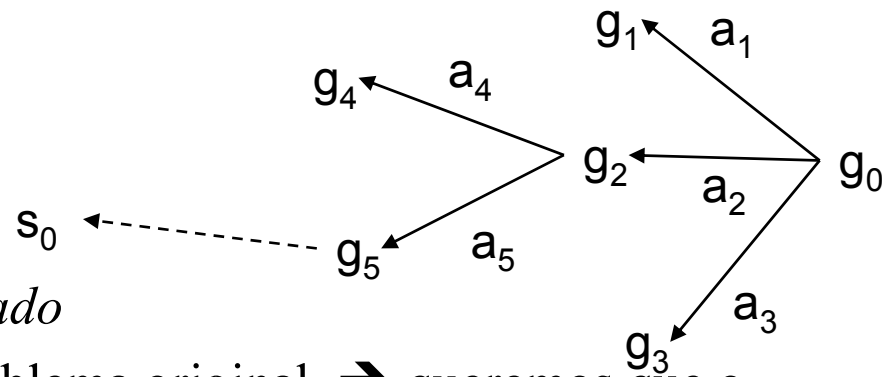
Grafos de Planejamento

Grafo de Planejamento: motivação

- Uma das razões da ineficiência dos algoritmos de busca é o ***fator de ramificação*** da árvore de busca, isto é, o número de filhos de cada nó
- Por exemplo: a busca regressiva pode selecionar muitas ações que não atingem o estado inicial
- Como reduzir o fator de ramificação?



1. Primeiro você cria um *problema relaxado*
   - Remova algumas restrições do problema original ➔ queremos que o problema relaxado seja mais fácil de se resolver (tempo polinomial)
   - As soluções para o problema relaxado incluirão todas as soluções do problema original
2. Depois, faça uma versão modificada da busca original
   - Restrinja o seu espaço de busca incluindo somente as ações que ocorrem no problema relaxado

Outline

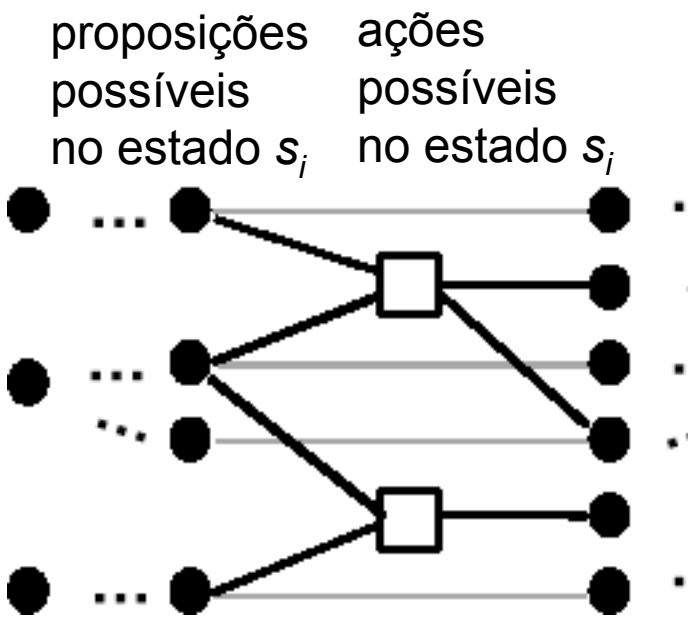
- O algoritmo Graphplan
- Construção de grafos de planejamento
- Exclusão Mútua (mutex)
- Extração da solução

Procedimento Graphplan

- para $k = 0, 1, 2, \ldots$
  - *Expansão do Grafo:*
    - » crie um "grafo de planejamento" que contém k "níveis"
  - Cheque se o grafo satisfaz uma condição necessária (mas não suficiente) para existir um plano solução

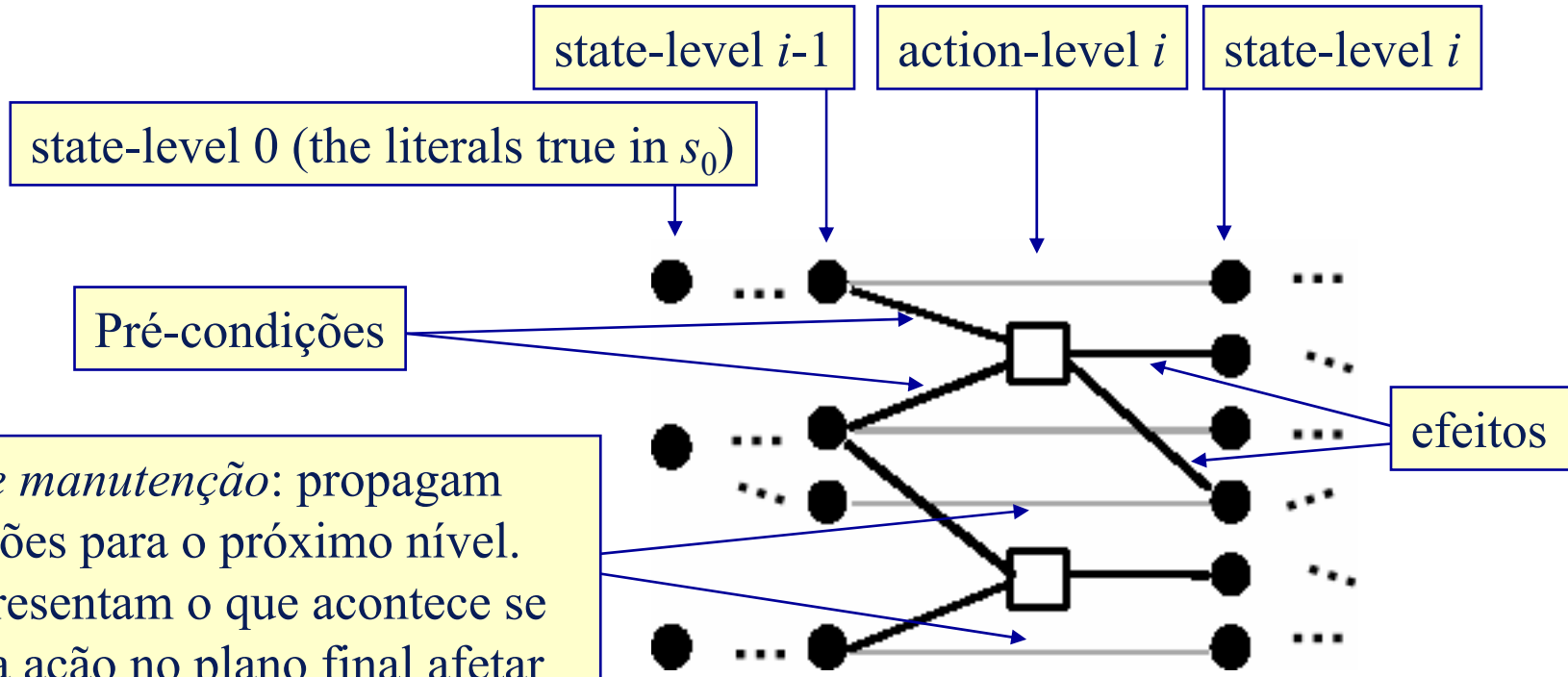
problem
relaxad

  - Se a condição for verdadeira, então
    - » Faça *extração da solução :*
      - Busca regressiva, modificada para considerar apenas as ações no grafo de planejamento
      - Se existir uma solução então devolva o plano



O grafo de planejamento

- Alterna níveis de proposições e ações
  - Todas as ações que podem ocorrer a cada instante (passo) do plano
  - Todas as proposições que podem ser adicionadas por aqueleas ações



Ações de manutenção: propagam proposições para o próximo nível. Elas representam o que acontece se nenhuma ação no plano final afetar uma determinada proposição.

anning: theory and practice (http://www.laas.fr/planning/).

Exemplo: Jantar Surpresa

» Daniel Weld (U. of Washington)

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

  s_0 = {garbage, cleanHands, quiet}

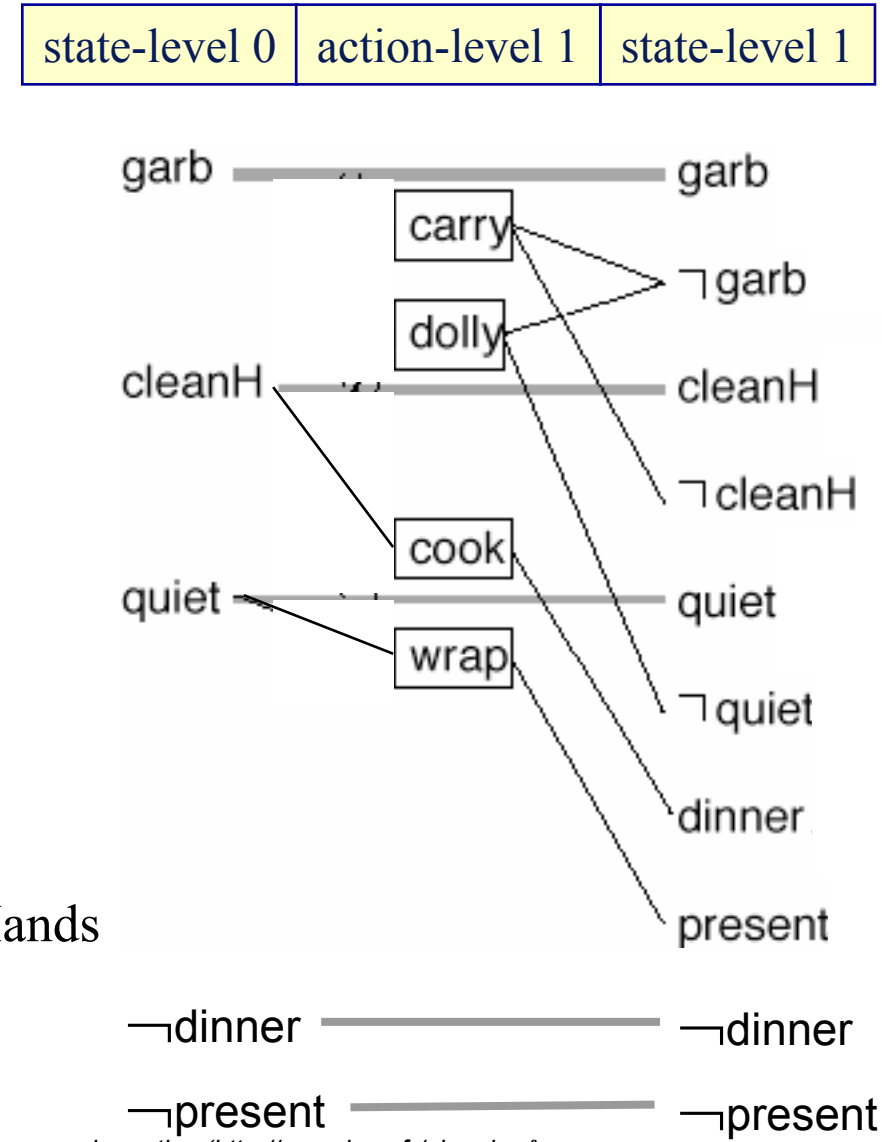
  g = {dinner, present, ¬garbage}

Ações	Pré-condições	Efeitos
cook()	cleanHands	dinner
wrap()	quiet	present
carry()	*none*	¬garbage, ¬cleanHands
dolly()	*none*	¬garbage, ¬quiet

Também existem ações de manutenção para cada proposição

Exemplo (continuação)

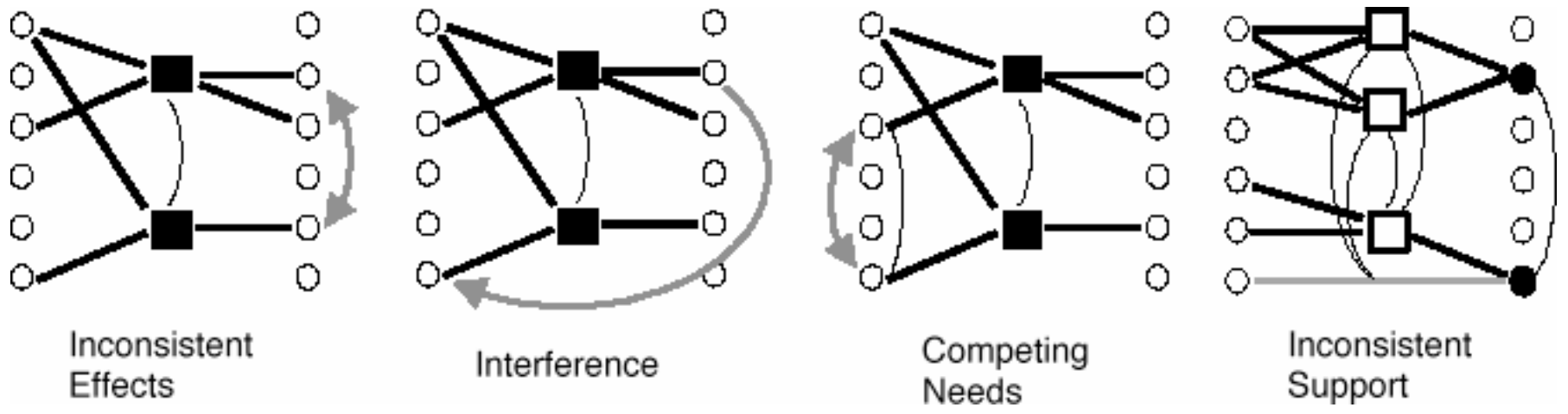
- *state-level* 0:
  {todos os átomos em s_0} U
  {negações de todos os átomos
  que não estão em s_0}
- *action-level* 1:
  {todas as ações cujas pré-condições
  são satisfeitas em s_0}
- *state-level* 1:
  {todos os efeitos de todas as
  ações em *action-level* 1} U
  {todos os átomos em *state-level* 0 }



Ações	Pré-condições	Efeitos
cook()	cleanHands	dinner
wrap()	quiet	present
carry()	*none*	¬garbage, ¬cleanHands
dolly()	*none*	¬garbage, ¬quiet

… e todas as ações de manutenção.

Exclusão Mútua (Mutex)



- Duas ações no mesmo nível são mutex se
  - *Efeitos inconsistentes:* um efeito de uma nega um efeito da outra
  - *Interferência:* uma elimina uma pré-condição da outra
  - *Necessidades que competem:* ações com pré-condições inconsistentes
- Caso contrário, as ações não interferem entre si, isto é:
  - Ambas podem fazer parte de um plano solução
- Duas proposições num mesmo nível são mutex se
  - *Suporte inconsistente:* uma é a negação de outra ou todas as maneiras de satisfazê-las são mutex

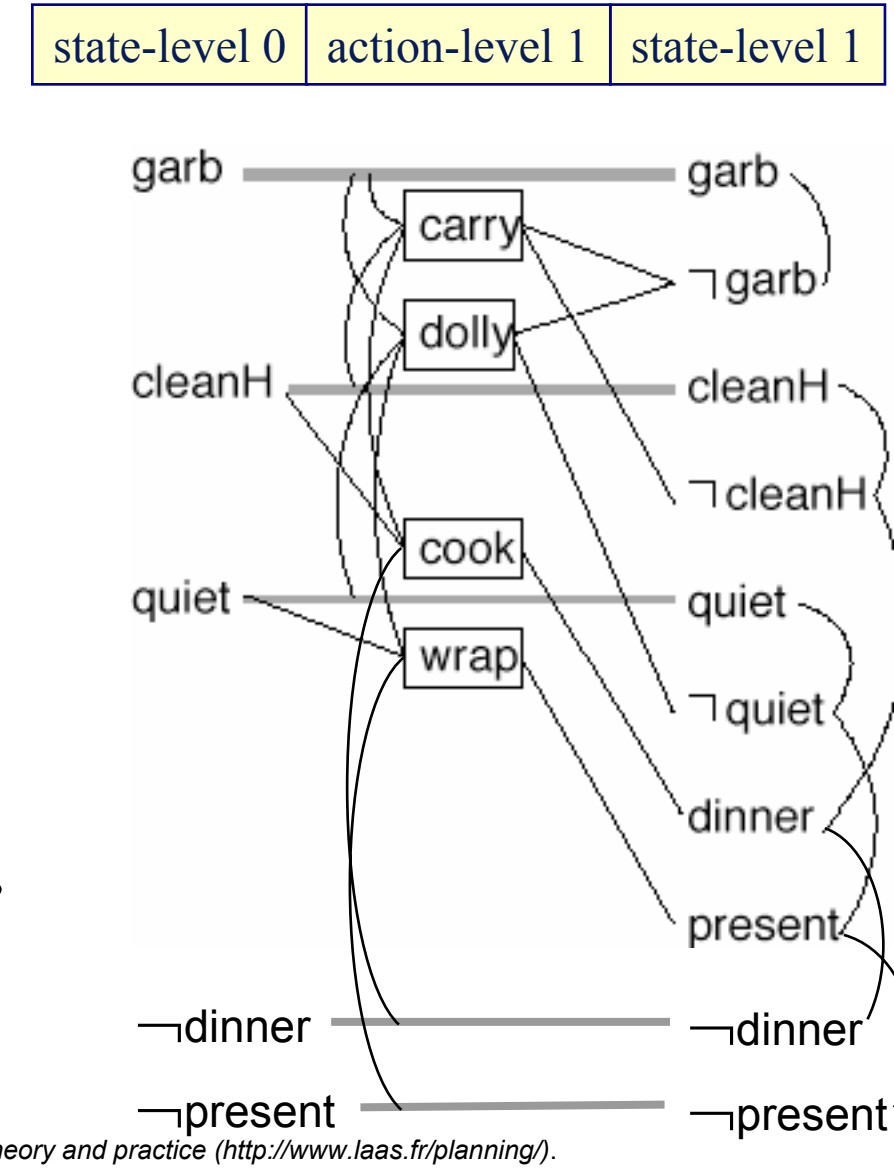
Exemplo (continuação)

Identificação de mutex no grafo:

- *carry* está em mutex com a ação de manutenção para *garbage*
  - efeitos inconsistentes
- *dolly* está em mutex com *wrap*
  - interferência
- *~quiet* está em mutex com *present*
  - suporte inconsistente
- as ações *cook* e *wrap* estão em mutex ações de manutenção

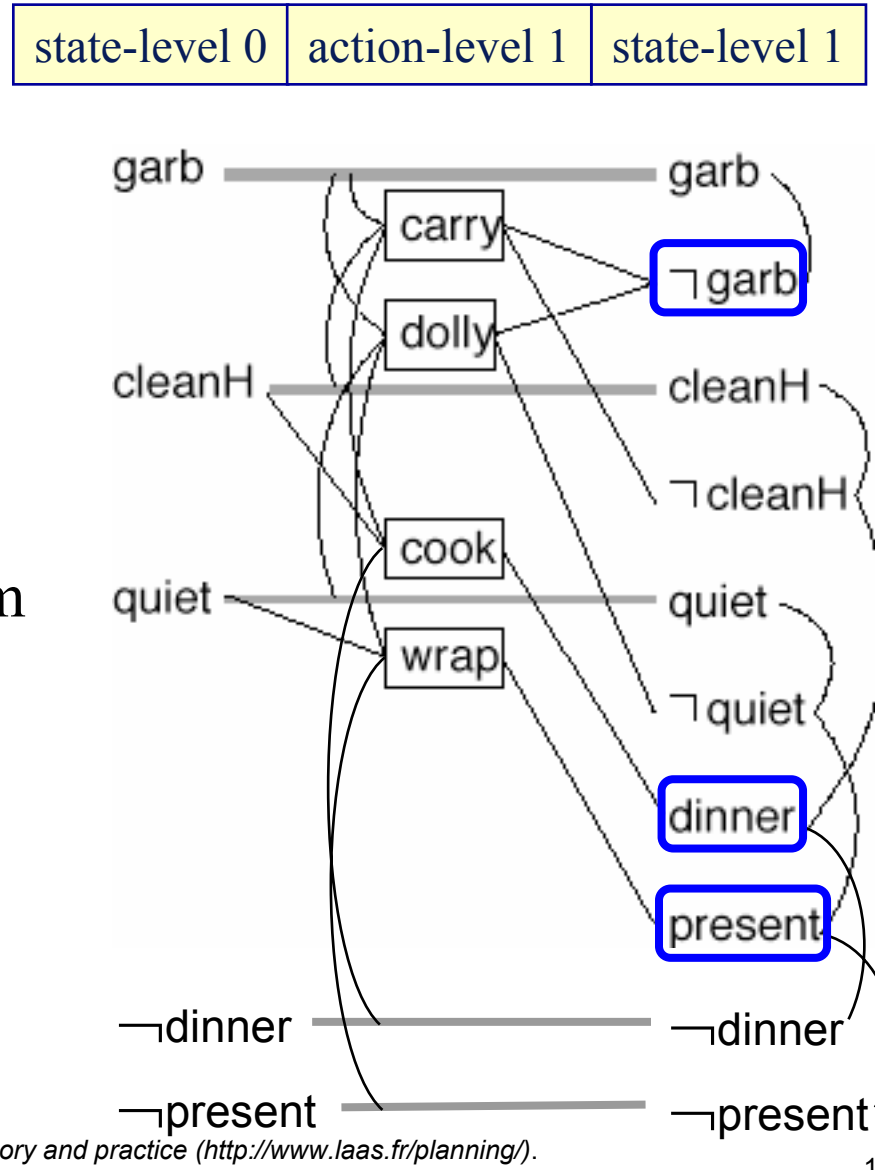
Ações	**Pré-condições**	**Efeitos**
cook()	cleanHands	dinner
wrap()	quiet	present
carry()	*none*	¬garbage, ¬cleanHands
dolly()	*none*	¬garbage, ¬quiet

… e todas as ações de manutenção.



Exemplo (continuação)

- Cheque se pode existir um plano para a meta do problema:
  - {¬*garbage, dinner, present*}
- Note que
  - Todas as proposições da meta ~são possíveis em *state-level* 1
  - Elas não estão em mutex
- Então há uma chance de existir um plano
  - Tente extrair uma solução



Extração da Solução

O conjunto de metas que estamos tentando alcançar

O nível do estado s_j

rocedimento Extração-da-Solução(g,j)

se j=0 então devolva a solução

Para cada átomo l em g

escolha não-determinística de uma ação para ser usada no estado s_{j-1} para atingir l

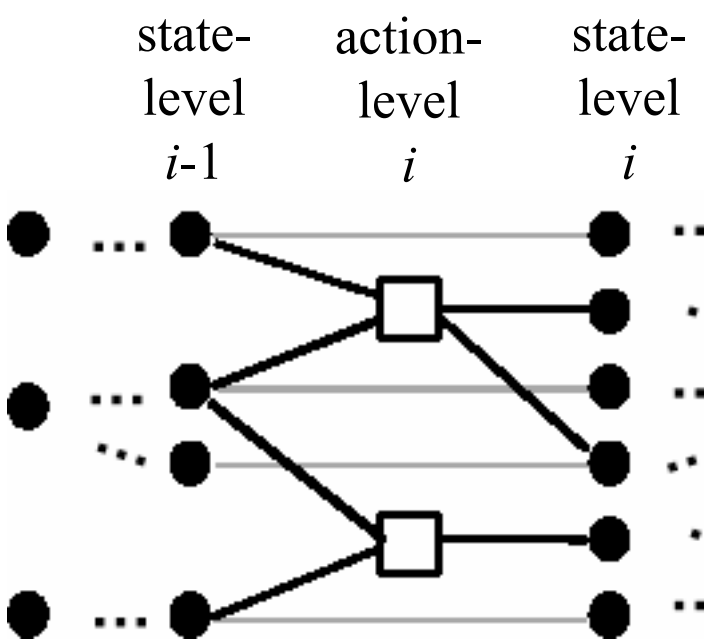
Se qualquer par possível de ações está em mutex então *backtracking*

g' := {pré-condições das ações selecionadas}

Extração-da-Solução(g', j–1)

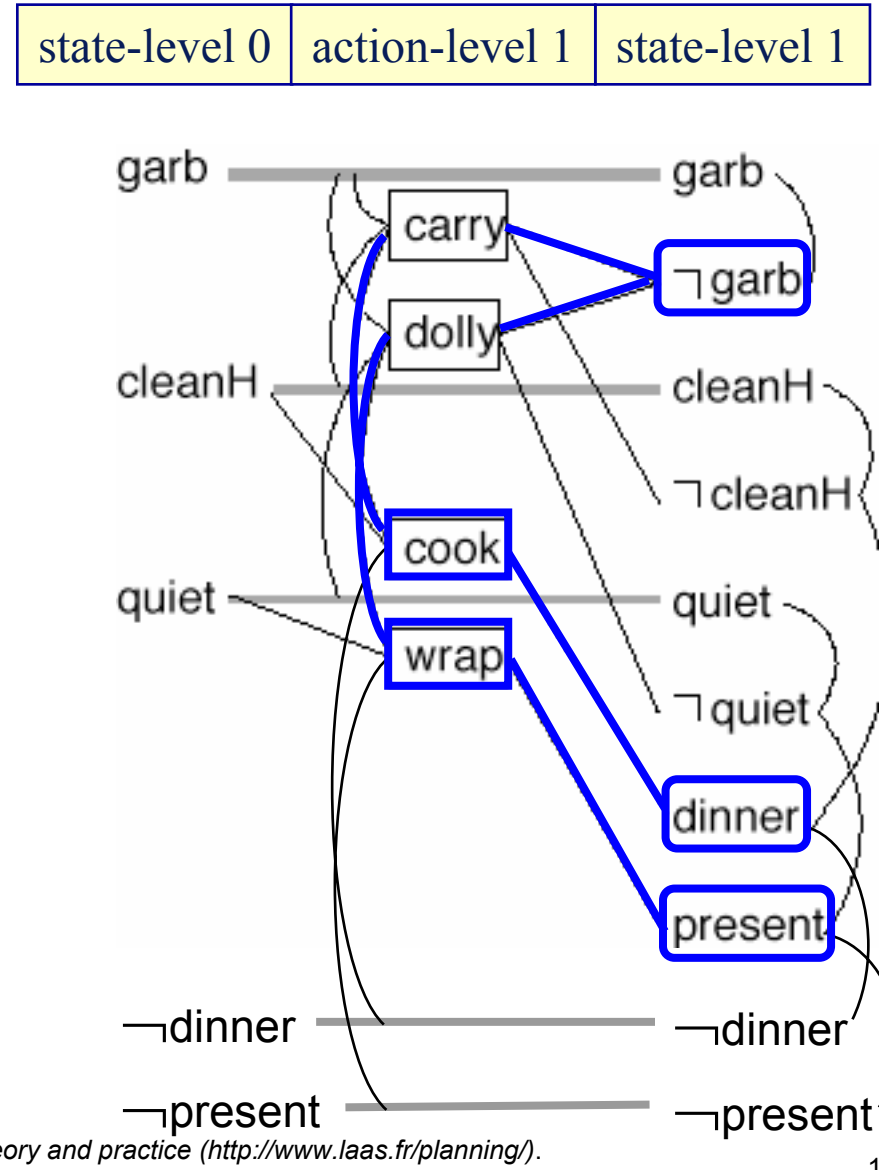
im

Uma ação real ou uma ação de manutenção



Exemplo (continuação)

- Existem 2 conjuntos de ações para as metas no ***state-level*** 1
- Nenhum deles serve: ambos contém ações que são ***mutex***



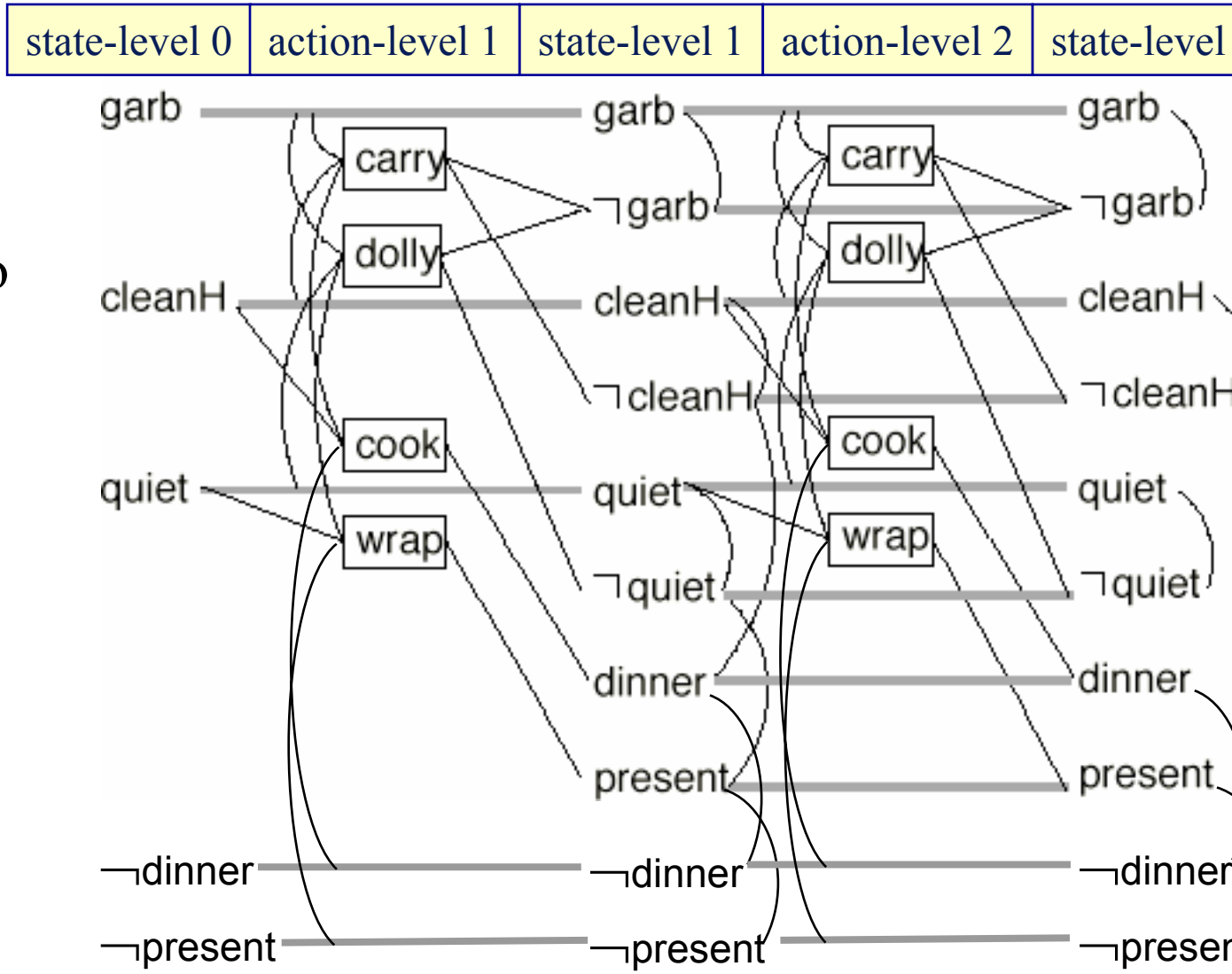
O que o algoritmo faz?

procedimento *Graphplan*:

- para $k = 0, 1, 2, \ldots$
  - *Expansão do Grafo:*
    - » crie um "grafo de planejamento" que contém k "níveis"
  - Cheque se o grafo satisfaz uma condição necessária (mas não suficiente) para existir um plano solução
  - Se a condição for verdadeira, então
    - » Faça *extração da solução :*
      - Busca regressiva, modificada para considerar apenas as ações no grafo de planejamento
      - Se existir uma solução então devolva o plano

Exemplo (continuação)

- Retroceda e faça mais uma expansão do grafo
- Gere outro ***action-level*** e outro ***state-level***

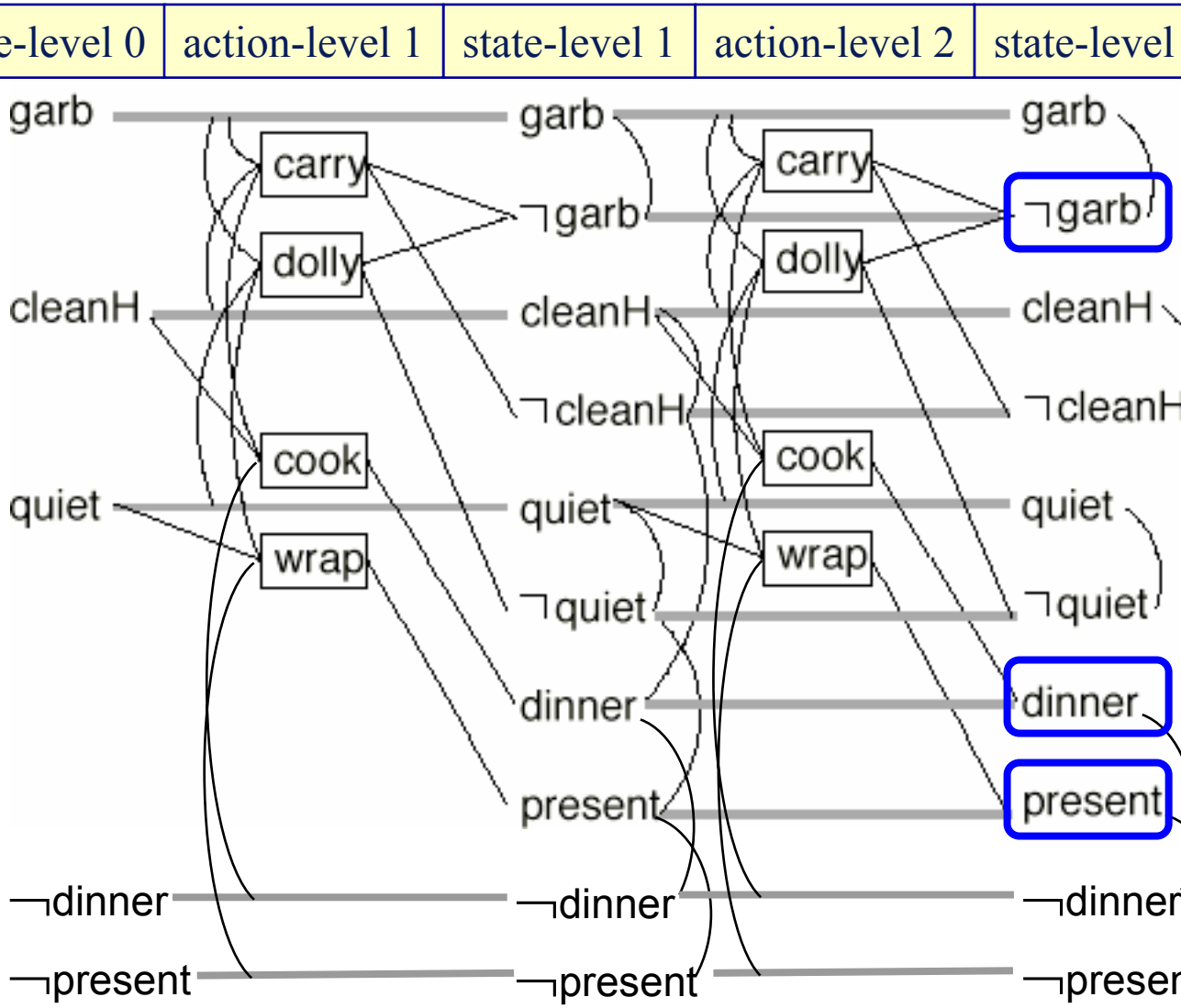


Exemplo (continuação)

Extração da solução

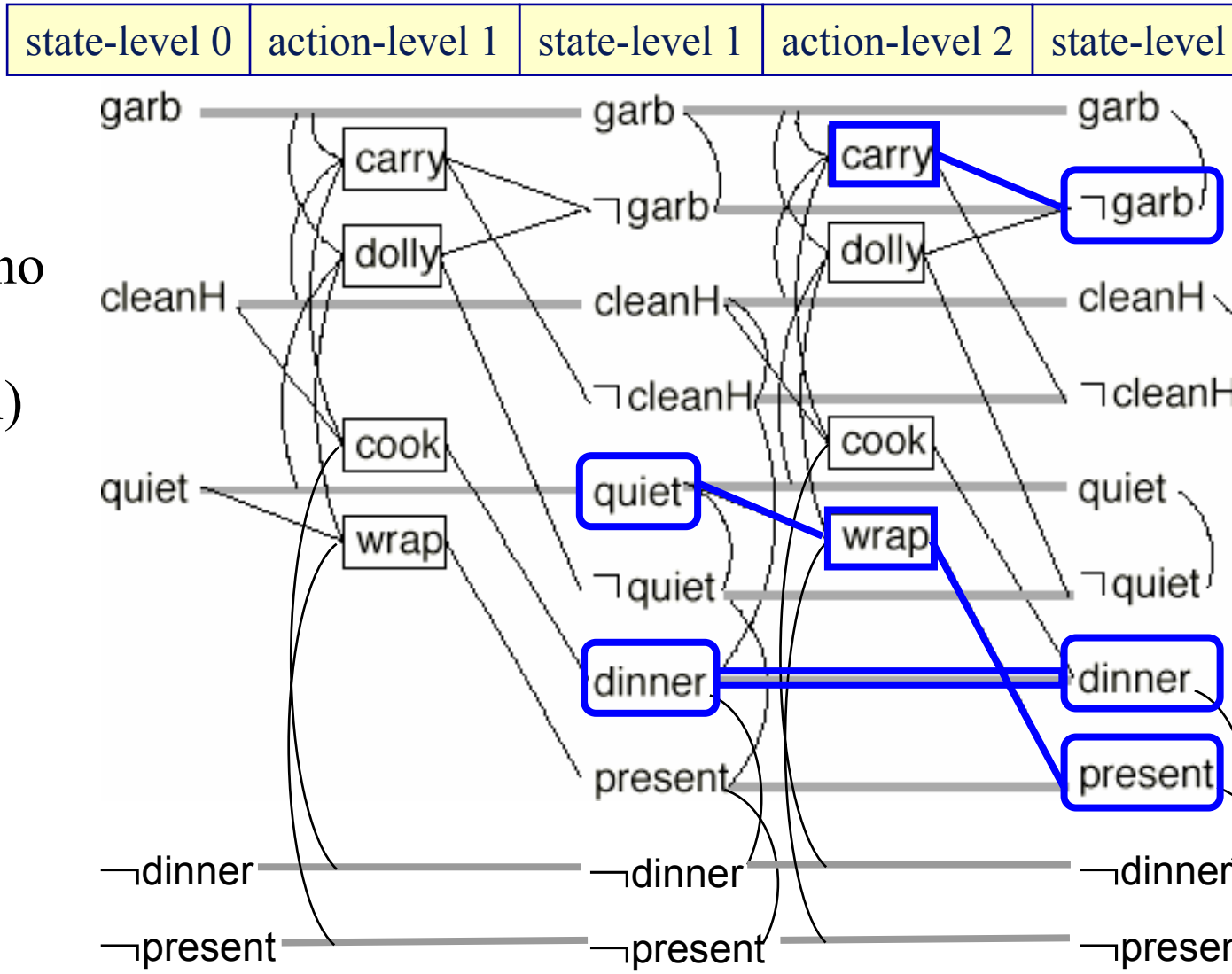
Existema 12 combinações no nível 4

- 3 para atingir ***¬garb***
- 2 para atingir ***dinner***
- 2 para atingir ***present***



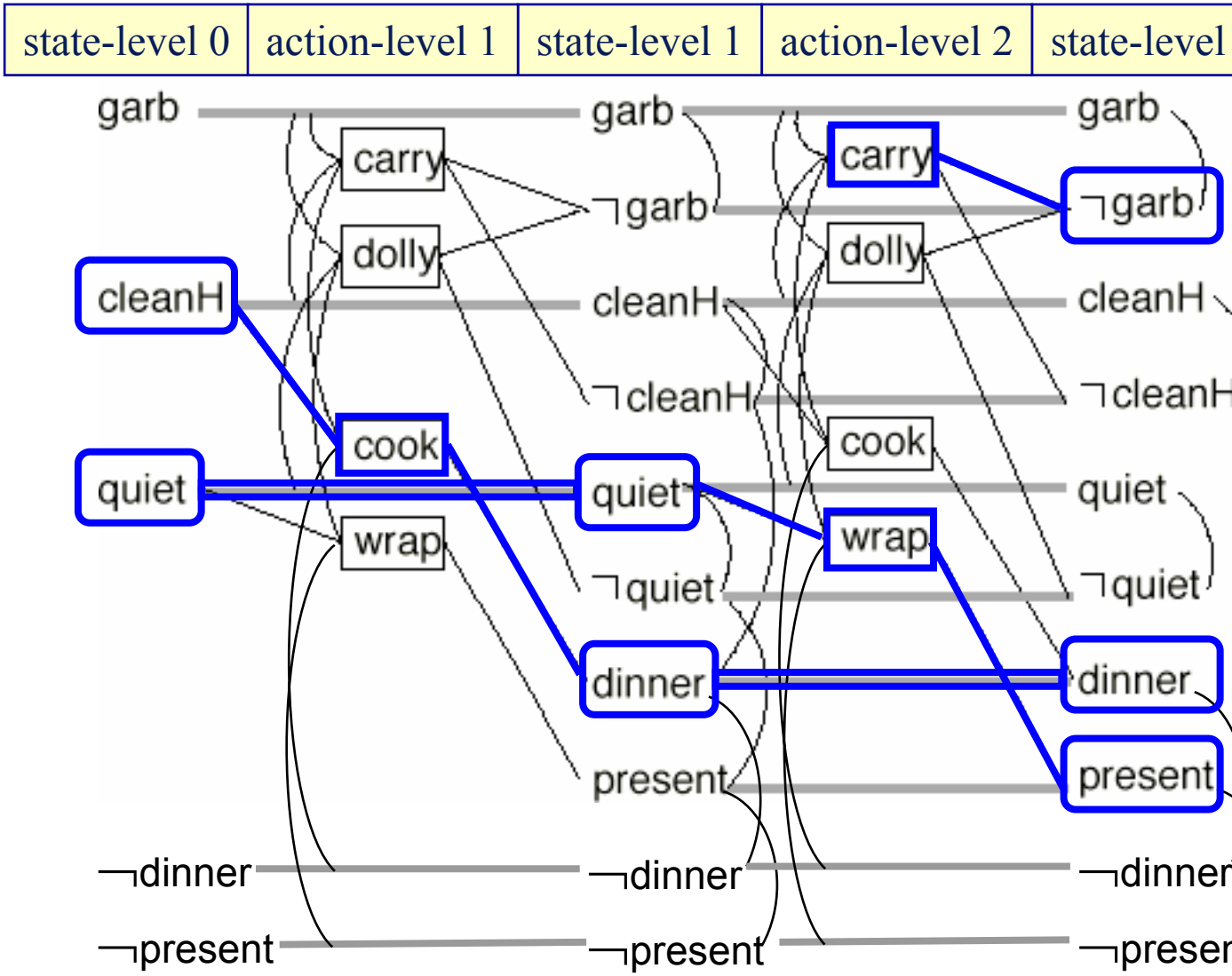
Exemplo (continuação)

- Várias das combinações parecem OK no nível 2 (vide ações em azul)



Exemplo (continuação)

- Chama Extração-da-Solução recursivamente no nível 2
- Sucesso!
- Solução de *tamanho paralelo* igual a 2



Comparação com o POP

- Diferentemente do POP, o planejador *Graphplan* cria instâncias *ground* de todos os operadores
  - Muitas delas podem ser irrelevantes !!
- Porém, a parte de busca regressiva do *Graphplan* — que é a parte mais difícil — somente olha as ações no grafo de planejamento!
  - Espaço de busca menor que o POP ➔ por isso é mais rápido

História

- Antes de *Graphplan*, muitos pesquisadores de planejamento estavam trabalhando com variações de planejadores do tipo POP
  - SNLP, UCPOP, etc.
- *Graphplan* causou a alvoroço porque ele é muito mais eficiente
- Existem muitos planejadores que se baseiam no Graphplan
  - IPP, STAN, GraphHTN, SGP, Blackbox, Medic, TGP, LPG
- … mas que são muito mais eficientes do que o *Graphplan* original