Planejamento de Decomposição Hierárquica

Capítulo 12

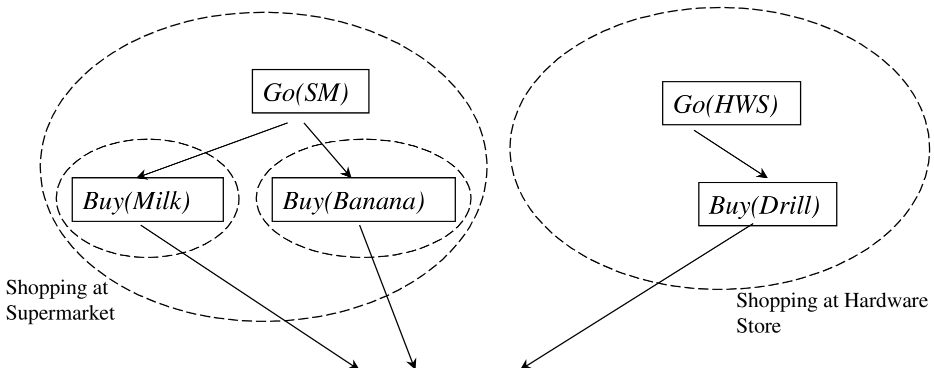
Practical Planners: O-Plan

Similar ao POP mas possue as seguintes características adicionais:

- representa tempo, recursos e planos hierárquicos
- Uso de O-Plan na indústria:
  - escalonamento para Job shop
  - escalonamento para missões espaciais

Composição de ações

Initial State: *At(Home) and Sells(HWS,Drill) and Sells(SM,Milk) and Sells(SM,Banana)*



Goal: *Have(Milk) and Have(Banana) and Have(Drill) and At(Home)*

Diferentes níveis de abstração

Ações de um plano com alto nível de abstração:

Shopping_at(SPM), Shopping_at(HWS),

Ações de um plano com baixo nível de abstração (instruções enviadas diretamente aos atuadores do agente):

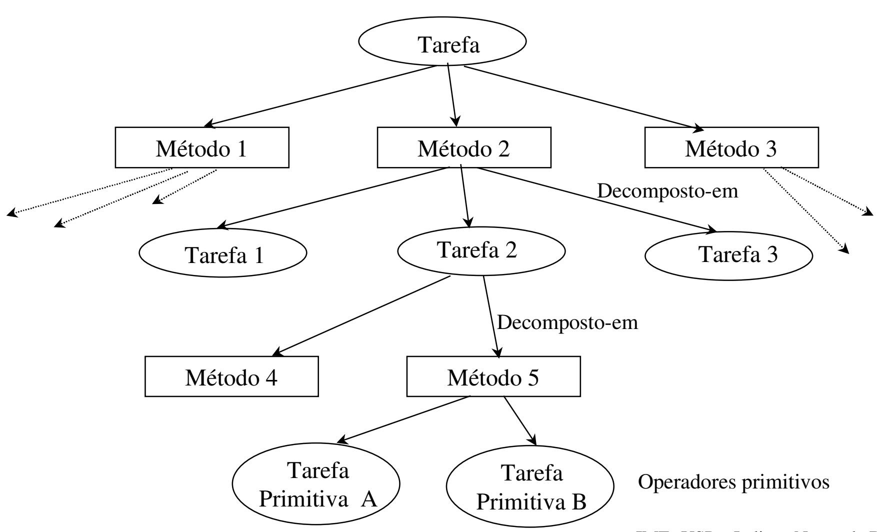
Forward(1cm), Turn(1degree), Forward(1cm), …

Problema:
plano com milhares de passos $\rightarrow$ espaço de planos muito grande

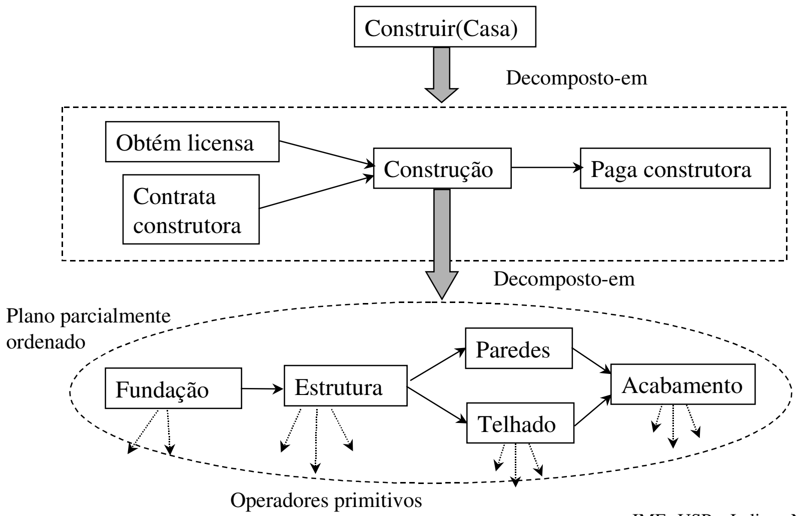
Decomposição Hierárquica

- Uma operador abstrato pode ser decomposto em um grupo de passos que formam um *sub-plano* que implementam o operador
- Decomposições são armazenadas em uma biblioteca de planos (também chamados de métodos de decomposição)
- Exemplo: agente construtor

Decomposição de tarefa



Decomposição Hierárquica



Decomposição Hierárquica

- Operadores podem ser decompostos de várias maneiras diferentes (métodos alternativos)
- Uma decomposição de um operador pode ser um plano de ordem total ou ordem parcial
- Operador primitivo: pode ser realizado diretamente pelo agente

Planejamento Hierárquico e POP

- Planos podem ser gerados e armazenados para serem re-usados em problemas futuros
- Operadores ainda devem ser introduzidos no plano para satisfazer condições antes de serem decompostos
- A decomposição introduz (pouca) interação entre as diferentes partes do plano

Planejamento Hierárquico

- Modificações no POP
  - Estender a linguagem Strips para permitir a descrição de operadores não primitivos (decomponíveis)
  - Modificar o algoritmo de planejamento para permitir a introdução de operadores não primitivos no plano com suas decomposições
  - Mudanças na especificação de goals (maior poder de expressão): uma ação composta pode ser um goal

Extensão da linguagem Strips

- Operadores podem ser primitivos ou não-primitivos
  - depende da capacitação do agente ou se a tarefa será delegada para outro agente (planejamento com multi-agentes)
- Cada método de decomposição é uma expressão do tipo: *Decompose(o, p)*
  - um operador não-primitivo que unifica com *o (que também deve descrito como um operador do tipo Strips)* pode ser decomposto no plano *p.*

Método

Decompose(Construct,

*Plan(*Steps: {S_1: *Build(Foundation)*, S_2: *Build(Frame)*, S_3: *Build(Roof)*, S_4: *Build (Walls)*, S_5: *Build(Interior)*}

Orderings: {$S_1 \prec S_2 \prec S_3 \prec S_5$, $S_2 \prec S_4 \prec S_5$},

Bindings: { },

Links:{$S_1 \xrightarrow{Foundation} S_2$, $S_2 \xrightarrow{Frame} S_3$, $S_2 \xrightarrow{Frame} S_4$, $S_3 \xrightarrow{Roof} S_5$, $S_4 \xrightarrow{Walls} S_5$}

))

Op(*Action: Construct,*

Preconditions: *Licensa, Contrato*

Effects: *Construction*

)

Decomposição correta

- Um plano *p* implementa um operador *o* se:
  - *p* é consistente (não existem contradições nas restrições de ordenação ou *binding* de *p*)
  - todo efeito de *o* é efeito de pelo menos um passo de *p* e não é negado por algum passo depois dele)
  - toda pré-condição dos passos em *p* deve ser realizada por um passo em p ou ser uma das pré-condições de *o*

HD-POP

function HD-POP(*plan, operators, methods)* **returns** *plan*

inputs: *plan,* a plan with start and goal steps (or other steps)

loop do

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

S_{need} , c ← Select-Subgoal(*plan*)

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

$S_{nonprimitive}$, c ← Select-Nonprimitive(*plan*)

Choose-Decomposition(*plan, methods,* $S_{nonprimitive}$)

Resolve-Threats(*plan)*

end

POP → HD-POP

- Solution?(*plan*): deve-se checar se todo passo do plano é primitivo
- Funções novas:

  Select-Nonprimitive → seleciona arbitrariamente um passo não primitivo para ser decomposto (ponto de escolha)

  Choose-Decomposition → seleciona um método de decomposição e aplica ao plano (ponto de escolha)
- As funções restantes de POP continuam iguais

Expansão do plano

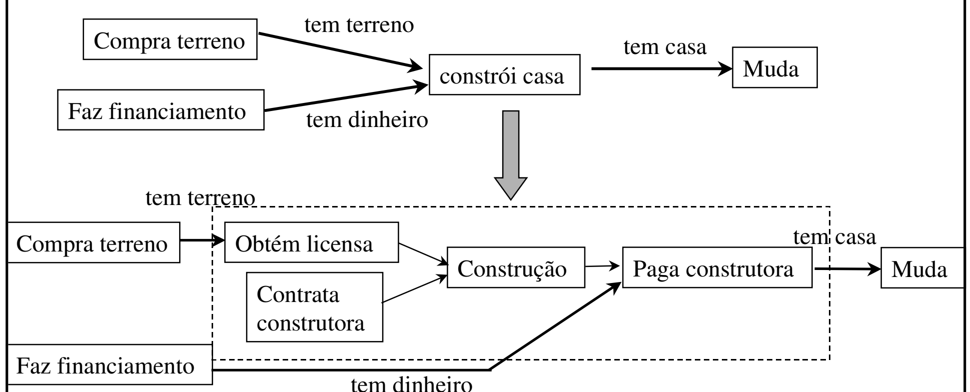
Se *method* foi escolhido para decompor o passo $S_{nonprimitive}$, então o plano corrente deve ser alterado da seguinte maneira:

- **Steps:** adicionar todos os passos de *method* removendo $S_{nonprimitive}$
- **Bindings:** adicionar todas as restrições de bindings de *method*. Falha se introduzir inconsistência

Expansão do plano

- **Orderings:** segue a estratégia de *least commitment:* (1) cada restrição de ordem do tipo $S_a \prec S_{nonprimitive}$ deve ser substituida por uma ou mais restrições que ordenam S_a antes do(s) último(s) passo(s) de *method* e (2) para cada restrição do tipo $S_{nonprimitive} \prec S_z$ deve ser substituida por uma ou mais restrições que ordenam S_z depois do(s) primeiros(s) passo(s) de *method* .
- **Causal Links**: se $S_i \xrightarrow{c} S_{nonprimitive}$ era um *CL* no plano, substitua-o por um conjunto de *CLs,* $S_i \xrightarrow{c} S_m$,onde S_m é um passo de *method* que tem *c* como pré-condição. E para $S_{nonprimitive} \xrightarrow{c} S_j$ substitua por um conjunto de *CLs* $S_i \xrightarrow{c} S_m$, *o*nde S_m é um passo de *method* que tem *c* como efeito.

Exemplo

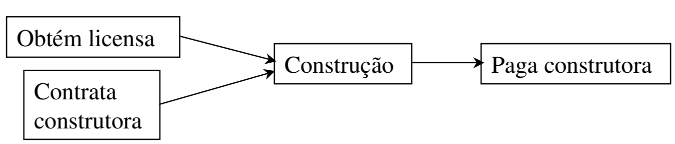


Decomposição Hierárquica

- analogia com subrotinas e macros (algoritmo <=> plano).
- permite especificar o problema em partes (de tamanho razoável) para serem combinadas e formarem planos complexos
- diminue a complexidade do problema = menor custo para a construção de planos complexos

Complexidade de um planejador hierárquico

- solução abstrata

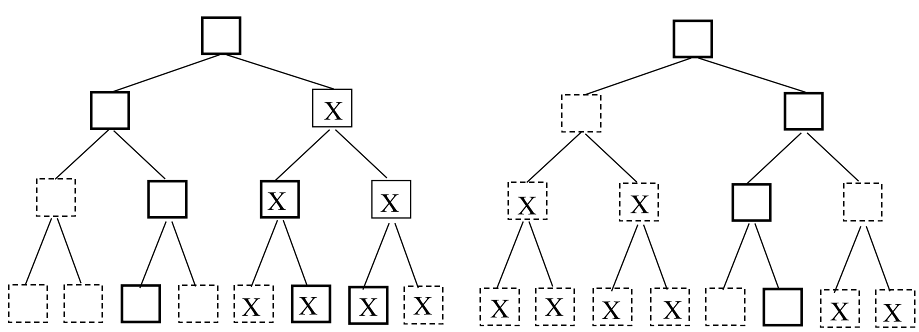


- Planejadores HTN devem ser capazes de encontrar uma solução abstrata (completa e consistente) para depois chegar até as suas primitivas sem muito *backtracking*

Propriedades desejadas de um planejador hierárquico

- *downward solution property*: uma vez que uma solução abstrata é encontrada pode-se cortar (prune) todos os outros planos abstratos da árvore de busca
- *upward solution property*: se um plano abstrato é inconsistente então não existe solução primitiva para ele e podemos podar todos os seus descendentes

Propriedades de corte em busca



downward solution property

upward solution property

possible abstract solutions

inconsistent plans

X pruned plans

Exercício

- Explicar porque o exemplo da Figura 12.6 é um exemplo de planejamento hierárquico que viola a *upward solution property*.
- Data de entrega: próxima aula depois da prova

HDP: como compartilhar passos de métodos ?

É preciso estender o algoritmo de Decomposição Hierárquica para compartilhar passos de métodos (para torná-lo completo)

Exemplo:

Goal: enjoy_a_honeymoon and raise_a_baby

Method(enjoy_a_honeymoon): get_married and go_on_honeymoon

Method(raise_a_baby): get_married and have_a_baby

Se a precondição de *get_married* for *not_married* e ~divorce, então os passos devem ser compartilhados

Estendendo Choose-Decomposition

- Compartilhamento de passos: modificação em *Choose-Decomposition* com mais um ponto de escolha: utilizar um passo já existente no plano ou adicionar um passo novo da decomposição
  - antes de expandir um *get-married*, tentar utilizar o outro para satisfazer a condição *married.*
- Outra maneira: através de críticas. Conhecimento explícito sobre planos que pode ser usado para detectar ações redundantes.

Planejamento hierárquico aproximado

- Ações com níveis de criticalidade: condições com números baixos são consideradas mais críticas.

Op*(Action: Buy(x),*

Effects: *Have(x)* ^ ¬ *Have(Money),*

Preconditions: 1: *Sells(store,x)* ^

2: *At(store)* ^

3: *Have(Money))*

- O planejador Abstrips propõe Hierarquias de “abstrações” chamadas no livro AIMA de *Hierarquias de Aproximação*

Planejamento hierárquico aproximado

- Mais fácil encontrar uma solução “abstrata”, ignorando certos detalhes
  - se a loja vende um certo produto é mais importante do que planejar como ir até a loja.
- Maneiras de se estender o HTN garantindo a satisfação da *upward solution property*:

  solução 1: decomposições de operadores que adicionam as pre-condições de níves de criticalidade maior, ou

  solução 2: adicionar uma heurística que faz com o algoritmo faça backtracking nos pontos de escolha que envolvam pré-condições de nível mais baixo de criticalidade

Planejamento hierárquico aproximado: Solução 1

Op(*Action: Buy1(x),*
Effects: *Have(x)* ^ ¬ *Have(Money),*
Preconditions: 1: *Sells(store,x)* ^

Op(*Action: Buy2(x),*
Effects: *Have(x)* ^ ¬ *Have(Money),*
Preconditions: 2: *At(store)* ^

Op(*Action: Buy3(x),*
Effects: *Have(x)* ^ ¬ *Have(Money),*
Preconditions: 3: *Have(Money))*

Buy1 é resolvido e depois espandido em *Buy2* que por sua vez é resolvido e espandido em *Buy3*

Planejamento hierárquico aproximado

- Os níveis de criticalidade correspondem a uma informação de controle que guiam a busca de planejamento
- Problemas:
  - a especificação de níveis de criticalidade nos operadores dependem do domínio (informações devem ser extraídas do especialista no domínio)
  - a especificação de níveis de criticalidade nos operadores dependem da situação (ter dinheiro pode ser mais crítico do que saber se é a loja certa). Uma possível solução é substituir os níveis de criticalidade com uma descrição da probabilidade do operador ser bem sucedido de acordo nas diferentes situações

Discussão

- Diferença entre Planejamento de Decomposição Hierárquica e Planejamento Hierarquico Aproximado