Planejamento apoiado em Inteligência Artificial

Sistemas de Informação Inteligente

Prof. Leandro C. Fernandes

Adaptação dos materiais de:
Edirlei Soares e outros autores

Introdução

- **Planejamento** consiste na tarefa de apresentar uma sequência de ações para alcançar um determinado objetivo.
  - Ir(Mercado), Comprar(Biscoito), Ir(Farmácia), Comprar(Remédio), Ir(Casa)
- Dado um objetivo, um **agente planejador** deve ser capaz de construir um plano de ação para chegar ao seu objetivo.
- Após planejar, o agente deve **executar as ações do plano** uma a uma.

Funcionamento de um Agente Planejador

- Inicialmente um agente planejador gera um objetivo a ser alcançado.
- A partir disto constrói um plano para atingir esse objetivo a partir do estado atual do ambiente.
- Em seguida executa o plano do início.
- Após conquistá-lo, gera um novo objetivo com base no novo estado do ambiente.

Planejamento

- No planejamento clássico assumimos que o ambiente do problema possui as seguintes características:
  - Observável;
  - Determinístico;
  - Finito;
  - Estático;

Resolução de Problemas vs Planejamento

- **Algoritmos de busca** tendem a tomar ações irrelevantes.
  - Grande fator de ramificação.
  - Pouco conhecimento para guiar a busca.
- **Planejador** não considera ações despropositais.
  - Faz conexões diretas entre estados (sentenças) e ações (pré-condições + efeitos)
  - Objetivo: Ter(Leite).
    - Ação: Comprar(Leite) => Ter(Leite)

Resolução de Problemas vs Planejamento

- Para os problemas do mundo real é difícil definir uma boa heurística para algoritmos de busca heurística.
- Um planejador tem acesso a representação explícita do objetivo.
  - **Objetivo**: conjunção de sub-objetivos que levam ao objetivo final.
  - **Heurística única**: o número de elementos da conjunção que ainda não foram satisfeitos.

Resolução de Problemas vs Planejamento

- Enquanto os algoritmos de busca não tiram proveito da decomposição do problema...
- Os planejadores aproveitam a estrutura do problema. É possível decompor com facilidade sub-objetivos.
  - Exemplo:
    - Ter(A,B,C,D) = Ter(A) Λ Ter(B) Λ Ter(C) Λ Ter(D)

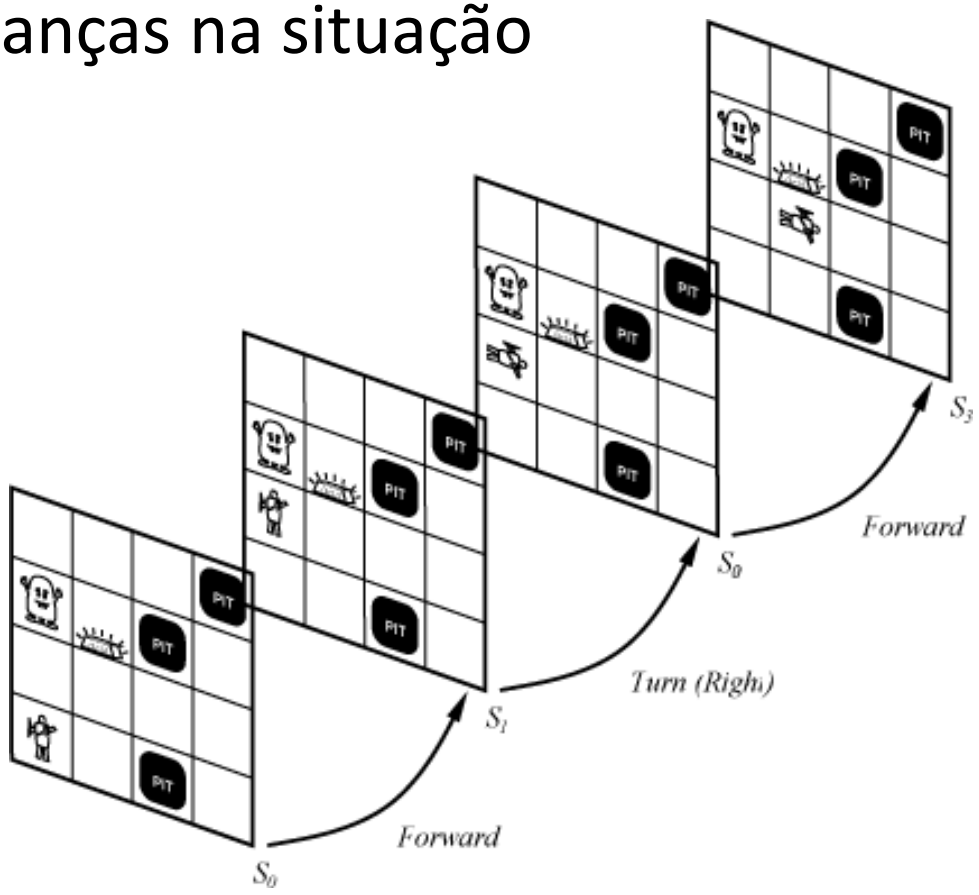
Planejamento: 3 idéias principais

- Representação dos estados, objetivos e ações usando LPO (descrições parciais dos estados)
  - Pode conectar diretamente estados e ações.
    - Ex. estado: Have(Milk), ação: Buy(milk) $\rightarrow$ Have(Milk)
- Adiciona ações ao plano quando forem necessárias
  - ordem de planejamento ≠ ordem de execução
  - primeiro, o que é importante : Buy(Milk) – pode-se colocar esta ação no plano, mesmo sem saber como chegar ao supermercado.
  - diminui fator de ramificação
- Uso da estratégia de dividir-e-conquistar
  - Definição de sub-planos: sub-plano supermercado, sub-plano loja de ferramentas (sub-metas)

Cálculo Situacional

- **Mundo**: sequência de situações (estados)
- **Ações**: provocam mudanças na situação

- Representação das mudanças no mundo:



Result(Forward,S0) = S1

Result(Turn(Right),S1) = S2

Result(Forward,S2) = S3

STANFORD RESEARCH INSTITUTE PROBLEM SOLVER (STRIPS)

STRIPS

- **STRIPS**: *STanford Research Institute Problem Solver*
- Planejador desenvolvido em 1971por Richard Fikes e Nils Nilsson;
- Modelo formal para representar ações e estados do mundo e um algoritmo que trata o problema como uma busca no espaço de estados;
  - Linguagem formal para a especificação de problemas de planejamento.

Linguagem STRIPS

- **Representação de estados:** conjunção de literais positivos sem variáveis.
  - **Inicial**: Em(Casa)
  - **Final**: Em(Casa) ^ Ter(Leite) ^ Ter(Bananas) ^ Ter(Furadeira)

- Trabalha com a **Hipótese do mundo fechado**, isto é, qualquer condição não mencionada em um estado é considerada negativa.
  - Ex: ¬Ter(Leite) ^ ¬Ter(Bananas) ^ ¬Ter(Furadeira)

Linguagem STRIPS

- **Objetivos**: conjunção de literais e possivelmente variáveis:
  - Em(Casa) ^ Ter(Leite) ^ Ter(Bananas) ^ Ter(Furadeira)
  - Em(x) ^ Vende(x, Leite)

- **Ações** são especificadas em termos de pré-condições e efeitos:
  - *Descritor da ação*: predicado lógico
  - *Pré-condição*: conjunção de literais positivos
  - *Efeito*: conjunção de literais (positivos ou negativos)

Linguagem STRIPS

- Operador para ir de um lugar para outro:
  - Ação:
    - **Ir** para um Destino,
  - Pré-condição
    - Estar **Em** Ponto de Partida
    - Haver um **caminho** da Partida para o Destino,
  - Efeito:
    - Estar **Em** Destino
    - Não estar **Em** Partida

- Em linguagem STRIPS, teríamos:

  Op (*ACTION*: **Ir**(destino),
  PRECOND: Em(partida) ^ Caminho(partida, destino),
  EFECT: Em(destino) ^ ¬ Em(partida))

Exemplo: Transporte Aéreo de Carga

Início(Em(C1, SFO) ^ Em(C2,JFK) ^ Em(A1,SFO) ^ Em(A2,JFK) ^ Carga(C1) ^ Carga(C2) ^ Avião(A1) ^ Avião(A2) ^ Aeroporto(JFK) ^ Aeroporto(SFO))

Objetivo(Em(C1,JFK) ^ Em(C2,SFO))

Ação(**Carregar**(c,a,l)

PRÉ-CONDIÇÃO: Em(c,l) ^ Em(a,l) ^ Carga(c) ^ Avião(a) ^ Aeroporto(l)

EFEITO: ¬Em(c,l) ^ Dentro(c,a))

Ação(**Descarregar**(c,a,l)

PRÉ-CONDIÇÃO: Dentro(c,a) ^ Em(a,l) ^ Carga(c) ^ Avião(a) ^ Aeroporto(l)

EFEITO: Em(c,l) ^ ¬Dentro(c,a))

Ação(**Voar**(a,de,para)

PRÉ-CONDIÇÃO: Em(a,de) ^ Avião(a) ^ Aeroporto(de) ^ Aeroporto(para)

EFEITO: ¬ Em(a,de) ^ Em(a,para))

Tipos de Planejadores

- Formas de Buscas de Planos:
  - Progressivo: estado inicial -> objetivo.
  - Regressivo: objetivo -> estado inicial.
    - Mais eficiente (há menos caminhos partindo do objetivo do que do estado inicial)
    - Problemático se existem múltiplos objetivos
- Espaços de busca:
  - Espaço de situações: Funciona da mesma forma que na resolução de problemas por meio de busca (nó = estado do mundo).
  - Espaço de planos: planos parciais (nó = plano parcial).
    - mais flexível.

Busca em Espaço de Situações

- A busca em espaço de situações é **ineficiente** devido a ela não considerar o problema das ações irrelevantes. Todas as opções de ações são testadas em cada estado.
- Isso faz com que a complexidade do problema cresça muito rapidamente.
- **Solução?** Busca no espaço de planos parciais **(planejamento de ordem parcial)**.

Planejamento de Ordem Parcial

- Subdivisão do problema.
- Ordem de elaboração do plano flexível.
- Compromisso mínimo.
  - Adiar decisões durante a procura.

- O planejador de ordem parcial pode inserir duas ações em um plano sem especificar qual delas deve ser executada primeiro.

Exemplo dos Sapatos

Inicio()

Objetivo(SapatoDireitoCalçado ^ SapatoEsquerdoCalçado)

Ação(**SapatoDireito**,

PRECOND: MeiaDireitaCalçada,

EFFECT: SapatoDireitoCalçado)

Ação(**MeiaDireita**,

EFFECT: MeiaDireitaCalçada)

Ação(**SapatoEsquerdo**,

PRECOND: MeiaEsquerdaCalçada,

EFFECT: SapatoEsquerdoCalçado)

Ação(**MeiaEsquerda**,

EFFECT: MeiaEsquerdaCalçada)

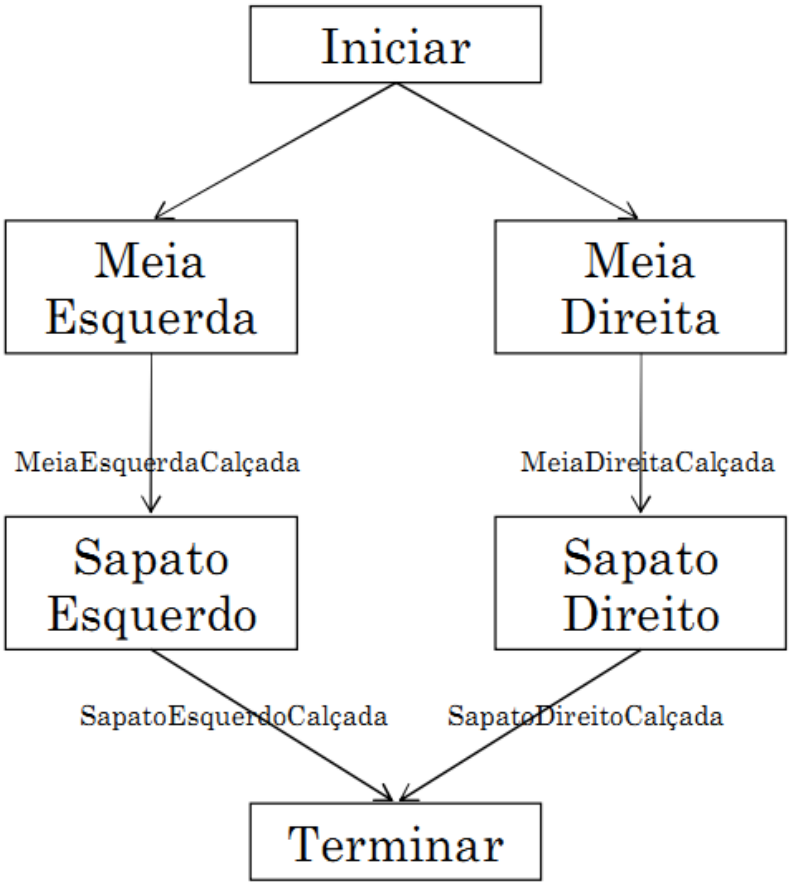
Exemplo dos Sapatos

- Para este exemplo, um planejador de ordem parcial deve ser capaz de **chegar a duas sequências** de ações:
  - *MeiaDireita* seguido por *SapatoDireito*;
  - *MeiaEsqueda* seguido por *SapatoEsquerdo*.

- As duas sequências podem ser **combinadas** para produzir o plano final.

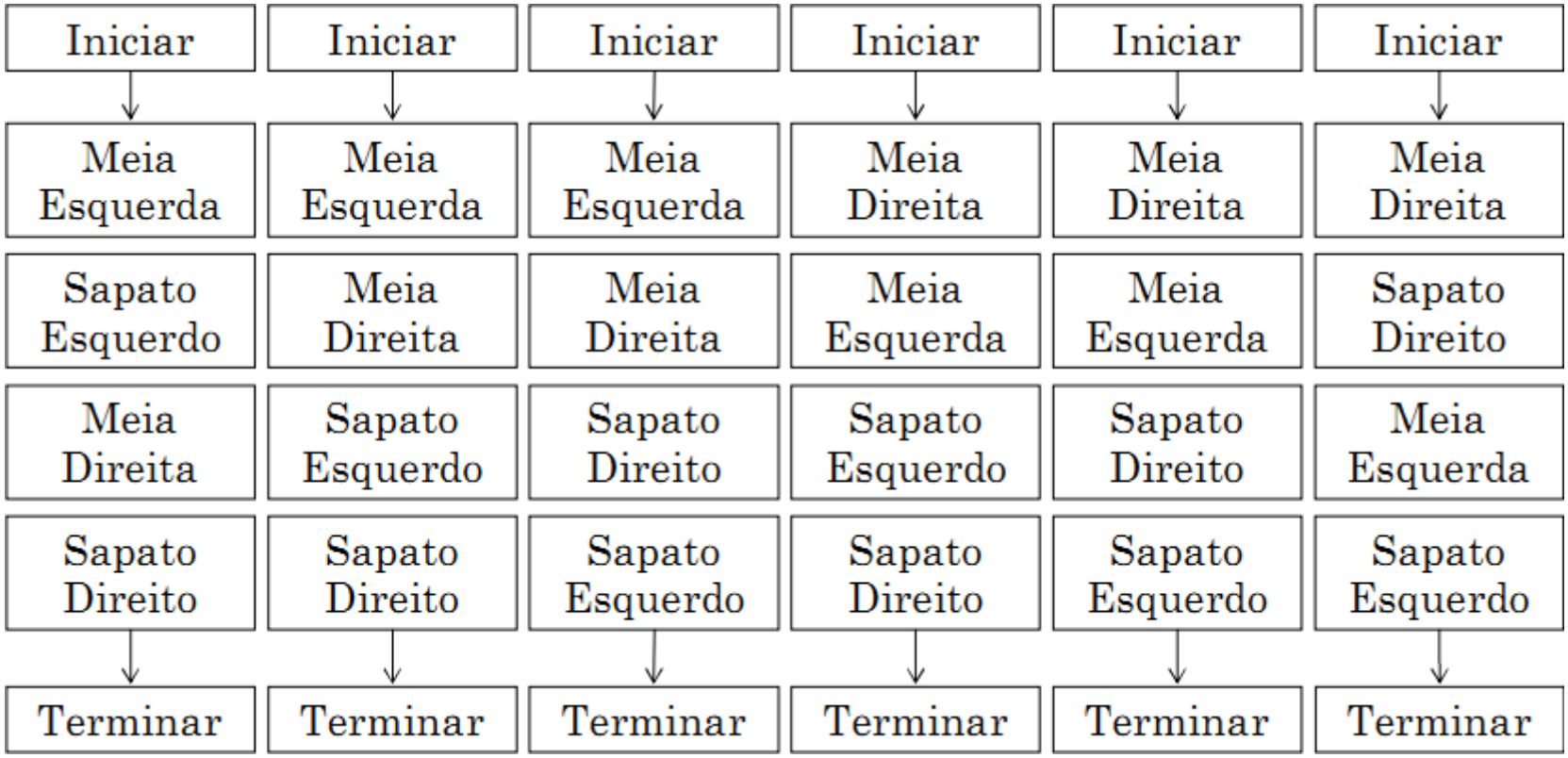
Exemplo dos Sapatos

- Plano de Ordem Parcial



Exemplo dos Sapatos

- Plano de Ordem Total



Planejador de Ordem Parcial (POP)

- O planejamento de ordem parcial pode ser implementado como uma busca no espaço de ordem parcial de planos.
  - **Ideia**:
    - Busca-se um plano desejado em vez de uma situação desejada (meta-busca).
    - Parte-se de um plano inicial (parcial) e aplica-se 2 tipos de operadores até chegar a um plano final (completo)
  - **Plano Final**:
    - **Completo**: todas as pré-condições de todas as ações são alcançada por meio de alguma outra ação.
    - **Consistente**: não há contradições.

Planejador de Ordem Parcial (POP)

- Operador de **refinamento**:
  - Adicionar um novo passo;
  - Instanciar a(s) variável(is);
  - Ordenar os passos;

- Operador de **modificação**:
  - Operadores de revisão (para corrigir planos);
  - Operadores de decomposição hierárquica;

Planejador de Ordem Parcial (POP)

- Na estratégia de **compromisso mínimo** a ordem e instanciações totais são decididas quando necessário.
  - Exemplo:
    - Para objetivo Ter(Leite), a ação Comprar(Produto, Loja), instancia-se somente item: **Comprar(Leite, Loja)**
    - Para o problema de colocar meias e sapatos: colocar cada meia antes do sapato, sem dizer por onde começar (esquerda ou direita)

Planejamento de Ordem Parcial

- Planos são definidos por 4 componentes:
  - **Ações/Passos:**
    - Ação(x, Precondição y, Efeito z)
  - **Restrições de Ordenação:**
    - S1 < Sk < Sn, o que não significa que entre S1 e Sk não exista outro passo
  - **Vínculos Causais:** $\{Si \xrightarrow{c} Sj\}$
    - Efeitos da ação Si = pré-condições da ação Sj. Não existe nenhum passo entre eles que negue a pré-condição c.
  - **Pré-condições Abertas:**
    - Não é alcançada por nenhuma ação do plano.

Exemplo dos Sapatos

Objetivo:

SapatoDireitoCalçado ^ SapatoEsquerdoCalçado

Operadores:

Ação(SapatoDireito, PRECOND: MeiaDireitaCalçada, EFFECT: SapatoDireitoCalçado)

Ação(MeiaDireita, EFFECT: MeiaDireitaCalçada)

Ação(SapatoEsquerdo, PRECOND: MeiaEsquerdaCalçada, EFFECT: SapatoEsquerdoCalçado)

Ação(MeiaEsquerda, EFFECT: MeiaEsquerdaCalçada)

Plano Inicial:

Plano(Passos:{S1: Operador(Ação: Inicio),
S2: Operador(Ação: Fim, Precondição: SapatoDireitoCalçado ^ SapatoEsquerdoCalçado)},
ORDENAÇÃO: { S1 < S2 },
VINCULOS: {},
PRECONDIÇÕES_ABERTAS: {}
)

Planejador de Ordem Parcial (POP)

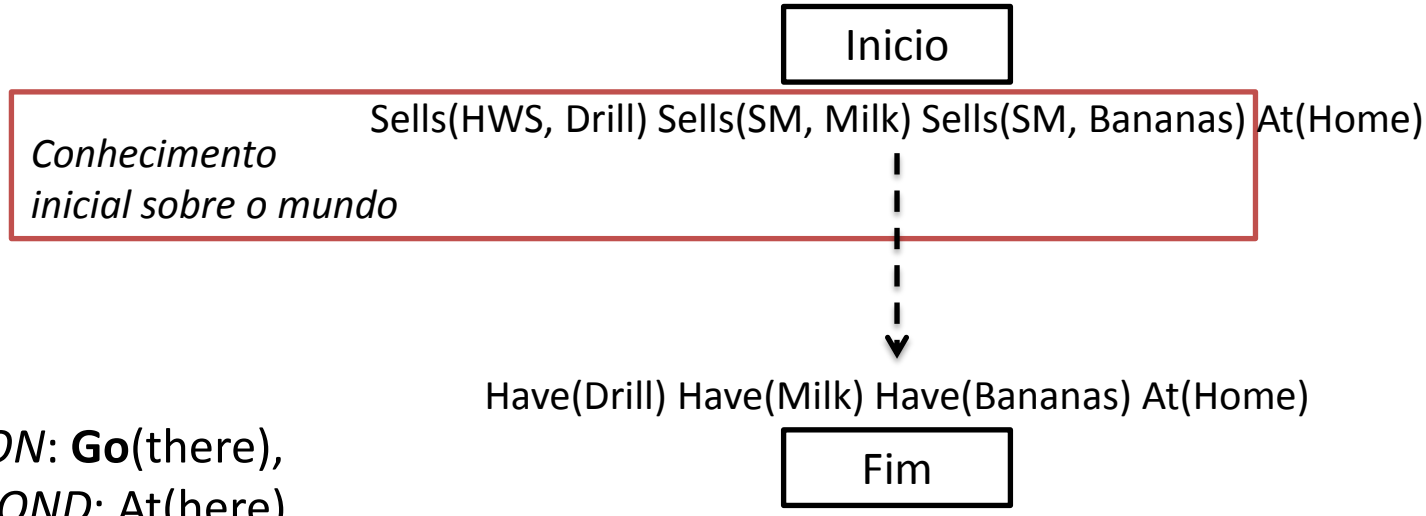
- Características do planejador de ordem parcial:
  - Algoritmo não determinístico;
  - A inserção de um passo só é considerada se ele atender uma precondição não atingida;
  - O planejador é regressivo (do objetivo para o início).
  - É correto e completo, assumindo busca em largura ou em profundidade iterativa.

Planejador de Ordem Parcial (POP)

- Ideia do algoritmo:
  - Identificar um passo com a pré-condição (sub-objetivo) não satisfeita.
  - Introduzir um passo cujo efeito satisfaz a pré-condição.
  - Instanciar as variáveis e atualizar as ligações causais, a partir dos valores instanciados.
  - Verificar se há conflitos e, se necessário, corrigir o plano.

Exemplo

- Plano Inicial:

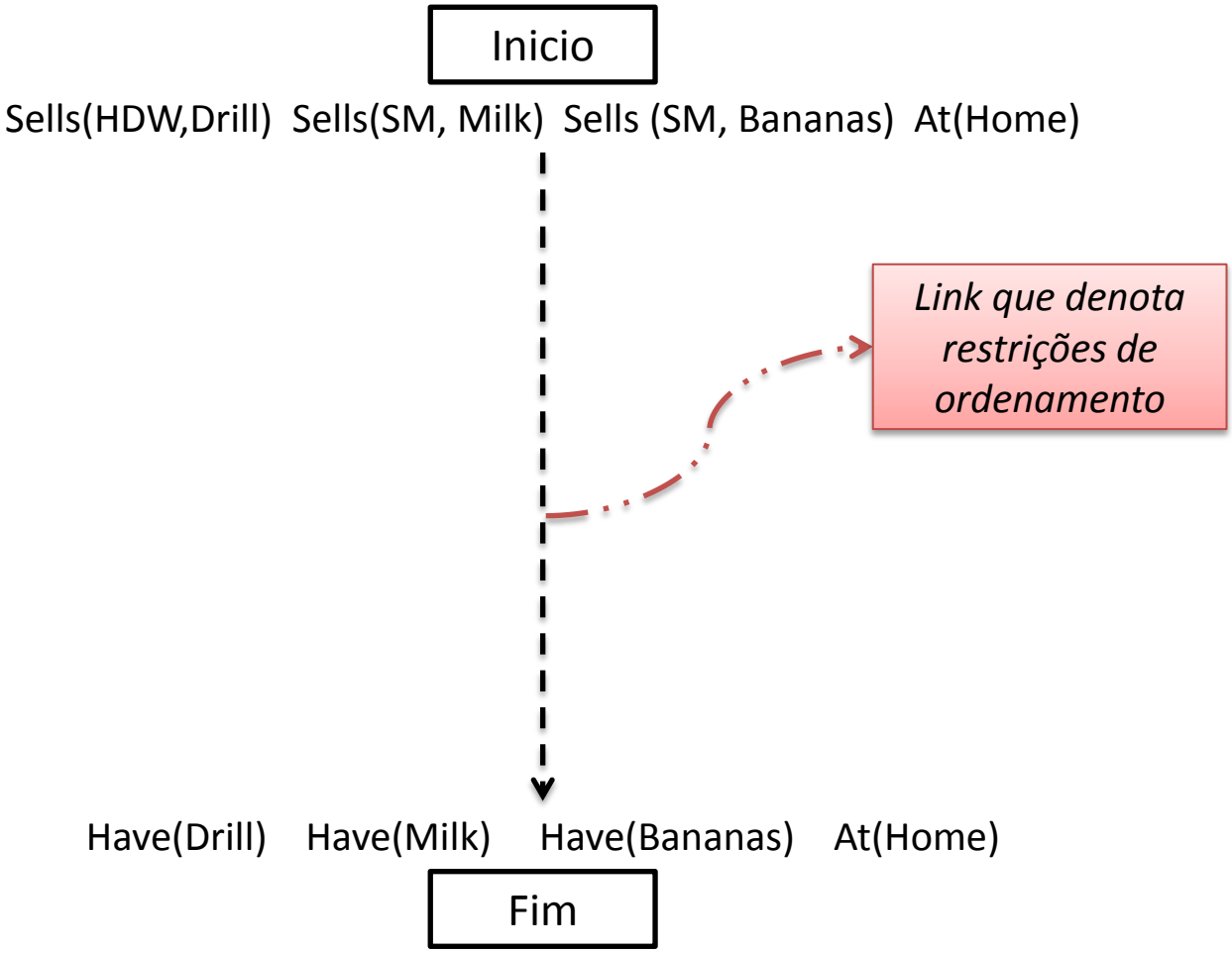


- Ações:

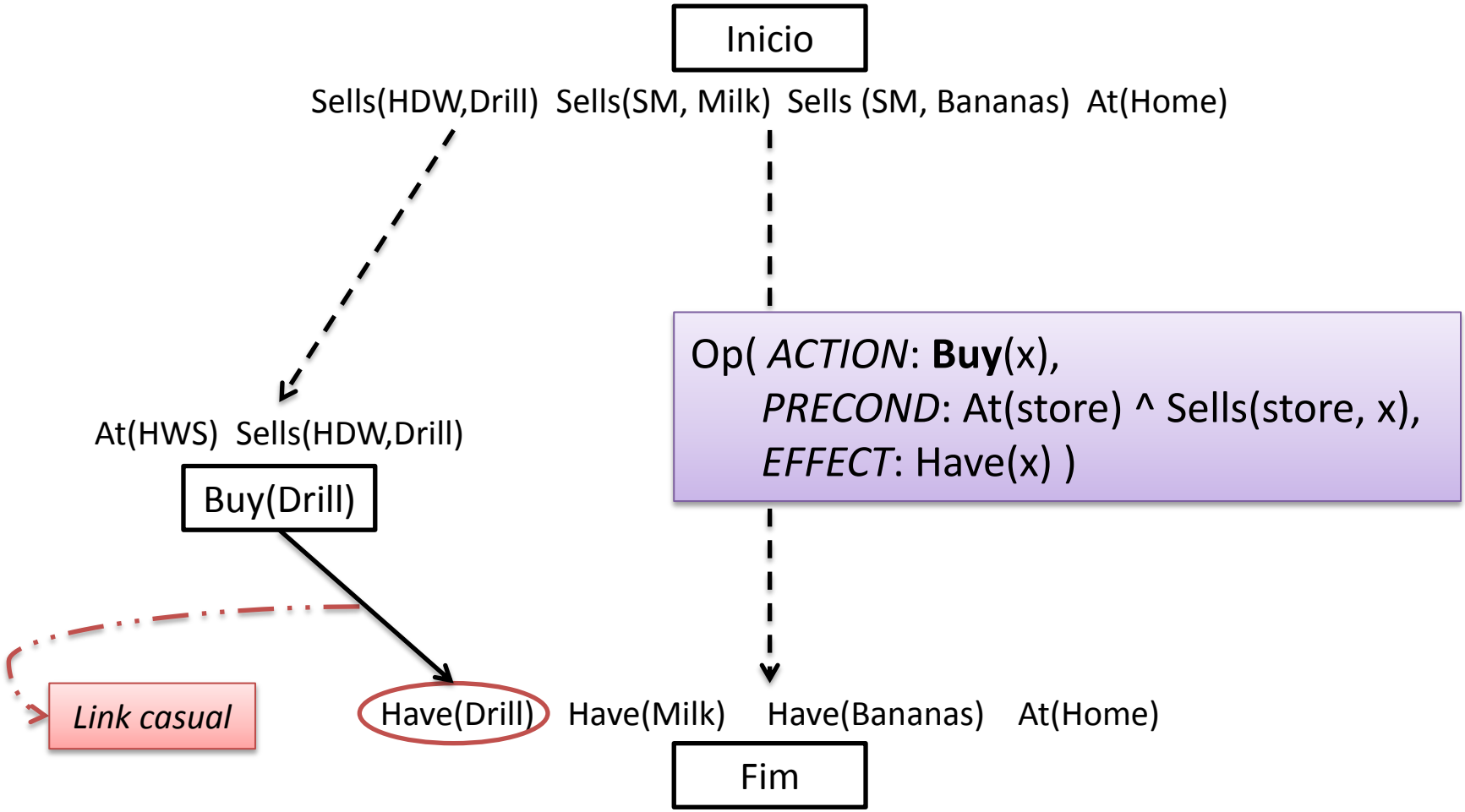
  Op(*ACTION*: **Go**(there),
  PRECOND: At(here),
  EFFECT: At(there) ^ ¬ At(here))

  Op(*ACTION*: **Buy**(x),
  PRECOND: At(store) ^ Sells(store, x),
  EFFECT: Have(x))

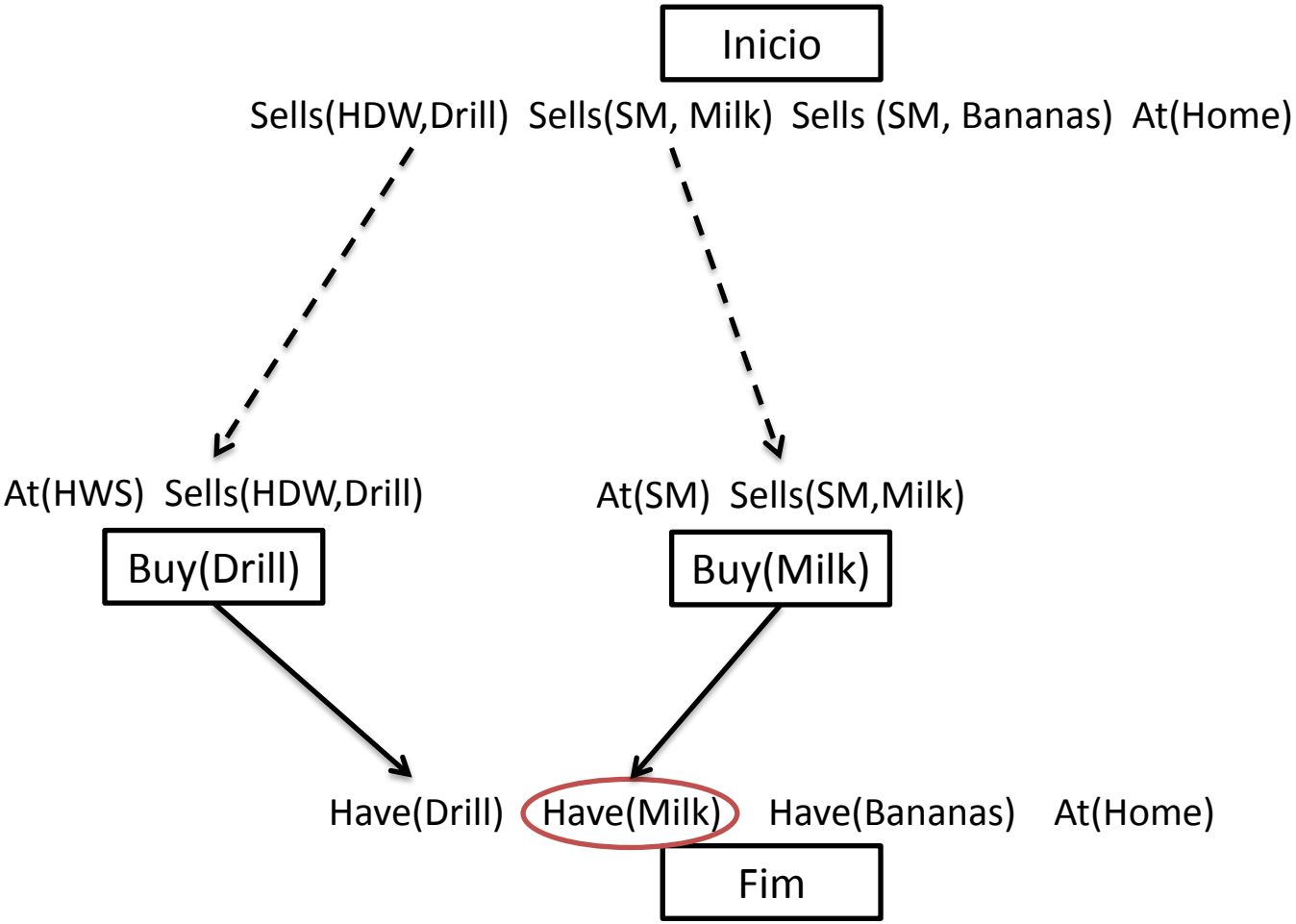
Exemplo


Inicio
Sells(HDW,Drill) Sells(SM, Milk) Sells (SM, Bananas) At(Home)
Link que denota restrições de ordenamento
Have(Drill) Have(Milk) Have(Bananas) At(Home)
Fim

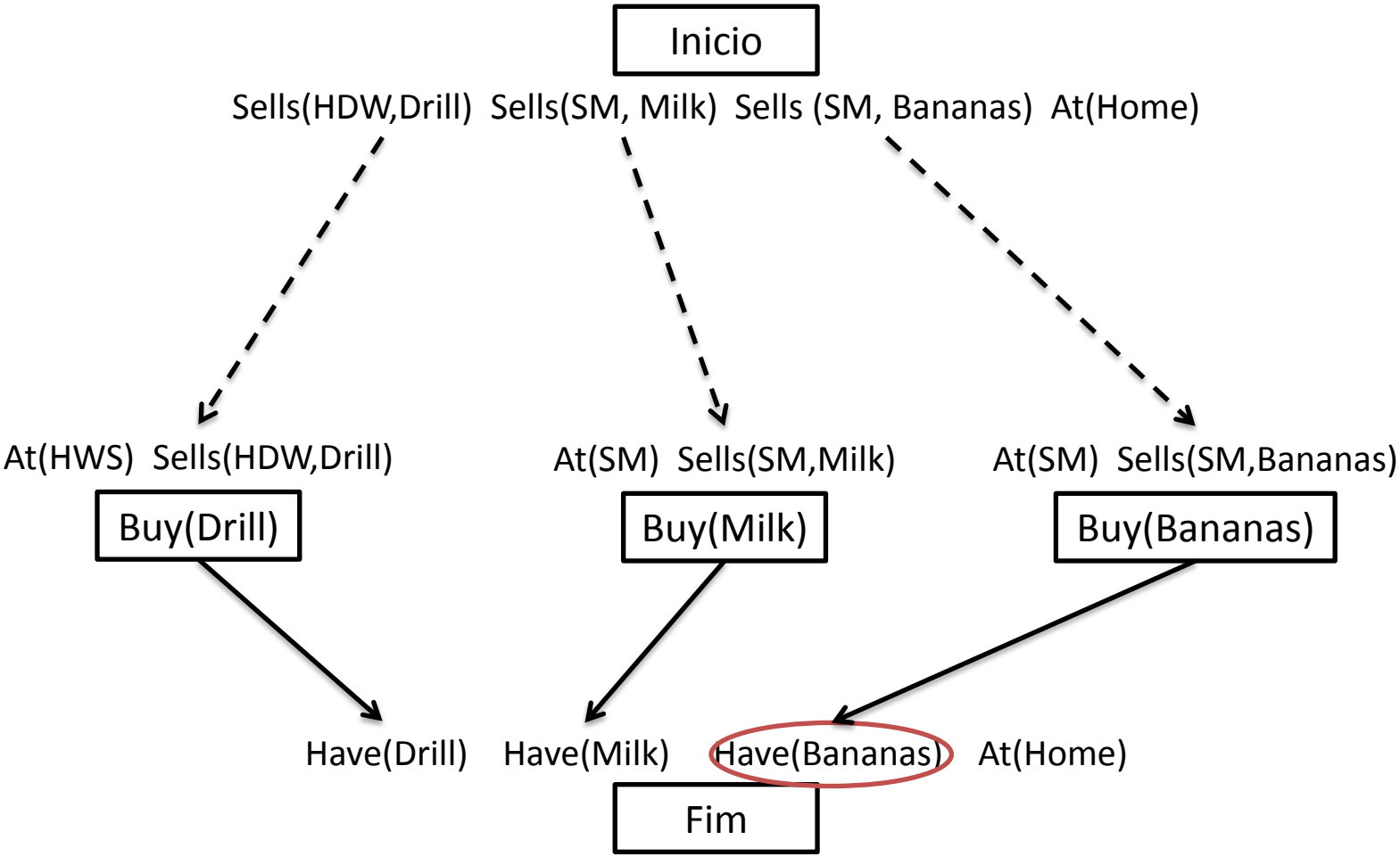
Exemplo


Inicio
Sells(HDW,Drill) Sells(SM, Milk) Sells (SM, Bananas) At(Home)
Op(ACTION: Buy(x),
PRECOND: At(store) ^ Sells(store, x),
EFFECT: Have(x))
At(HWS) Sells(HDW,Drill)
Buy(Drill)
Link casual
Have(Drill)
Have(Milk)
Have(Bananas)
At(Home)
Fim

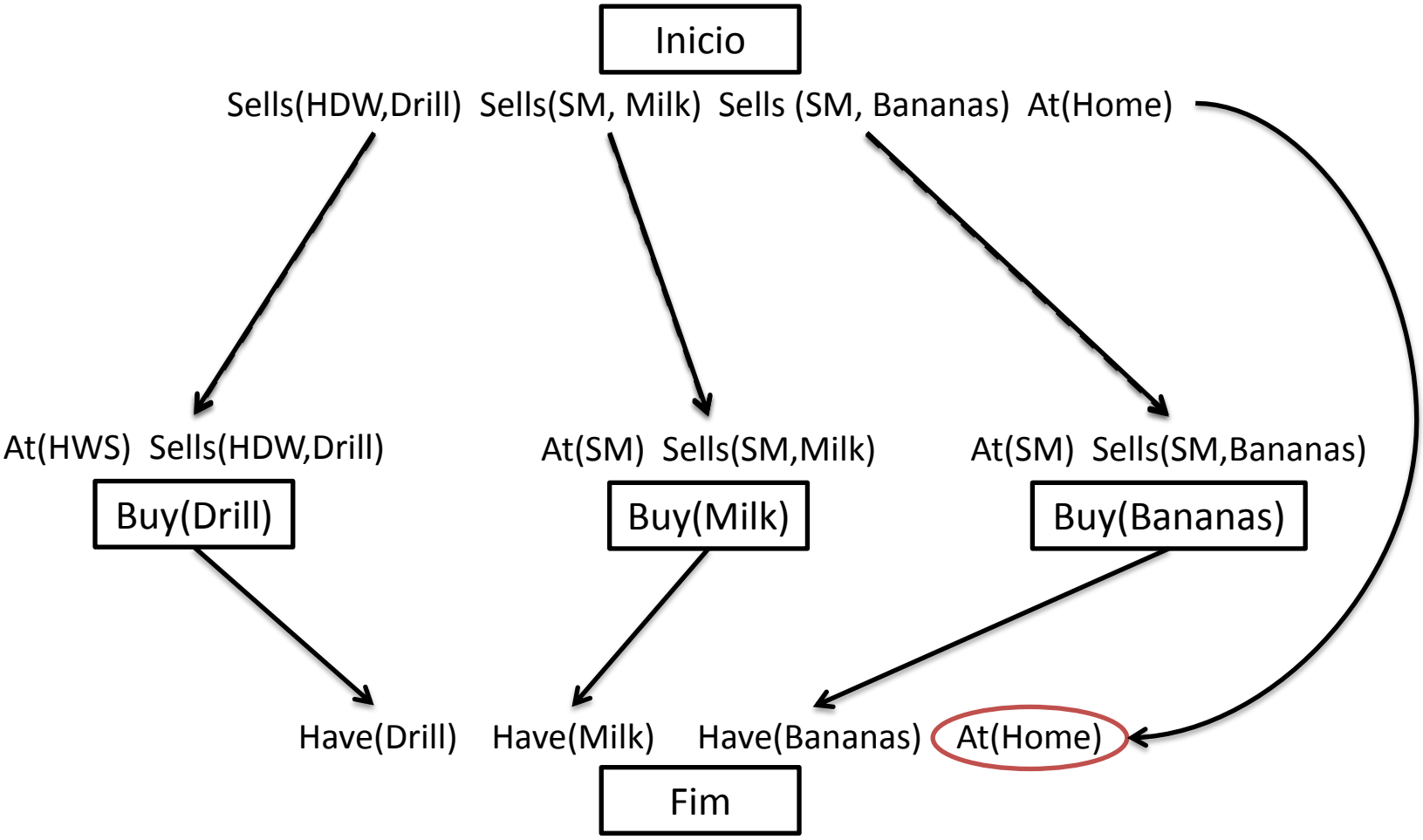
Exemplo


Inicio
Sells(HDW,Drill) Sells(SM, Milk) Sells (SM, Bananas) At(Home)
At(HWS) Sells(HDW,Drill)
At(SM) Sells(SM,Milk)
Buy(Drill)
Buy(Milk)
Have(Drill) Have(Milk) Have(Bananas) At(Home)
Fim

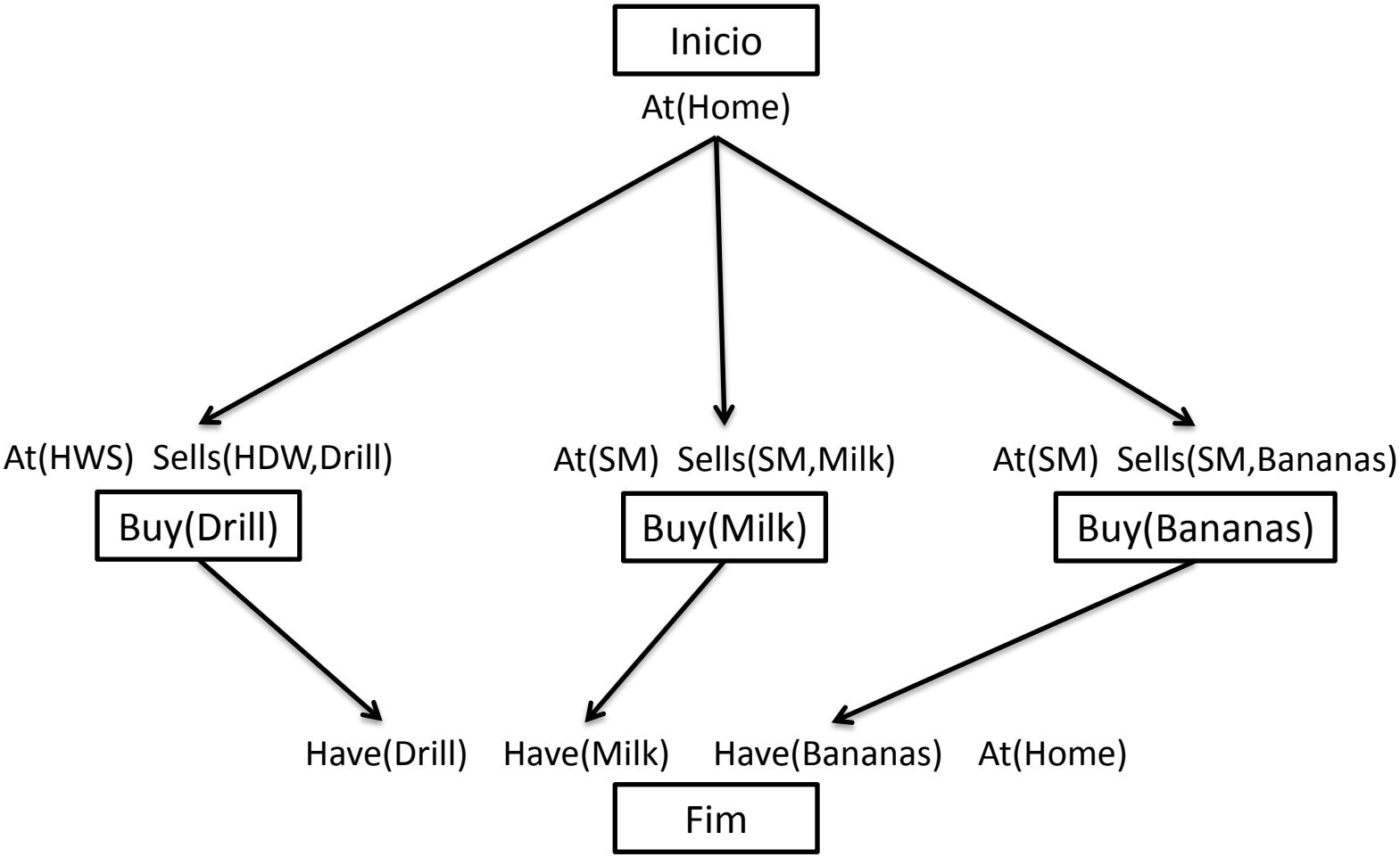
Exemplo


Inicio
Sells(HDW,Drill)
Sells(SM, Milk)
Sells (SM, Bananas)
At(Home)
At(HWS)
Sells(HDW,Drill)
At(SM)
Sells(SM,Milk)
At(SM)
Sells(SM,Bananas)
Buy(Drill)
Buy(Milk)
Buy(Bananas)
Have(Drill)
Have(Milk)
Have(Bananas)
At(Home)
Fim

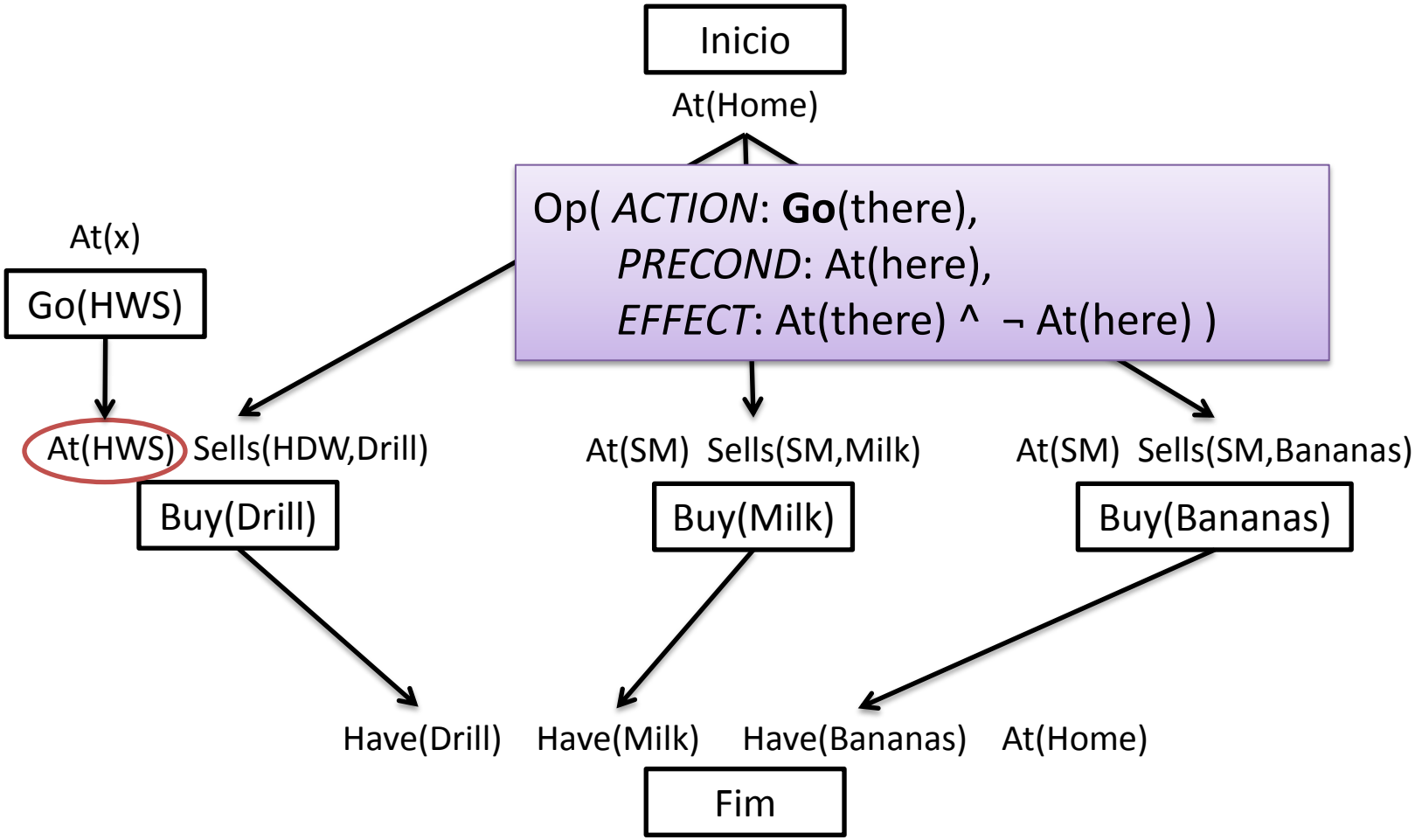
Exemplo


Inicio
Sells(HDW,Drill)
Sells(SM, Milk)
Sells (SM, Bananas)
At(Home)
At(HWS) Sells(HDW,Drill)
At(SM) Sells(SM,Milk)
At(SM) Sells(SM,Bananas)
Buy(Drill)
Buy(Milk)
Buy(Bananas)
Have(Drill)
Have(Milk)
Have(Bananas)
At(Home)
Fim

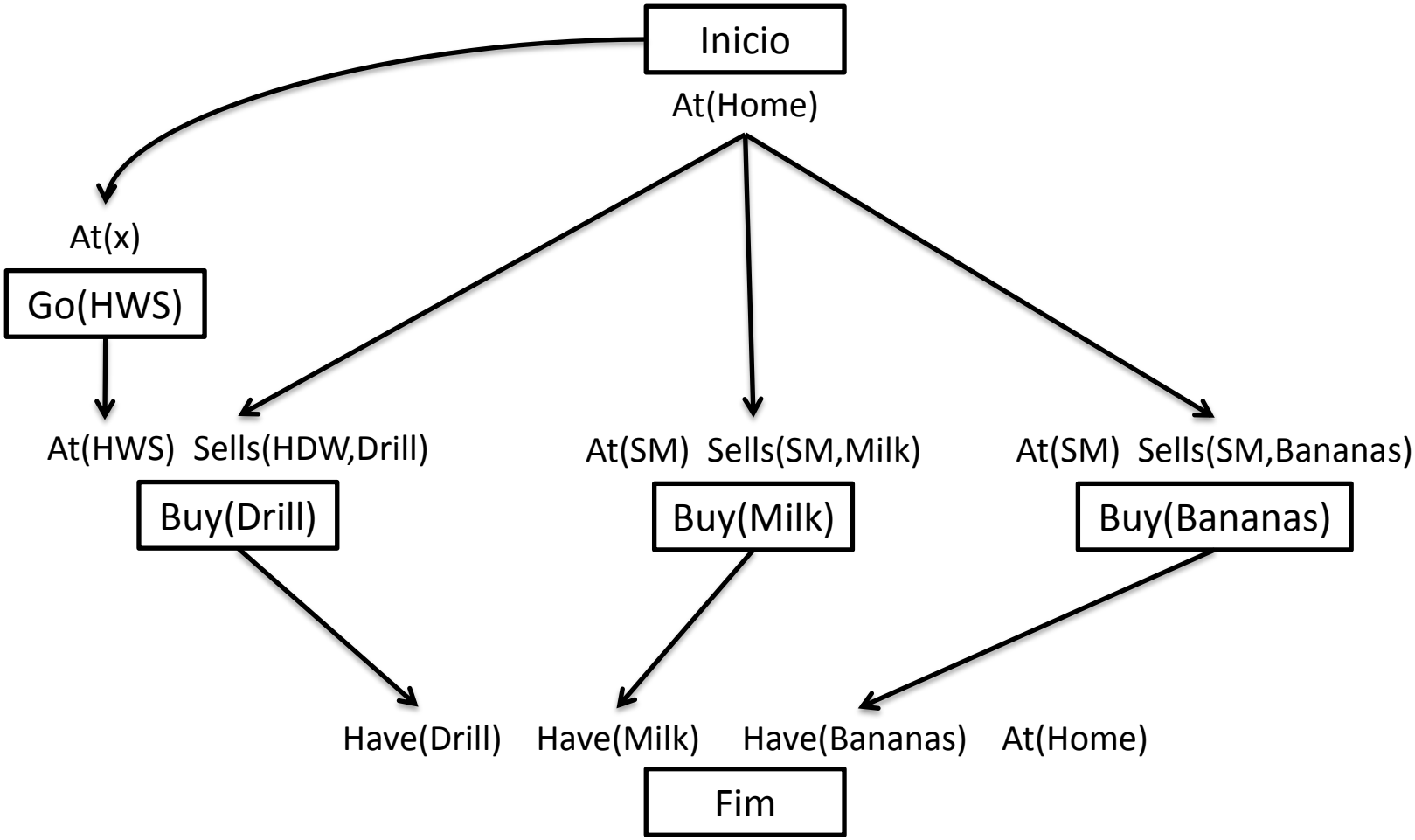
Exemplo


Inicio
At(Home)
At(HWS) Sells(HDW,Drill)
Buy(Drill)
At(SM) Sells(SM,Milk)
Buy(Milk)
At(SM) Sells(SM,Bananas)
Buy(Bananas)
Have(Drill)
Have(Milk)
Have(Bananas)
At(Home)
Fim

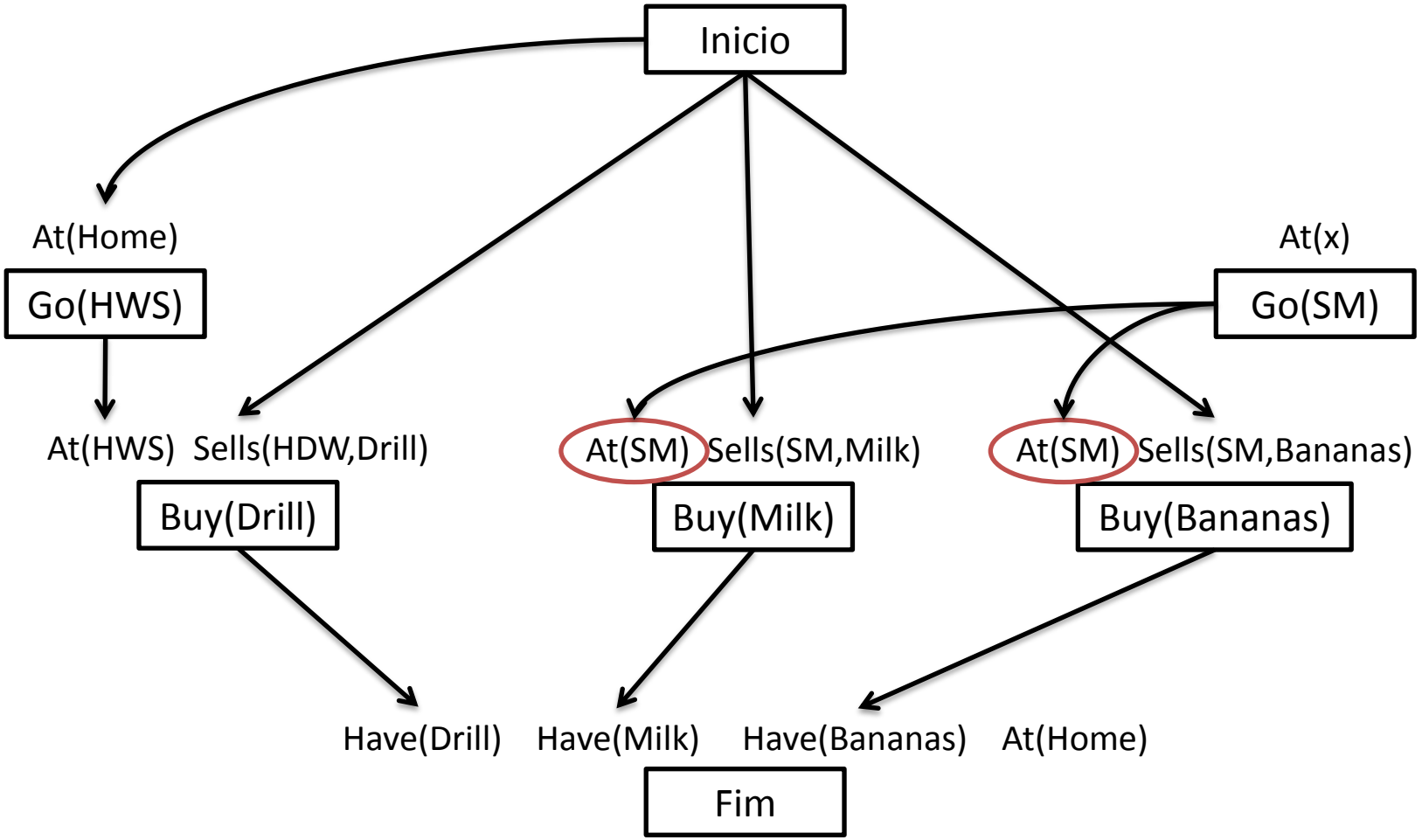
Exemplo


Inicio
At(Home)
Op(ACTION: Go(there),
PRECOND: At(here),
EFFECT: At(there) ^ ¬ At(here))
At(x)
Go(HWS)
At(HWS)
Sells(HDW,Drill)
Buy(Drill)
At(SM)
Sells(SM,Milk)
Buy(Milk)
At(SM)
Sells(SM,Bananas)
Buy(Bananas)
Have(Drill)
Have(Milk)
Have(Bananas)
At(Home)
Fim

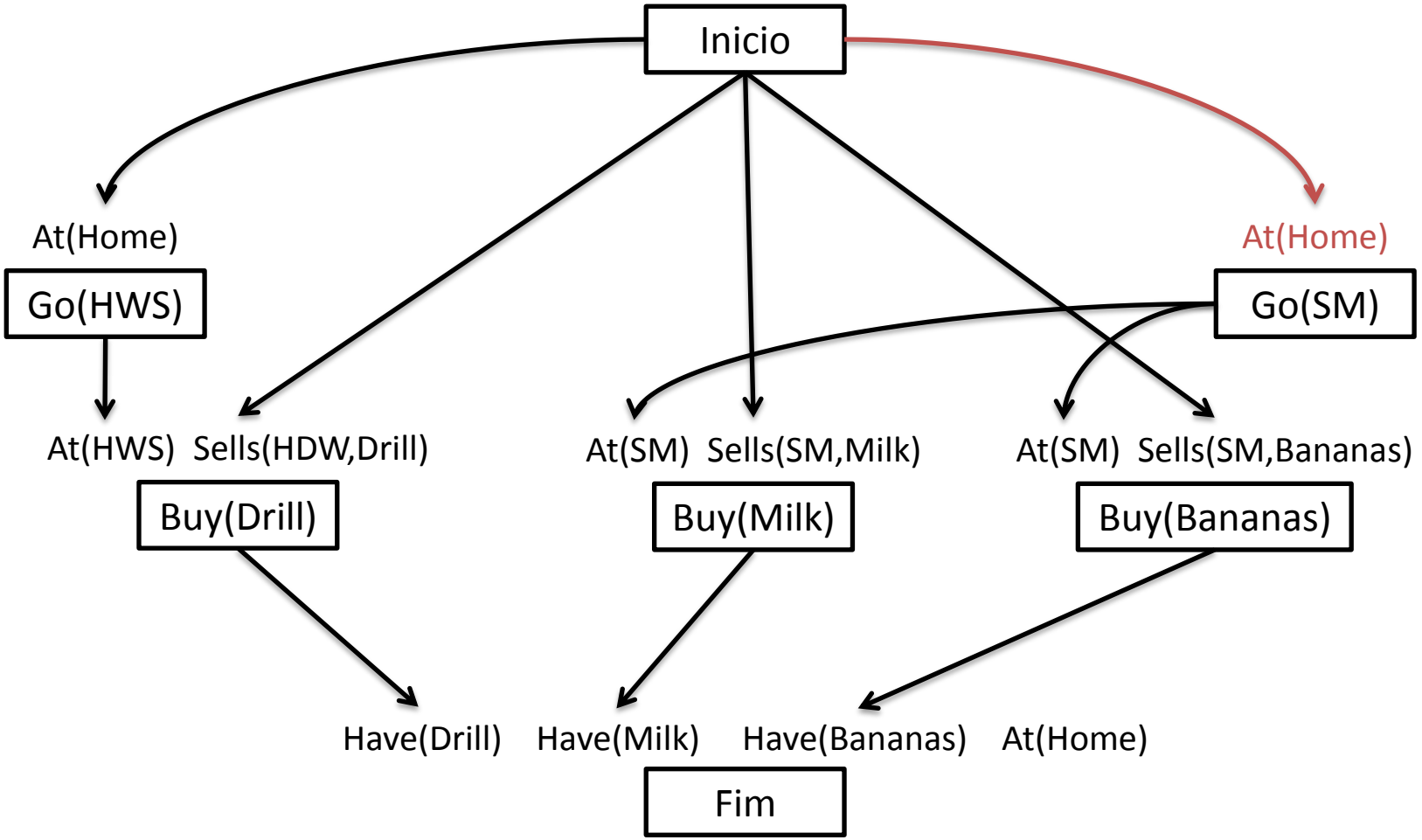
Exemplo


Inicio
At(Home)
At(x)
Go(HWS)
At(HWS) Sells(HDW,Drill)
Buy(Drill)
At(SM) Sells(SM,Milk)
Buy(Milk)
At(SM) Sells(SM,Bananas)
Buy(Bananas)
Have(Drill)
Have(Milk)
Have(Bananas)
At(Home)
Fim

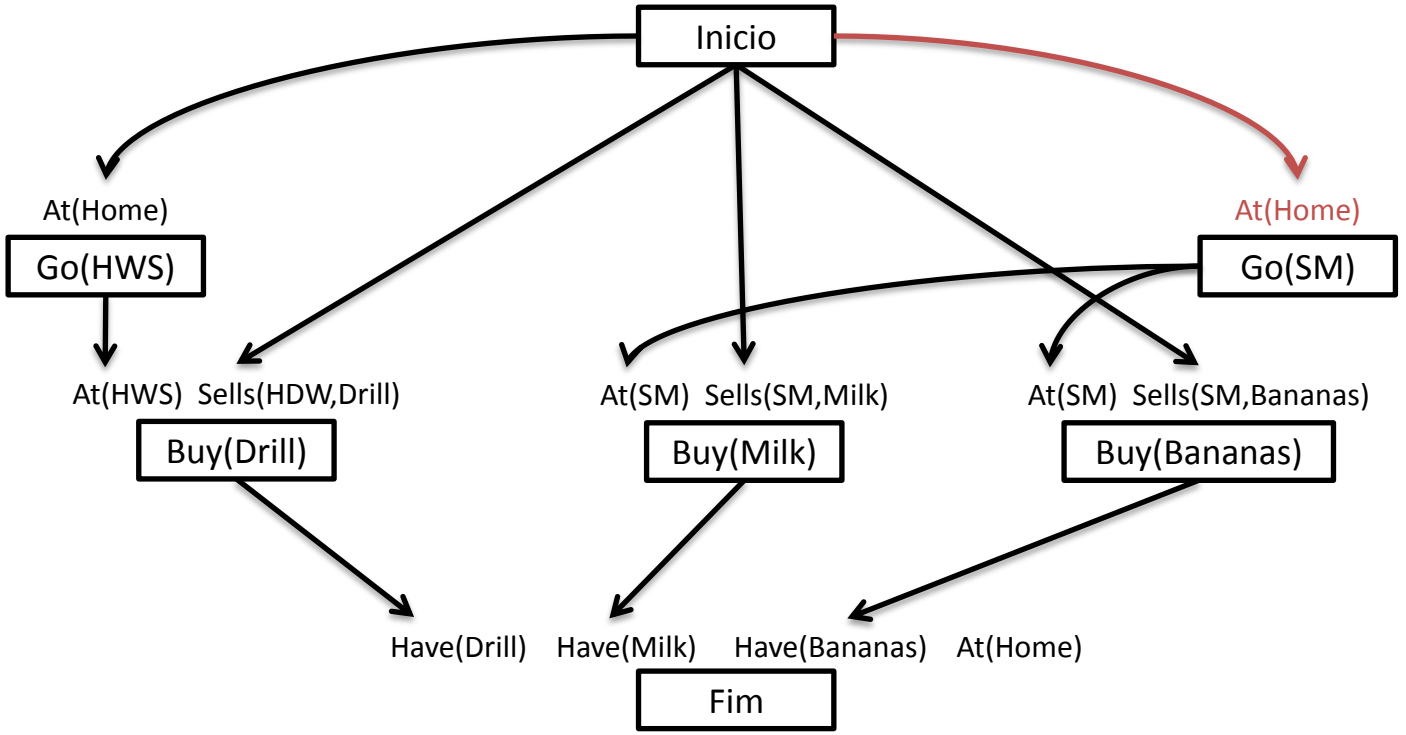
Exemplo


Inicio
At(Home)
Go(HWS)
At(x)
Go(SM)
At(HWS)
Sells(HDW,Drill)
At(SM)
Sells(SM,Milk)
At(SM)
Sells(SM,Bananas)
Buy(Drill)
Buy(Milk)
Buy(Bananas)
Have(Drill)
Have(Milk)
Have(Bananas)
At(Home)
Fim

Exemplo


Inicio
At(Home)
Go(HWS)
At(Home)
Go(SM)
At(HWS)
Sells(HDW,Drill)
At(SM)
Sells(SM,Milk)
At(SM)
Sells(SM,Bananas)
Buy(Drill)
Buy(Milk)
Buy(Bananas)
Have(Drill)
Have(Milk)
Have(Bananas)
At(Home)
Fim

Exemplo



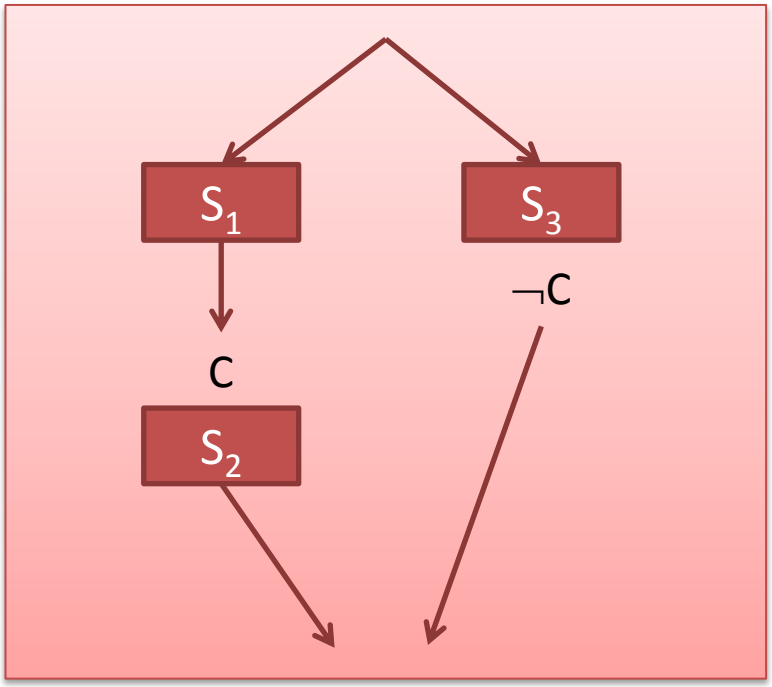
PROBLEMA: Considere que a pré-condição At(x) da ação Go(SM) foi satisfeita através de uma ligação à condição At(Home) do Start; se o agente decidir ir primeiro à HWS, ele não mais poderá sair de casa para ir ao SM, pois Go(HWS) adiciona At(HWS), mas também remove At(Home)!!! (e vice-versa: indo de casa ao SM, não mais consegue ir de casa à HWS) → CONFLITO

Conflitos no POP

- Um conflito ocorre quando os efeitos de uma ação põem em risco as pré-condições de outra ação.
  - No caso anterior, os operadores **Go**(HWS) e **Go**(SM) apagam **At**(Home).

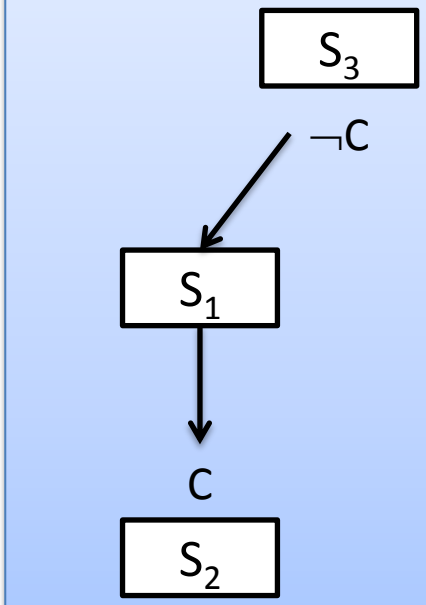
- Com testar?
  - O novo passo é inconsistente com condição protegida (link causal)
  - O passo antigo é inconsistente com nova condição protegida

Solução de Conflitos

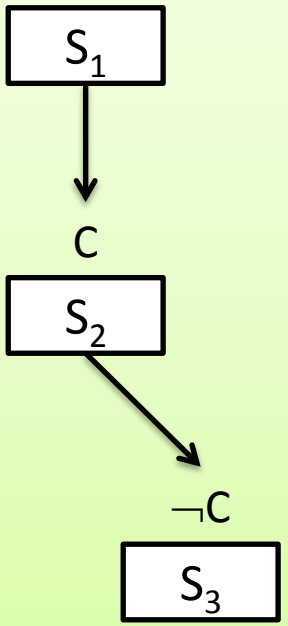


SITUAÇÃO DE CONFLITO:
S3 ameaça a condição c estabelecida por de S1 e protegida pelo link causal S1 para S2

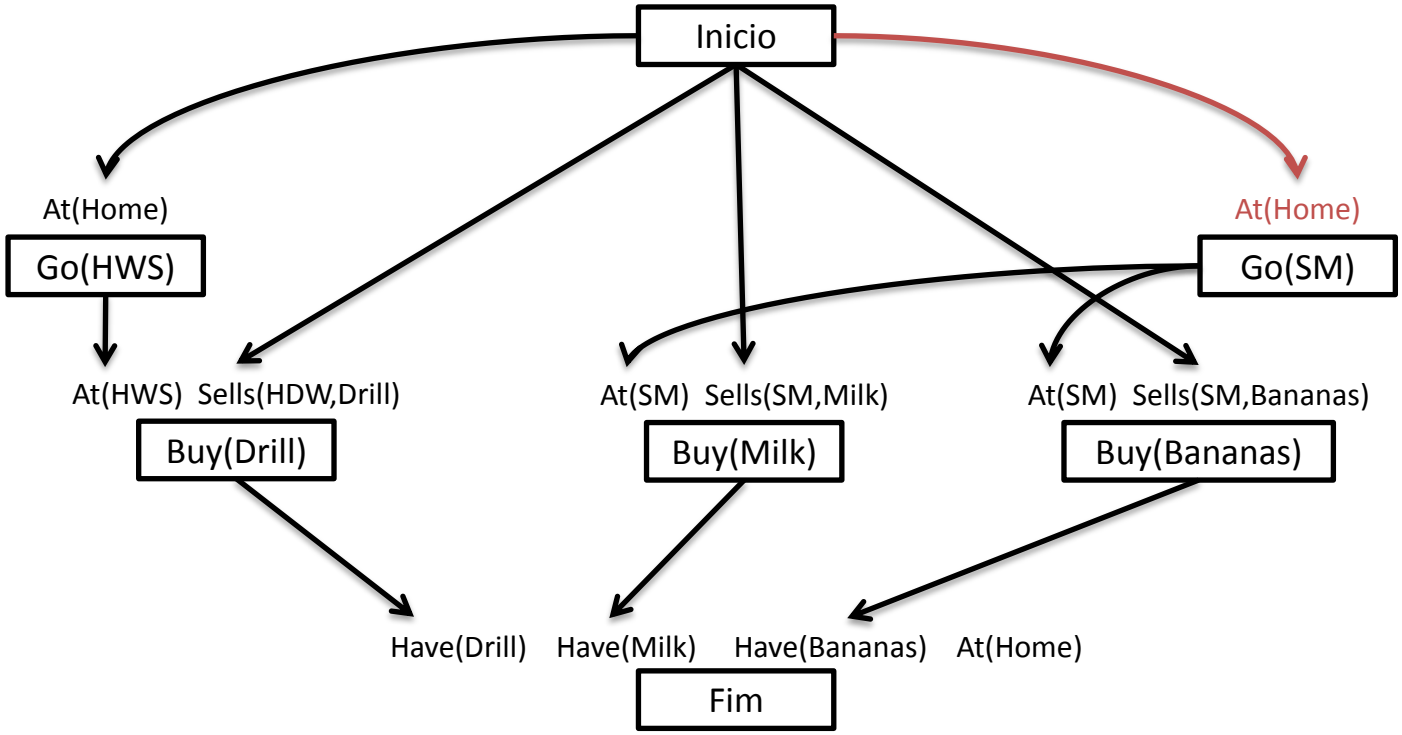
Demotion



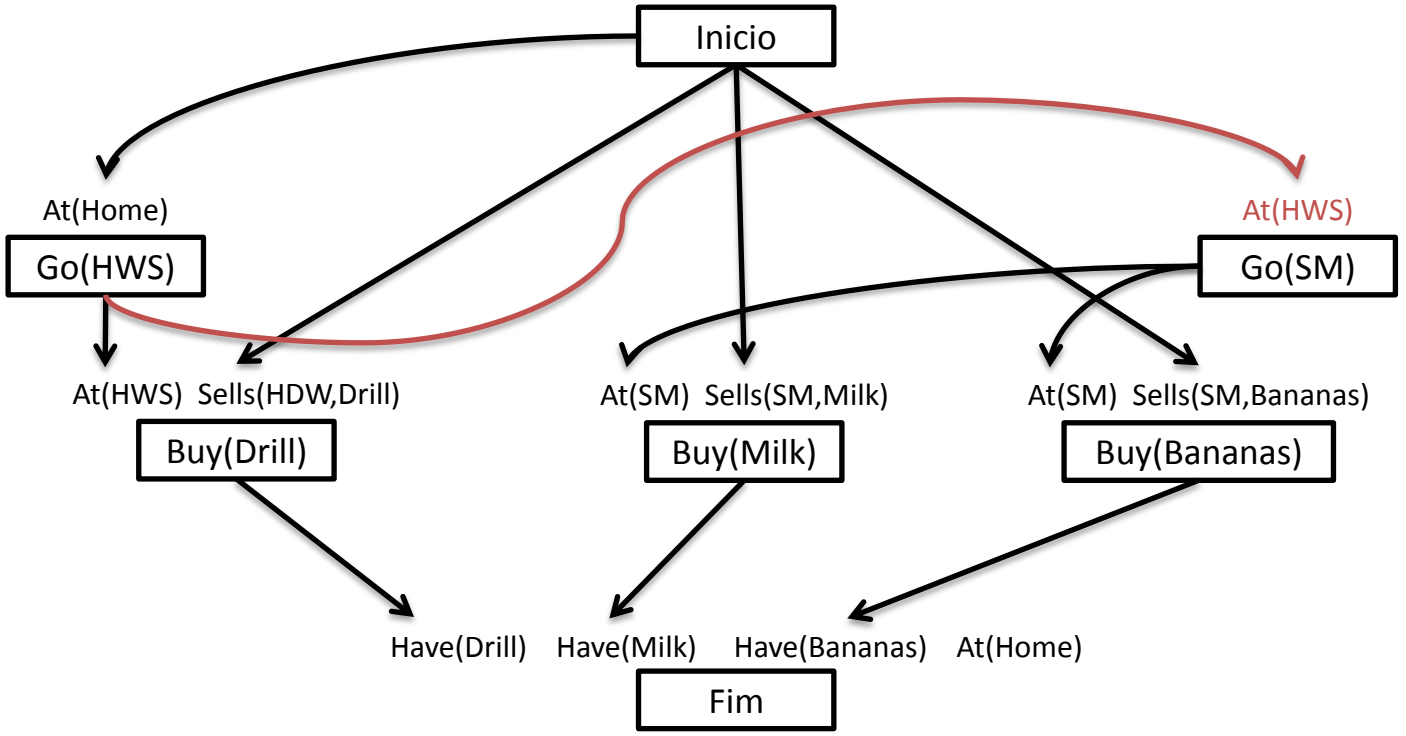
Promotion



Exemplo


Inicio
At(Home)
Go(HWS)
At(Home)
Go(SM)
At(HWS) Sells(HDW,Drill)
Buy(Drill)
At(SM) Sells(SM,Milk)
Buy(Milk)
At(SM) Sells(SM,Bananas)
Buy(Bananas)
Have(Drill)
Have(Milk)
Have(Bananas)
At(Home)
Fim

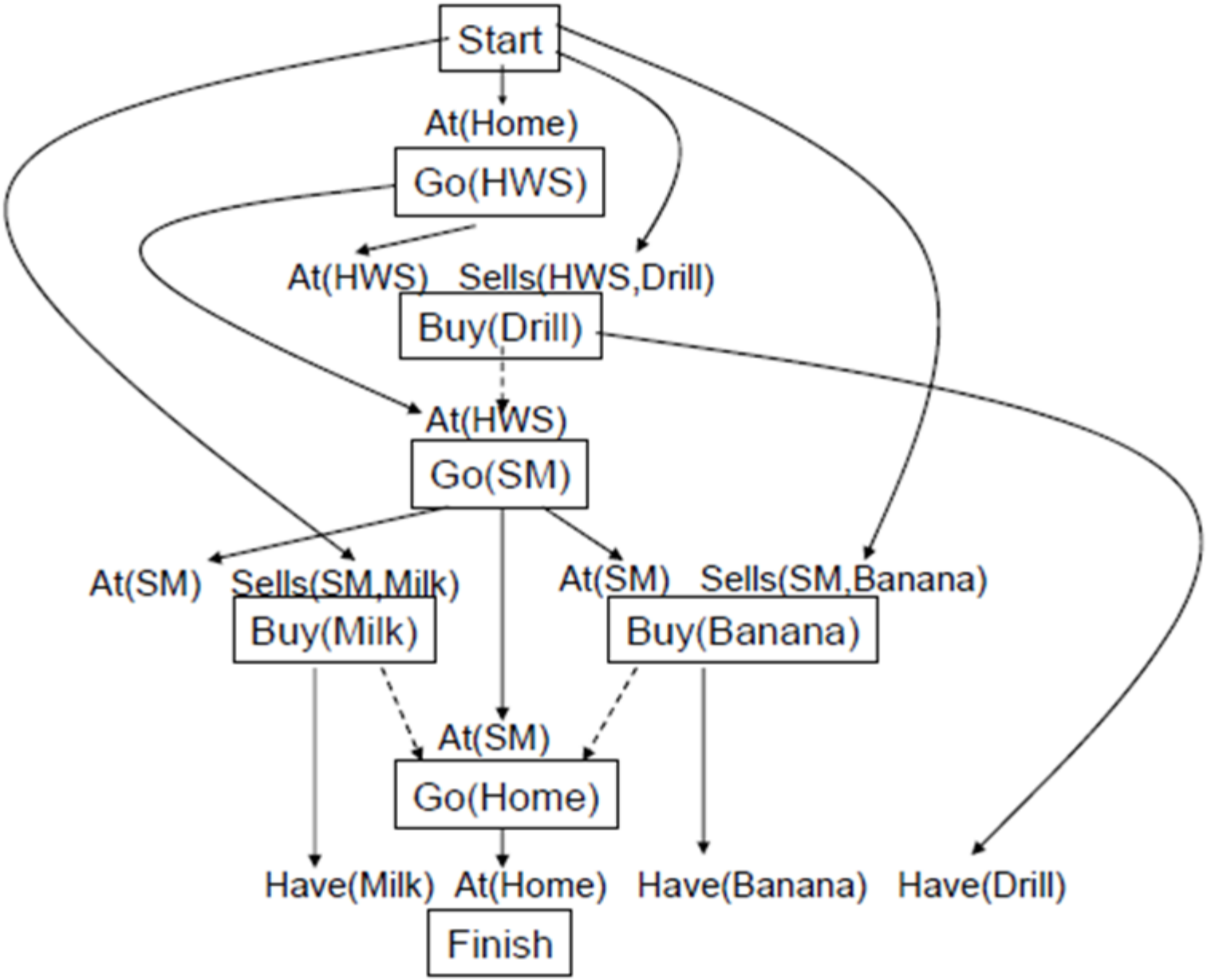
Exemplo



Tentativas de solução:

1. **Demotion:** não é possível aplicar, pois Go(SM) ficaria antes de Start!!!
2. **Promotion:** sairia de Home, iria à HWS e, logo em seguida, ao SM ... porém sem comprar o Drill!!! Isso implica num novo conflito, pois Have(Drill) não é incluído.

Exemplo : Plano Final



STRIPS : Limitações

- A simplicidade desse modelo formal restringe em muito o número de problemas que podem ser modelados como problemas de planejamento.
- A linguagem STRIPS não suporta predicados de igualdade e não é uma linguagem tipada.

ADL

- ADL (*Action Descripition Language*)
- Surgiu na tentativa de suprir as limitações da linguagem STRIPS.
- Através da linguagem ADL é possível representar uma gama maior de problemas.

STRIPS vs ADL

Linguagem STRIPS

- Usa apenas literais positivos nos estados
- Hipótese de Mundo Fechado: literais não mencionados são falsos
- Efeito P ^ ¬Q significa adicionar P e remover Q
- Objetivos são conjunções
- Efeitos são conjunções
- Não oferece suporte para igualdade
- Não oferece suporte para tipos

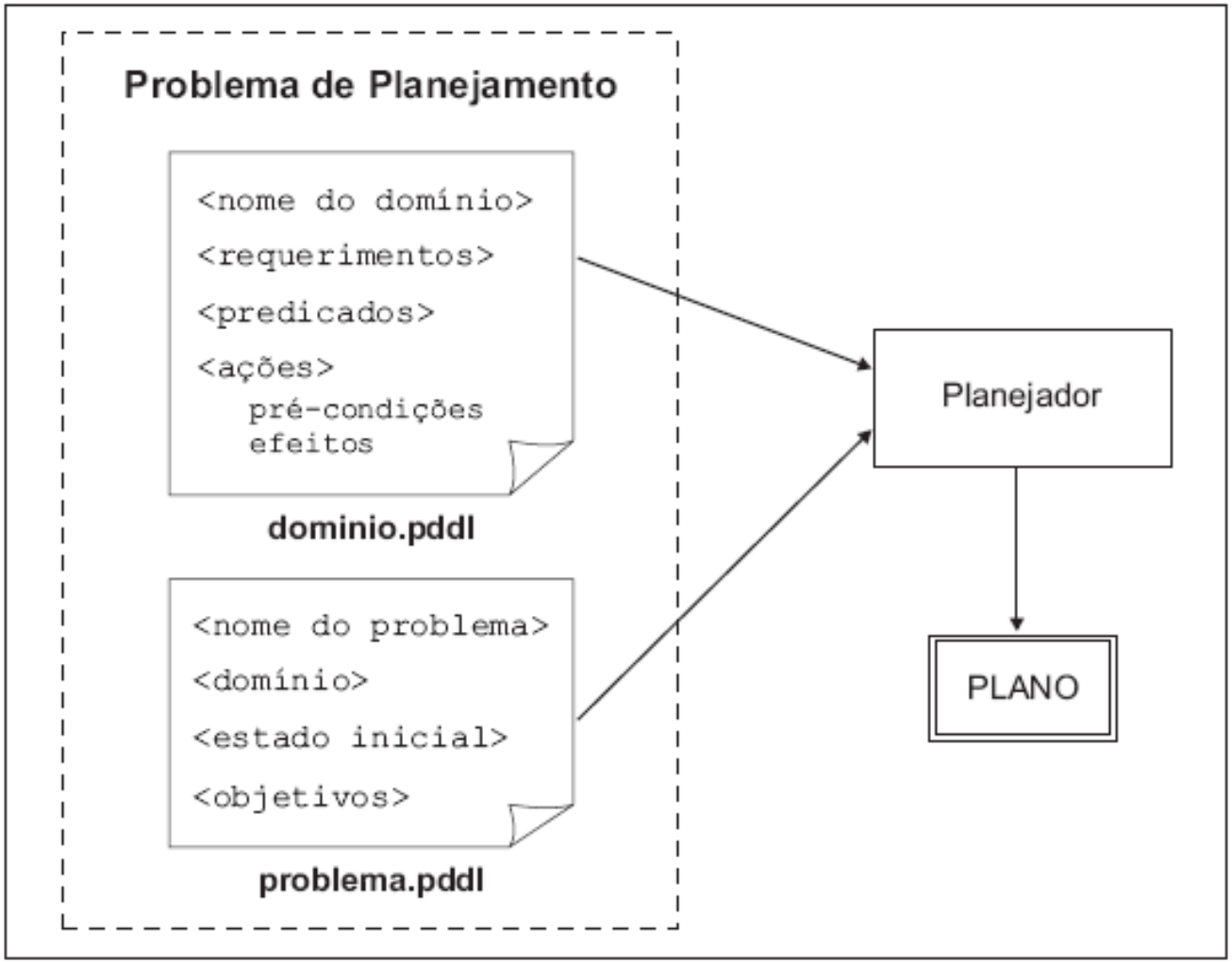
Linguagem ADL

- Usa tanto literais positivos quanto negativos nos estados
- Hipótese de Mundo Aberto: literais não mencionados são desconhecidos
- Efeito P ^ ¬Q significa adicionar P e ¬Q e remover ¬ P e Q
- Objetivos permitem conjunções e disjunções
- Permite o uso Efeitos condicionais (quando P = E é efeito se P válido)
- Dispõe de um predicado de igualdade
- Variáveis podem ter tipos

PDDL

- PDDL (*Planning Domain Definition Language*).
- Combinação de linguagens STRIPS e ADL.
- Especificação padrão para representar os problemas de planejamento.
- Para a representação de um problema de planejamento em PDDL, são necessários dois arquivos, o arquivo de domínio e o arquivo de problema.

PDDL


Problema de Planejamento
<nome do domínio>
<requerimentos>
<predicados>
<ações>
pré-condições
efeitos
dominio.pddl
<nome do problema>
<domínio>
<estado inicial>
<objetivos>
problema.pddl
Planejador
PLANO

Aplicações de Planejamento

- Qualquer problema que necessite de passos/ações para chegar a um determinado objetivo.
- Exemplos:
  - Robôs que realizam tarefas.
  - Personagens de jogos direcionados a objetivos.
  - Geração de histórias para storytelling interativo.