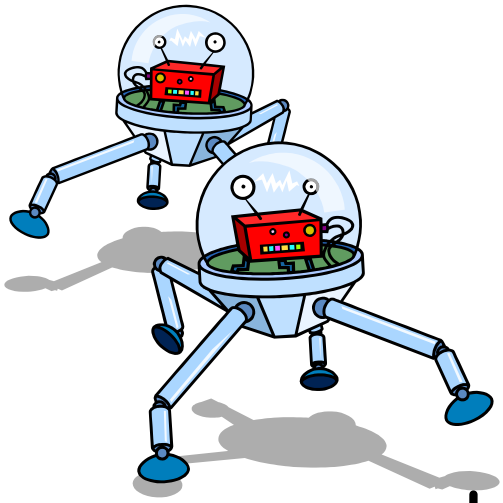
Planejamento em Inteligência Artificial



Algoritmos de Planejamento Clássico

Leliane Nunes de Barros
leliane@ime.usp.br

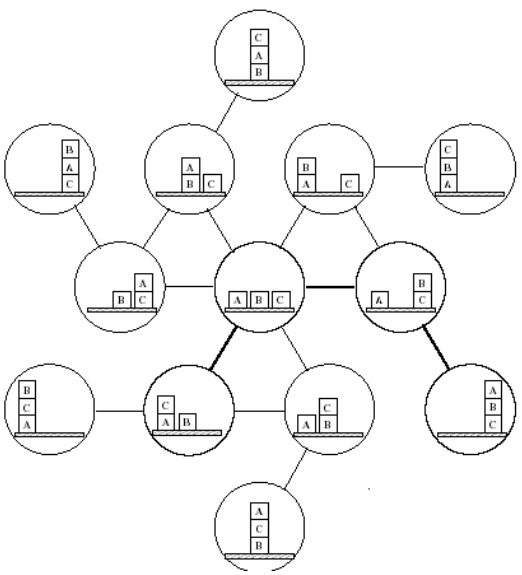
Como resolver problemas de planejamento clássico?

- Algoritmo STRIPS (70s):
- Busca no espaço de estados: planejamento progressivo e regressivo
- Busca no espaço de planos: Planejamento de Ordem Parcial (80's) (UCPOP, 1992)
- Busca no grafo de planejamento: Graphplan (1995)
- Busca no espaço de interpretações lógicas: SatPlan (1996)
- Busca heurística: planejamento como uma busca heurística no espaço de estados (1996 ...)
- Planejamento como Verificação de Modelo (1998)

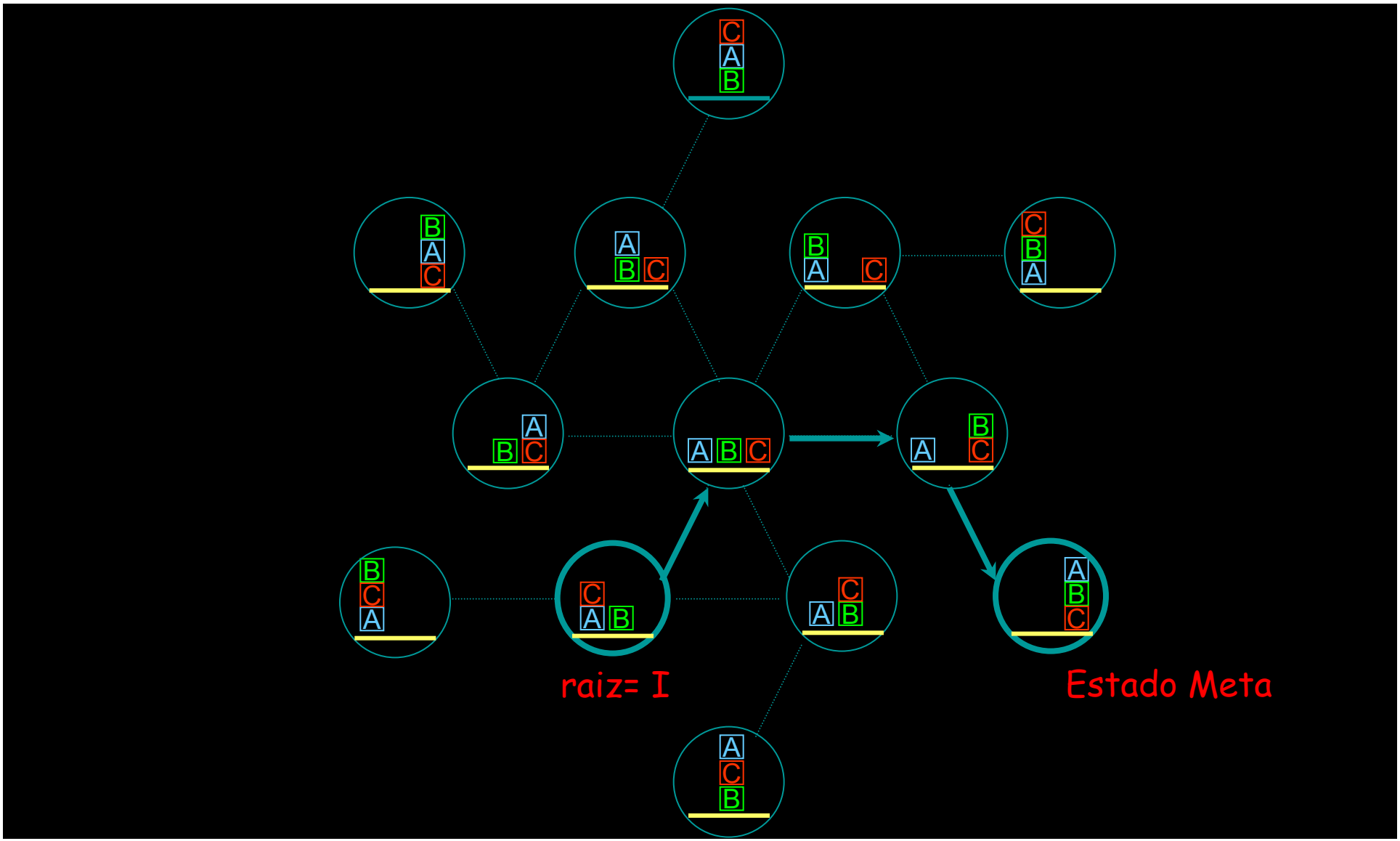
Planejamento como Busca

- Conceitos básicos
- Nó raiz
- Nós sucessores
- Caminho
- Teste de meta

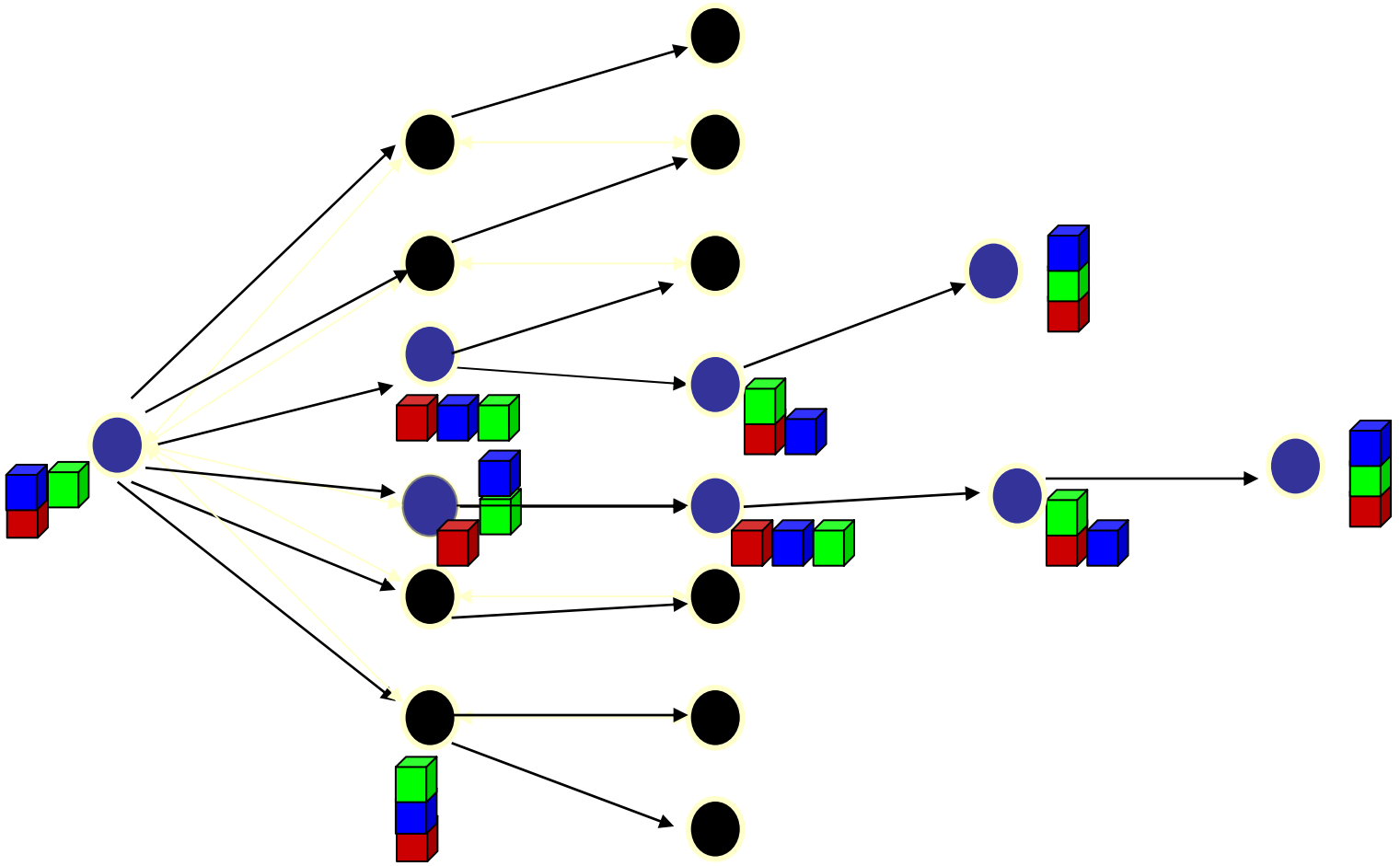
Planejamento como busca no no espaço de estados



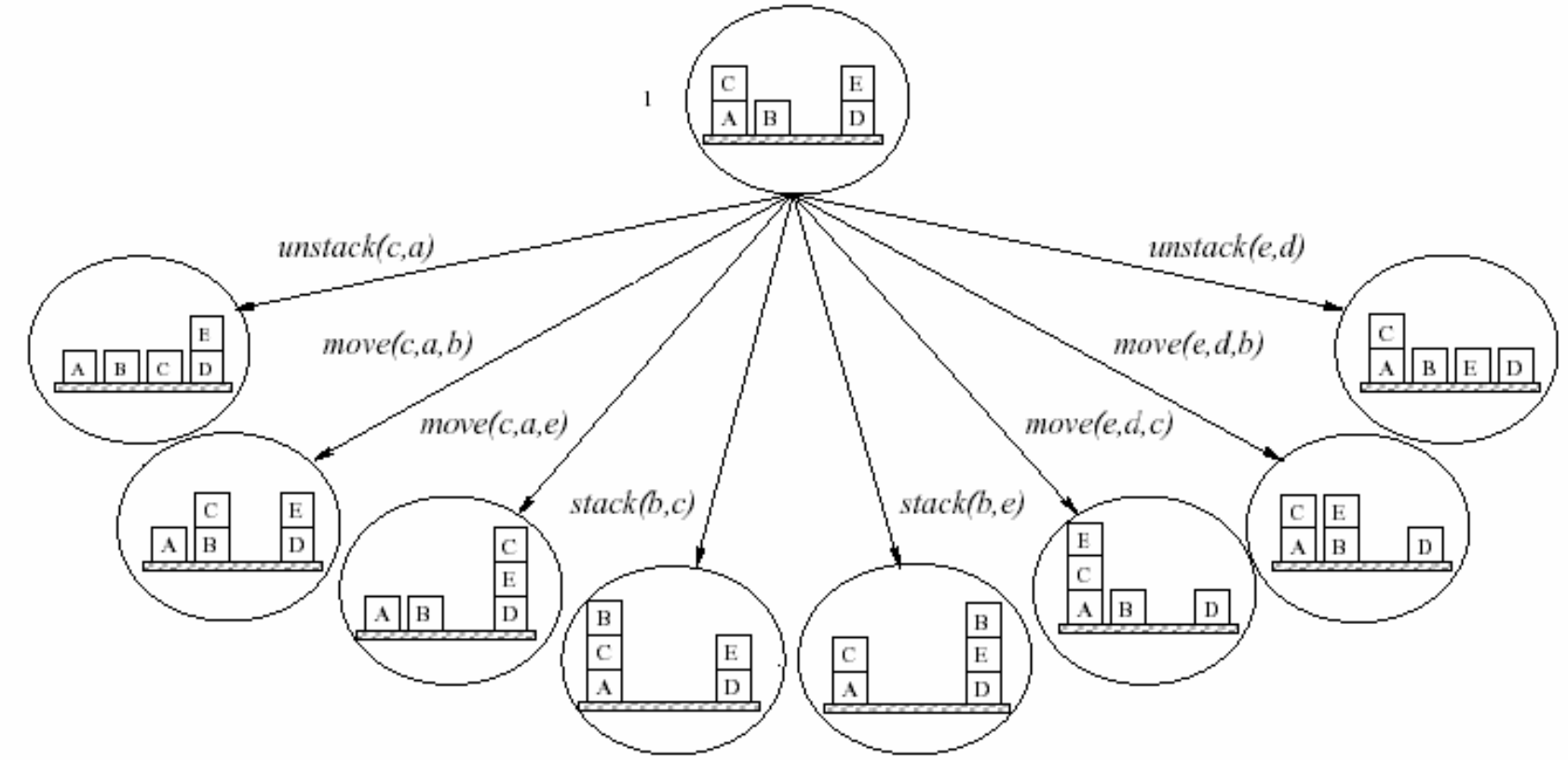
Planejamento Progressivo no Espaço de Estados



Planejamento Progressivo



Geração de Nós Sucessores



Todas as ***ações aplicáveis*** são usadas para gerar sucessores de s

Representação implícita: todas as transições mas não todos os estados

Uma linguagem proposicional

recordando

Problema de planejamento é dada por $\langle P, A, I, G \rangle$

- P: conjunto de proposições (literais constantes)
- $I \subseteq P$: estado inicial (subconjunto de P)
- $G \subseteq P$: conjunto de proposições da descrição da meta
- A: conjunto de ações
  - $pre(a) \subseteq P$: precondições
  - $eff+(a) \subseteq P$: efeitos positivos
  - $eff-(a) \subseteq P$: efeitos negativos

Representação implícita: todas as transições mas não todos os estados

Uma linguagem proposicional

recordando

O estado s é um subconjunto de P

- s={p | p ∈ P}, fazendo a suposição de mundo fechado (CWA)

Algumas propriedades da semântica de ações:

- A ação a é aplicável em s, appl(a, s) : pre(a) ⊆ s
- $s' = \gamma(a_n, \ldots \gamma(a_2, \gamma(a1, s)) \ldots) = \gamma(\{a_1, a_2, \ldots, a_n\}, s)$
- Se $\gamma(\{a_1, a_2, \ldots, a_n\}, I) \supseteq G$, então $\{a_1, a_2, \ldots, a_n\}$ é um plano solução

A geração de estados sucessores depende das estratégias de busca empregadas

Planejamento Progressivo

- Ações aplicáveis em um estado s

$$a_i \in \{a \mid a \in A \wedge pre(a) \subseteq s\}$$

- Cálculo do estado sucessor $s' = \gamma(a, s)$

$$s' = s \cup eff+(a_i) \setminus eff-(a_i)$$

- Teste de meta para um estado s

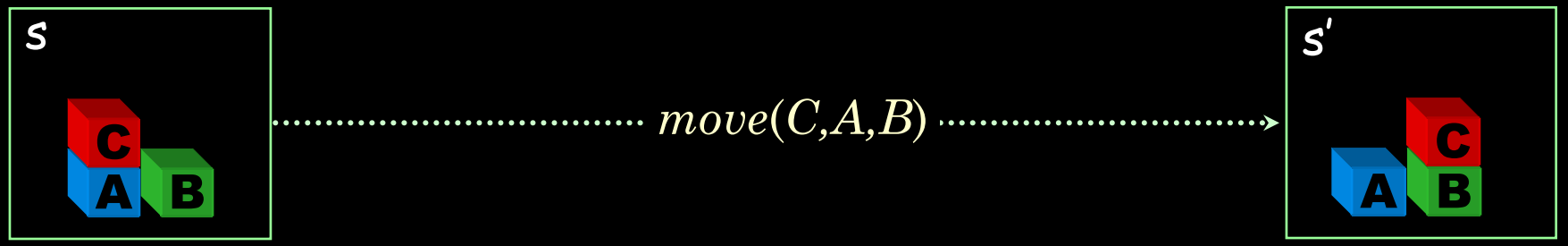
$$G \subseteq s$$

- A concatenação de uma ação selecionada é feita no fim do plano π

$$\pi \circ a_i$$

Geração de nós sucessores na busca progressiva (recordação)

action: (*move*(C,A,B),
pre: { *limpo*(C), *limpo*(B), *sobre*(C,A) },
eff+: { *limpo*(A), *sobre*(C,B)},
eff–: {*limpo*(B), *sobre*(C,A)})

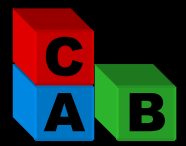


limpo(B)
limpo(C)
sobre (A,mesa)
Sobre(B,mesa)
sobre(C,A)

Geração de nós sucessores na busca progressiva (recordação)

action: (*move*(C,A,B),
pre: { *limpo*(C), *limpo*(B), *sobre*(C,A) },
eff+: { *limpo*(A), *sobre*(C,B)},
eff–: {*limpo*(B), *sobre*(C,A)})

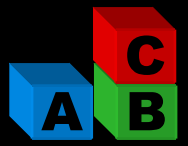
s



s' = s ∪ eff+(a) \ eff-(a)

move(C,A,B)

s'



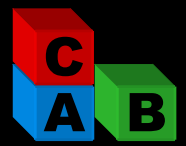
limpo(B)
limpo(C)
sobre (A,mesa)
Sobre(B,mesa)
sobre(C,A)

limpo(B)
limpo(C)
sobre(A,mesa)
sobre(B,mesa)
sobre(C,A)

Geração de nós sucessores na busca progressiva (recordação)

action: (*move*(C,A,B),
pre: { *limpo*(C), *limpo*(B), *sobre*(C,A) },
eff+: { *limpo*(A), *sobre*(C,B)},
eff–: {*limpo*(B), *sobre*(C,A)})

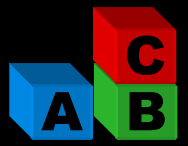
s



s' = s ∪ eff+(a) \ eff-(a)

move(C,A,B)

s'



limpo(B)
limpo(C)
sobre (A,mesa)
Sobre(B,mesa)
sobre(C,A)

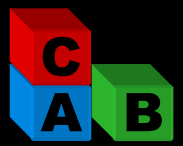
limpo(B)
limpo(C)
sobre(A,mesa)
sobre(B,mesa)
sobre(C,A)

limpo(B)
limpo(C)
sobre(A,mesa)
sobre(B,mesa)
sobre(C,A)
limpo(A)
sobre(C,B)

Geração de nós sucessores na busca progressiva (recordação)

action: (*move*(C,A,B),
pre: { *limpo*(C), *limpo*(B), *sobre*(C,A) },
eff+: { *limpo*(A), *sobre*(C,B)},
eff–: {*limpo*(B), *sobre*(C,A)})

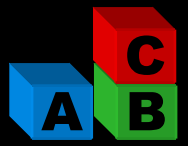
s



s' = s ∪ eff+(a) \ eff-(a)

move(C,A,B)

s'



limpo(B)
limpo(C)
sobre (A,mesa)
Sobre(B,mesa)
sobre(C,A)

limpo(B)
limpo(C)
sobre(A,mesa)
sobre(B,mesa)
sobre(C,A)

limpo(B)
limpo(C)
sobre(A,mesa)
sobre(B,mesa)
sobre(C,A)
limpo(A)
sobre(C,B)

~~*limpo(B)*~~
limpo(C)
sobre(A,mesa)
sobre(B,mesa)
~~*sobre(C,A)*~~
limpo(A)
sobre(C,B)

Planejamento Progressivo: algoritmo

Função PROG(A,s,G, π) devolve Falha ou um plano completo π

Entrada:

o conjunto de ações A

o estado atual s // inicializa com o Estado Inicial

a descrição da meta G

um plano parcialmente especificado π

se $G \subseteq s$ então devolva π

selecione $a_i \in \{a \mid a \in A \wedge pre(a) \subseteq s\}$ // pto. de retrocesso

$s' = s \cup eff+(a_i) \setminus eff-(a_i)$ // cálculo de $\gamma(s, a_i)$

$\pi' = PROG(A, s', G, \pi o a_i)$

se $\pi' \neq$ Falha então devolva π'

retroceda

devolva Falha

Planejamento Progressivo: algoritmo

Função PROG(A,s,G, π) devolve Falha ou um plano completo π

Entrada:

o conjunto de ações A

o estado atual s // inicializa com o Estado Inicial

a descrição da meta G

um plano parcialmente especificado π

se $G \subseteq s$ então devolva π

selecione $a_i \in \{a \mid a \in A \wedge pre(a) \subseteq s\}$ // ordem de acesso

$s' = s \cup eff+(a_i) \setminus eff-(a_i)$

$\pi' = PROG(A, s', G, \pi o a_i)$

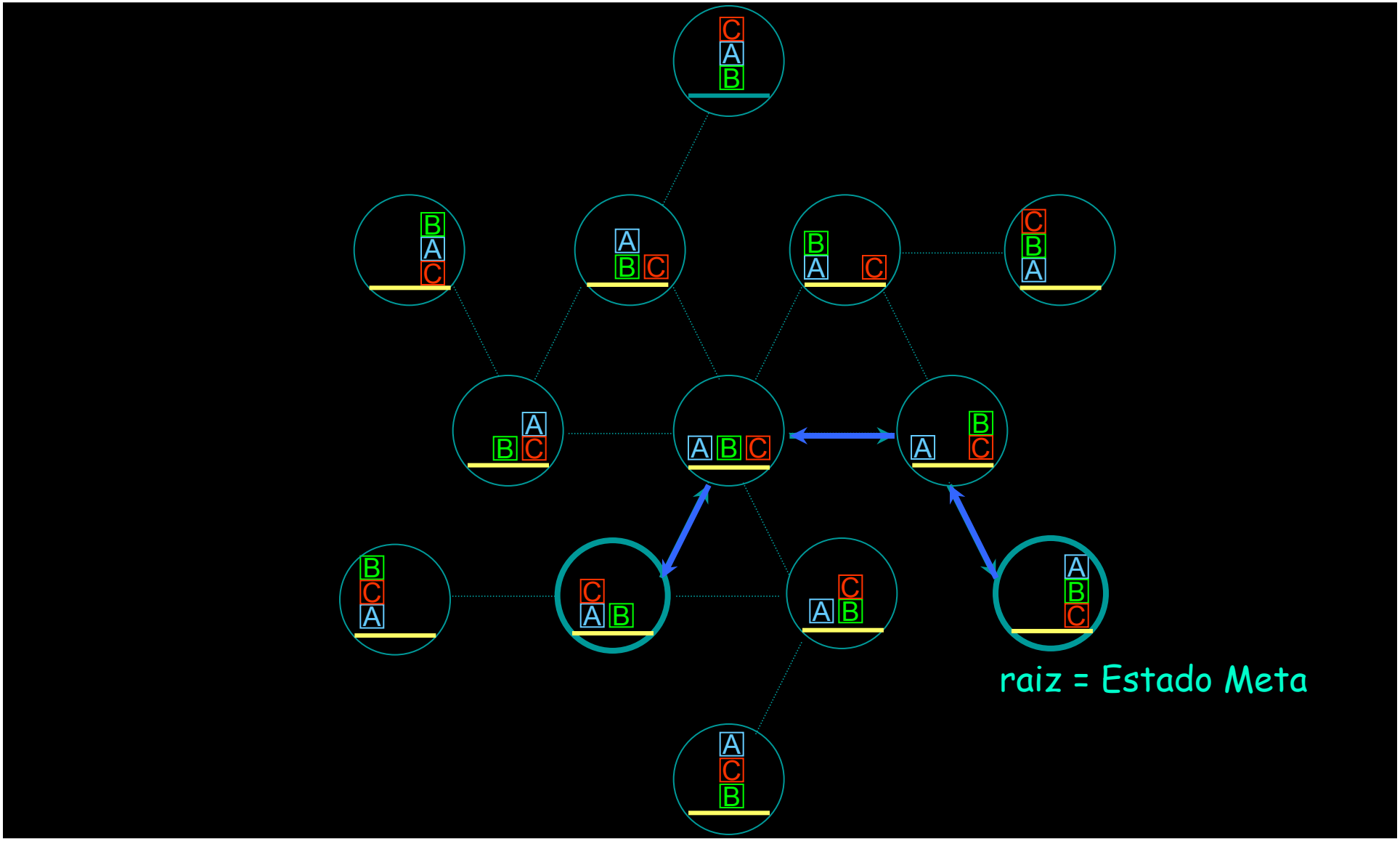
se $\pi' \neq$ Falha então devolva π'

retroceda

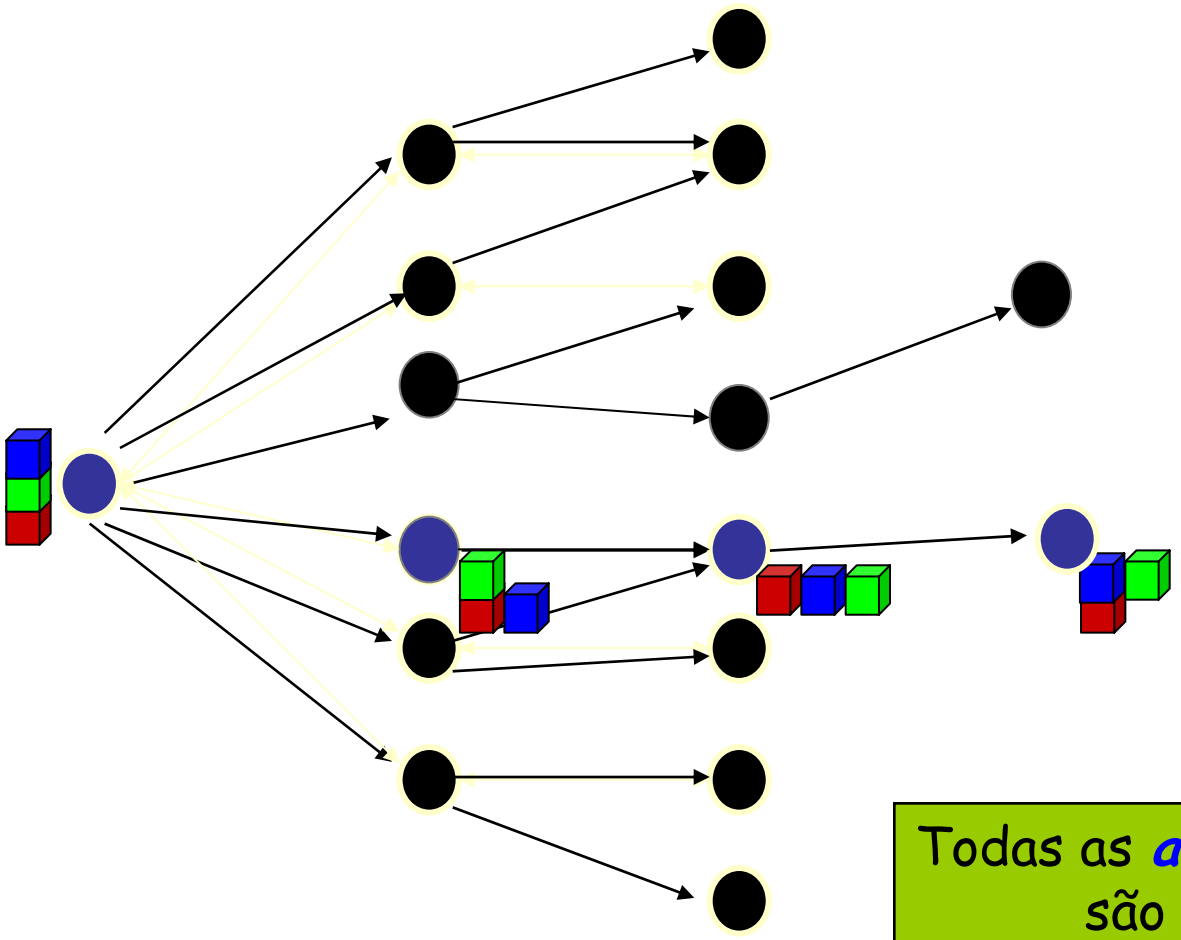
devolva Falha

busca em profundidade: outras estratégias de busca também podem ser usadas

Planejamento Regressivo no Espaço de Estados



Planejamento Regressivo



Todas as ***ações relevantes*** são aplicadas

Planejamento Regressivo: estados parciais

- Uma vez que na linguagem adotada o Estado Inicial I é completo mas a **Meta G é parcial**, isso significa que teremos muitos Estados Meta completos para considerarmos como **raiz** da busca regressiva:
  - **Busca paralela** a partir de todos os estados meta e escolher a solução ótima (o menor plano) ou
  - Busca com **conjunto de estados**

- Para calcular o estado sucessor de *s* para cada ação, é preciso aplicar a função de transição inversa

$$s' = \gamma^{-1}(a, s)$$

Planejamento Regressivo

- Ações relevantes em um estado s

$a_i \in \{a \mid a \in A \ \wedge \ eff+(a) \cap s \neq \emptyset \ \wedge \ eff-(a) \cap s = \emptyset\}$

- Cálculo do estado sucessor $s' = \gamma^{-1}(a, s)$

$$s' = s \cup pre(a) \cup eff-(a) \setminus eff+(a)$$

- Teste de meta para um estado s

$$I = s$$

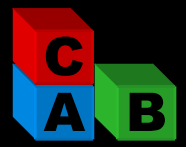
- A concatenação de uma ação selecionada é feita no início do plano π

$$a_i \, o \, \pi$$

Geração de nós sucessores na busca regressiva

ação: (*move*(*C*,A,B),
pre: { *limpo*(C), *limpo*(B), *sobre*(C,A) },
eff+: { *limpo*(A), *sobre*(C,B)},
eff–: {*limpo*(B), *sobre*(C,A)})

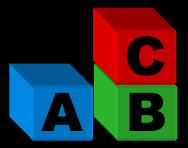
s'



s' = γ^{-1}(a, s) = s ∪ pre(a) ∪ eff–(a) \ eff+(a)

◄····· *move*(*C*,*A*,*B*) ·····

s

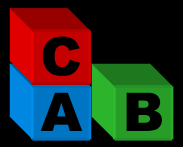


limpo(C)
sobre(A,mesa)
sobre(B,mesa)
limpo(A)
sobre(C,B)

Geração de nós sucessores na busca regressiva

ação: (*move*(*C*,A,B),
pre: { *limpo*(C), *limpo*(B), *sobre*(C,A) },
eff+: { *limpo*(A), *sobre*(C,B)},
eff–: {*limpo*(B), *sobre*(C,A)})

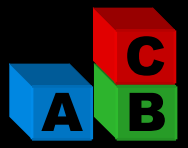
s'



$s' = \gamma^{-1}(a, s) = s \cup pre(a) \cup eff–(a) \setminus eff+(a)$

move(*C*,*A*,*B*)

s



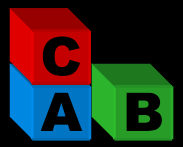
limpo(C)
sobre(A,mesa)
sobre(B,mesa)
limpo(A)
sobre(C,B)
limpo(B)
limpo(C)
sobre(C,A)

limpo(C)
sobre(A,mesa)
sobre(B,mesa)
limpo(A)
sobre(C,B)

Geração de nós sucessores na busca regressiva

ação: (*move*(*C*,A,B),
pre: { *limpo*(C), *limpo*(B), *sobre*(C,A) },
eff+: { *limpo*(A), *sobre*(C,B)},
eff–: {*limpo*(B), *sobre*(C,A)})

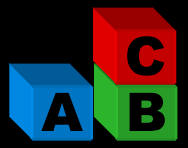
s'



$s' = \gamma^{-1}(a, s) = s \cup pre(a) \cup eff-(a) \setminus eff+(a)$

move(*C*,*A*,*B*)

s

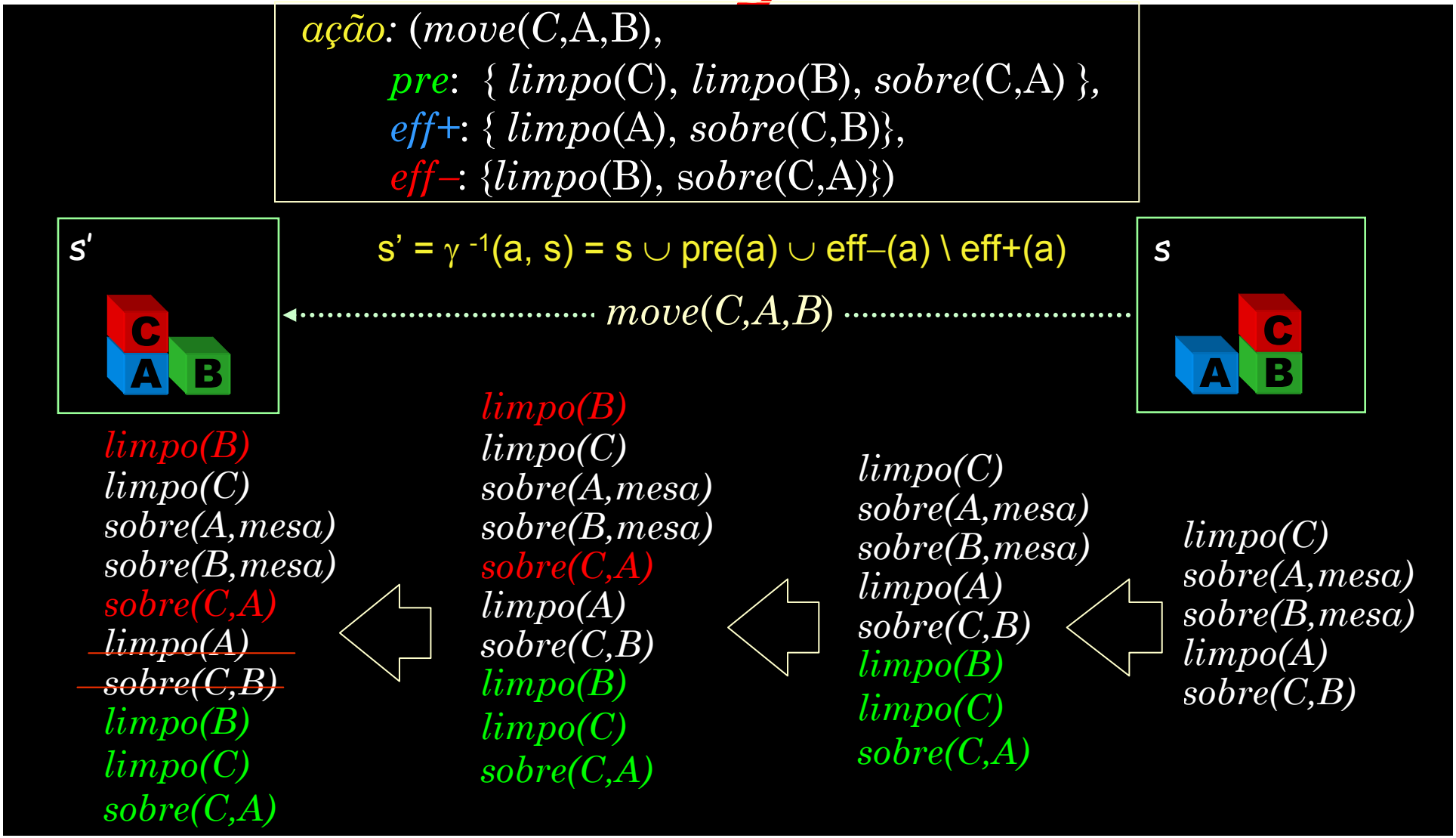


limpo(B)
limpo(C)
sobre(A,mesa)
sobre(B,mesa)
sobre(C,A)
limpo(A)
sobre(C,B)
limpo(B)
limpo(C)
sobre(C,A)

limpo(C)
sobre(A,mesa)
sobre(B,mesa)
limpo(A)
sobre(C,B)
limpo(B)
limpo(C)
sobre(C,A)

limpo(C)
sobre(A,mesa)
sobre(B,mesa)
limpo(A)
sobre(C,B)

Geração de nós sucessores na busca regressiva

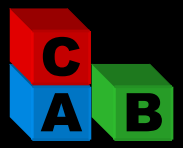


Geração de nós sucessores na busca regressiva

ação: (*move*(*C*,A,B),
pre: { *limpo*(C), *limpo*(B), *sobre*(C,A) },
eff+: { *limpo*(A), *sobre*(C,B)},
eff–: {*limpo*(B), *sobre*(C,A)})

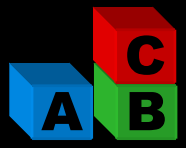
s' = γ^{-1}(a, s) = s $\cup$ pre(a) $\cup$ eff–(a) \ eff+(a)

s'



move(*C*,*A*,*B*)

s



sobre(A,mesa)
sobre(B,mesa)
sobre(C,A)
limpo(B)
limpo(C)

limpo(B)
limpo(C)
sobre(A,mesa)
sobre(B,mesa)
sobre(C,A)
limpo(A)
sobre(C,B)
limpo(B)
limpo(C)
sobre(C,A)

limpo(C)
sobre(A,mesa)
sobre(B,mesa)
limpo(A)
sobre(C,B)
limpo(B)
limpo(C)
sobre(C,A)

limpo(C)
sobre(A,mesa)
sobre(B,mesa)
limpo(A)
sobre(C,B)

Planejamento Regressivo: algoritmo

Função REGR(A,s,I,π) devolve Falha ou um plano completo π

Entrada:

o conjunto de ações A

o estado atual s // inicializa com um estado meta (completo)

a descrição do Estado Inicial I

um plano parcialmente especificado π

se $I \subseteq s$ então devolva π

selecione $a_i \in \{a \mid a \in A \wedge eff^+(a) \cap s \neq \emptyset \wedge eff^-(a) \cap s = \emptyset\}$

$s' = s \cup pre(a_i) \cup eff^-(a_i) \setminus eff^+(a_i)$ // cálculo de $\gamma^{-1}(s, a_i)$

$\pi' =$ REGR(A,s', I, $a_i o \pi$)

se $\pi' \neq$ Falha então devolva π'

retroceda

devolva Falha

Planejamento Regressivo: algoritmo

Função REGR(A,s,I,π) devolve Falha ou um plano completo

Entrada:

o conjunto de ações A

o estado atual s // inicializa com (completo)

a descrição do Estado Inicial I

um plano parcialmente especificado

se $I \subseteq s$ então devolva π

selecione $a_i \in \{a \mid a \in A \wedge eff^+(a) \cap s \neq \emptyset \wedge eff^-(a) \cap s = \emptyset\}$

$s' = s \cup pre(a_i) \cup eff^-(a_i) \setminus eff^+(a_i)$ cálculo de $\gamma^{-1}(s, a_i)$

$\pi' =$ REGR(A,s', I, a_ioπ)

se $\pi' \neq$ Falha então devolva π'

retroceda

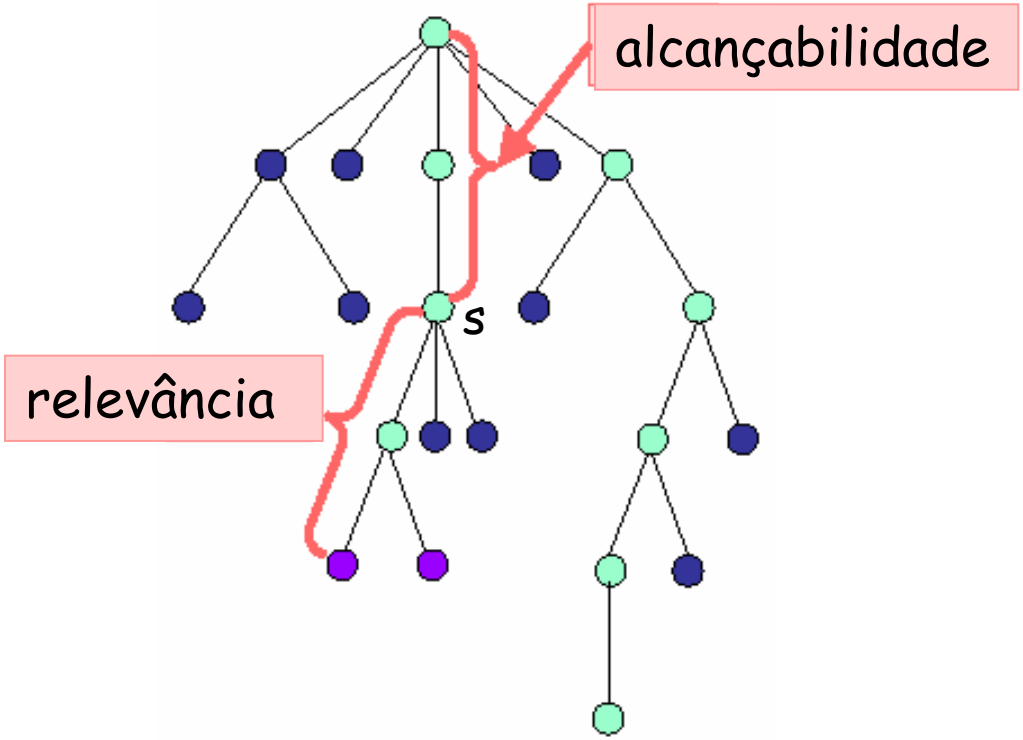
devolva Falha

busca em profundidade: outras estratégias de busca também podem ser usadas

Relevância e Alcançabilidade

Alcançabilidade: Dado um problema [D,I,G], um estado (parcial) s_p é chamado de alcançável se existe uma sequencia de ações $(a_1,a_2,\ldots,a_k)$ que quando executada a partir de I chegará a um estado que satisfaz s_p $(s_p \subseteq s)$.

Relevância: Dado um problema [D,I,G], um estado s é chamado relevante se existe uma sequencia de ações $(a_1,a_2,\ldots,a_k)$ que quando executada a partir de s chegará a um Estado Meta.



Relevância e Alcançabilidade

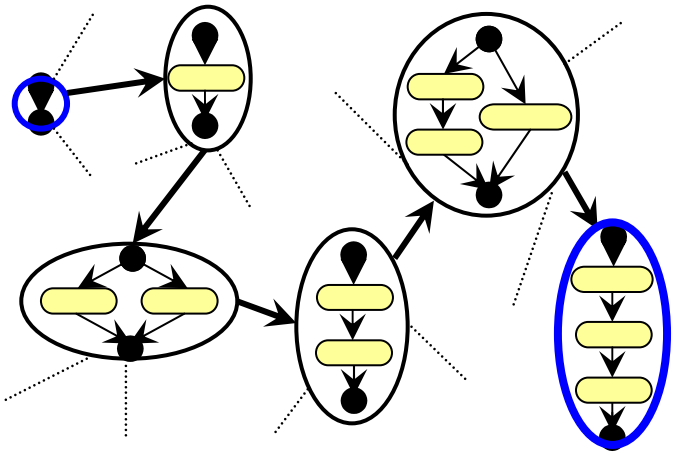
Alcançabilidade: Dado um problema [D,I,G], um estado (parcial) s_p é chamado de alcançável se existe uma sequencia de ações $(a_1,a_2,\ldots,a_k)$ que quando executada a partir de I chegará a um estado que satisfaz s_p $(s_p \subseteq s)$.

Relevância: Dado um problema [D,I,G], um estado s é chamado relevante se existe uma sequencia de ações $(a_1,a_2,\ldots,a_k)$ que quando executada a partir de s chegará a um Estado Meta.

- Planejamento Progressivo: garante que todo estado na busca é alcançável mas não tem nenhuma idéia se os estados são relevantes
  - Heurísticas para progressão precisam ajudar a estimar a relevância dos estados
- Planejamento Regressivo: garante que todo estado na busca é relevante mas não tem nenhuma idéia se os estados são alcançáveis
  - Heurísticas para regressão precisam ajudar a estimar a alcançabilidade dos estados

Idéia: considerar a alcançabilidade e relevância simultaneamente
➔ planejamento no espaço de planos e no grafo de planejamento

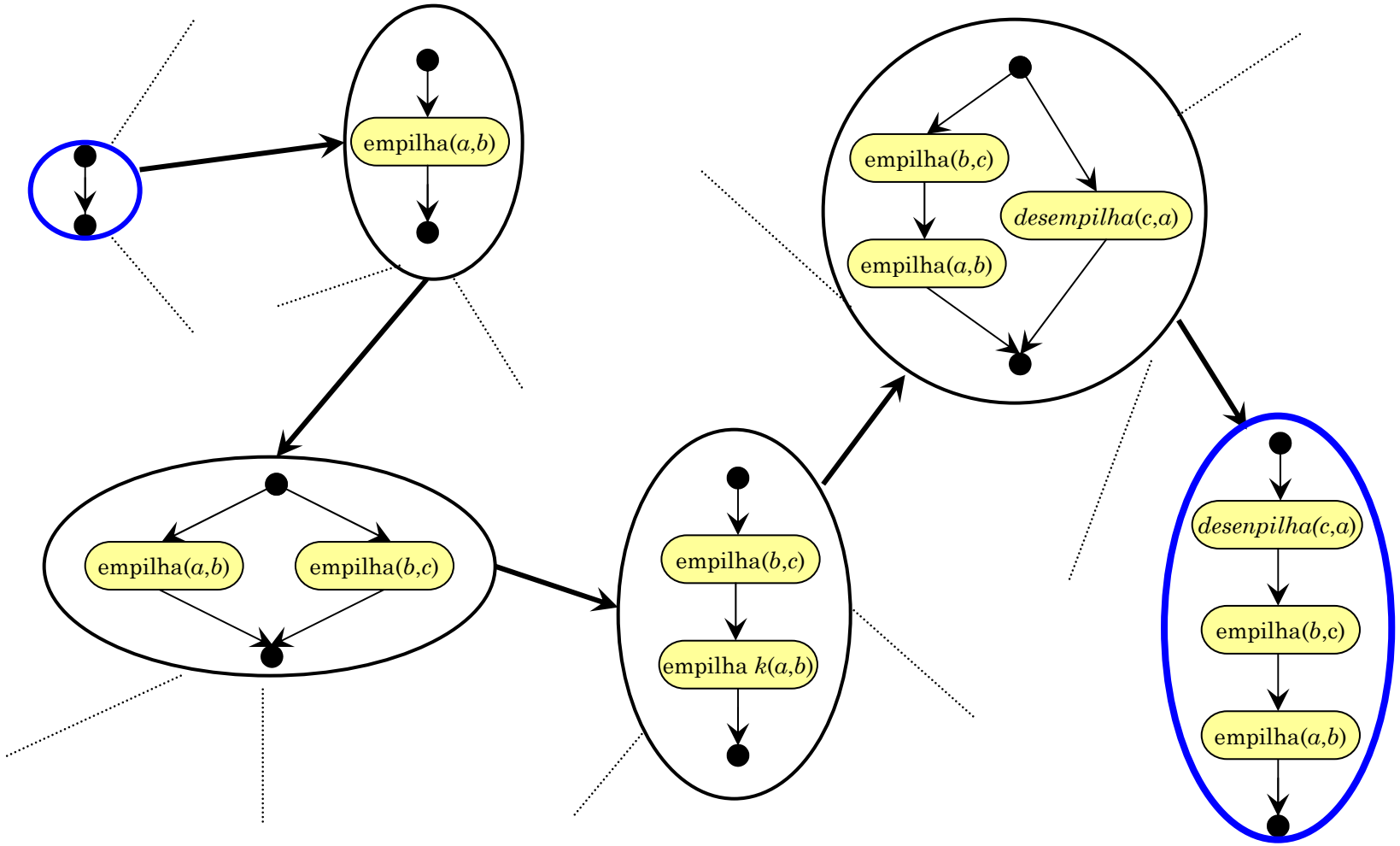
Planejamento como busca no no espaço de planos



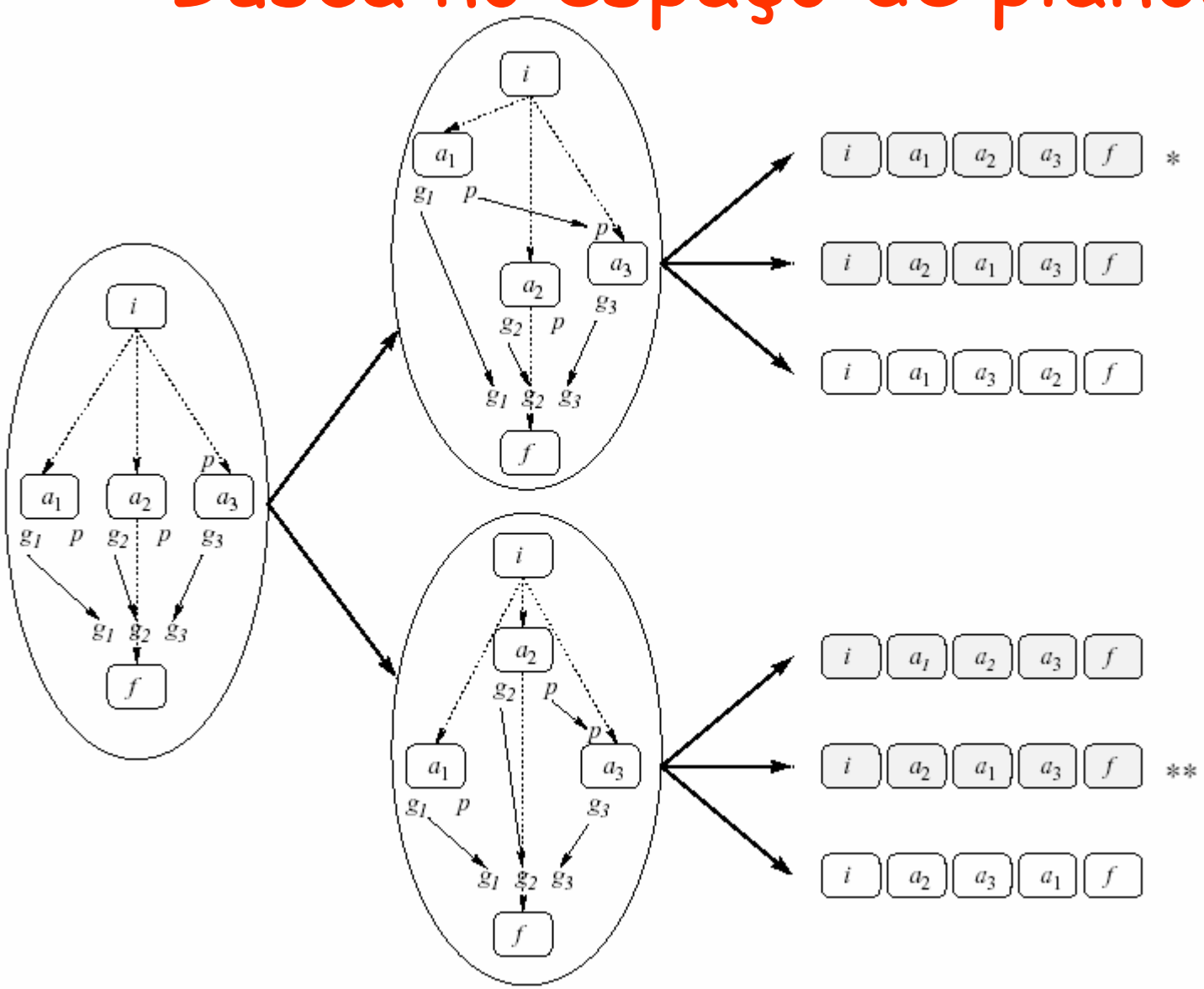
Planejamento como busca no no espaço de planos

Idéia: considerar a alcançabilidade e relevância simultaneamente
➔ planejamento no espaço de planos e no grafo de planejamento

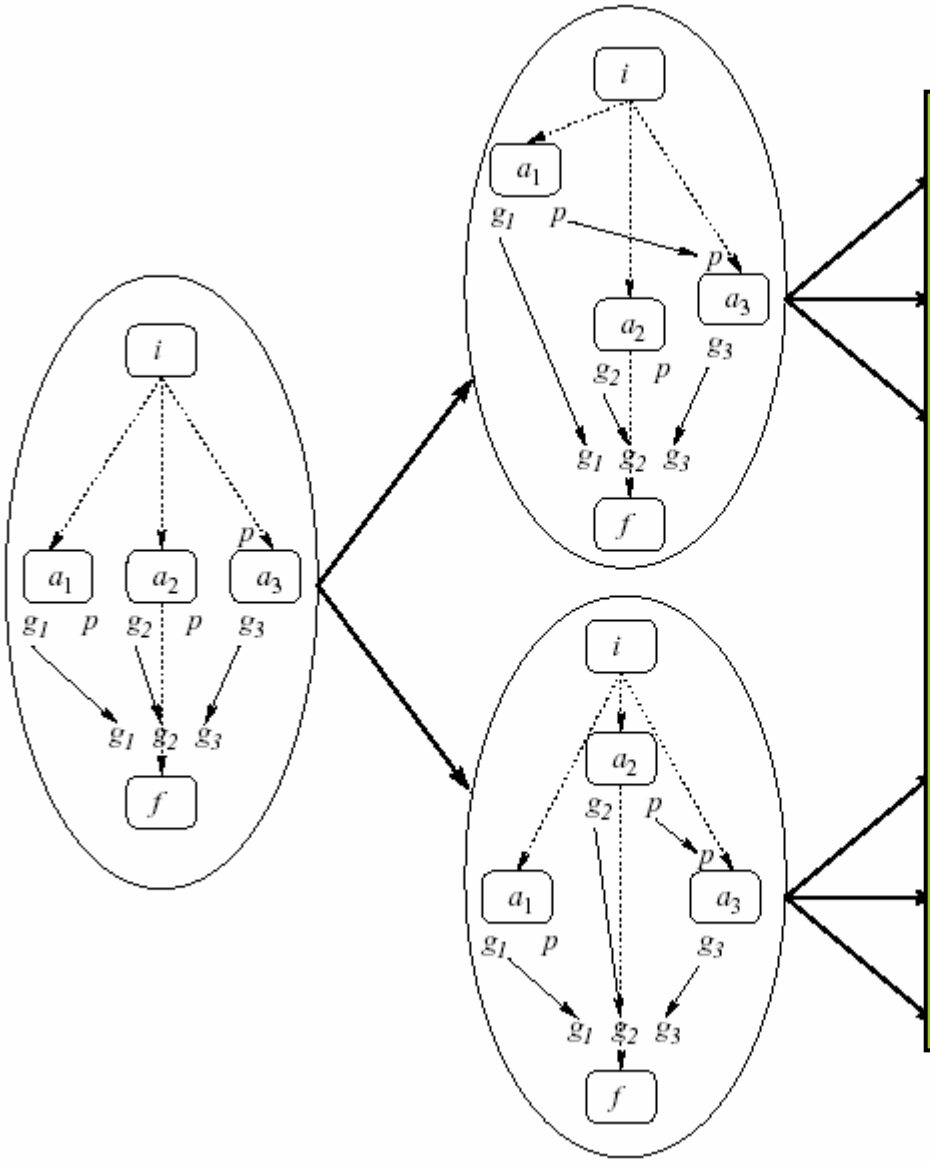
Busca no espaço de planos



Busca no espaço de planos



Busca no espaço de planos



Estados da busca descrevem um plano parcial em termos de:

- passos do plano, mapeados às ações do problema
- restrições de ordem parcial entre ações: definem um conjunto de ordens legais
- vínculo causal: compromisso entre uma ação que adiciona uma condição para satisfazer a precondição de outra ação
- lista de submetas $g \subseteq$ pre(a): todas as precondições de ações do plano, que ainda precisam ser satisfeitas (resolvidas)

Verificar a corretude de um plano parcial: a abordagem causal

- Prova causal: *verificar cada uma das submetas e precondições das ações do plano*
  - *estabelecer submeta g* :
    " Existe uma ação com efeito g, que antecede a ação que tem g como precondição"
  - *submeta g protegida* :
    "Se a submeta g foi estabelecida por uma ação a, não deixe nenhuma outra ação do plano torná-la falsa"

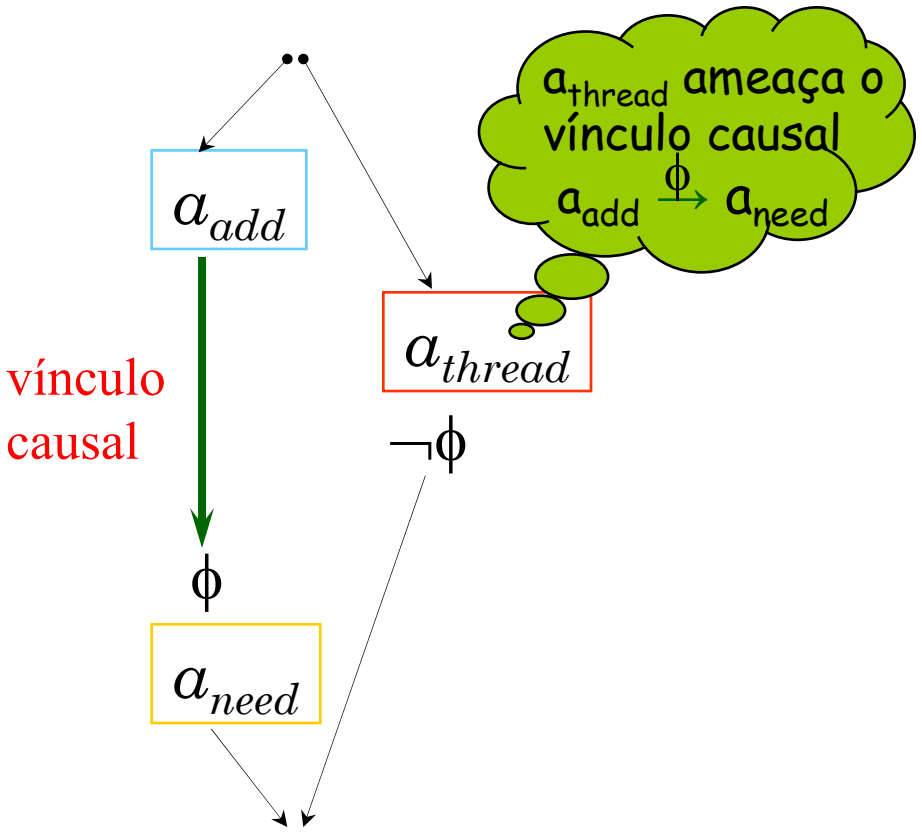
Quando a submeta g também for protegida de efeitos positivos para evitar que 2 ações satisfaçam a mesma submeta, garante-se que o mesmo plano não é visitado duas vezes (busca sem repetição). Ex.: SNLP [McAllester, 89].

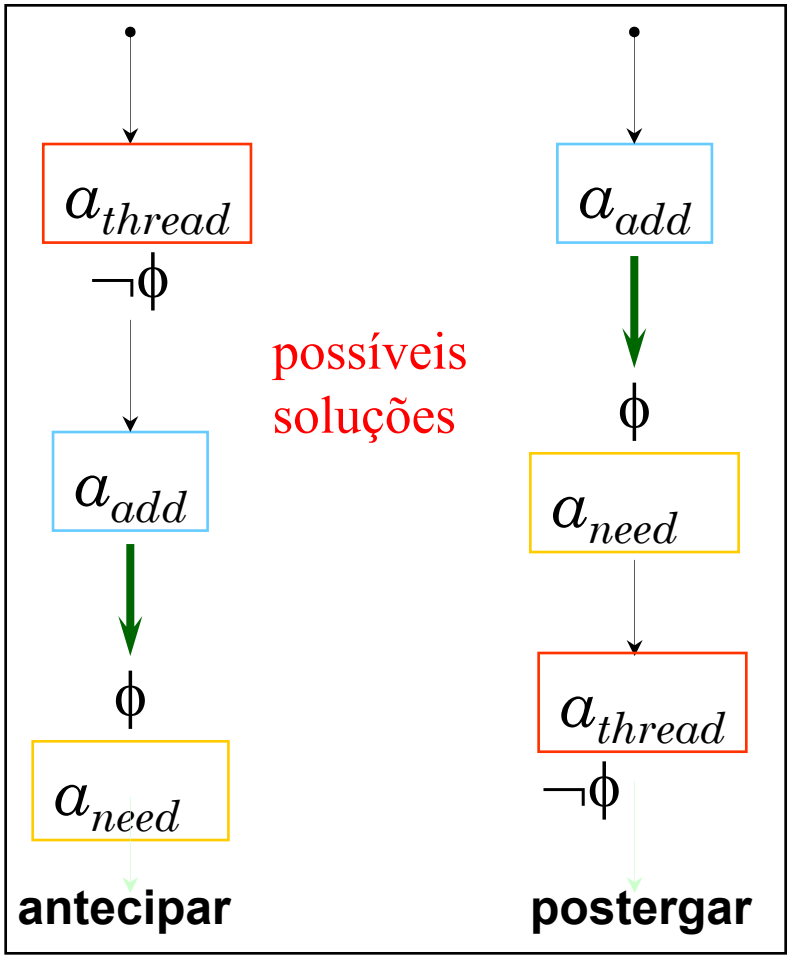
Verificar corretude de um plano: a abordagem causal

- Vantagens da prova causal:
  - local (verifica uma condição por vez)
  - incremental com respeito à inserção de ações
  - independente do estado (não precisa saber quais estados precedem a ação)
    - Fácil de ser extendida para ações durativas e replanejamento

    (se alguma coisa der errado na execução, replaneja utilizando o máximo possível do plano que falhou)

Proteção de vínculo causal





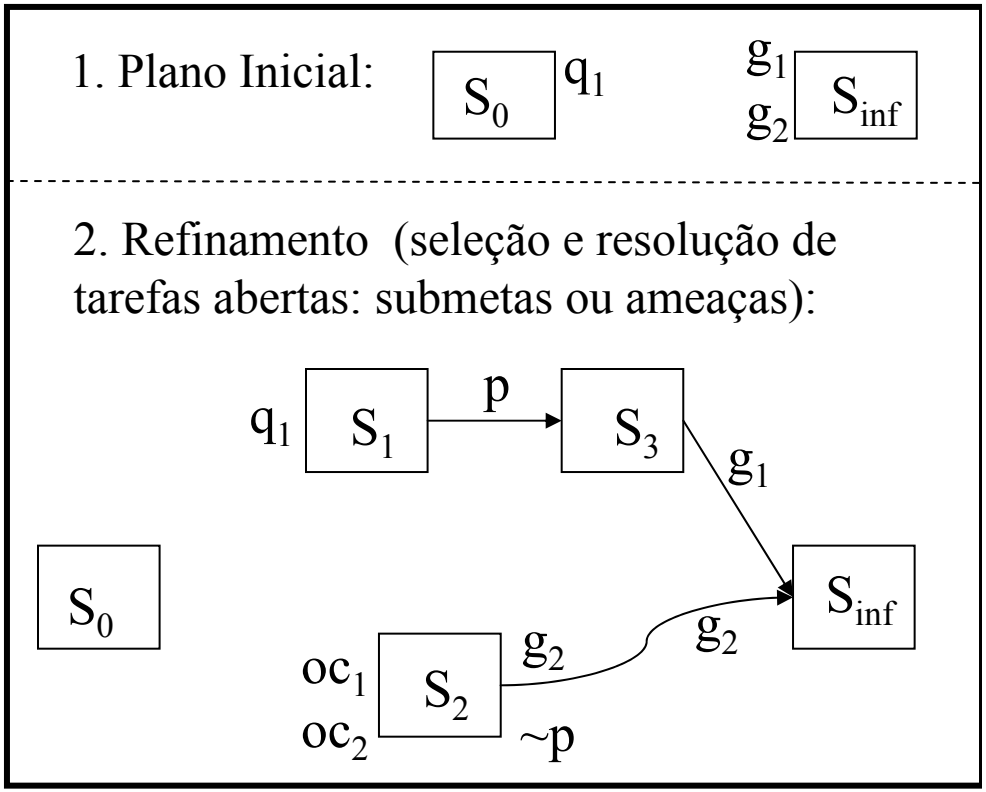
Vínculo causal (*causal link*): $a_{add} \xrightarrow{\phi} a_{need}$

Busca no Espaço de Planos: Geração de Nós Sucessores

- Estado: plano parcial
- O estado sucessor não é gerado com a aplicação de ações do domínio (de modo progressivo ou regressivo) mas com modificações no plano, do tipo:
  - *estabelecer submeta g*
    - adicionar uma ação ao plano
    - adicionar um novo vínculo causal
  - *proteger submeta g (adição de restrição de ordem)*
    - antecipar
    - postergar

Busca no espaço de planos: algoritmo POP (*Partial Order Planning*)

1. *Seja P um plano inicial*
2. *Seleciona uma tarefa (flaw)*: Escolha *f* :
   uma precondição aberta ou uma ameaça
3. *Resolva tarefa*:
   - *Se f é* uma precondição aberta ,
     escolha uma ação *S que adiciona f*
   - *Se f é uma ameaça*,
     escolha antecipar or postergar *f*
   - *Utualiza P*
   - *Devolva* NULL, caso não exista uma solução
4. *Se não existirem mais tarefas*
   então devolva P
   senão *go to* 2.



Pontos de escolha

- Seleção de tarefa *(precondição aberta? ameaça?)*
- Resolução de tarefas *(como selecionar planos parciais sucessores?)*
  - Seleção de ações (*ponto de retrocesso*)
  - Seleção de tratamentos de ameaças (*ponto de retrocesso*)

O algoritmo POP

Idéia geral:

tarefas = SubmetasAbertas(π) $\cup$ Ameaças(π)
// tarefas: submetas sem vínculos causas ou ameaças
se tarefas = $\emptyset$ então devolva π
selecione t $\in$ tarefas
resoluções = Resolvedor(t,π)
// resolver tarefas: adiciona ação, postergar ou antecipar
selecione não-deterministicamente um $\rho \in$ resoluções
π' = Refina(ρ, π)
devolva POP(π')

POP (PSP) é correto e completo (80's). Existem muitas variações desse algoritmo. UCPOP (1992): extensão para tratar efeitos condicionais e quantificação de variáveis de estado

Algoritmo POP (AIMA)

função POP(I, *G, Ações)* **devolve** *plano*
    plano ← Construa-Plano-Minimal(I, *G*)
    laço
        se *Solução?* **então devolva** *plano*
        $Step_{need}$, c ← Seleciona-Submeta(*plano*)
        plano ← Escolha-Ação(*plano, Ações,* $Step_{need}$, c)
        plano ← Resolve-Ameaças(*plano)*
    fim

função Seleciona-Submeta(*plano*) **devolva** $Step_{need}$, c
        selecione um passo do plano $Step_{need}$
        com uma precondição c que não foi satisfeita
        // uma condição c para quel não existe um vínculo causal
    devolva S_{need} , c

POP algorithm (cont.)

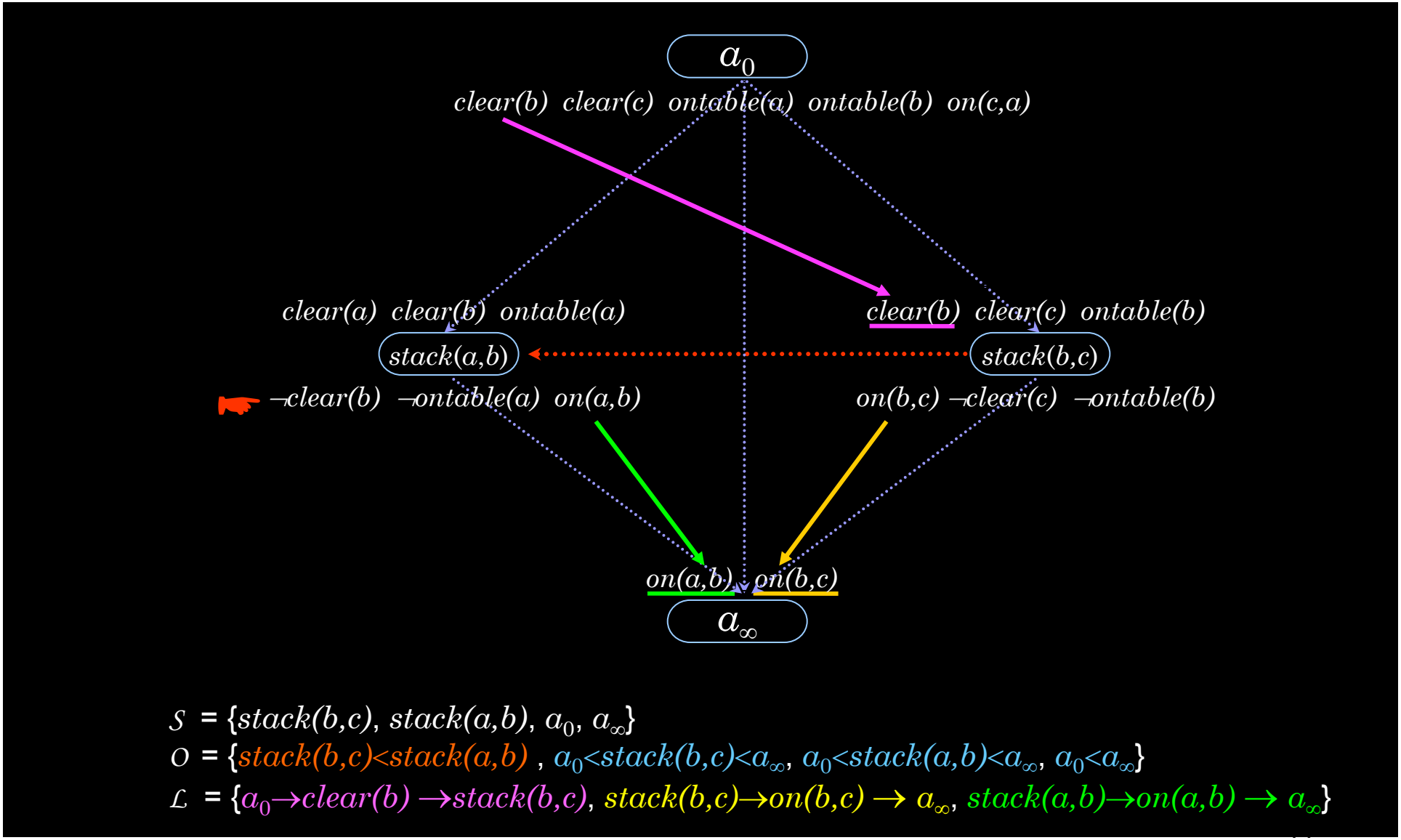
função Escolha-Ação(*plano,Ações, Step*$_{need}$, c) **devolva** plano
selecione uma ação para *Step*$_{add}$ que tem como efeito positivo *c*
// sempre que possível selecione uma ação que já pertence ao plano

se não existe um passo **então falha**
adicione o vínculo causal *Step*$_{add} \xrightarrow{c}$ *Step*$_{need}$ em Links(*plano*)
adicione a restrição de ordem *Step*$_{add}$ < *Step*$_{need}$ em Orderings(*plano*)
se *Step*$_{add}$ é um passo novo no plano **então**
adicione *S*$_{add}$ em Steps(*plano*)
adicione *Start* < *S*$_{add}$ < *Finish* em Orderings(*plano*)

POP algorithm (cont.)

função Resolve-Threats(*plan*) **devolva** plano
  para todo $Step_{threat}$ que ameaçe um vínculo causal $Step_i \xrightarrow{c} Step_j$ em Links(*plano*)
    faça escolha uma das opções:
      Postergar: adicione $Step_{threat} < Step_i$ em Orderings(plano)
      Antecipar: adicione $Step_i < Step_{threat}$ to Orderings(plan)
    se inconsistente(*plano*) **então falha**
  fim

POP: exemplo



S = {$stack(b,c)$, $stack(a,b)$, a_0, a_∞}

O = {$stack(b,c) < stack(a,b)$, $a_0 < stack(b,c) < a_\infty$, $a_0 < stack(a,b) < a_\infty$, $a_0 < a_\infty$}

$\mathcal{L}$ = {$a_0 \rightarrow clear(b) \rightarrow stack(b,c)$, $stack(b,c) \rightarrow on(b,c) \rightarrow a_\infty$, $stack(a,b) \rightarrow on(a,b) \rightarrow a_\infty$}

Domínio do jantar surpresa

Suponha que você quer preparar um jantar surpresa para sua esposa (que está dormindo)

Domínio do jantar surpresa

ação(cozinhar)
    prec: {mãos_limpas}
    eff+: {jantar}

ação(embrulhar)
    prec: {silêncio}
    eff+: {presente}

ação(carregar_lixo)
    prec: {lixo}
    eff-: {lixo, mãos_limpas}

ação(triturar_lixo)
    prec: {lixo}
    eff-: {lixo, silêncio}

Domínio do jantar surpresa

Exercício: faça a simulação do algoritmo POP para o problemas do Jantar Surpresa, em que:

s_0 = {lixo, mãos_limpas, silêncio}

g = {jantar, presente, $\neg$lixo}

Porque Planejamento em "Inteligência Artificial"?

- Solução independente de domínio (GPS!)
- Uso de linguagem com forte fundação na lógica formal
- Melhores algoritmos busca (heurística) informada
- Construção de heurística baseada no conhecimento de planejamento ➔ heurística independente de domínio
- Outras técnicas de IA muito usadas em planejamento:
  - SAT-solvers, raciocínio baseado em casos, planejamento na robótica cognitiva (baseado em prova de teoremas), etc.

Planejamento vs. Busca

- Algoritmos de busca assumem que existem as funções geração-de-nós-sucessores e teste-de-meta que "sabem" quais são os estados legais e como gerar novos estados

- Planejamento faz uma suposição *adicional*: os *estados e ações* podem ser representados em termos de *variáveis de estados e seus valores*
  - Estados inicial e meta são especificados em termos de atribuição de valores às variáveis de estado

A função **teste-de-meta** não é um procedimento do tipo "caixa-preta"

  - As ações modificam os valores das variáveis de estado

A função **geração-de-nós-sucessores** é baseada na semântica da linguagem de descrição de ações

Planejamento vs. Busca

- Dadas as suposições feitas em planejamento sobre a representação de estados e ações, algumas funções genéricas de teste-de-meta e geração-de-nós-sucessores podem ser definidas:
  - função de geração-de-nós-sucessores chamada de "Progressão"
  - função de geração-de-nós-sucessores chamada de "Regressão" e
  - função de geração-de-nós-sucessores chamada de "Ordem-parcial"

- Note que as suposições adicionais feitas em planejamento não mudam os algoritmos de busca (A*, IDS, IDA*,etc)— elas só mudam na maneira de como especificam as funções geração-de-nós-sucessores e teste-de-meta

Planejamento como Busca

- Em planejamento, a busca ainda é realizada em termos de nós de busca que apontam para os seus pais.

- O nó de busca representará:
  - Atribuições completas às variáveis de estado no caso da Progressão
  - Atribuições parciais às variáveis de estado no caso da Regressão (ou atribuições completas se considerarmos um único estado meta)
  - Uma coleção de passos, ordenações, vínculos causais e condições-abertas (sub-goals sem vínculos causais) no de Ordem-parcial

Bibliografia

- M. Ghallab, Dana Nau, and Paulo Traverso. *Automated Planning: Theory and Practice.* Morgan Kaufmann Publishers, 2004
- Stuart Russel and Peter Norvig. Artificial Intelligence: a Modern Approach. Elsevier (2nd edition), 2003.
- Daniel S. Weld. "Recent Advances in AI". Artificial Intelligence Magazine. Ed.: AAAI, 1999.
- Subarao Kambhampati. Notas de aula na Web http://rakaposhi.eas.asu.edu/cse471/
- Silvio do Lago Pereira. Planejamento no Cálculo de Eventos. Dissertação de Mestrado em Ciência da Computação, IME-USP, 2002.