Pontifícia Universidade Católica
do Rio de Janeiro

INF 1771 – Inteligência Artificial

Aula 11 – Planejamento

Edirlei Soares de Lima
<elima@inf.puc-rio.br>

visionlab
visualization laboratory

Agentes Vistos Anteriormente

- **Agentes Baseados em Busca.**
  - Busca cega;
  - Busca heurística;
  - Busca local;

- **Agentes Lógicos.**
  - Lógica proposicional;
  - Lógica de primeira ordem;
  - Prolog;

Planejamento

- **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 alcançar.
- **Constrói um plano** para atingir o objetivo a partir do estado atual do ambiente.
- **Executa o plano** do começo ao fim.
- **Gera um novo objetivo** com base no novo estado do ambiente.

Planejamento

- Em **planejamento clássico** o ambiente do problema possui as seguintes características:
  - Observável
  - Determinístico
  - Finito
  - Estático

Resolução de Problemas X Planejamento

- **Algoritmos de busca** tendem a tomar ações irrelevantes.
  - Grande fator de ramificação.
  - Pouco conhecimento para guiar a busca.

- **Planejador** não considera ações irrelevantes.
  - Faz conexões diretas entre estados (sentenças) e ações (pré-condições + efeitos)
  - Objetivo: Ter(Leite).
    - Ação: Comprar(Leite) => Ter(Leite)

Resolução de Problemas X Planejamento

- Em 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**: número de elementos da conjunção não-satisfeitos.

Resolução de Problemas X Planejamento

- **Algoritmos de busca** não tiram proveito da decomposição do problema.

- **Planejadores** aproveitam a estrutura do problema. É possível decompor com facilidade sub-objetivos.

  – Exemplo: Ter(A) Λ Ter(B) Λ Ter(C) Λ Ter(D)

Linguagem STRIPS

- **Linguagem formal** para a especificação de problemas de planejamento.

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

  - **Hipótese do mundo fechado:** qualquer condição não mencionada em um estado é considerada negativa.

    - Exemplo: ¬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(Destino),
  Pré-condição Em(Partida) ^ Caminho(Partida, Destino),
  Efeito 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))

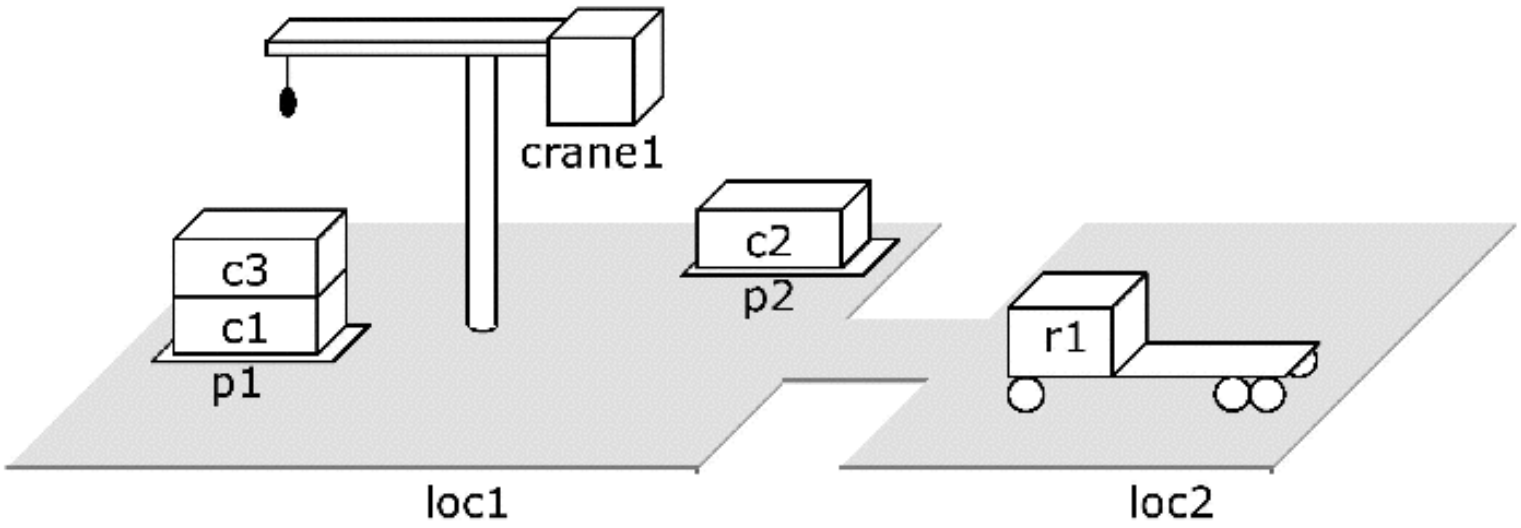
Objetivo(Em(C1,JFK) ^ Em(C2,SFO))

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))

Exemplo – Doca Automatizada



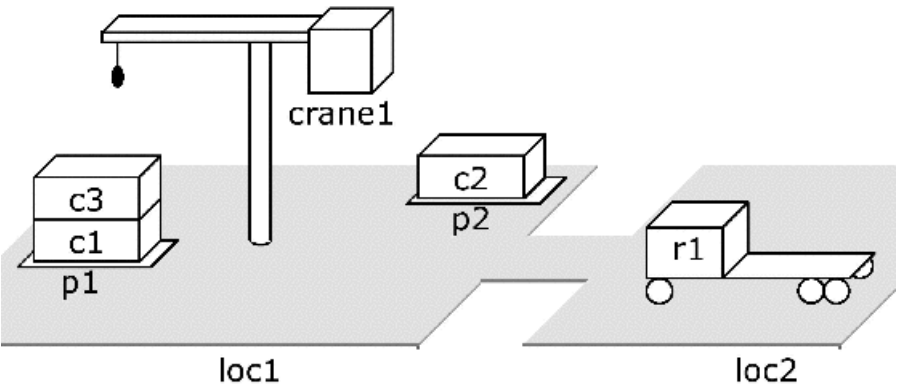
Exemplo de Estado:

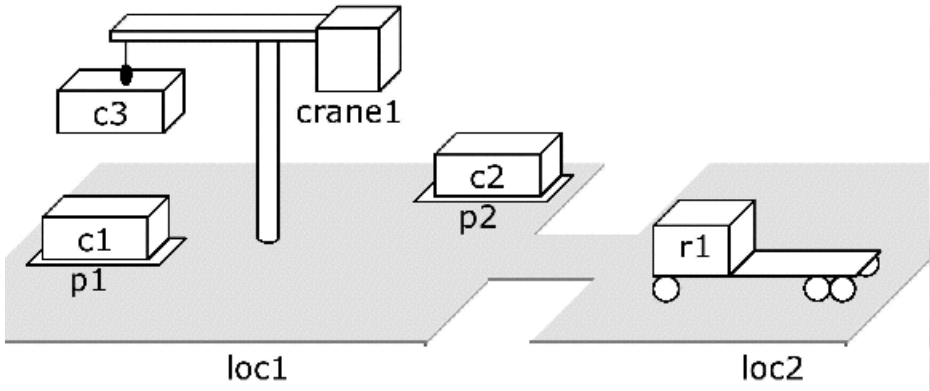
s_1={attached(p1,loc1), in(c1,p1), in(c3,p1), top(c3,p1), on(c3,c1), on(c1,pallet), attached(p2,loc1), in(c2,p2), top(c2,p2), on(c2,pallet), belong(crane1,loc1), empty(crane1), adjacent(loc1,loc2), adjacent(loc2,loc1), at(r1,loc2), occupied(loc2), unloaded(r1)}.

visionlab
visualization laboratory

Exemplo – Doca Automatizada

take(crane1,loc1,c3,c1,p1)
;; crane crane1 at location loc1 takes c3 off c1 in pile p1
precond: belong(crane1,loc1), attached(p1,loc1),
empty(crane1), top(c3,p1), on(c3,c1)
effects: holding(crane1,c3), ¬empty(crane1), ¬in(c3,p1),
¬top(c3,p1), ¬on(c3,c1), top(c1,p1)





Exemplo – Doca Automatizada

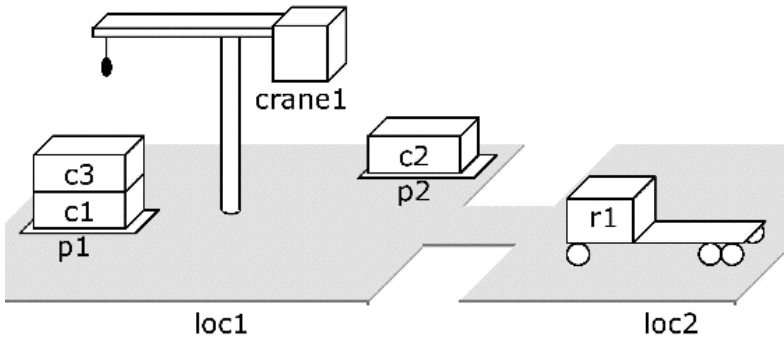
move(r, l, m)
;; robot r moves from location l to location m
precond: adjacent(l, m), at(r, l), $\neg$ occupied(m)
effects: at(r, m), occupied(m), $\neg$ occupied(l), $\neg$ at(r, l)

load(k, l, c, r)
;; crane k at location l loads container c onto robot r
precond: belong(k, l), holding(k, c), at(r, l), unloaded(r)
effects: empty(k), $\neg$ holding(k, c), loaded(r, c), $\neg$ unloaded(r)

unload(k, l, c, r)
;; crane k at location l takes container c from robot r
precond: belong(k, l), at(r, l), loaded(r, c), empty(k)
effects: $\neg$ empty(k), holding(k, c), unloaded(r), $\neg$ loaded(r, c)

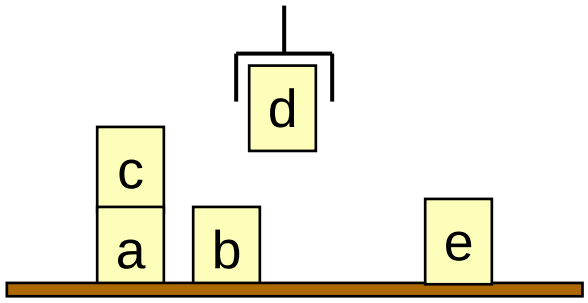
put(k, l, c, d, p)
;; crane k at location l puts c onto d in pile p
precond: belong(k, l), attached(p, l), holding(k, c), top(d, p)
effects: $\neg$ holding(k, c), empty(k), in(c, p), top(c, p), on(c, d), $\neg$ top(d, p)

take(k, l, c, d, p)
;; crane k at location l takes c off of d in pile p
precond: belong(k, l), attached(p, l), empty(k), top(c, p), on(c, d)
effects: holding(k, c), $\neg$ empty(k), $\neg$ in(c, p), $\neg$ top(c, p), $\neg$ on(c, d), top(d, p)

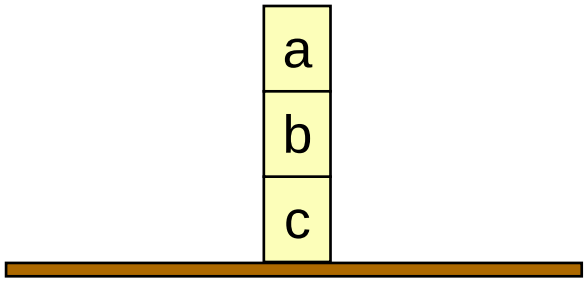


Exemplo - Mundo dos Blocos

- Mesa infinitamente larga, número finito de blocos;
- Ignora a posição em que um bloco está sobre a mesa;
- Um bloco pode estar sobre a mesa ou sobre um outro bloco;
- Os blocos devem ser movidos de uma configuração para outra;



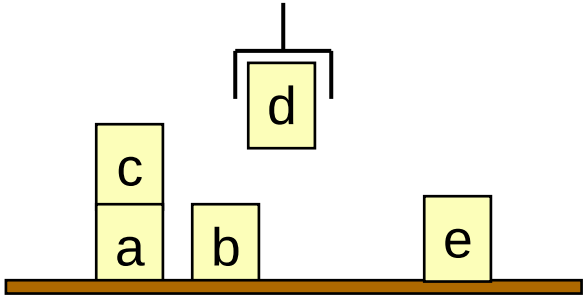
Estado Inicial



Estado Objetivo

Exemplo - Mundo dos Blocos

- **Símbolos constantes:**
  - Os blocos: a, b, c, d, e



- **Predicados:**
  - ontable(x) - bloco x está sobre a mesa
  - on(x,y) - bloco x está sobre o bloco y
  - clear(x) - bloco x não tem nada sobre ele
  - holding(x) - a garra do robô está segurando o bloco x
  - handempty - a garra do robô não está segurando nada

Exemplo - Mundo dos Blocos

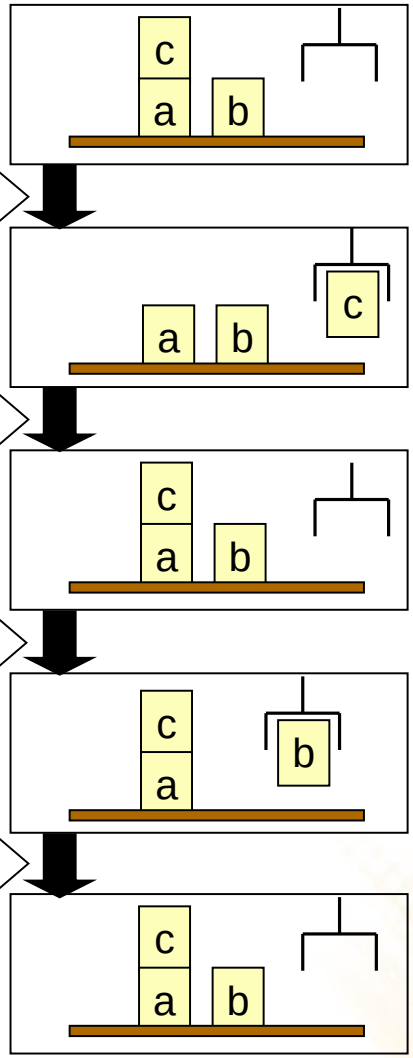
- Operadores:

unstack(*x*,*y*)
Precond: on(*x*,*y*), clear(*x*), handempty
Effects: ~on(*x*,*y*), ~clear(*x*), ~handempty, holding(*x*), clear(*y*)

stack(*x*,*y*)
Precond: holding(*x*), clear(*y*)
Effects: ~holding(*x*), ~clear(*y*), on(*x*,*y*), clear(*x*), handempty

pickup(*x*)
Precond: ontable(*x*), clear(*x*), handempty
Effects: ~ontable(*x*), ~clear(*x*), ~handempty, holding(*x*)

putdown(*x*)
Precond: holding(*x*)
Effects: ~holding(*x*), ontable(*x*), clear(*x*), handempty



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)

- Espaços de busca:
  - **Espaço de situações:** Funciona da mesma forma que na resolução de problemas por meio de busca.
  - **Espaço de planos:** planos parciais.
    - mais flexível.

Planejamento Progressivo

Forward-search(O, s_0, g)

$s \leftarrow s_0$

$\pi \leftarrow$ the empty plan

loop

if s satisfies g then return π

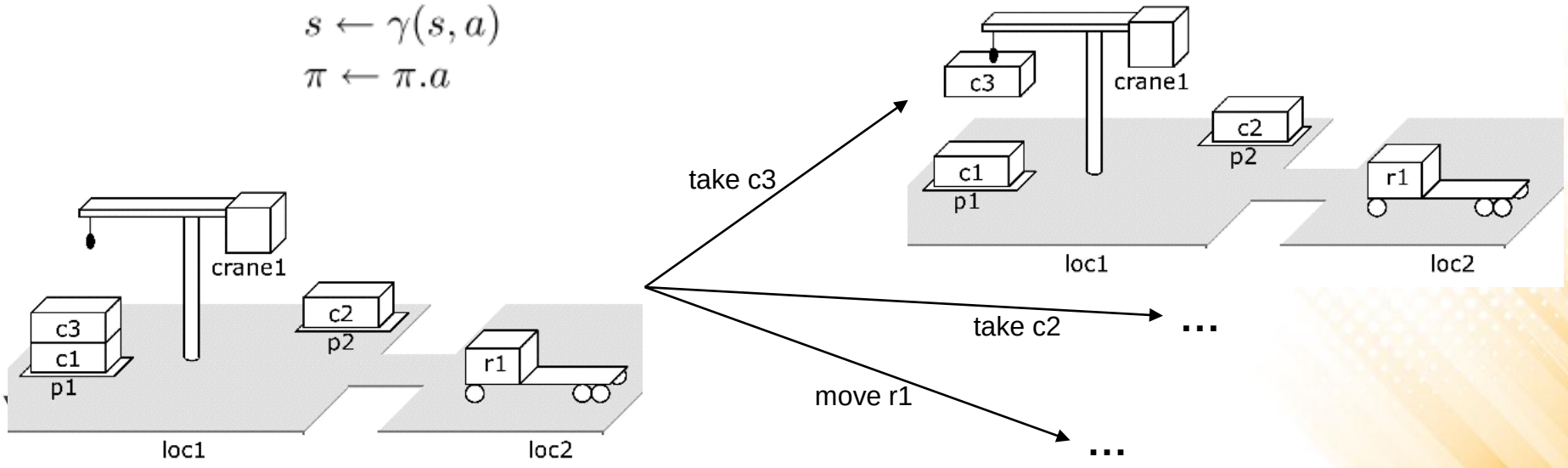
$E \leftarrow \{a | a$ is a ground instance an operator in O,
and $\text{precond}(a)$ is true in $s\}$

if $E = \emptyset$ then return failure

nondeterministically choose an action $a \in E$

$s \leftarrow \gamma(s, a)$

$\pi \leftarrow \pi.a$



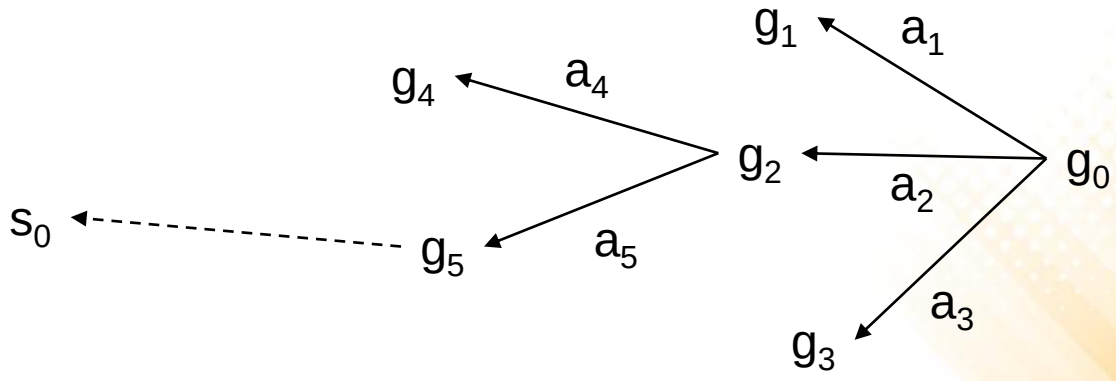
Planejamento Progressivo

- **Algoritmos de busca clássicos:**
  - Busca em profundidade;
  - Busca em largura;
  - Busca de custo uniforme;

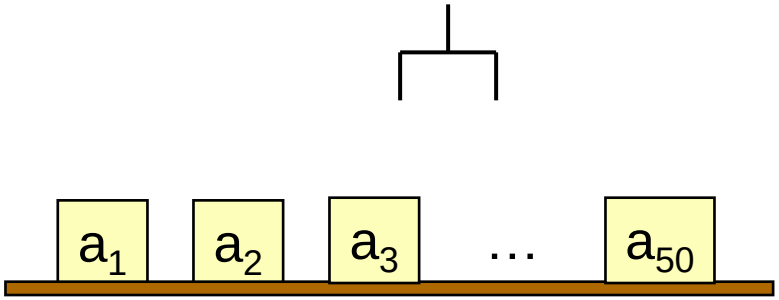
- Pode ter um fator de ramificação muito grande.

Planejamento Regresivo

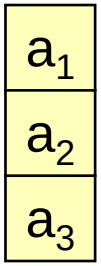
Backward-search(O, s_0, g)
  $\pi \leftarrow$ the empty plan
  loop
    if s_0 satisfies g then return π
    $A \leftarrow \{a | a$ is a ground instance of an operator in O
        and $\gamma^{-1}(g, a)$ is defined$\}$
    if $A = \emptyset$ then return failure
    nondeterministically choose an action $a \in A$
    $\pi \leftarrow a.\pi$
    $g \leftarrow \gamma^{-1}(g, a)$



Planejamento Regresivo



Estado inicial



Estado Objetivo

- O fator de ramificação da busca para trás é menor, mas existem casos onde pode ainda ser muito grande.
  - Muitas instâncias de operadores são avaliadas.

Busca em Espaço de Estados

- A **busca em espaço de estados** 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,

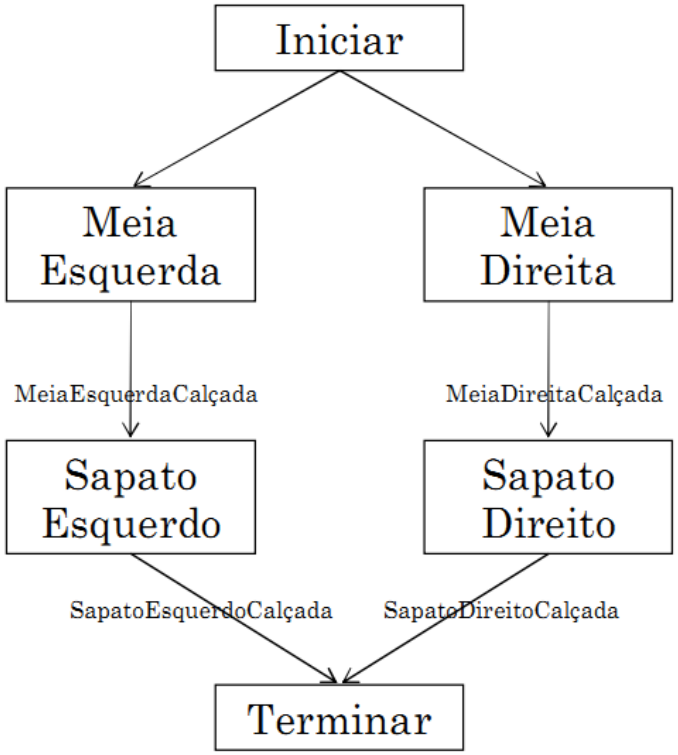
EFFECT: MeiaEsquerdaCalçada)

Exemplo dos Sapatos

- 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.

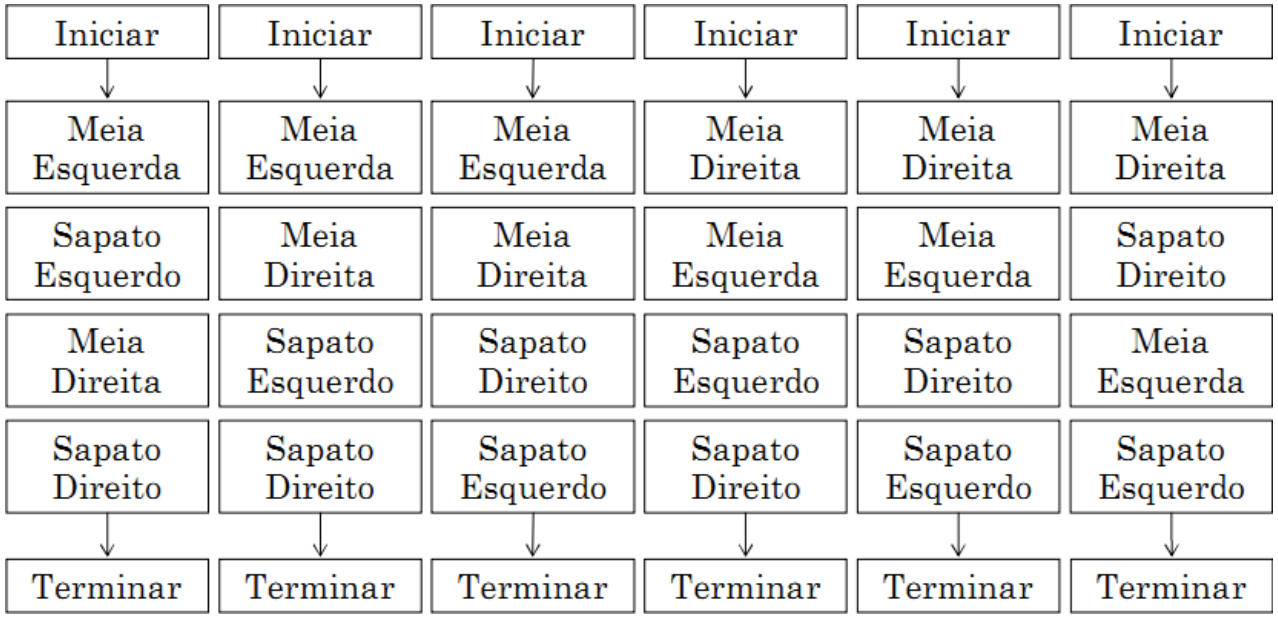
Exemplo dos Sapatos

- Plano de Ordem Parcial



Exemplo dos Sapatos

- Plano de Ordem Total



Planejamento de Ordem Parcial

- 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 as ações 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.

Planejamento de Ordem Parcial

- 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)**

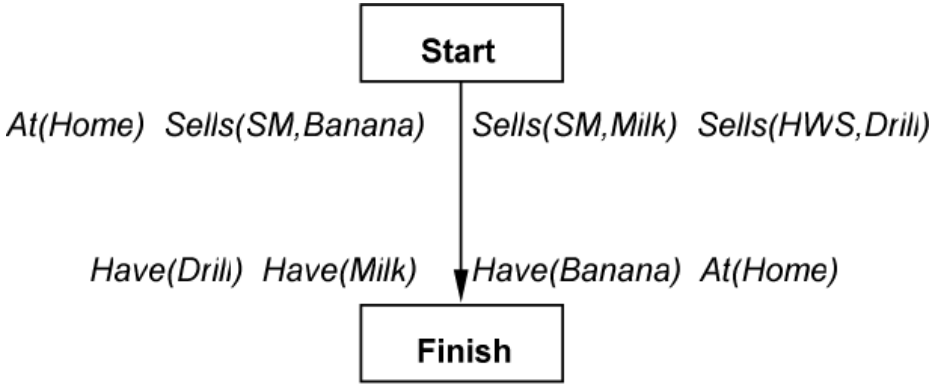
  - Para o problema de colocar meias e sapatos: colocar cada meia antes do sapato, sem dizer por onde começar (esquerda ou direita)

Planejamento de Ordem Parcial

- **Algoritmo de planejamento de ordem parcial:**
  - Identifica-se um passo com a pré-condição (sub-goal) não satisfeita.
  - Introduz-se um passo cujo efeito satisfaz a pré-condição.
  - Instancia-se variáveis e atualiza-se as ligações causais.
  - Verifica-se se há conflitos e corrige-se o plano se for o caso.

Exemplo

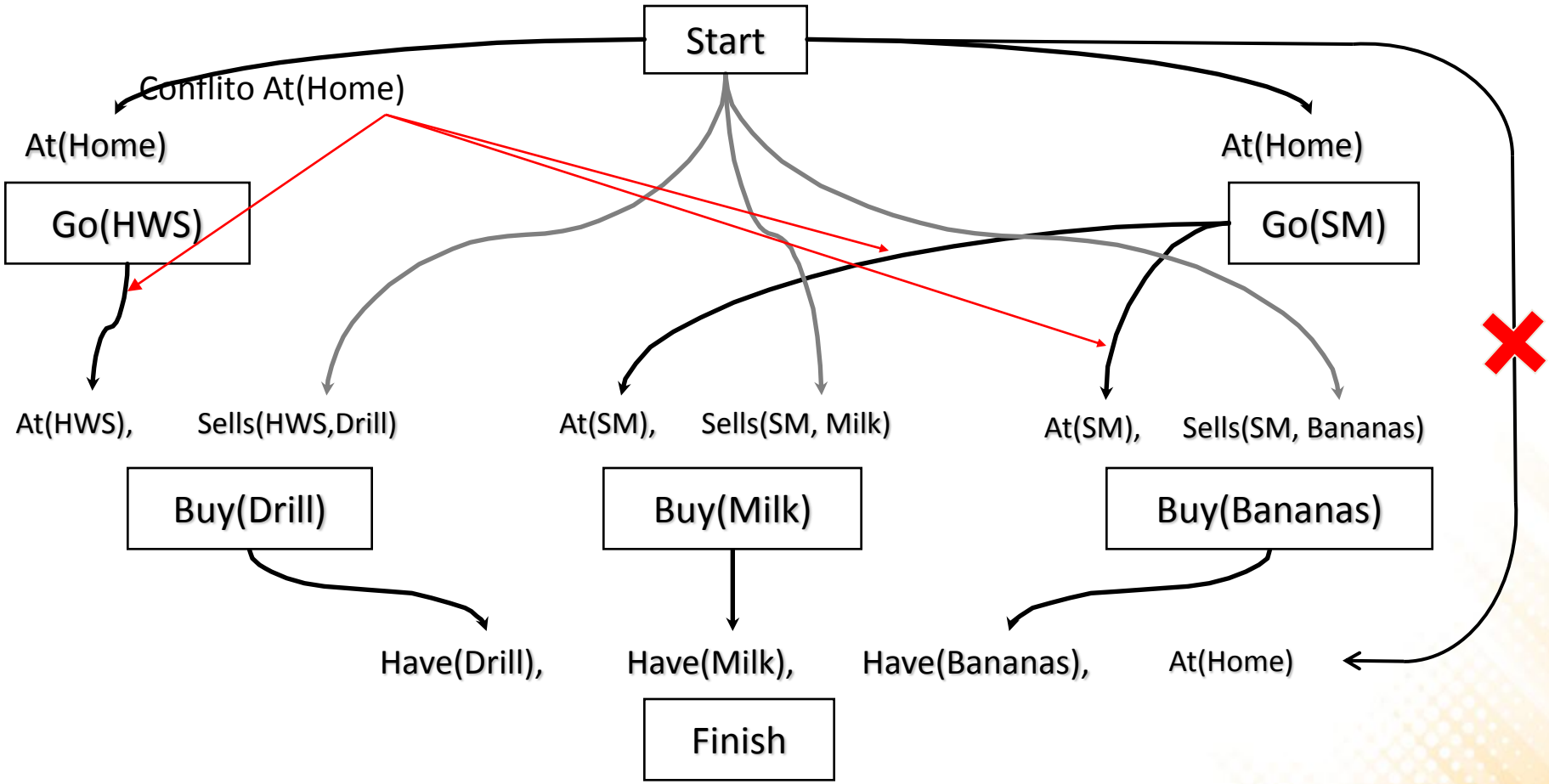
- Plano Inicial:



- Ações:

Op(ACTION: **Go**(there),
PRECOND: At(here),
EFFECT: At(there) $\wedge$ $\neg$ At(here))
Op(ACTION: **Buy**(x),
PRECOND: At(store) $\wedge$ Sells(store, x),
EFFECT: Have(x))

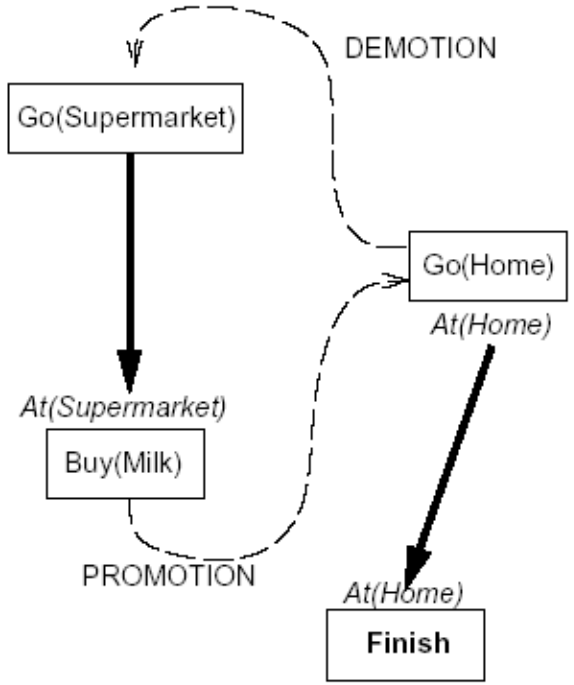
Exemplo



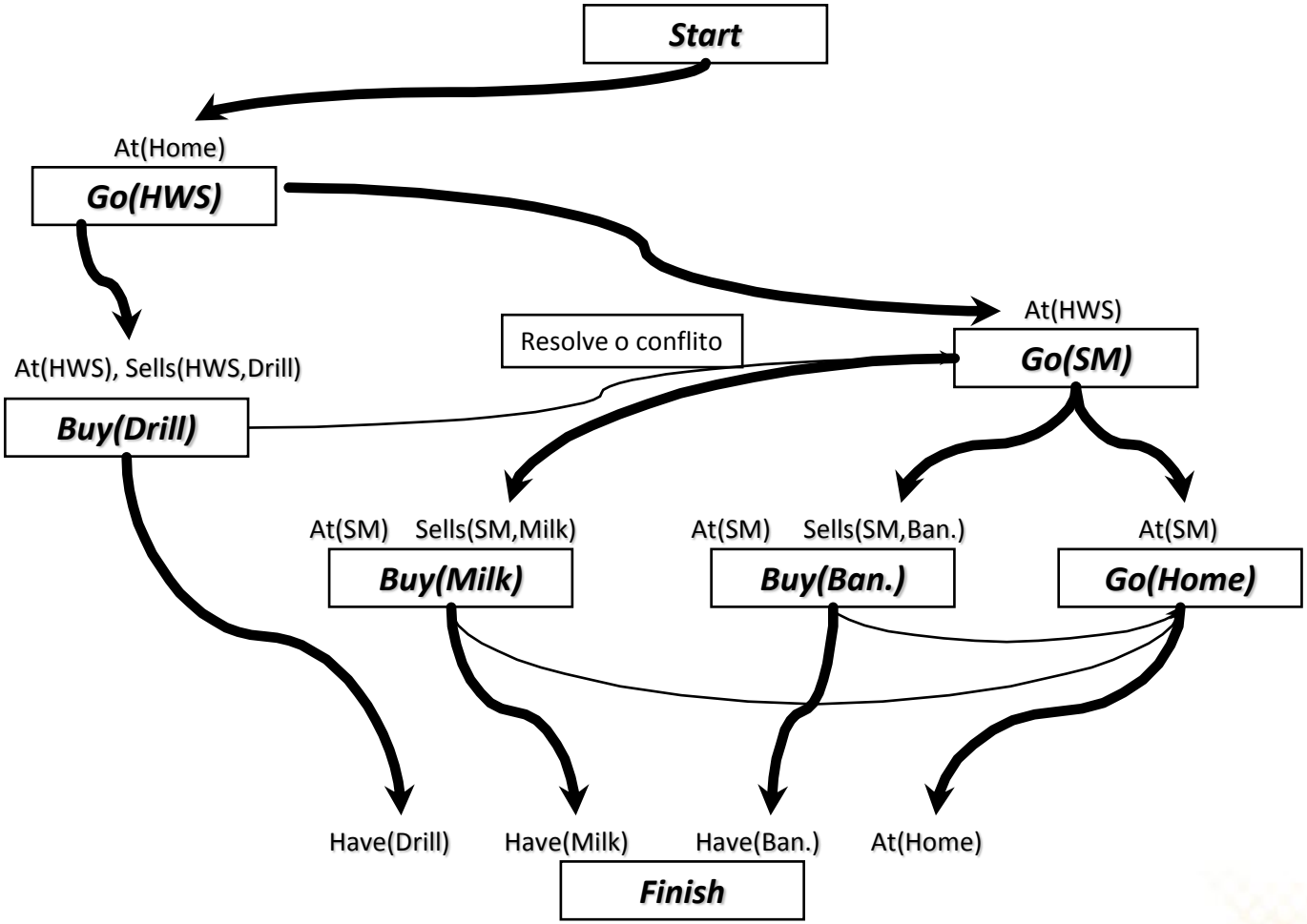
visionlab
visualization laboratory

Conflito em Planejamento de Ordem Parcial

- 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).
- Demotion e Promotion:

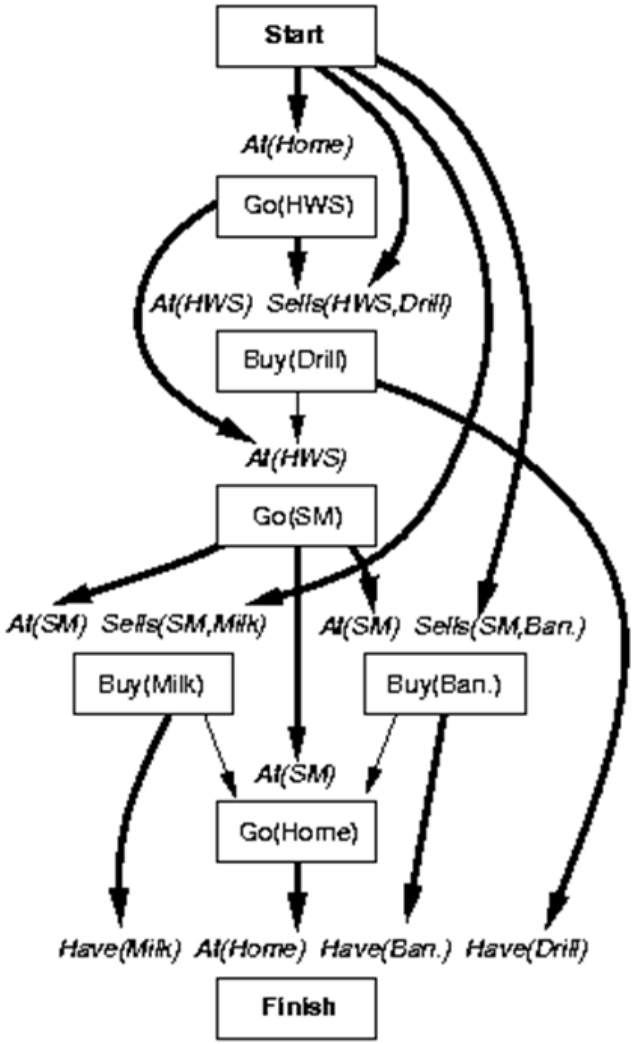


Exemplo



Exemplo

- Plano de Ordem Parcial



Planejamento Hierárquico

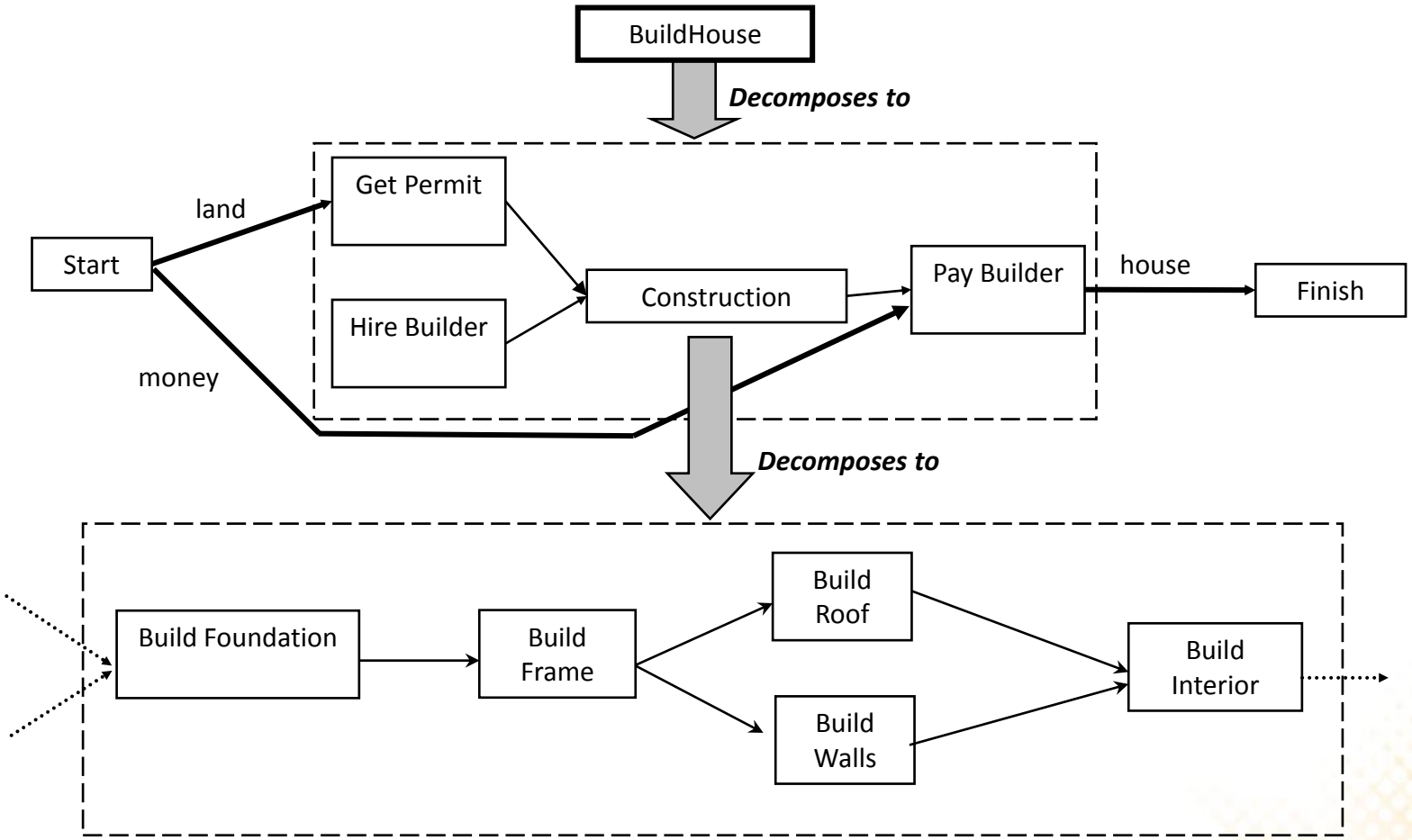
- **Hierarchical Task Network (HTN) Planning**
  - Planejamento que busca refinar um plano com a **decomposição hierárquica** de operadores abstratos.

- Em planejamento HTN, o plano inicial que descreve o problema, é visto como uma **descrição de alto nível** do que deve ser feito.

- Faz uma **busca no espaço de redes de tarefas** através das diferentes decomposições de ações compostas.

  - Ações compostas representam sub-metas de alto nível.
  - Ações primitivas representam ações.

Exemplo



Planejamento Hierárquico

- **Plan library:**
  - Contém várias decomposições de ações abstratas em menos abstratas ou mesmo planos inteiros pré-concebidos.
  - Cada ação abstrata tem pré-condições e efeitos que são comuns a todas as instanciações dela.

- As decomposições podem ser expressadas da seguinte maneira **Decompose(a, d)** - uma ação **a** um pode ser decomposta em plano **d**.

Planejamento Hierárquico

- **Planejamento hierárquico híbrido**
  - Na prática, se mistura operadores de decomposição **HTN** com outros operadores do **planejamento de ordem parcial**.

Decompose(Contruction,
Plan(STEPS:{S_1: Build(Foundation),S_2:Build(Frame),
S_3: Build(Roof), S_4:Build(Walls),
S_5: Build(Interior)}
Orderings:{$S_1<S_2<S_3<S_5$, $S_2<S_4<S_5$},
Bindings:{},
Links:{$S_1 \xrightarrow{\text{Foundation}} S_2$, $S_2 \xrightarrow{\text{Frame}} S_3$, $S_2 \xrightarrow{\text{Frame}} S_4$,
$S_3 \xrightarrow{\text{Roof}} S_5$, $S_4 \xrightarrow{\text{Walls}} S_5$}))

Planejamento Hierárquico

- **Algoritmo:**
  - Constrói-se um plano de ordem parcial inicial no **maior nível de abstração**.
  - Recursivamente **decompõem-se ações abstratas** até o plano de ordem parcial final conter apenas operadores primitivos (que podem ser executados pelo agente).
  - **Resolve-se ameaças** e verifica-se a **consistência global** do plano de ordem parcial final.

Aplicações de Planejamento

- Qualquer problema que necessite de **passos/ações** para chegar a um determinado **objetivo**.

- Exemplos:

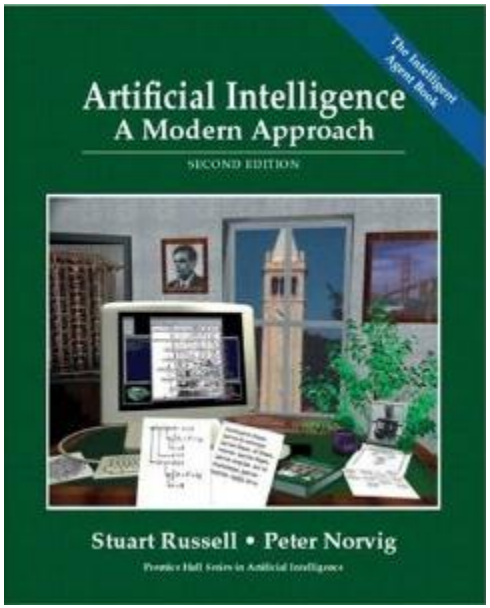
  - Robôs que realizam tarefas.

  - Personagens de jogos direcionados a objetivos.

  - Geração de histórias para storytelling interativo.

Leitura Complementar

- Russell, S. and Novig, P. **Artificial Intelligence: a Modern Approach**, 2nd Edition, Prentice-Hall, 2003.



- **Capítulo 11: Planning**
- **Capítulo 12: Planning and Acting in the Real World**