Lecture slides for
Automated Planning: Theory and Practice

Técnicas de Grafos de Planeamento

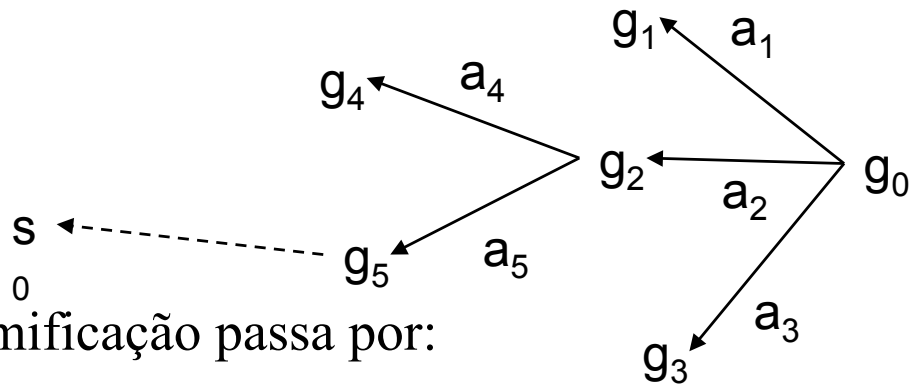
Adaptado por Carlos Viegas Damásio a partir dos slides desenvolvidos por
Dana S. Nau

História

- Antes da aparição do Graphplan, a maior parte da investigação em planeamento estava centrada em planeadores com ordem parcial
  - POP, SNLP, UCPOP, etc.
- Graphplan foi proposto em 1995 por Blum e Furst, provocando admiração porque era ordens de grandeza mais rápido
- Diversos sistemas de planeamento a seguir utilizaram algumas das ideias do Graphplan
  - IPP, STAN, GraphHTN, SGP, Blackbox, Medic, TGP, LPG

Motivação

- Uma fonte de ineficiência é o factor de ramificação elevado em qualquer dos tipos de planeamento analisados anteriormente



- Uma técnica para reduzir o factor de ramificação passa por:
- Criar um *problema relaxado*
  - Remover algumas restrições do problema original
    - » Deseja-se que o problema relaxado seja de fácil resolução (tempo polinomial)
  - As soluções do problema relaxado incluirão todas as soluções do problema original
- Depois efectuar uma versão modificada da procura original
  - Restringe o espaço de procura de forma a incluir apenas as soluções que ocorrem no problema relaxado

Resumo

- O algoritmo Graphplan
- Contrução de grafos de planeamento
- Exclusão Mútua
- Extracção da Solução
- Heurísticas para o POP
- Discussão

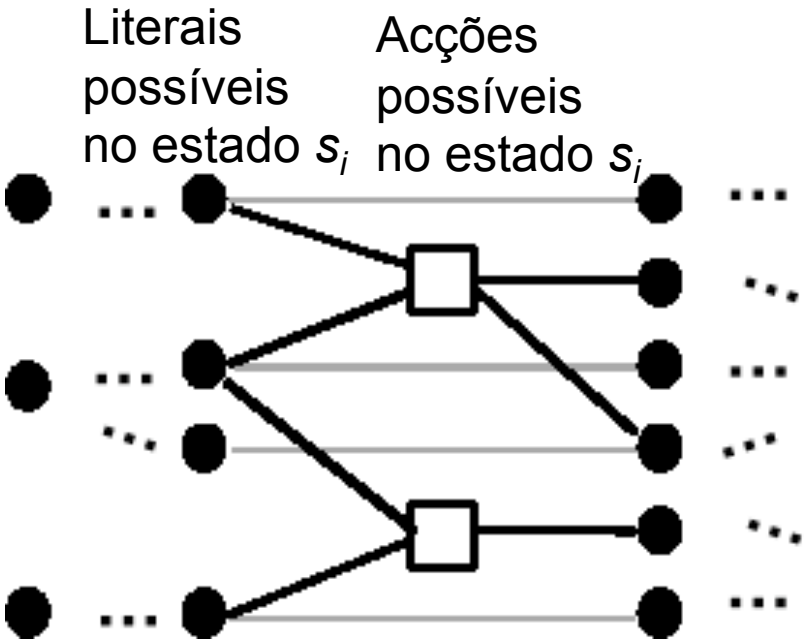
Graphplan

procedure Graphplan:

- for $k = 0, 1, 2, \ldots$
  - *Expansão do grafo:*
    - » criar um "grafo de planeamento" contendo k "níveis"
  - Verificar se o grafo de planeamento satisfaz uma condição necessária (mas insuficiente) para a existência de um plano

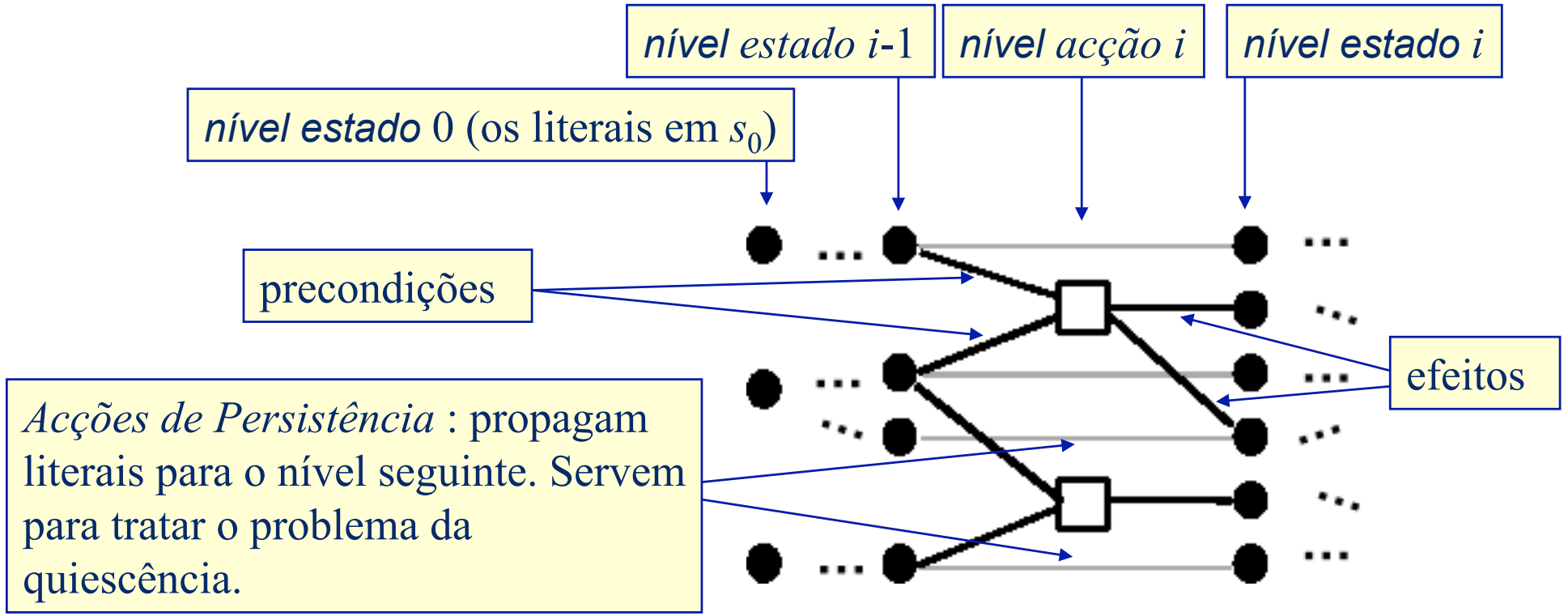
problema relaxado

  - Se satisfizer, então
    - » *Extrair solução:*
      - Procura regressiva, modificada para considerar apenas as acções no grafo de planeamento
      - Caso se encontre uma solução, então devolvê-la



O Grafo de Planeamento

- Camadas alternadas de literais e acções concretos (ground)
  - Todas as acções que possivelmente podem ocorrer em cada instante de tempo
  - Todos os literais produzidos por essas acções



Exemplo (Dan Weld – U. of Washington)

- Suponha-se que pretende preparar um jantar de surpresa para o seu/ sua cara-metade (que está a dormir e não deve ser acordado)

  s_0 = {garbage, cleanHands, quiet}

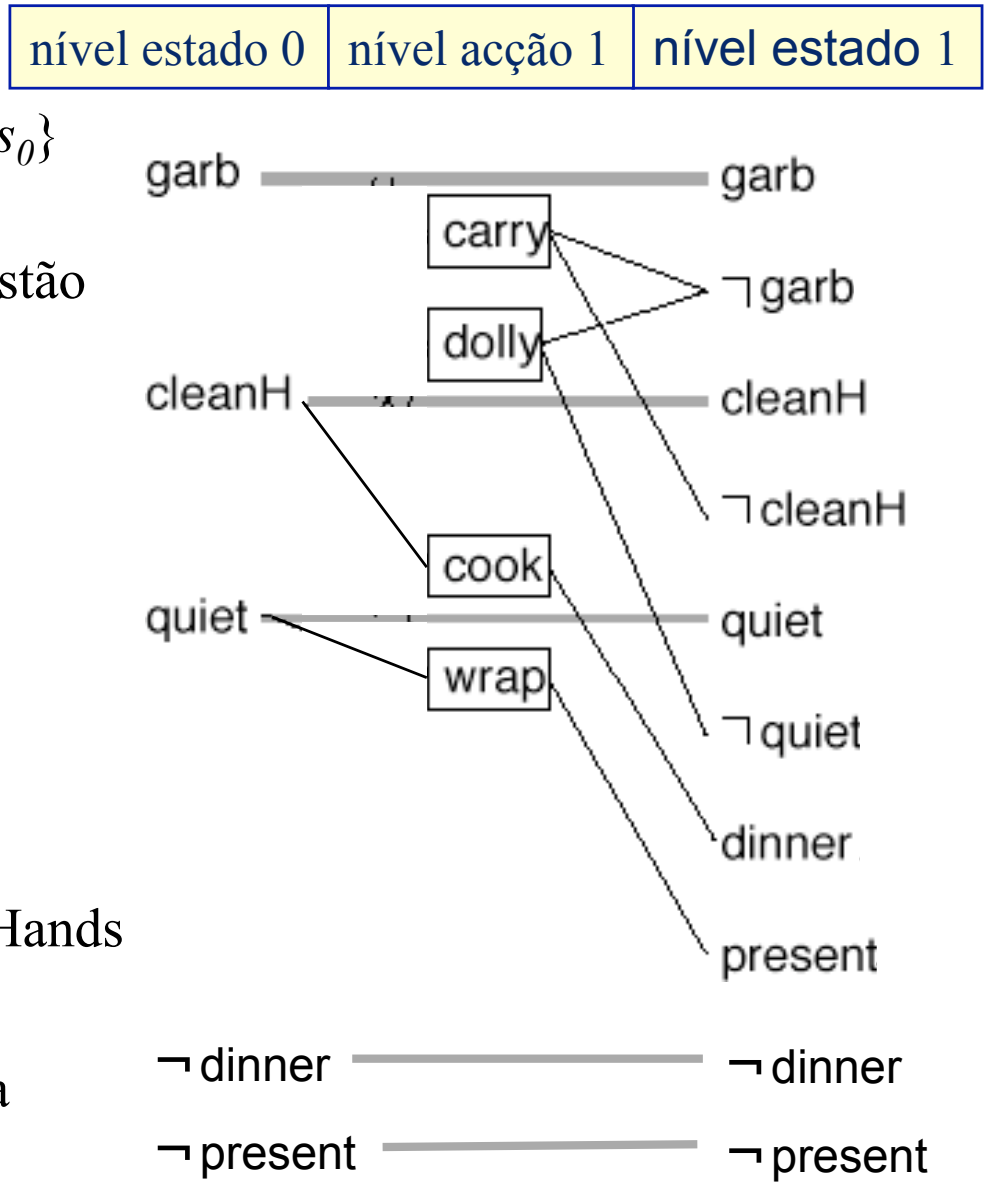
  g = {dinner, present, ¬garbage}

Acção	Precondições	Efeitos
cook()	cleanHands	dinner
wrap()	quiet	present
carry()	*nenhuma*	¬garbage, ¬cleanHands
dolly()	*nenhuma*	¬garbage, ¬quiet

  Também se adicionam acções de persistência: uma para cada literal L, com precondição L e efeito L.

Exemplo (continuação)

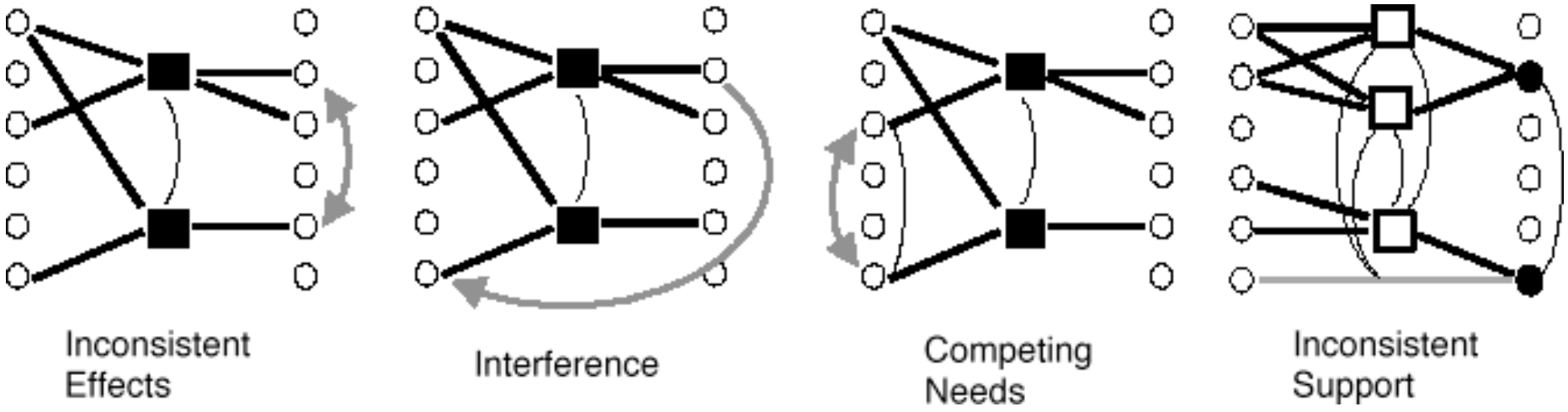
- nível estado 0:
  {todos os átomos em s_0} U
  {negação de todos os átomos não em s_0}
- nível acção 1:
  {todas as acções cujas precondições estão satisfeitas em s_0}
- nível estado 1:
  {todos os efeitos de todas as acções no nível acção 1}



Acção	Precondições	Efeitos
cook()	cleanHands	dinner
wrap()	quiet	present
carry()	*nenhum*	¬garbage, ¬cleanHands
dolly()	*nenhum*	¬garbage, ¬quiet

Juntamente com as acções de persistência

Exclusão Mútua



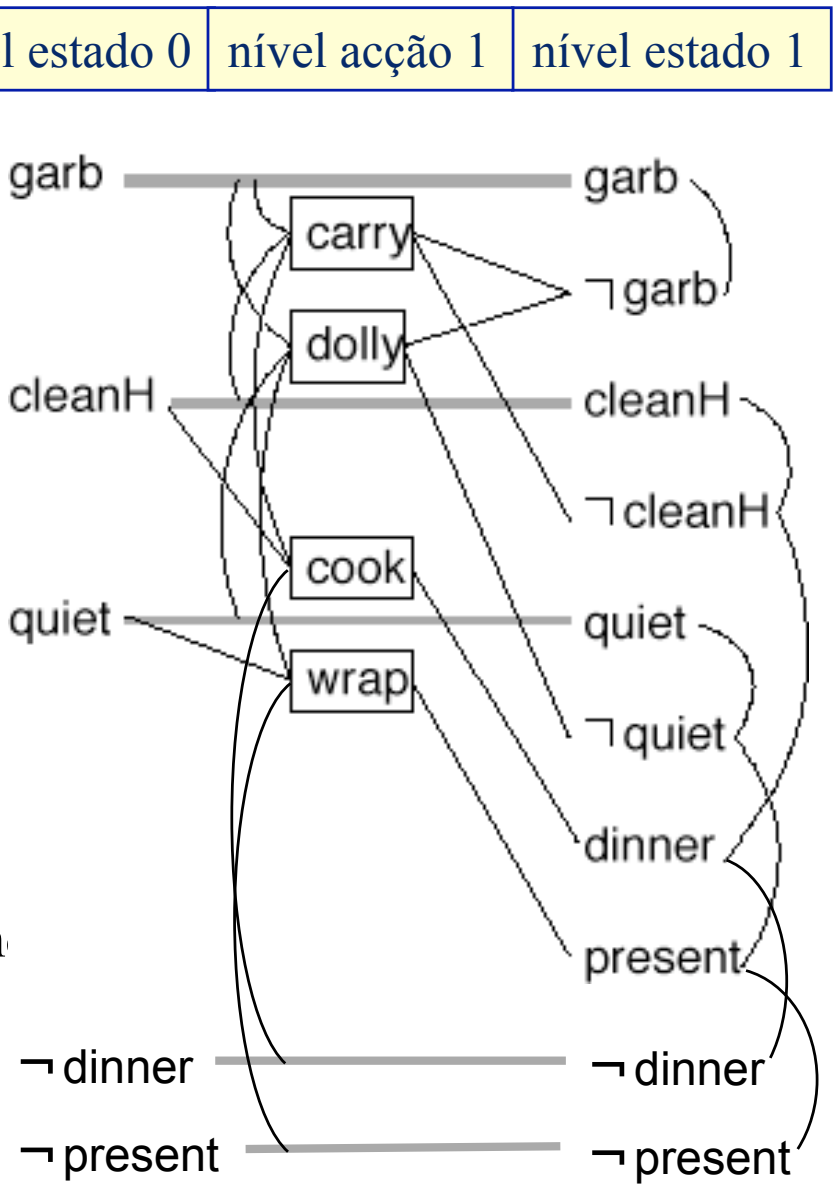
- Duas acções no mesmo nível de acção são mutex se
  - *Efeitos inconsistentes :* o efeito de uma nega o efeito da outra
  - *Interferência:* uma remove a precondição da outra
  - *Necessidades em competição:* têm precondições mutuamente exclusivas
- Caso contrário, não interferem uma com a outra
  - Ambas podem aparecer num plano solução
- Dois literais no mesmo nível estado são mutex se
  - *Suporte inconsistente:* um é a negação do outro, ou todas as maneiras de os alcançar são mutex

Exemplo (continuação)

- Aumentar o grafo para indicar mutexes
- *carry* é mutex com acção de persistência para *garbage* (efeitos inconsistentes)
- *dolly* é mutex com *wrap*
  - interferência
- *~quiet* é mutex com *present*
  - suporte inconsistente
- quer *cook* quer *wrap* é mutex com uma acção de persistência

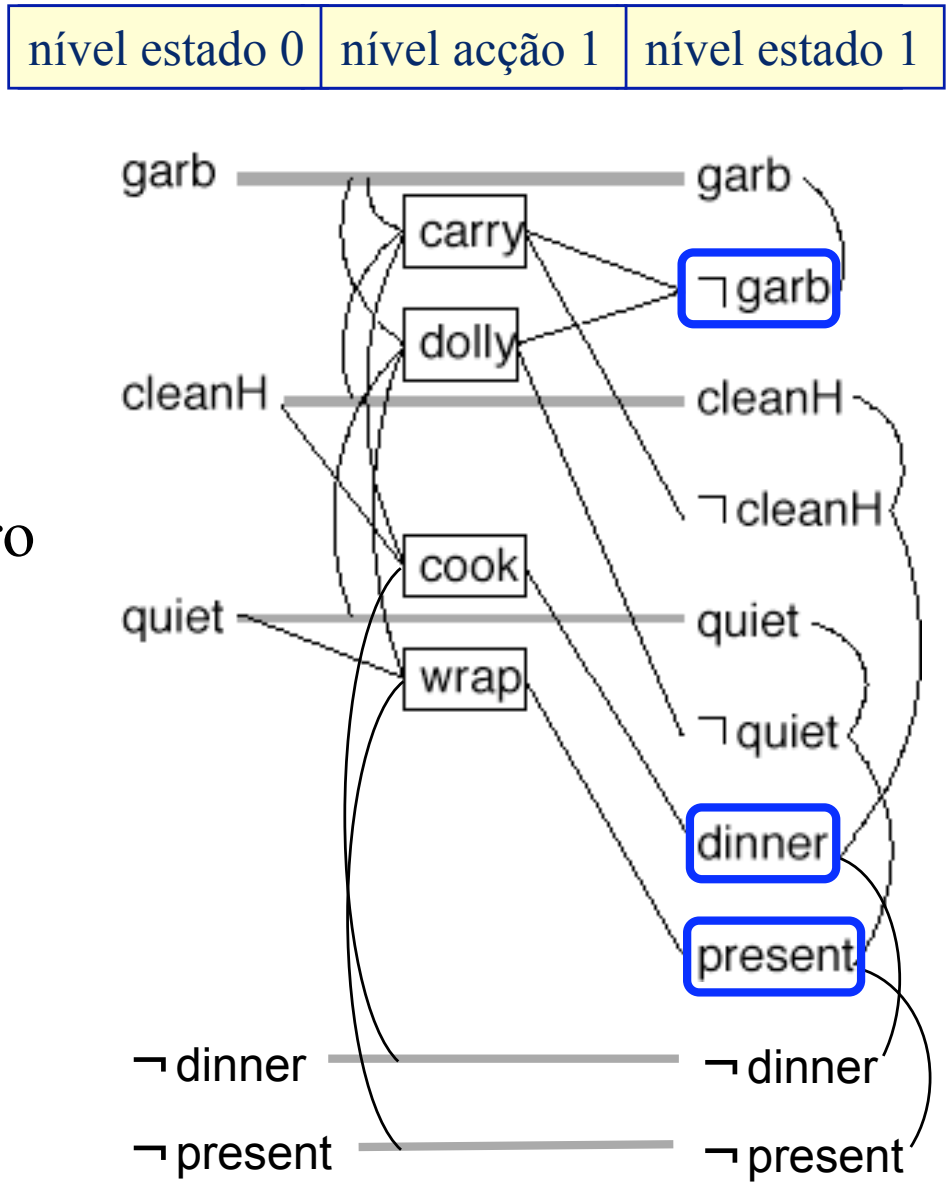
Acção	Precondições	Efeitos
cook()	cleanHands	dinner
wrap()	quiet	present
carry()	*nenhum*	¬garbage, ¬cleanHan
dolly()	*nenhum*	¬garbage, ¬quiet

Juntamente com as acções de persistência



Exemplo (continuação)

- Verificar se existe um plano possível
- O objectivo é
  - $\{\neg garbage, dinner, present\}$
- Repare-se que
  - Todos são possíveis em s_1
  - Nenhum é mutex com qq. outro
- Logo, há hipóteses de um plano existir
- Tentar encontrá-lo
  - Extracção da solução



Extracção da solução

Os objectivos que estamos a tentar atingir

O nível de estado s_j

Uma acção real ou de persistência

procedure Solution-extraction(g,j)

se j=0 então devolver solução

para cada literal l em g

escolher não deterministicamente uma acção a usar no estado s_{j-1} para alcançar l

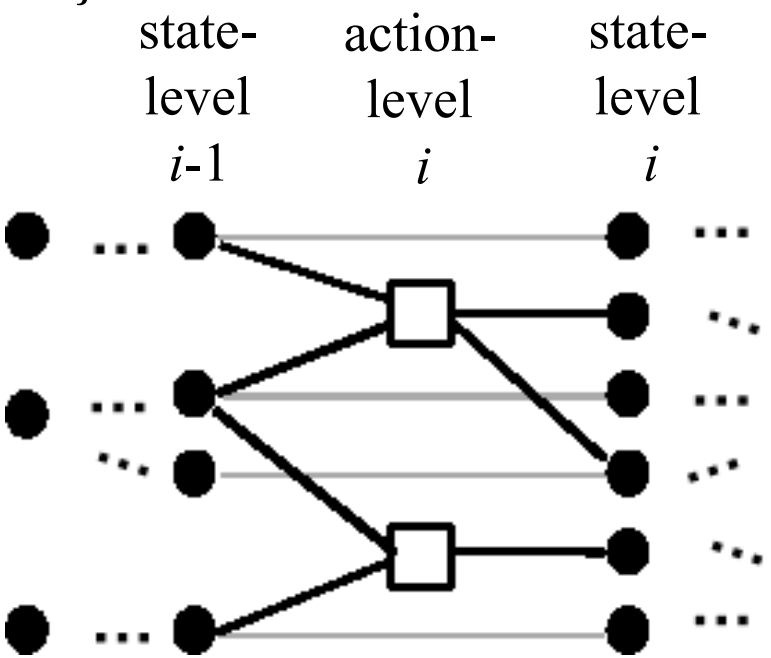
se qualquer par de acções é mutex

então retroceder

g' := {as precondições das acções escolhidas}

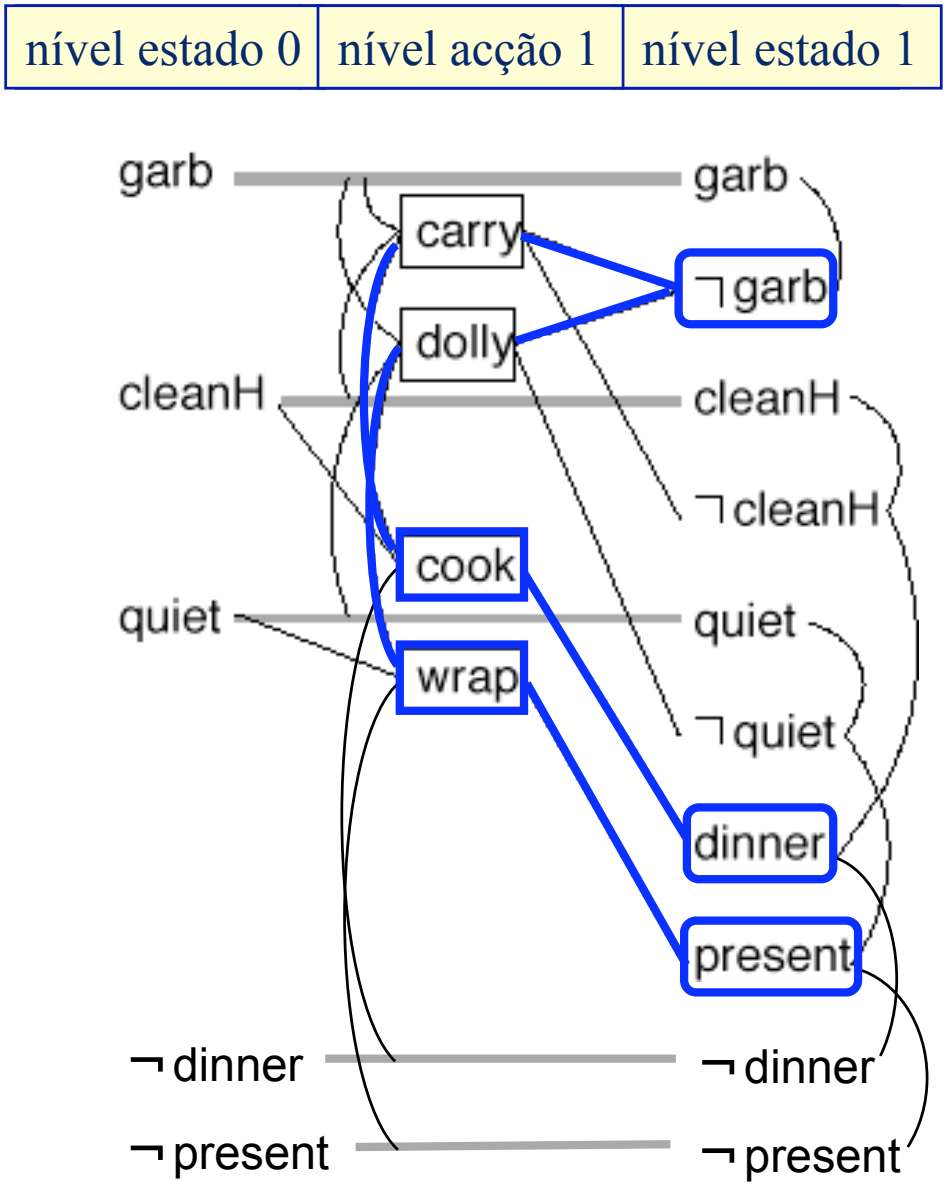
Solution-extraction(g', j–1)

end Solution-extraction



Exemplo (continuação)

- Dois conjuntos de acções para os objectivos no nível estado 1
- Nenhum serve: ambos os conjuntos contêm acções que são mutex



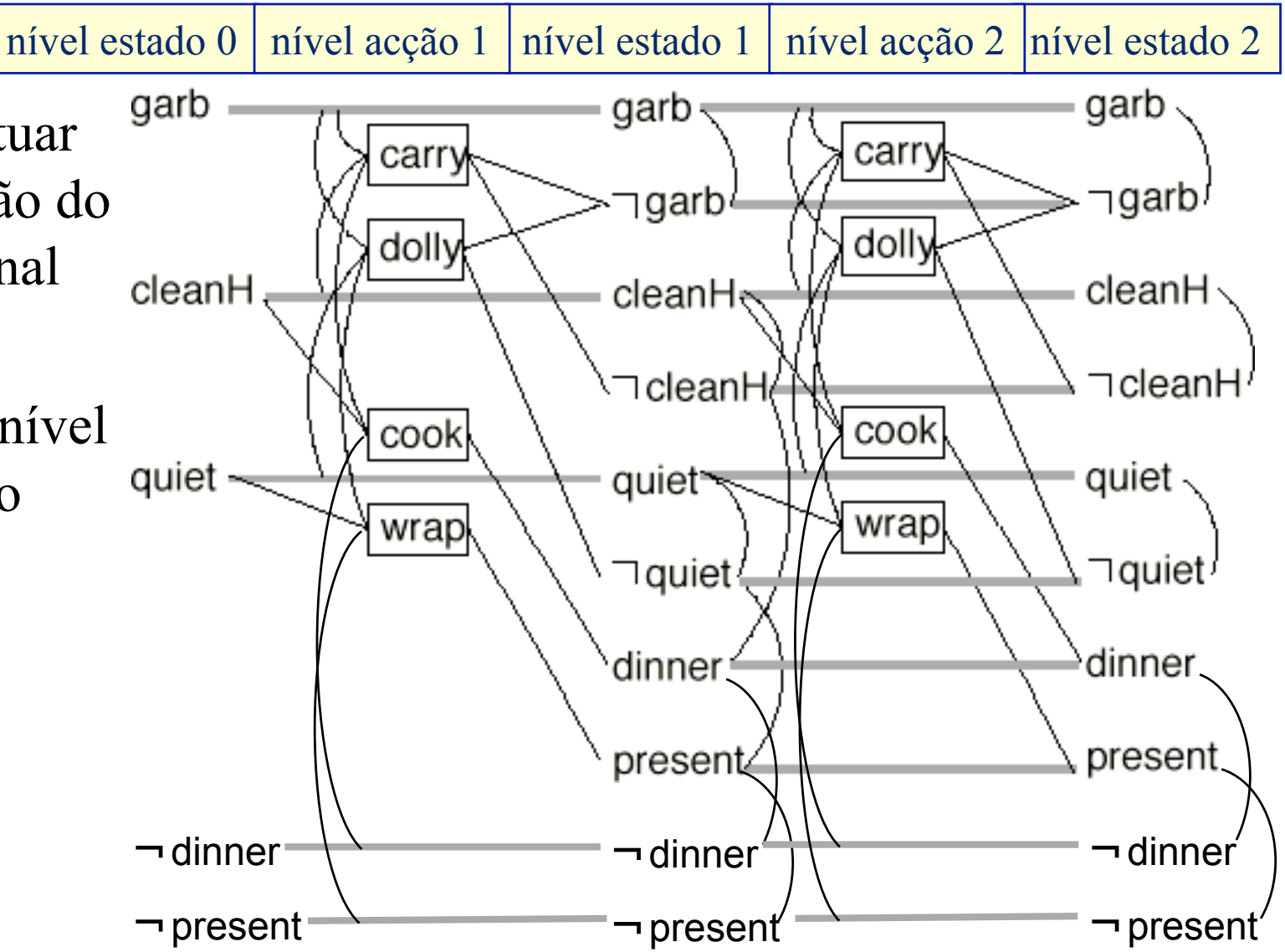
Relembrar como o algoritmo funciona

procedure Graphplan:

- for $k = 0, 1, 2, \ldots$
  - *Expansão do grafo:*
    - » criar um "grafo de planeamento" contendo k "níveis"
  - Verificar se o grafo de planeamento satisfaz uma condição necessária (mas insuficiente) para a existência de um plano
  - Se satisfizer, então
    - » *Extrair solução:*
      - Procura regressiva, modificada para considerar apenas as acções no grafo de planeamento
      - Caso se encontre uma solução, então devolvê-la

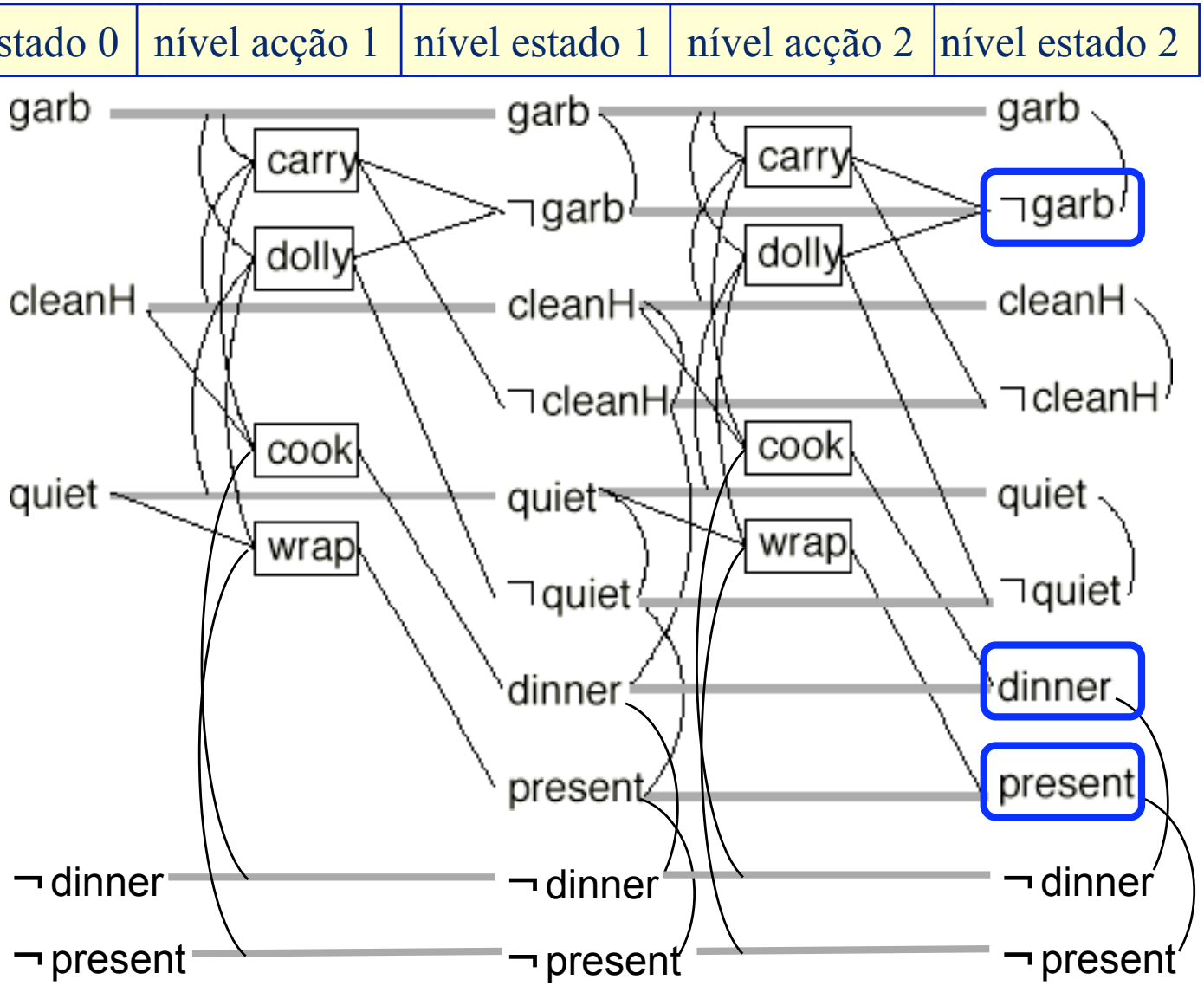
Exemplo (continuação)

- Iterar e efectuar uma expansão do grafo adicional
- Gerar outro nível acção e outro nível estado



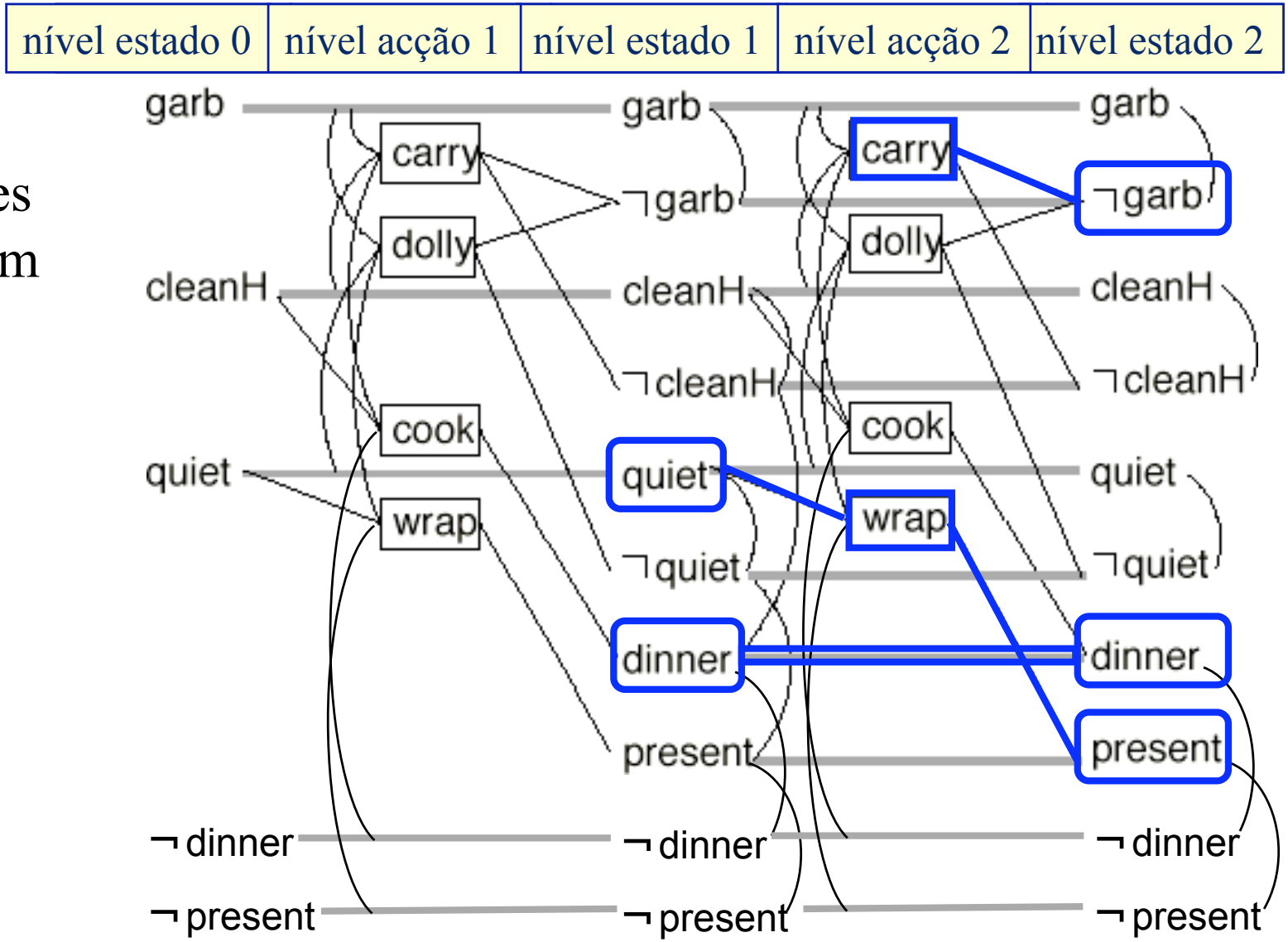
Exemplo (continuação)

- Extracção da solução
- 12 combinações
  - Três maneiras para ¬*garb*
  - Duas maneiras para *dinner*
  - Duas maneiras para *present*



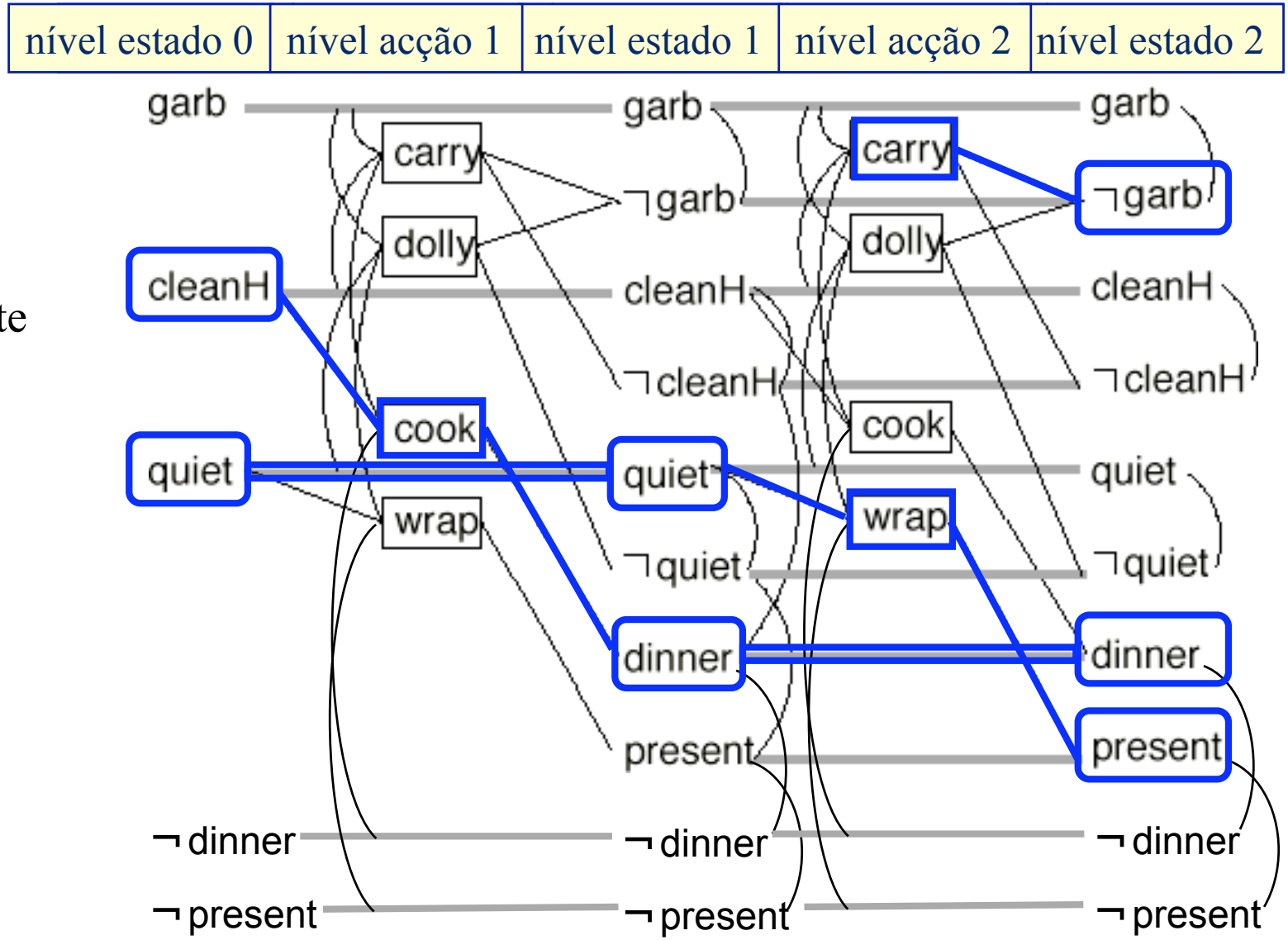
Exemplo (continuação)

- Algumas combinações parecem bem no nível 2
- Assinala-se uma delas



Exemplo (continuação)

- Chamar Solution-Extraction recursivamente no nível 2
- Tem sucesso
- Solução cujo *comprimento paralelo* é 2



Comparação com POP

- Vantagem: a procura regressiva do Graphplan— que é a parte difícil—só olha para as acções no grafo de planeamento
  - Espaço de procura mais pequeno do que o do POP; logo mais rápido

- Desvantagem: para gerar o grafo de planeamento, Graphplan cria um número enorme de literais concretos (instanciados)
  - Muitos deles podem ser irrelevante
- Pode-se aliviar este problema (mas não eliminar) atribuindo tipos de dados a variáveis e constantes
  - Só se instanciam variáveis com termos do mesmo tipo de dados

Heurísticas para o POP

- O grafo de planeamento pode ser gerado em tempo polinomial e termina sempre após um número finito de passos:
  - Critério de paragem: quando dois níveis são idênticos.
- Pode-se utilizar o grafo de planeamento para obter heurísticas para o POP.
- Constrói-se o grafo de planeamento até obter um nível estado em que todos os objectivos sejam alcançados e não sejam mutex dois a dois.
- A heurística assim obtida é admissível e pode ser utilizada conjuntamente com o POP, como por exemplo no sistema RePOP de Nguyen e Kambhampati (2001)

Planeamento proposicional

- Com os sucessos obtidos com os algoritmos para resolver o problema da satisfatibilidade surgiu a ideia de os utilizar para atacar o problema do planeamento
- Assim, surgem uma série de algoritmos de planeamento baseados na tradução para problemas de satisfatibilidade em lógica proposicional.
- A junção com o grafos de planeamento resulta em grandes ganhos de performance…

Tradução genérica

- Proposicionalizar as acções
- Definir o estado inicial: juntar F^0 para cada fluente no estado inicial e $\neg F^0$ para cada fluente não mencionado no estado inicial
- Proposicionalizar o objectivo
- Adicionar os axiomas de estado sucessor
- Adicionar axiomas de precondição: para cada acção concreta A juntar $A^t \Rightarrow PRE(A)^t$
- Adicionar axiomas de exclusão de acções.

Aproximação Genérica

- Um *problema de planeamento limitado* é um par (P,n):
  - P é um problema de planeamento; n é um inteiro positivo
  - Qualquer solução de comprimento n para P é solução para (P,n)

- Algoritmo de planeamento:
- Efectuar aprofundamento progressivo ou procura em profundidade limitada (SATPLAN):
  - for $n = 0, 1, 2, \ldots,$
    - » codificar (P,n) como problema de satisfatibilidade Φ
    - » se Φ é satisfazível, então
      - A partir do modelo que satisfaz Φ, um plano solução pode ser construído, logo constrói-o e devolve-o

Problema principal

- A dimensão da fórmula Φ é o principal problema.
- Por exemplo, o número de símbolos de proposição para representar acções é limitado por $n * |Acc| * |O|^P$ em que n é o comprimento do plano, $|Acc|$ é o número de esquemas de acções no domínio, $|O|$ o número de objectos e P a aridade máxima dos fluentes.
- O número de cláusulas é ainda maior…
- Uma das técnicas consiste em substituir cada fluente de aridade m por m símbolos proposicionais
  - Por exemplo, move(r1,l2,l1,0) pode ser codificado através de:
    $move_1(r1)^0$
    $move_2(l2)^0$
    $move_3(l1)^0$
  - Esta técnica permite na prática limitar muito o número de cláusulas geradas

BlackBox

- O sistema BlackBox combina expansão do grafo de planeamento com verificação de satisfatibilidade
  - Resumidamente:

- for $n = 0, 1, 2, \ldots$
  - *Expansão do grafo:*
    - » Criar um "grafo de planeamento" com n "níveis"
  - Verificar se o grafo de planeamento satisfaz uma condição necessária (mas insuficiente) para a existência de um plano
  - Se satifazer, então
    - » Codificar (P,n) como um problema de satisfatibilidade Φ mas incluindo apenas as acções no grafo de planeamento
    - » Se Φ for satisfazível então devolver a solução

Mais acerca do BlackBox

- Os requisitos de memória continuam a ser enormes, mas menos do que a técnica de satisfatibilidade sozinha
- Foi um dos planeadores mais rápidos na competição de 1998
- O seu sucessor SatPlan 2006 ficou em primeiro lugar empatado no ICAPS-2006 Planning Competition (IPC-6) na categoria de planeamento óptimo.

Planeamento com ASP

- Técnicas semelhantes (mas mais declarativas) podem ser utilizadas para reduzir problemas de planeamento ao cálculo de modelos estáveis de problemas de planeamento limitados.
- Existem diversas traduções que não abordaremos aqui.