Planejamento
Capítulo 11

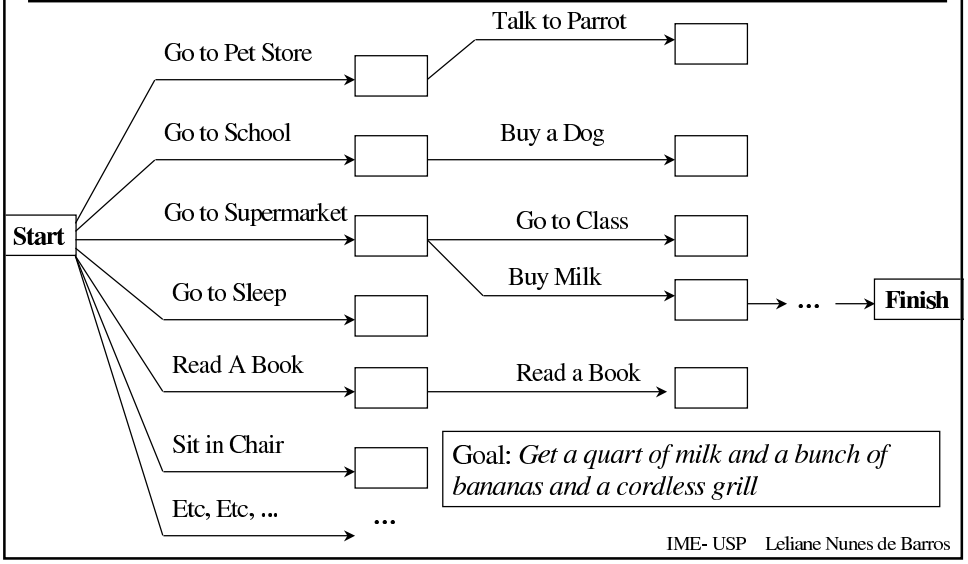
Parte III

Partial Order Regression Planner (revisão)

- Busca em um espaço de planos
- Plano Inicial contém 2 passos: *Start* e *Finish*
- A cada iteração o (i) planejador adiciona passos ou (ii) modifica a ordem de passos e instanciação de variáveis
- *Backtracks* se plano inconsistente
- Sistematicidade alcançada com a proteção de *causal-links*: só adiciona passos que servem para realizar uma precondição ainda não realizada

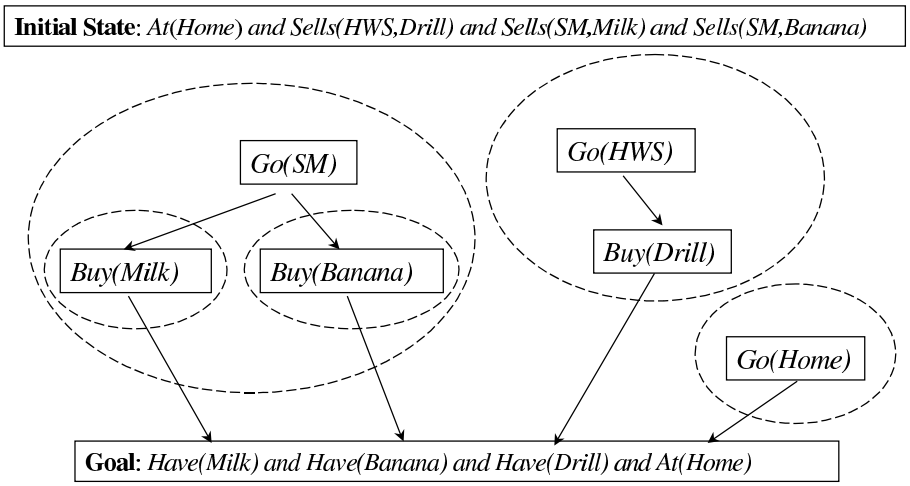
3

Exemplo do Supermercado



4

Exemplo do Supermercado: *divide and conquer*



5

Exemplo do Supermercado: operadores

- Operadores que especificam o problema:

  Op(Action: *Start,* Effect: *At*(Home), *Sells*(*HWS,Drill*), *Sells*(*SM,Milk*), *Sells*(*SM,Banana*))

  Op(Action: *Finish*, Precond: *Have(Drill), Have(Milk), Have(Banana), At*(Home))

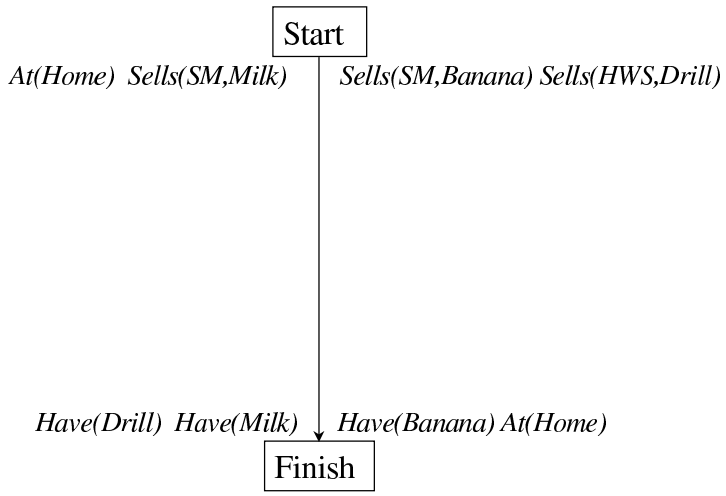
- Operadores que especificam o domínio:

  Op(Action: *Go(there),* Precond: *At*(here), Effect: *At(there),* ¬ *At*(*there*))

  Op(Action: *Buy(x),* Precond: *At*(store), *Sells(store,x),* Effect: *Have*(*x*)))

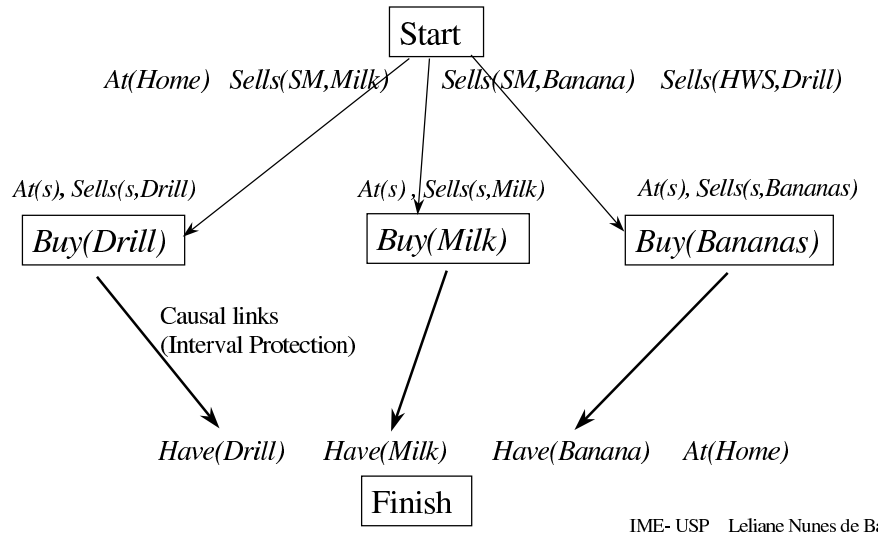
6

Plano Inicial



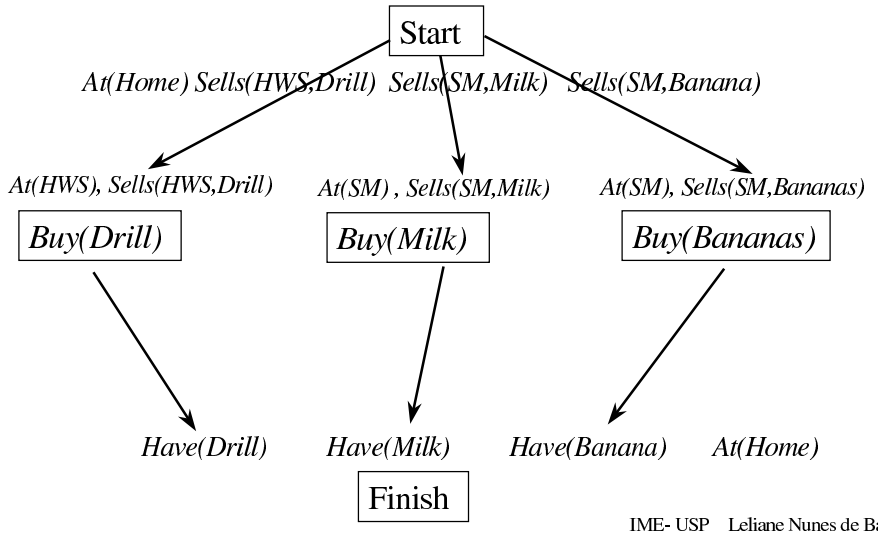
7

Plano Parcial



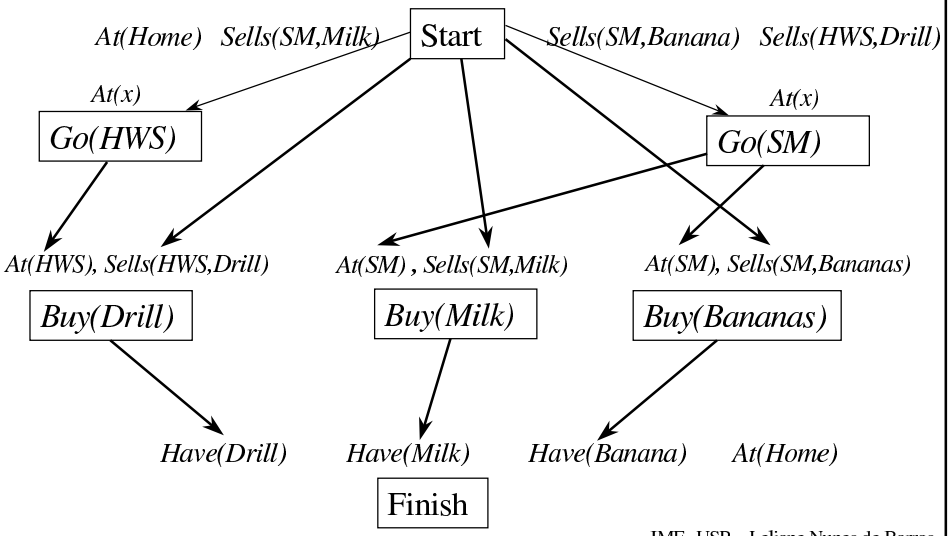
8

Plano Parcial Refinado



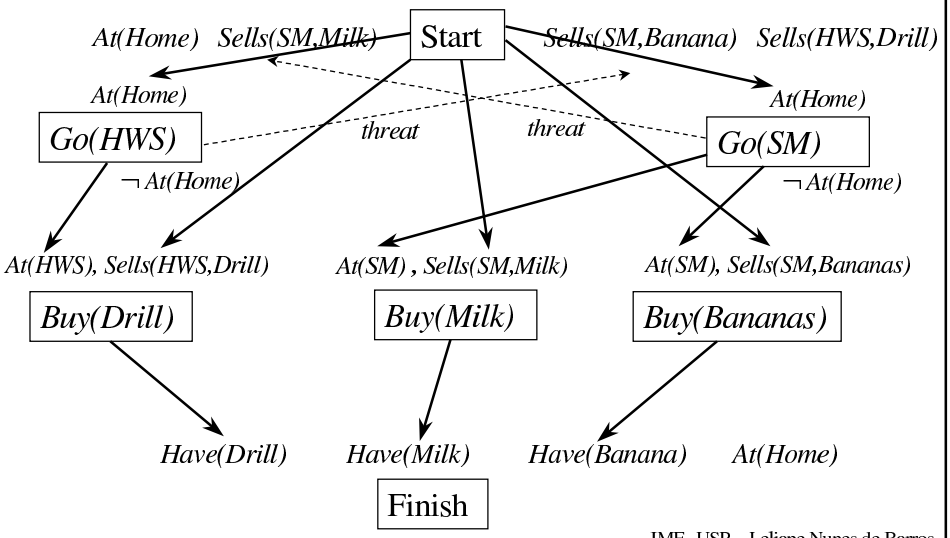
9

Plano Parcial Refinado

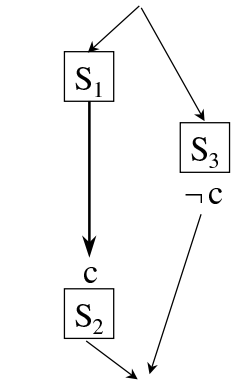


10

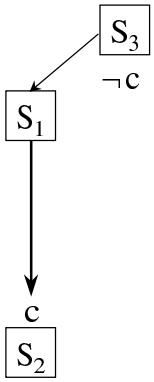
Ações que interagem



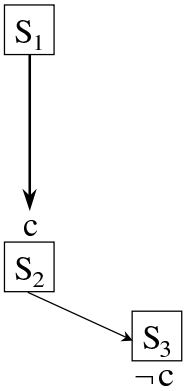
Protegendo causal links



S_3 "ameaça" a condição c



demotion



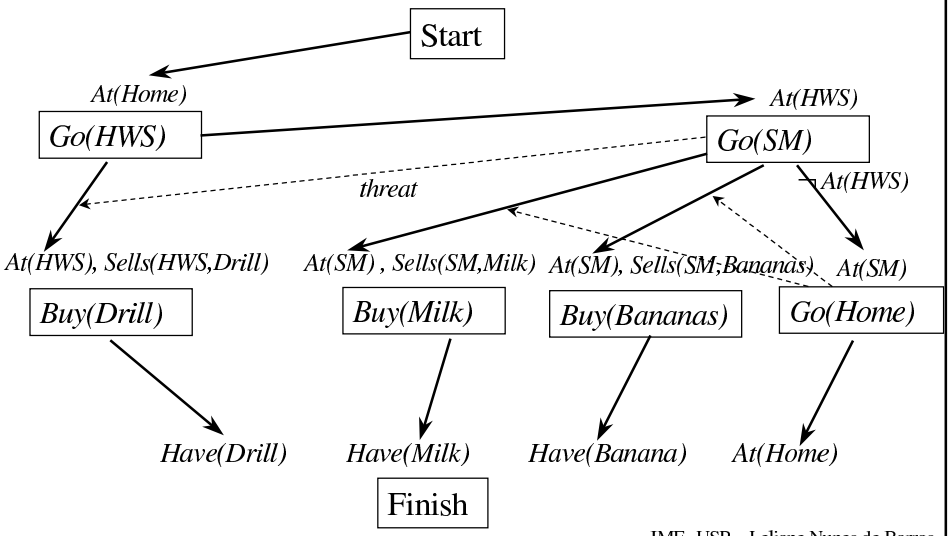
promotion

Resolvendo o *threat*

- O planejador *backtracks* (ou pode modificar o plano) sempre que não for possível resolver um *threat* com *promotion* ou *demotion*
  - escolher uma outra maneira de realizar a pré-condição *At(x)* da ação *Go(SM)*: criar um causal-link de *Go(HWS)* para *Go(SM)*
  - Novo *threat* : *Go(SM) threats* a pré-condição *At(HWS)* da ação *Buy(Drill)* que é protegida por um causal-link => *promotion*

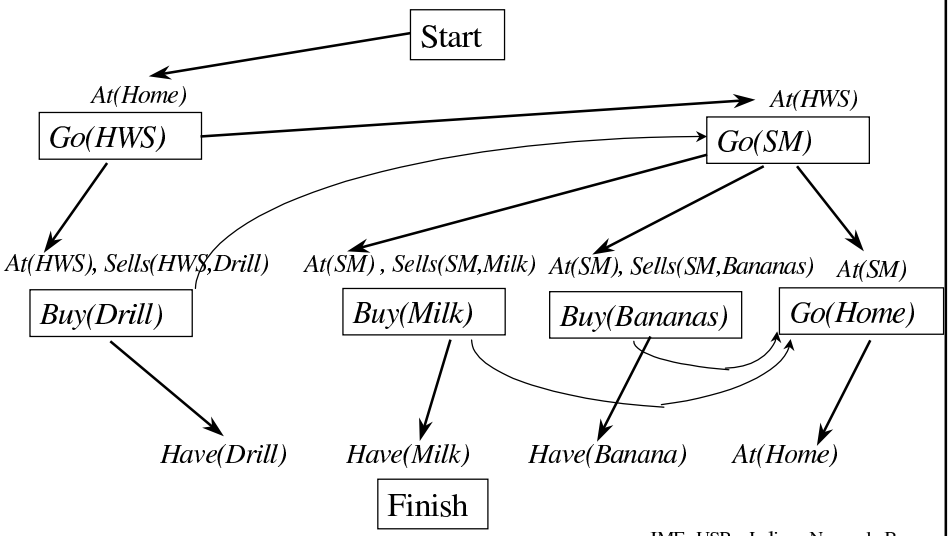
13

Ações que interagem



14

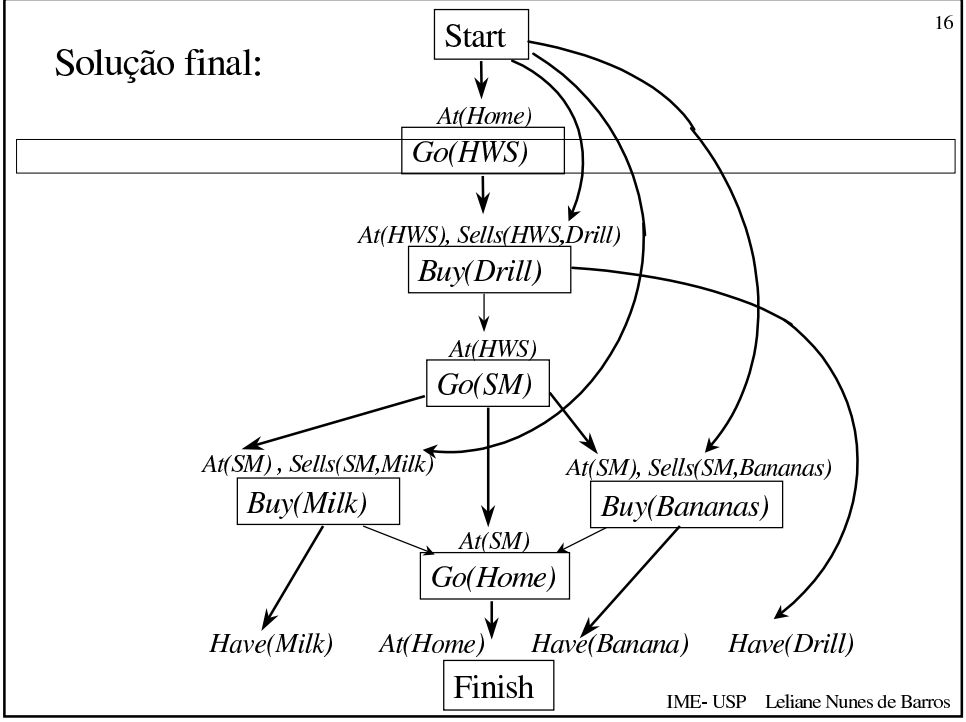
Ações que interagem



Proteção de Causal-link

- O causal-link *Go(HWS)* $\xrightarrow{At(HWS)}$ *Buy(Drill)* é protegido ordenando-se o passo *Go(SM)* depois de *Buy(Drill)*
- O causal-link *Go(SM)* $\xrightarrow{At(SM)}$ *Buy(Milk/Bananas)* é protegido ordenando-se o passo *Go(Home)* depois de *Buy(Milk)* e *Buy(Bananas)*

Solução final:



17

Planejadores

- Permitem reduzir o espaço de busca (raciocinam explicitamente sobre o plano)
- Realizam busca somente onde sub-planos interagem entre si (*least commitment strategy*)
- Causal-links permitem abandonar planos que não levam a solução

18

Análise de domínios

- O desempenho de algoritmos de planejamento depende de características do domínio
- Domínios de aplicação de planejamento podem ser caraterizados através seus sub-goals, que podem ser:
  - independentes
  - serializáveis
    - trivialmente serializáveis
    - laboriozamente serializáveis
  - não-serializáveis

Sub-goals independentes

- Domínios em que uma solução ótima global pode ser encontrada concatenando-se, em qualquer ordem, soluções ótimas individuais de sub-problemas (ideal mas raro).
- A complexidade de problemas cresce linearmente com o número de goals

Ação: A_i
Precondição: $\{I_i\}$
Adição: $\{G_i\}$
Eliminação: $\{\}$

Domínio: D^0S^1

Sub-goals serializáveis

Uma conjunto de sub-goals é *serializável* se existe uma ordem entre eles na qual sub-goals podem ser resolvidos (planejados) sequencialmente sem nunca violarem um sub-goal previamente resolvido.

Ação: A_i
Precondição: $\{I_i\}$
Adição: $\{G_i\}$
Eliminação: $\{I_j \mid j < i\}$

Domínio: D^mS^1

Sub-goals trivialmente serializáveis

Um conjunto de sub-goals é *trivialmente serializável* se os sub-goals podem ser resolvidos (planejados) em qualquer ordem sem nunca violarem um sub-goal previamente resolvido (engloba sub-goals independentes mas não mantém a propriedade de composição).

Ação: A_i

Precondição: $\{I_i\}$

Adição: $\{G_i, I_{i+1}\}$

Eliminação: $\{\}$

Sub-goals laboriozamente serializáveis

Um conjunto de sub-goals é *laboriozamente serializável* se existir pelo menos uma ordem serializável, sendo que pelo menos em 1/n ordens, sub-goals não podem ser resolvidos sem violarem um sub-goal previamente resolvido.

Ação: A_i

Precondição: $\{I_i\}$

Adição: $\{G_i\}$

Eliminação: $\{I_{i-1}\}$

Domínio: D^1S^1

Sub-goals laboriozamente serializáveis

Ação: A_i^1
Precondição: $\{I_i\}$
Adição: $\{P_i\}$
Eliminação: $\{I_j \mid j < i\}$

Ação: A_i^2
Precondição: $\{P_i\}$
Adição: $\{G_i\}$
Eliminação: $\{I_j \mid \forall j\} \cup \{P_j \mid j < i\}$

Domínio: $D^m S^2$

Sub-goals laboriozamente serializáveis

Ação: A_i^1
Precondição: $\{I_i\}$
Adição: $\{P_i\}$
Eliminação: $\{P_j \mid j < i\}$

Domínio: $D^m S^{2*}$

Ação: A_i^2
Precondição: $\{P_i\}$
Adição: $\{G_i\}$
Eliminação: $\{P_j \mid j < i\}$

Ação: A_*
Precondição: $\{I_*\}$
Adição: $\{G_*\}$
Eliminação: $\{I_i \mid \forall i\} \cup \{G_i \mid \forall i\}$

Exercício em sala de aula:

- Utilize o algoritmo POP e o algotitmo TOPI para resolver os seguintes problemas:

Domínio	Problema
D^mS^1	$\{g_2, g_1, g_3\}$
D^1S^1	$\{g_3, g_2, g_1\}$
D^mS^{2*}	$\{g_3, g_2, g_1, g^*\}$

Exemplo

- Expanda os operadores de cada domínio em uma tabela (incluindo Start e Finish)

Passos	Vínculos Causais	Restrições de Ordem	Goals
S_0 : Start		$S_0 < S_\infty$	$< \ldots , S_\infty>$
S_∞ : Finish		$S_0 < S_1$ $S_1 < S_\infty$	
S_1 : …			

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

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

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 - comentários

- POP é correto, completo e sistemático (sem repetição): todo plano que ele retorna é solução e não examina o mesmo plano duas vezes.
- Não é preciso fazer *backtracking* em *Select-subgoal* : resultados de pesquisa mostram que realizar o sub-gol c_1 antes de c_2 ou vice-versa afeta somente a velocidade e não a possibilidade de se encontrar uma solução

POP - variações

- Planejadores com multi-contribuidores: ao invés de tratar ameaças positivas, guarda-as em uma lista de contribuidores.
  - Diminuição de backtrackings e de compromissos de ordem para tratar as ameaças positivas
- MTC - pode ser bom para domínios muito redundantes.

Total-order, Prior Insertion (TOPI)

TOPI (<*S*, *O*, *B*>, *I*, *G*)

(1) Teste de fim de planejamento: Se G⊆ I, sucesso

(2) Seleção de operador: Seja S_{add} o novo passo que adiciona um conjunto de condições *A* tal que $(A \cap G) \neq \emptyset$, que não elimina $g \in G$ e tem um conjunto de precondições *C*. Seja *S'* = *S* ∪{S_{add}}, *O'* = *O* ∪{restrições de ordem que façam S_{add} vir depois de S_0 e antes de todos os demais} (*Backtracking point)*

(3)Atualiza o conjunto de goals: Seja *G'* = (*G* - *A*) ∪ *C*

(4) Chamada recursiva: TOPI(<*S'*, *O'*, *B'*>, *I*, *G'*)