Planejamento
Capítulo 11

Parte II

Operador Strips

Ação: *Go(there)*

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

Efeito: *At(there), ¬ At(here)* *conjunção de literais positivos ou negaivos (a versão original de Strips dividia os efeitos em* **add list** *e* **delete list**)

Notação gráfica:

At(here), Path(here,there)

Go(there)

At(there), ¬ At(here)

3

Linguagem do tipo Strips

- Não existe variável explícita do estado do mundo (como por exemplo, tempo ou situação)
  - Tudo o que for declarado na pré-condição se refere à situação imediatamente antes da ação ser executada
  - Tudo o que for declarado no efeito se refere à situação imediatamente após a ação ser executada
- Operador *schema*: operador que 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.

4

Operador Strips

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

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

- Estado resultante:
  - valem os literais positivos de *Efeito(o)*
  - e aqueles em *s* exceto os pertencentes à *Delete List* (literais negativos de *Efeito(o)*)

Espaço de Estados (Situações)

- *Espaço de estados*: o plano é o caminho a partir do estado inicial ao estado goal
  - *planejador progressivo*: busca para frente (forward) a partir da situação inicial à situação goal 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 goal para o estado inicial (goals 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 goal G se G $\subset$ *Efeito(o)*
  - Problema: goals conjuntivos nem sempre são independentes
  - Strips: *situation-regression planner* (incompleto)

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.
- Operadores da busca são operações de extensão do plano: adicionar um passo no plano, adicionar restrições de ordem, instanciar variáveis, detectar e corrigir interações
- Operações no plano podem ser de dois tipos: *refinamento* e *modificação*

Exemplo

- Goal:

  LeftShoeOn, RightShoeOn

- Operadores:

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

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

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

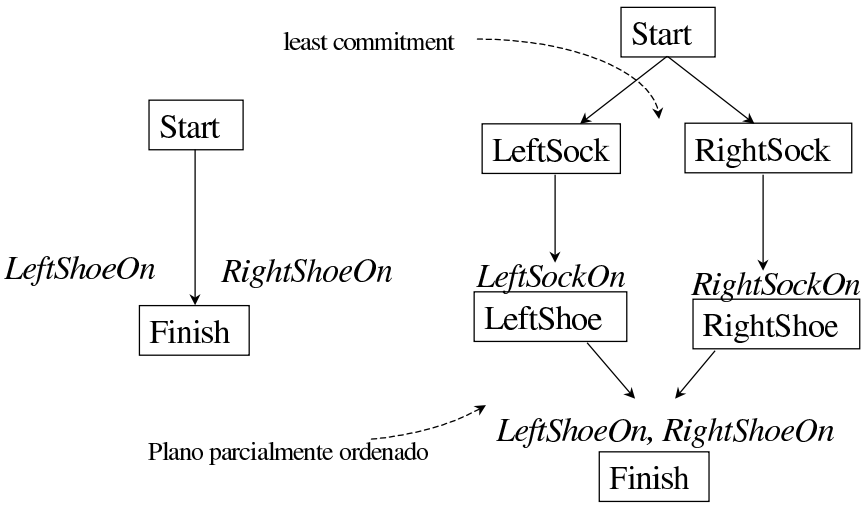
  *Op(*Action:*LeftSock*,Effect:*LeftSockOn*)

- Plano parcial inicialmente incompleto

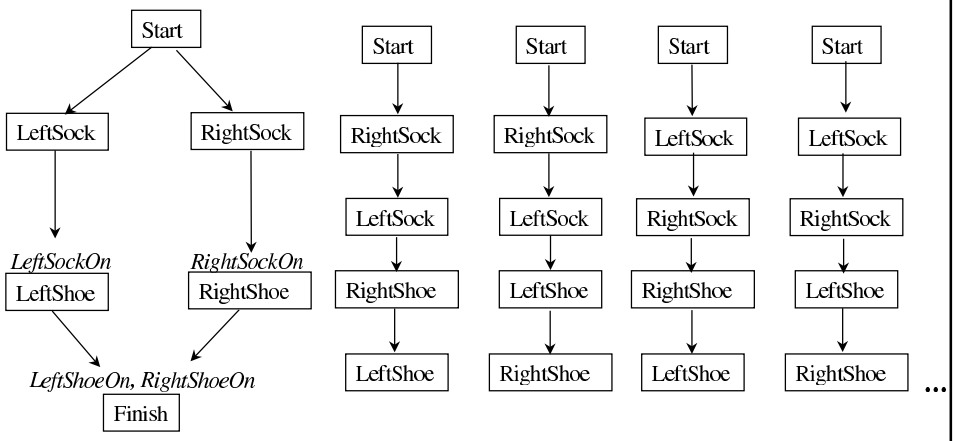
  LeftSshoe RightShoe

  em qual ordem?

Representações de planos



Linearizações de *p*



Partial Order Plan Total Order Plans

Partial Order Vs Total Order Planners

Least commitment (com relação a ordem)

- *Partial order planner*: representa planos onde alguns passos são ordenados e outros não
- *Total order planner*: representa planos onde todos os passos são totalmente ordenados (lista de passos)

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

11

Least Commitement Planners

- *Least commitement* pode ser feito com relação a:
  - a ordenação de passos de plano e
  - a instanciação de variáveis (*variable binding*)

12

Representação de Plano

Estrutura de dados contendo 4 componentes:

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

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

- $S_i < S_j$, 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 instanciações do tipo $v = x$ ou $v \neq x$, sendo x uma constante ou uma outra variável (codesignação ou não-codesignação).

Representação de Plano

④ Um conjunto de *ligações 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"

Causal links servem para registrar o propósito dos passos no plano, onde o propósito de S_i é realizar a pré-condição c de S_j.

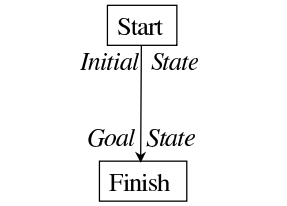
Também chamados por alguns autores de ***intervalos de proteção (protection intervals)***

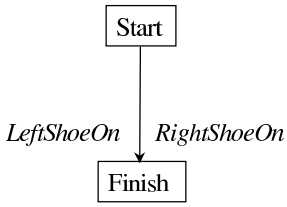
Plano inicial

- 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-condição o goal 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 realizar o goal
  - plano totalmente instanciado e totalmente ordenado ou
  - plano totalmente instanciado e parcialmente ordenado (corrresponde a um conjunto de planos totalmente ordenados)

17

Plano solução (def.)

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

18

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 $S_{add} < S_{need}$ to Orderings(*plan*)
  if S_{add} is a newly added step from *operators* **then**
    add S_{add} to Steps(*plan*)
    add $Start < S_{add} < Finish$ to Orderings(*plan*)

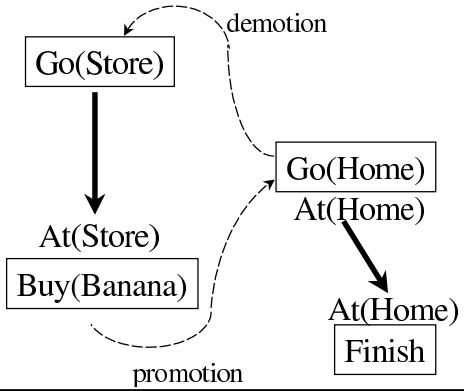
POP algorithm (cont.)

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, completo e sistemático (sem repetição)

Promotion /demotion

- *Clobberer* ou *threaten* é um passo de plano que destrói a condição realizada por um causal link

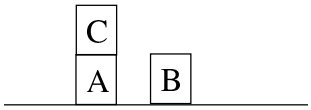


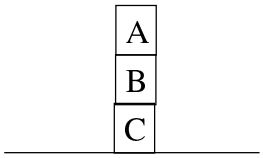
Demotion: coloque antes de Go(Store)
Promotion: coloque depois de Buy(Banana)

Mundo dos Blocos

- *Move(x,y,z)* move bloco x de cima de y para cima de z e z pode ser a mesa ou outro bloco
- *Move* é aplicável somente se x e z estiverem *clear* e x estiver sobre y.
- A mesa possue sempre espaço *clear*
- Um bloco é *clear* se não possue nenhum outro bloco em cima dele

Mundo dos Blocos





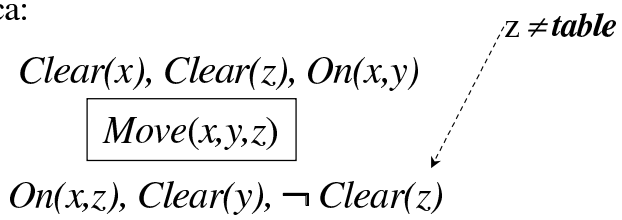
Estado Inicial:
On(B,table)
On(A,table)
On(C,A)
Clear(C)
Clear(B)

Ação: *Move(x,y,z)*

...

goal:
On(A,B) ^ On(B,C)

Mundo dos Blocos: Operador Strips

Ação: *Move(x,y,z)*
Pré-condição: *Clear(x), Clear(z), On(x,y)*
Efeito: *On(x,z), Clear(y),* ¬ *Clear(z)*

Notação gráfica: