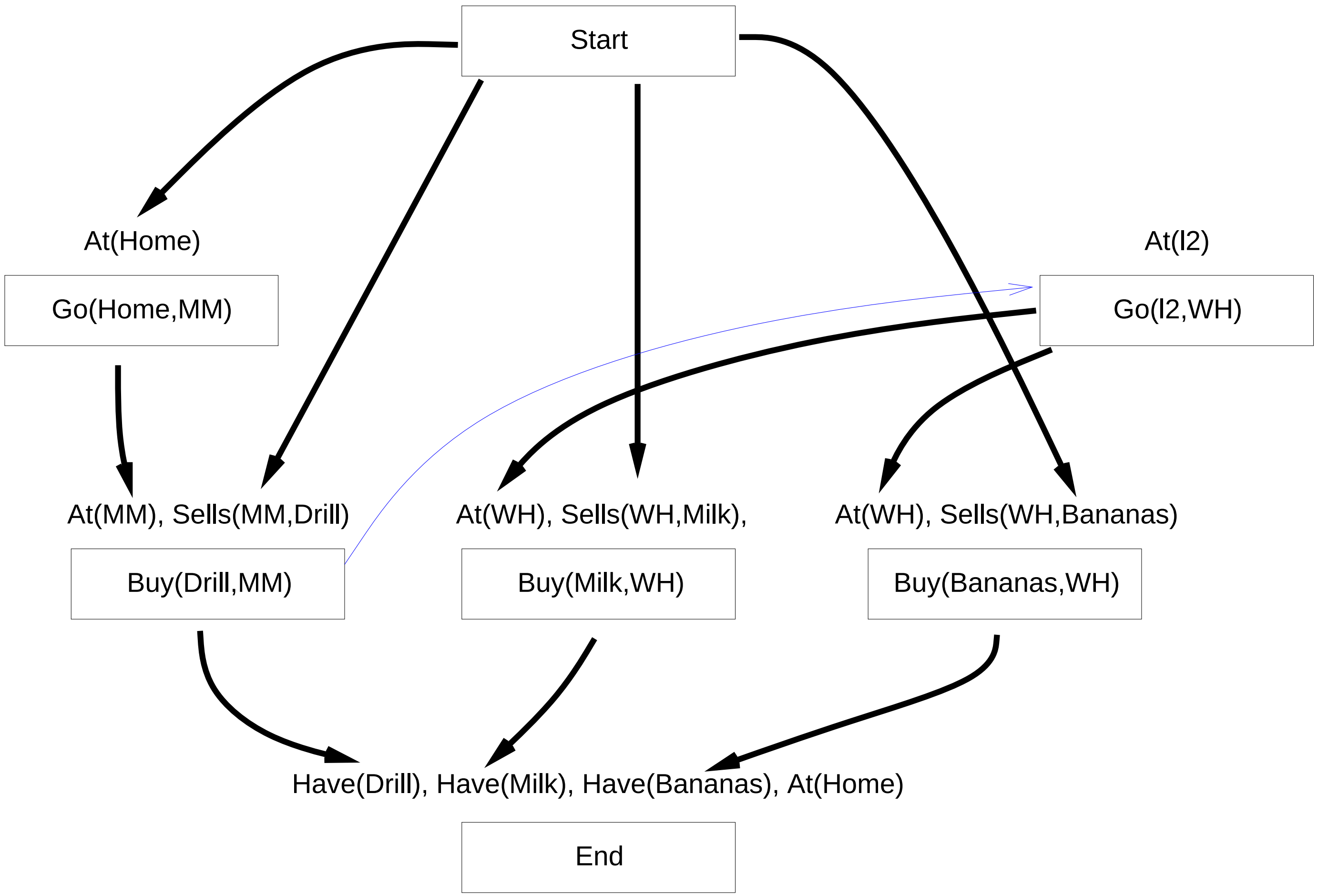
Example

To resolve the 3rd threat, order Go($l2$, WH) after Buy(Drill).
This resolves all three threats.



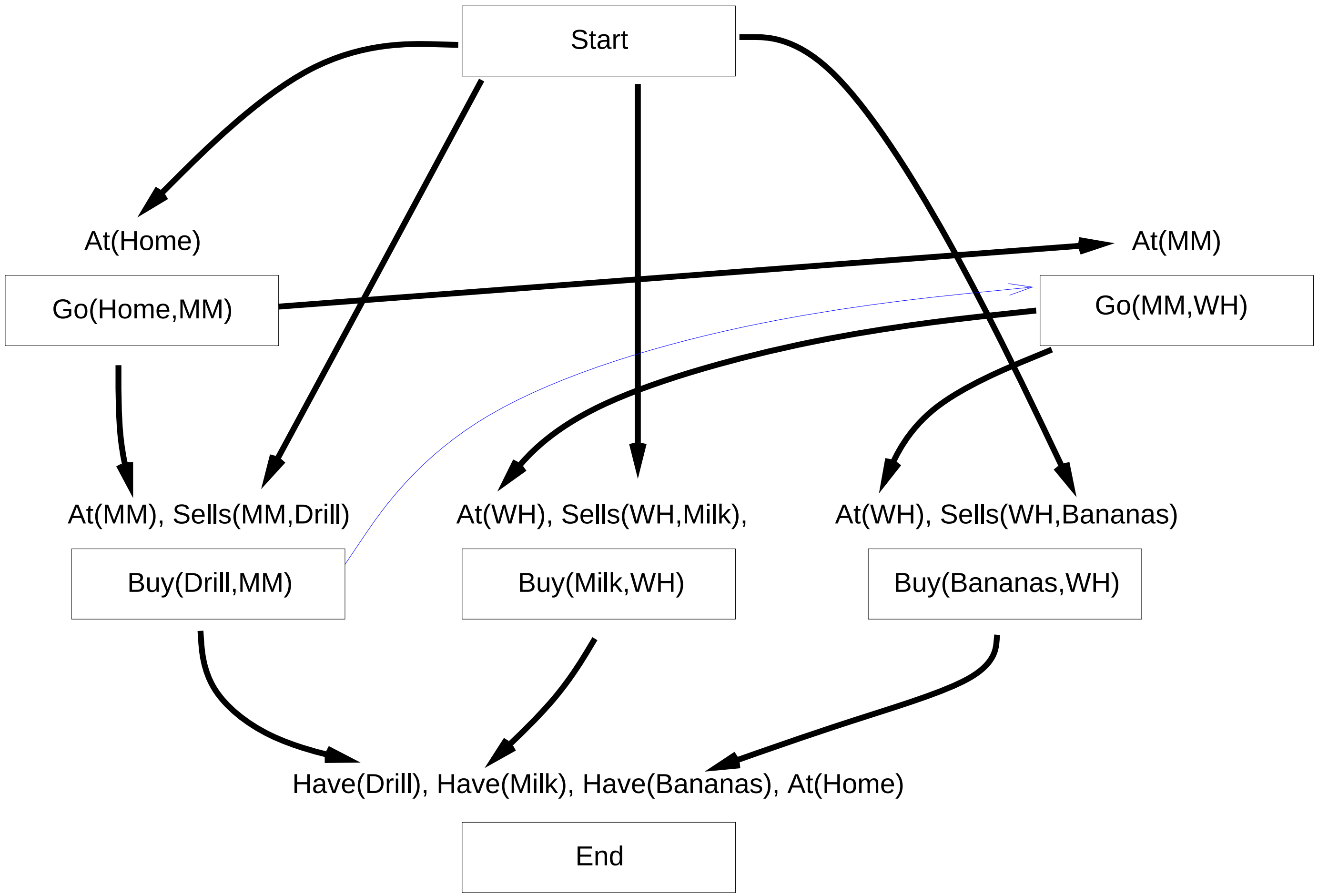
Example

Add binding constraint $l1 =$ Home and causal link to establish At$(l1)$



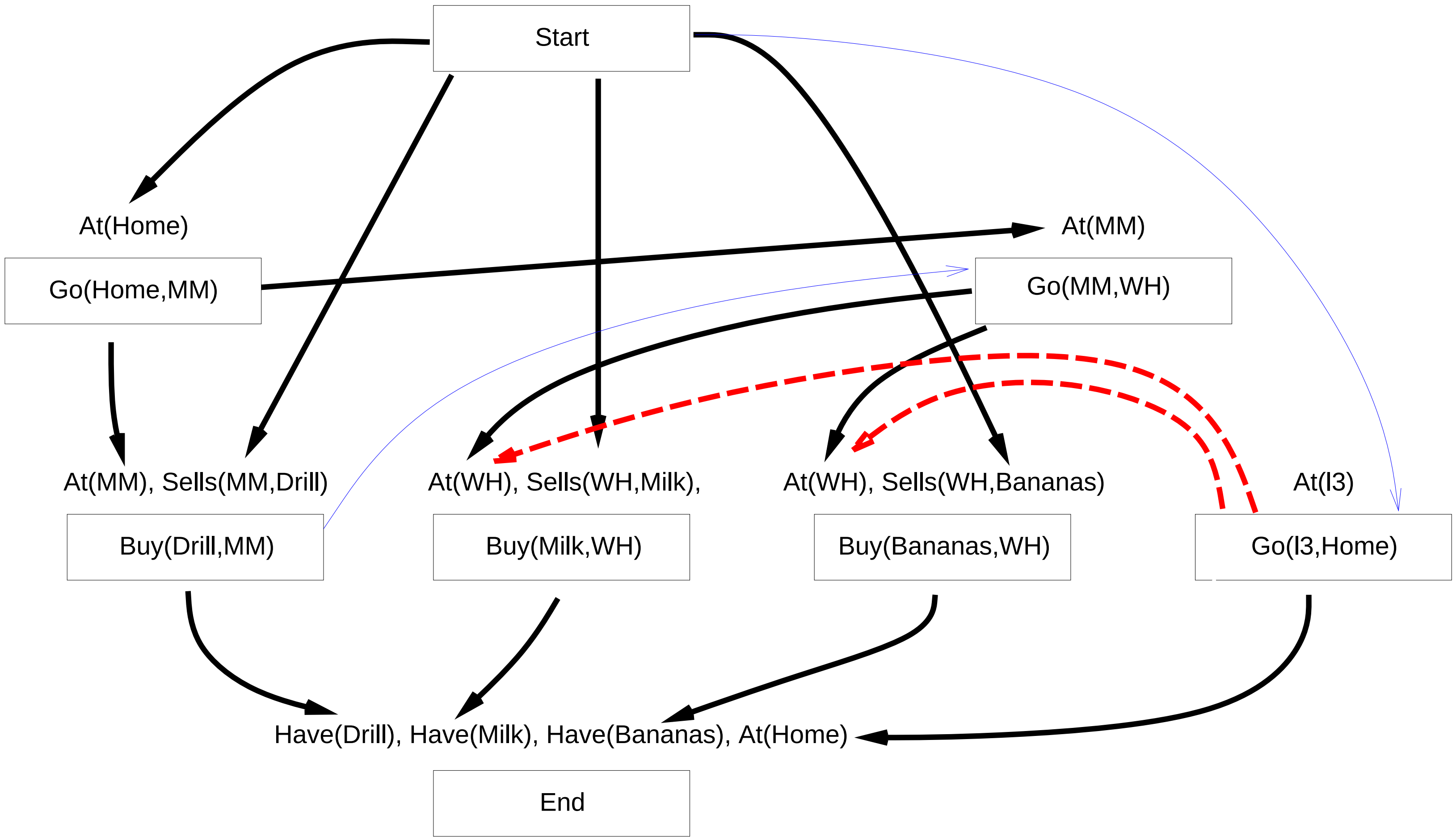
Example

Add binding constraint $l2 = \mathrm{MM}$ and causal link to establish $\mathrm{At}(l2)$



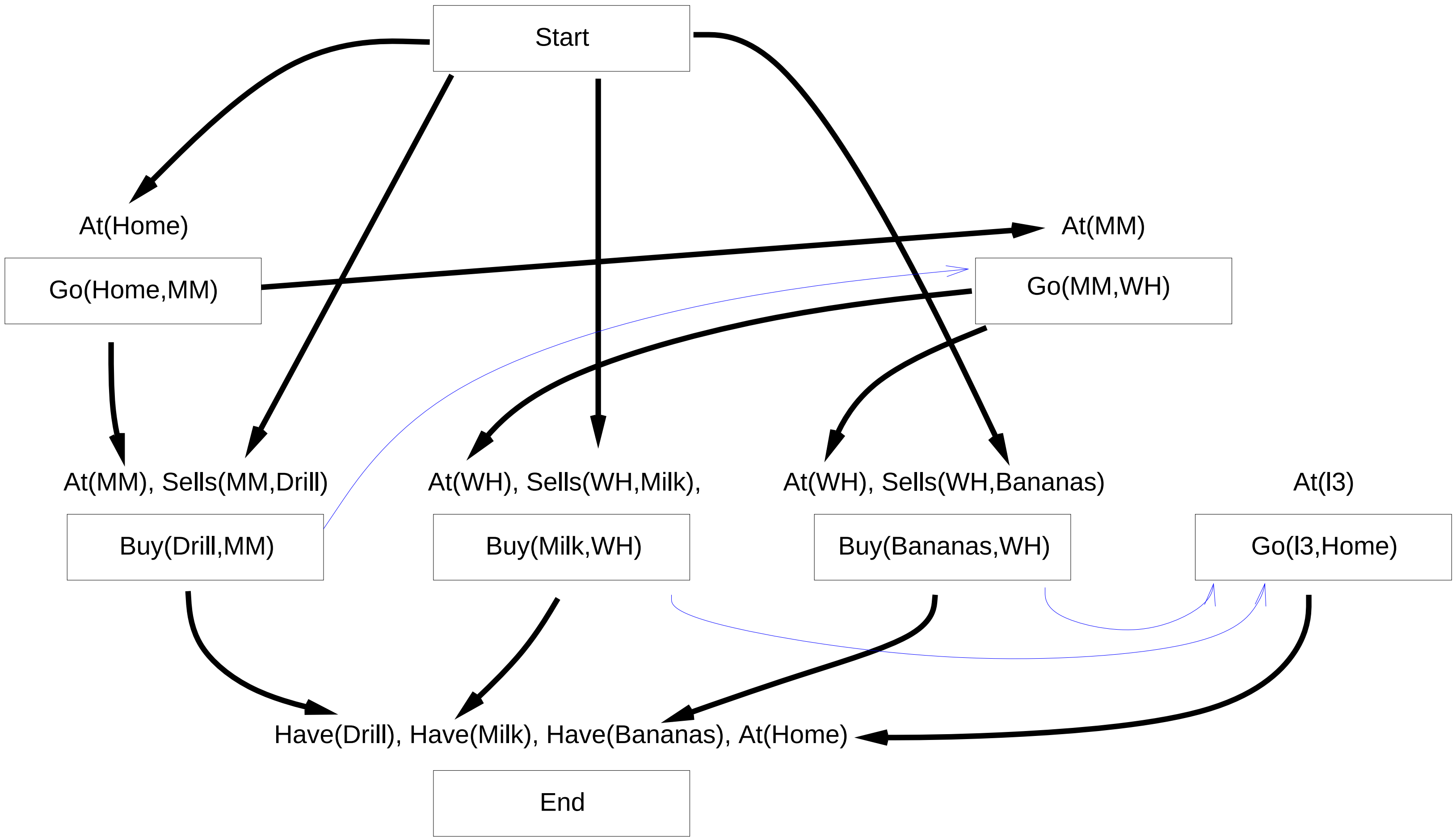
Example

Establish At(Home) for end.
Note the threats.



Example

Order Go(Home) after Buy(Milk) and Buy(Banana) to remove the threats.



Example

Add binding constraint $l3 = \text{WH}$ and causal link to establish At$(l3)$.
The plan is flawless.