MAC5788
Planejamento em Inteligência Artificial

Introdução

(Esta apresentação possui algumas páginas do curso de Introdução à IA e portanto, contém exemplos e termos em inglês extraídos do livro AIMA, adotado neste curso.)

Leliane Nunes de Barros

Introdução

- O que é planejamento?
- Artigo recomendado: *A Review of AI Planning Techniques.* Austin Tate, James Hendler e Mark Drummond ==> mostra os problemas ainda não resolvidos no início dos anos 90.
- No final dos anos 90 houve uma grande evolução da área. Este curso pretende esclarecer os problemas apontados neste artigo e apresentar as técnicas de planejamento modernas.

Suposições de planejamento clássico

- tempo atômico
- existência de um único agente
- determinismo
- onisciência
- causa de mudança única
- função objetivo: número de passos do plano

Modelo de Estados

O planejamento clássico pode ser caracterizado como um Modelo de Estados

- um espaço de estados S, finito e não-vazio
- um estado inicial $s_0 \in S$
- um conjunto de estados meta $S_G \subseteq S$
- um conjunto de ações aplicáveis $A(s) \subseteq A$, para $\forall s \in S$
- uma função de transição $s' = f(a,s)$, para $\forall s', s \in S$ e $\forall a \in A(s)$
- custos das ações $c(a,s) > 0$

A linguagem Strips

- Strips (*Stanford Research Institute Problem-Solving*) é a mais antiga, simples e usada linguagem de planejamento (Fikes e Nilsson, 1971)
- evolução: Strips $\rightarrow$ ADL $\rightarrow$ PDDL2.1 $\rightarrow$ PDDL+
- Um problema em Strips é uma tupla $\langle A, O, I, G \rangle$
  - A é o conjunto de todos os átomos (variáveis booleanas como descritores de estados),
  - O é um conjunto de todos os operadores (ações proposicionais), e
  - $I \subseteq A$ representa a situação inicial
  - $G \subseteq A$ representa as situações meta

A linguagem Strips

- Os operadores $\boldsymbol{o} \in \boldsymbol{O}$ são representados por três listas:
  - lista Pre, $Pre \subseteq A$
  - lista Add, $Add(o) \subseteq A$
  - lista Del, $Del \subseteq A$

Intuitivamente: *Pre(o)* especifica os átomos que devem ser verdadeiros para *o* ser aplicável, *Add(o)* especifica os átomos que passam a ser verdadeiros após a execução de *o* e *Del(o)* especifica os átomos que passam a ser falsos após a execução de *o*

Strips: da linguagem aos modelos

Um problema Strips, ***P* = <*A, O, I, G*>** determina um modelo de estados ***S(P)*** em que:

- os estados $s \in S$ são coleções de átomos;
- o estado inicial s_0 é I
- os estados meta $s \in S_G$ são aqueles em que $G \subseteq s$
- as ações $a \in A(s)$ são operadores $o \in O$ tal que $Prec(o) \subseteq s$
- a função de transição *f* mapeia estados *s* em estados $s' = f(s, a)$, tal que $s' = s - Del(a) + Add(a)$ para $a \in A(s)$
- os custos das ações são iguais a 1
- a solução (ótima) de um problema de planejamento é a solução (ótima) do Modelo de Estados *S(P)*

Modelo de ações Strips

Nome da ação: *Go(there)*

Pré-condição: *At(here), Path(here,there)* *conjunção de literais positivos*

Add: *At(there)*

Del: *At(here)* *conjunção de literais positivos ou negaivos*

$$At(here),\ Path(here,there)$$

$$\boxed{Go(there)}$$

$$At(there),\ \neg\, At(here)$$

Linguagem do tipo Strips

Não existe variável explícita do estado do mundo (como por exemplo, tempo ou situação)

– Tudo o que for declarado na <u>pré-condição</u> se refere à situação imediatamente <u>antes</u> da ação a ser executada.

– Tudo o que for declarado no <u>efeito</u> (listas add and del) se refere à situação imediatamente <u>após</u> a ação ser executada.

Raciocínio sobre ações STRIPS

Enfoque no plano:

(there milk) → **get milk** → (here milk), ¬ (there milk)

(there cheerios) → **get cheerios** → ¬ (there cheerios), (here cheerios)

(here cheerios), (cereal cheerios), (here milk) → **pour milk cereal** → ¬ (here cheerios), ¬ (here milk), (here breakfast)

get milk → **get cheerios** → **pour milk cereal**

Enfoque no estado do mundo:

Initial State	*after (get milk)*	*after (get cheerios)*	*Goal State*
¬ (here cheerios)	¬ (here cheerios)	(here cheerios)	¬ (here cheerios)
(there cheerios)	(there cheerios)	¬ (there cheerios)	¬ (there cheerios)
(cereal cheerios)	(cereal cheerios)	(cereal cheerios)	(cereal cheerios)
¬ (here milk)	(here milk)	(here milk)	¬ (here milk)
(there milk)	¬ (there milk)	¬ (there milk)	¬ (there milk)
(time morning)	(time morning)	(time morning)	(time morning)
¬ (here breakfast)	¬ (here breakfast)	¬ (here breakfast)	(here breakfast)

Operador *schema*

- Operador *schema*: representação de um conjunto de ações (contém variáveis)
- Corresponde a uma família de ações
  - Assumi-se que variáveis são universalmente quantificadas
  - O algoritmo de planejamento deve garantir que cada variável tenha um valor no plano final. Somente ações totalmente instanciadas podem ser executadas.

Ação: *Go(there)*
Pré-condição: *At(here), Path(here, there)*
Efeito:
    Lista de Adição: *At(there)*
    Lista de Eliminação*: At(here)*

Operador *instance*

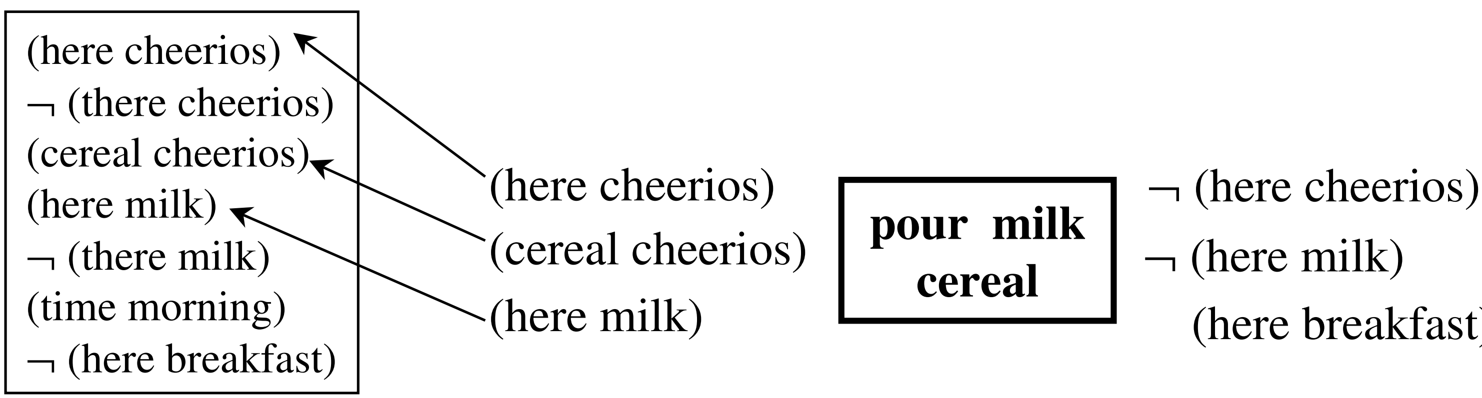
- Operador *instance* ou proposicional: representa uma única ação sobre objetos específicos do domínio (constantes).
- Alguns planejadores raciocinam diretamente sobre ações proposicionais. É preciso usar um *parser* para gerar operadores proposicionais a partir de um operador *schema*

Ação: *Go(BlocoB)*
Pré-condição: *At(BlocoA), Path(BlocoA, BlocoB)*
Efeito:
    Lista de Adição: *At(BlocoB)*
    Lista de Eliminação*: At(BlocoA)*

Aplicabiblidade de um *schema*

- Um operador *o* é ***aplicável*** em um estado (ou situação) *s* se existe uma maneira de se unificar as variáveis em *o* de forma que todos os literais da <u>pré-condição</u> de *o* são verdadeiros, ou seja:

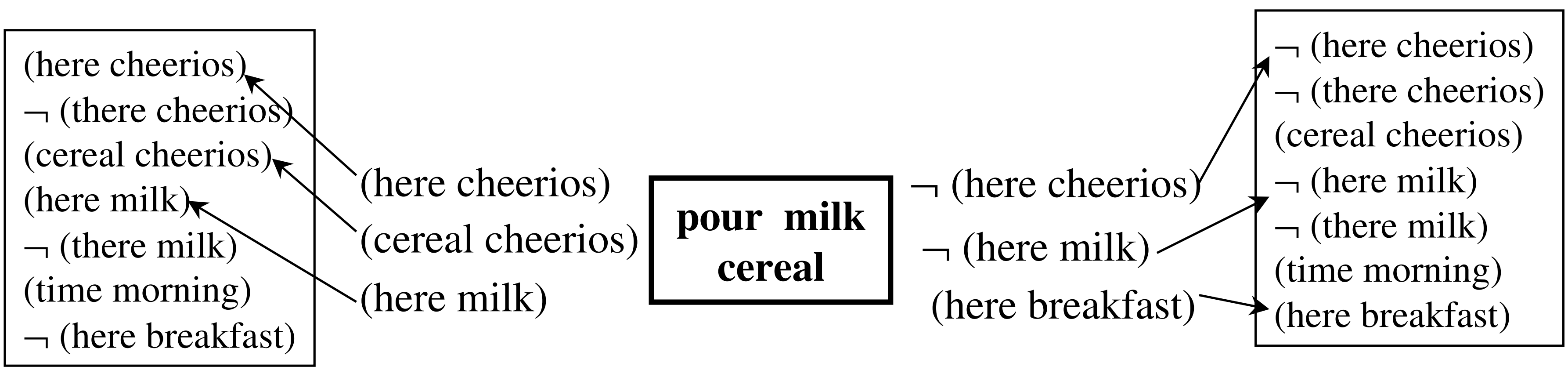
  $$\textit{Pré-condição}(o) \subset s$$



Efeitos de um *schema*

Estado resultante da aplicação de um operador no estado *s* :

- valem os literais positivos de *Efeito*(*o*) com as unificações feitas nas variáveis da pré-condição
- e aqueles em *s* exceto os pertencentes à lista de eliminação (literais negativos de *Efeito*(*o*))



Planejamento usando a linguagem STRIPS

- Planos podem ser representados:
  - ordem-total
  - ordem-parcial
  - grafo de planejamento
  - caminho no espaço de busca heurística (local ou global)
  - etc ...

Planejamento usando a linguagem STRIPS

- O raciocínio pode ser :
  - busca não-informada (planejamento progressivo ou regressivo)
  - busca num espaço de estados = planejadores de ordem-total (*total-order planners*) ~~*linear planners*~~
  - busca num espaço de planos = planejadores de ordem-parcial (*parcial-order planners*) ~~*non-linear planners*~~
  - busca num grafo de planejamento
  - busca informada
    - busca heurística
    - uso de conhecimento sobre interação entre ações
  - etc ...

Planejamento de ordem total: busca em espaço de estados (situações)

Espaço de estados: o plano é o caminho a partir do estado inicial ao estado meta

- *planejador progressivo*: busca para frente (forward) a partir da situação inicial à situação meta no espaço de situações possíveis (fator de ramificação grande)

  adiciona ações ***aplicáveis*** no fim do plano

- *planejador regressivo* : busca para trás (backward) a partir do estado meta para o estado inicial (metas tipicamente contém poucos literais (descrição parcial de estados) com poucos operadores aplicáveis)

  adiciona ações ***apropriadas*** no início do plano

Um operador *o* é apropriado para um estado meta G se G $\subset$ *Efeito(o)*

Planejamento de ordem total: busca em espaço de estados (situações)

- Problema: metas conjuntivos nem sempre são independentes
- Strips: *situation-regression planner* (incompleto)

- outras abordagens mais modernas: busca heurística (local ou global). Resolvem conflitos através do uso de heurísticas.
  - Hill Climbing e Hill Climbing Reforçado
  - WA*

Algoritmo geral de busca

função Busca-Geral(*problema*, *Queueing-FN*) devolve uma solução, ou falha

/* inicializa a árvore de busca com o estado inicial do *problema* */

nós ← CRIA-LISTA(Estado-Inicial[problema])

laço faça

se a lista de *nós* estiver vazia então devolva falha

nó ← Remove-Primeiro (*nós*)

se Testa-Meta(*nó*) então devolva *nó* (e o caminho da raiz até o nó)

nós ← *Queueing-FN* (nós, EXPANDA(nó, Ações[problema]))

fim

Estratégias em *Queueing-FN*:

inserção numa fila: busca em largura

inserção numa pilha: busca em profundidade

fila de prioridade: A* (ordenar a fila de acordo uma função de avaliação

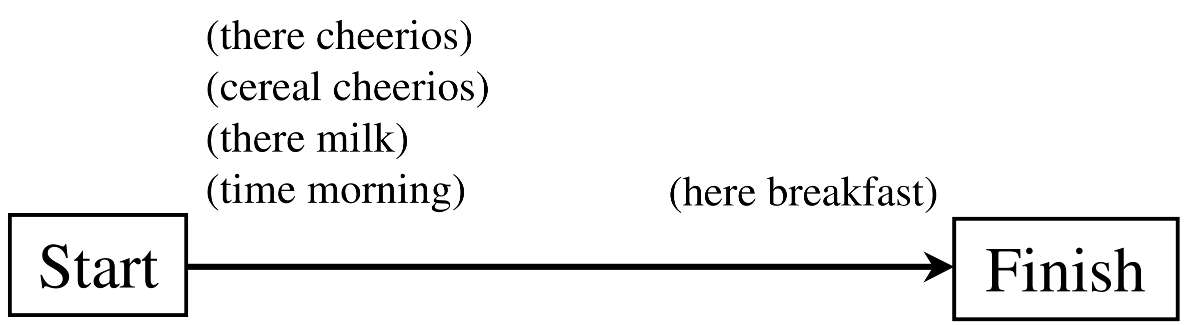
f(n)= g(n) + h(n))

Primeiro EP

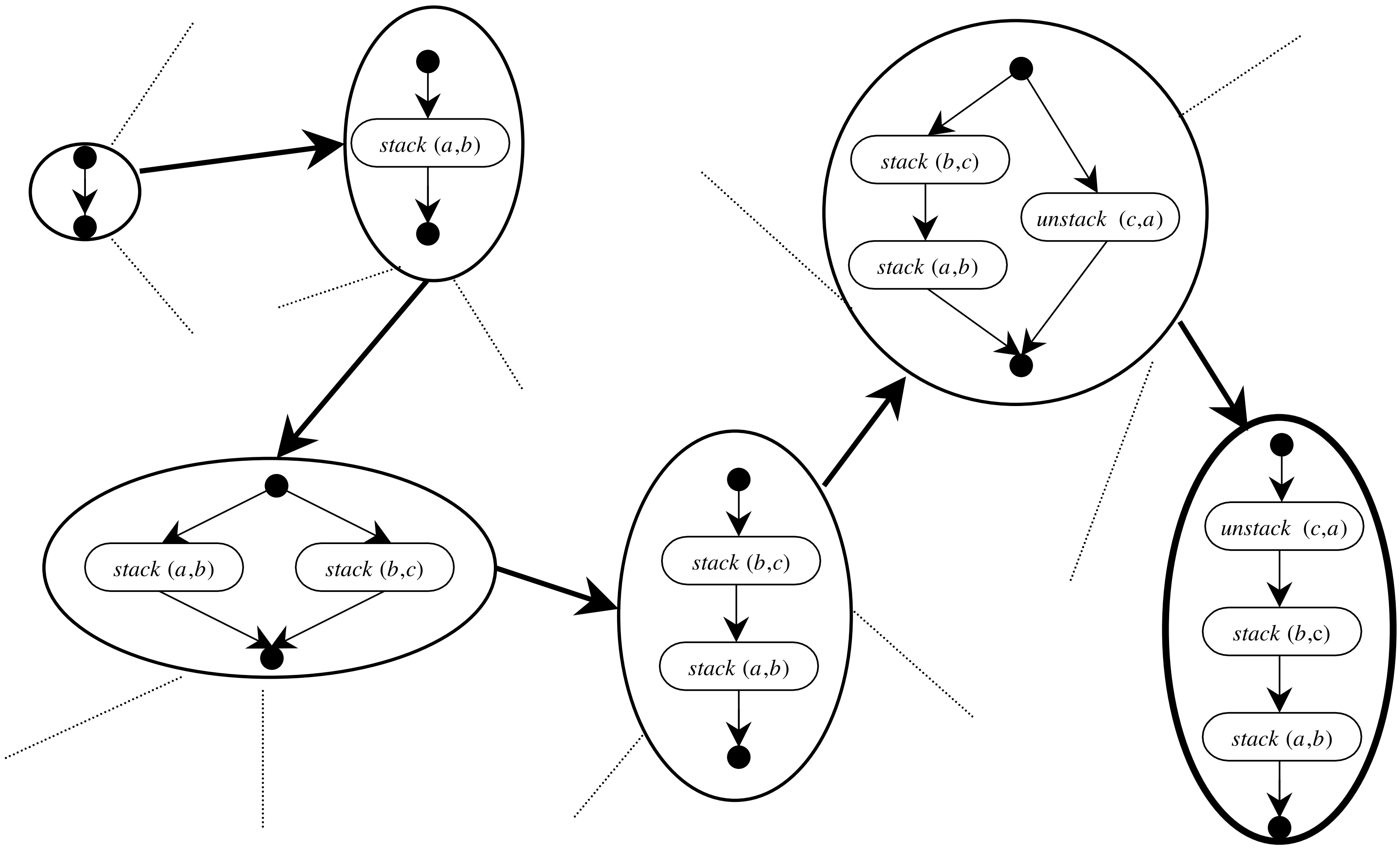
- Construir um planejador totalmente ordenado com as seguintes estratégias:
  - regressivo
  - progressivo
  - A* com função heurística igual ao numero de metas não realizadas

Busca no Espaço de Planos

- Começa-se com um plano simples, incompleto: *plano parcial*
- Expande-se o plano parcial até gerar um plano solução que resolve o problema.



Busca no espaço de planos



Leliane Nunes de Barros

Operadores de refinamento do plano

Operações no plano podem ser de dois tipos: *refinamento* e *modificação*

- adicionar um passo no plano
- adicionar um vínculo causal
- adicionar restrições de ordem
- adicionar restrições de valores das variáveis
- detectar inconsistências e interações
- corrigir interações (modificar ordem ou valores de variáveis)
- *backtracking*

Planejamento de compromisso fraco

Planejadores podem adotar uma estratégia de compromisso fraco *(least commitment planners)* com relação a:

- ordenação de passos de plano e
- unificação de variáveis (*codesignation*)
- vínculos causais e intervalos de proteção

Planejamento de ordem parcial X Planejamento de ordem total

Estratégia de compromisso fraco com relação a ordem

- *Planejamento de ordem total*: representa planos onde todo passo do plano é ordenado com relação a todos os demais passos do plano.
- *Planejamento de ordem parcial*: representa planos onde somente alguns passos são ordenados.

Um plano totalmente ordenado, derivado de um plano parcialmente ordenado *p* é chamado *linearização de p*

Exemplo

- Objetivo:

  LeftShoeOn, RightShoeOn

- Operadores:

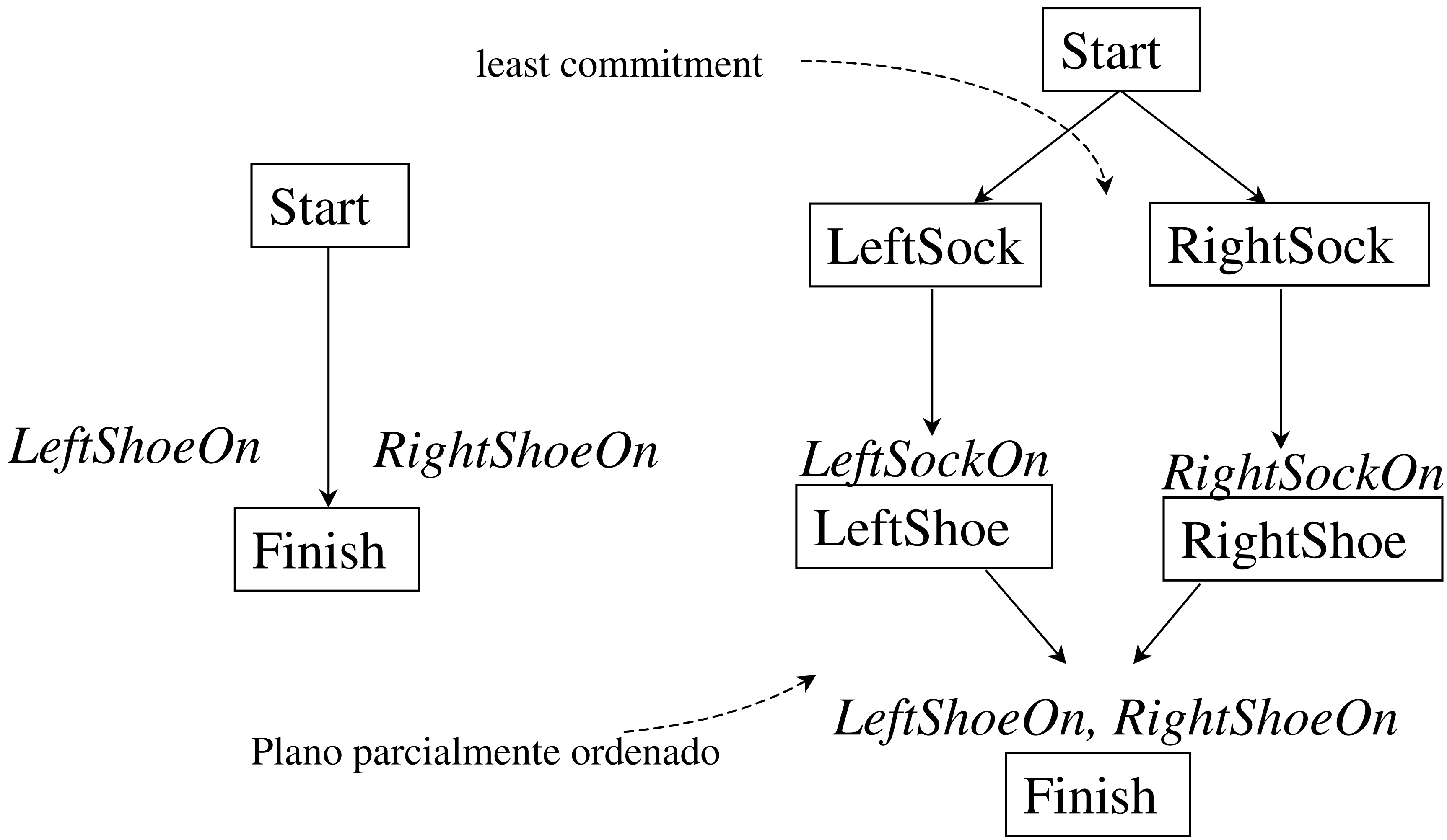
  *Op(*Action:*RightShoe,*Precond:*RightSockOn,*Effect:*RightShoeOn*)

  *Op(*Action:*RightSock,*Effect:*RightSockOn*)

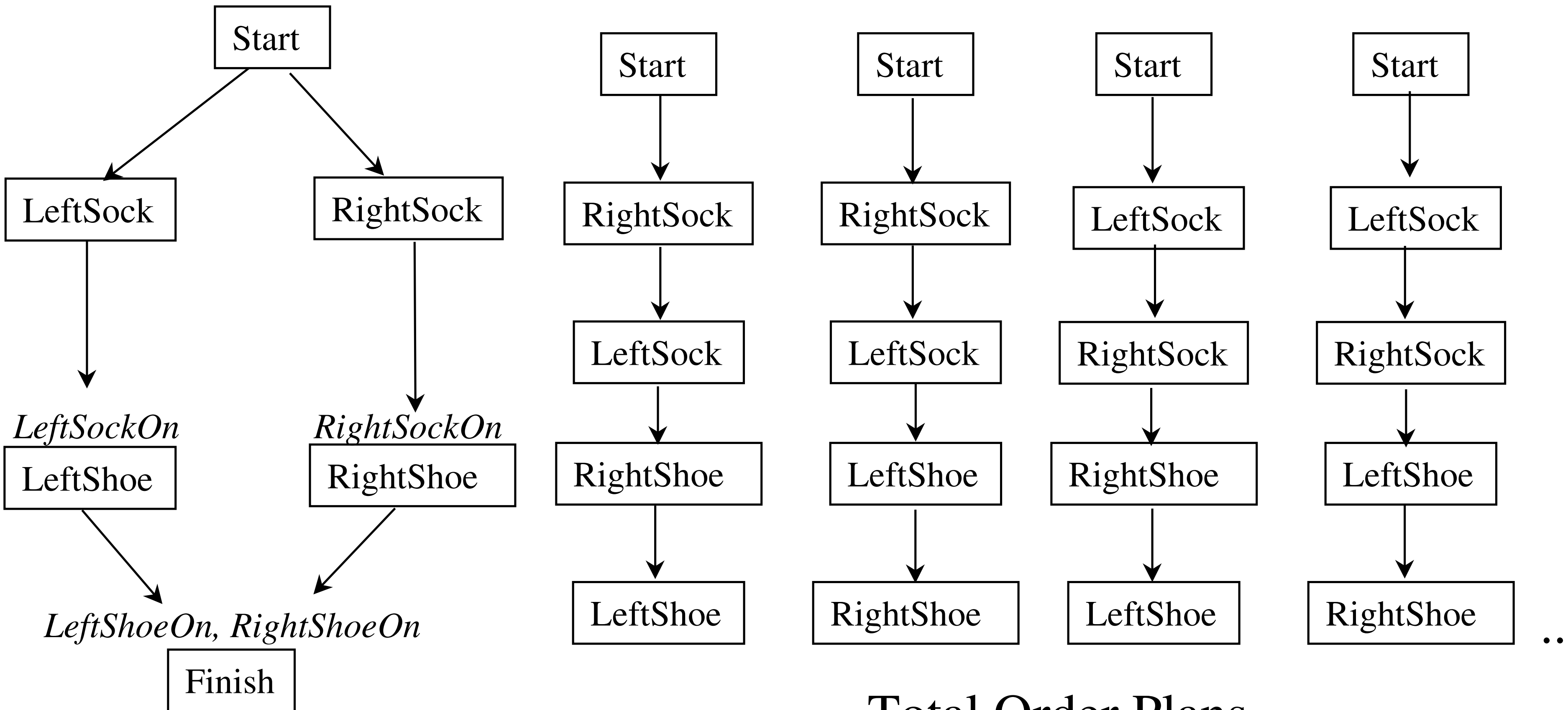
  *Op(*Action:*LeftShoe,*Precond:*LeftSockOn,*Effect:*LeftShoeOn*)

  *Op(*Action:*LeftSock,*Effect:*LeftSockOn*)

Representações de planos



Linearizações de *p*



Partial Order Plan

Total Order Plans

Representação de Plano

Composto por 4 componentes:

① Um conjunto de **passos de plano +** um mapeamento entre passos e operadores com suas unificações

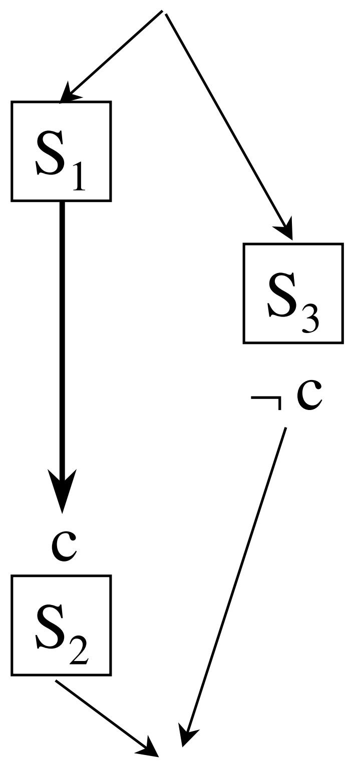
② Um conjunto de **restrições de ordem** sobre os passos do plano :

- $S_i \prec S_j$, que deve ser lido "S_i *ocorre antes de* S_j" , que significa: S_i deve ocorrer antes de S_j mas não necessariamente imediatamente antes.

③ Um conjunto de **restrições de unificação** do tipo $v = x$ ou $v = \neg x$ ou $v \neq x$, sendo x uma constante ou uma outra variável (codesignação ou não-codesignação).

④ Alguns planejadores incluem no plano restrições auxiliares (p.e., vínculos causais)

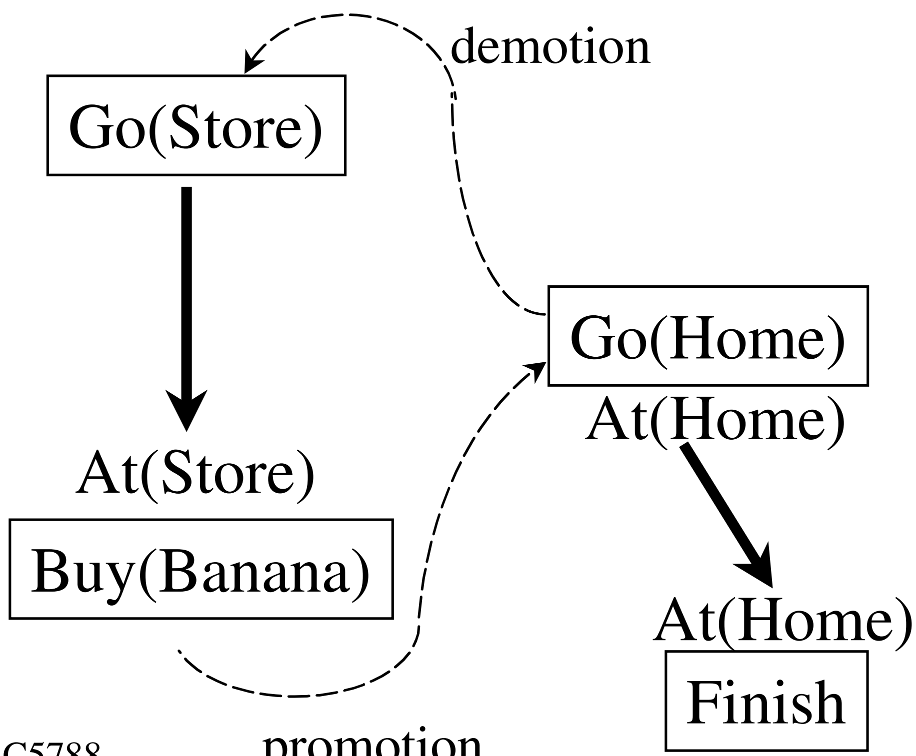
Verificando a satisfação de metas



Uma precondição ***c*** de um passo S_2 é satisfeita sse ela é efeito de um passo S_1 e <u>não existir</u> um passo S_3 , que ocorra possivelmente entre o intervalo S_1 *e* S_2 , *"ameaçando"* a satisfação de ***c*** **.**

Modificando o plano

- Um passo que ameaça *(threaten)* ou interage é um passo do plano que pode destruir uma pré-condição já satisfeita



Demotion (:
coloque
antes de Go(Store)
Promotion: coloque
depois de
Buy(Banana)

Um algoritmo para Planejamento de Ordem Parcial

[AIMA, 1996]

O Planejador POP

Partial-Order Planner

Representação de Plano

④ Um conjunto de *vínculos causais* (***causal links***):

$$S_i \xrightarrow{c} S_j$$

que deve ser lido "O passo contribuidor S_i, *realiza c para o passo que necessita* S_j"

Vínculos causais servem para: (i) registrar o propósito dos passos no plano, onde o propósito de S_i é realizar a pré-condição c de S_j; (ii) proteger a satisfação de uma meta ou pré-condição.

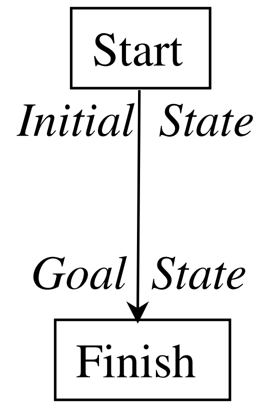
Também chamados por alguns autores de ***intervalos de proteção (protection intervals)***

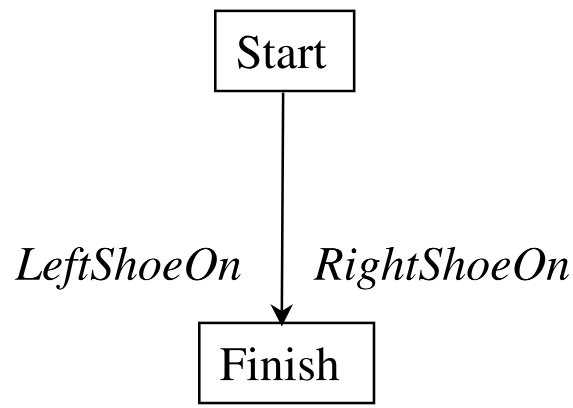
Plano inicial

- Corresponde à própria descrição do problema
- Dois passos: *Start* e *Finish*
  - onde, *Start* < *Finish*
  - estão associados a ações nulas

Start não tem pré-condição e seu efeito é o de adicionar as proposições verdadeiras no estado inicial

Finish tem como pré-condições as metas e nenhum efeito





Plano inicial

Plan(Steps: {S_1: *Op*(Action: *Start*),

S_2: *Op*(Acion: *Finish*,

Precond: *RightShoeOn* ^ *LeftShoeOn*)},

Ordering: {$S_1 < S_2$},

Bindings: { },

Causal Links: { })

Plano solução

- Um plano que o agente pode executar e que garante a realização do estado meta
  - plano totalmente instanciado e <u>totalmente ordenado</u> ou
  - plano totalmente instanciado e <u>parcialmente ordenado</u> (corresponde a um conjunto de planos totalmente ordenados). Cabe ao agente escolher uma linearização ou o plano pode ser executado por multi-agentes.

Plano solução (def.)

- Plano <u>completo</u> (teste de fim de planejamento):
  - toda precondição é realizada por algum passo do plano.
  - Uma precondição é realizada sse ela é efeito de um passo e nenhum <u>passo intermediário</u> a desfaz (teoria sobre interação de ações, por exemplo: MTC, axiomas de frame do SC ou axiomas do EC)
- Plano <u>consistente</u>:
  - não existem contradições nas restrições de ordenação ou codesignações

Algoritmo POP

function POP(*initial, goal, operators)* **returns** *plan*

plan ← Make-Minimal-Plan(*initial, goal*)

loop do

if Solution? (plan) **then return** *plan*

S_{need} , c ← Select-Subgoal(*plan*)

Choose-Operator(*plan, operators,* S_{need}, *c*)

Resolve-Threats(*plan)*

end

function Select-Subgoal(*plan*) **returns** S_{need} , c

pick a plan step S_{need} from Steps(*plan*)

with a precondition c that has not been achieved

return S_{need} , c

POP algorithm (cont.)

procedure Chose-Operator(*plan,operators,* S_{need} , c *)*

  choose a step S_{add} from *operators* or Steps(*plan*) that has *c* as an effect

  if there is no such step **then fail**

  add the causal link $S_{add} \xrightarrow{c} S_{need}$ to Links(*plan*)

  add the ordering constraint $S_{add} \prec S_{need}$ to Orderings(*plan*)

  if S_{add} is a newly added step from *operators* **then**

    add S_{add} to Steps(*plan*)

    add $Start \prec S_{add} \prec Finish$ to Orderings(*plan*)

POP algorithm (cont.)

procedure Resolve-Threats(*plan*)

for each S_{threat} that threatens a link $S_i \xrightarrow{c} S_j$ in Links(*plan*) **do**

choose either

Demotion: Add $S_{threat} < S_i$ to Orderings(plan)

Promotion: Add $S_i < S_{threat}$ to Orderings(plan)

if not consistent(*plan*) **then fail**

end

POP é correto e completo

Próximos tópicos

- Leitura obrigatória:

Systematic Nonlinear Planning. D. McAllester and D. Rosenblitt. In AAAI'91.

- Leitura recomendada:

Planning for conjunctive goals. D. Chapman. Artificial Intelligence, 32(333-379) 1987.

On the nature and role of modal truth criteria in planning. Subarao Kambhampati and Dana S. Nau.

Artificial Intelligence, 88(1-2). 1996.