Planejamento em Inteligência Artificial

Leliane Nunes de Barros

Motivação

- Um dos principais objetivos da IA foi/é o desenvolvimento de um ***Solucionador Geral de Problemas*** [Newell & Simon, 1961]

Problema $\Longrightarrow$ Linguagem $\Longrightarrow$ $\boxed{\text{Planejador}}$ $\Longrightarrow$ Solução

- **Idéia**: problemas são descritos numa linguagem de alto-nível de abstração e resolvidos automáticamente
- **Objetivo**: facilitar a modelagem de problemas com um prejuízo mínimo em termos de desempenho.

Escopo do planejador

- Planejamento não é tão geral como GPS!
- Exemplo:
  - a solução do Cubo Mágico é uma *sequência de ações*
  - a solução para um problema de diagnóstico é uma *estratégia de ações* sendo a determinação precisa dos testes a serem realizados (ações), uma função das observações coletadas ==> técnica de *planejamento condicional* ou técnicas de *planejamento e execução*
- Para definir uma linguagem e um algoritmo é preciso definir um escopo

Modelos Matemáticos

- Modelos definem o escopo de um planejador:
  - o que é um problema de planejamento
  - o que é uma solução (plano)
  - o que é uma solução ótima

Componentes de Planejamento

- **Modelos** para compreender os algoritmos e estabelecer classes de problemas e tipos de solução
- **Linguagens** para representação de problemas
- **Algoritmos** para resolver esses problemas (usando informações disponíveis na representação dos problemas)

Modelo de Estados

O planejamento clássico pode ser caracterizado como um Modelo de Estados

- um espaço de estados S , finito e não-vazio
- um estado inicial $s_0 \in S$
- um conjunto de estados meta $S_G \subseteq S$
- um conjunto de ações aplicáveis $A(s) \subseteq A$, para $\forall s \in S$
- uma função de transição $s' = f(a, s)$, para $\forall s', s \in S$ e $\forall a \in A(s)$
- custos das ações $c(a,s) > 0$

Modelo de Estados

Uma **solução** é uma sequência de ações aplicáveis que mapeiam s_0 no estado meta S_G .

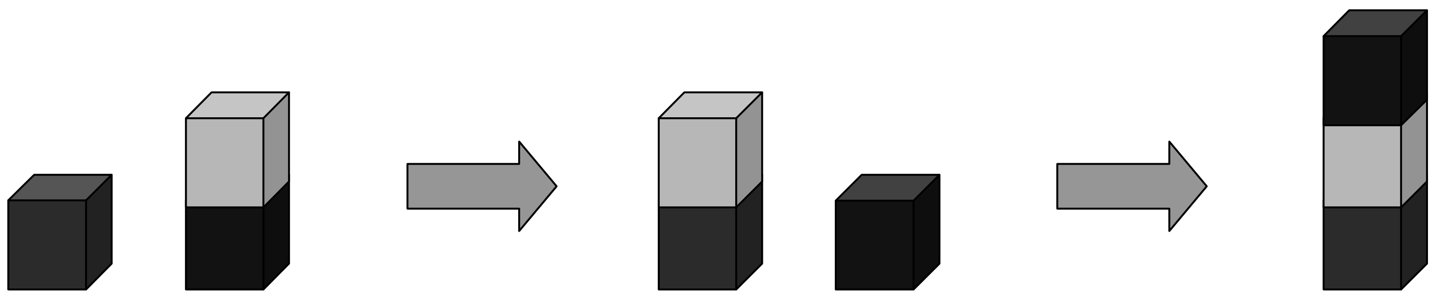
Uma **solução ótima** minimiza a soma dos custos das ações.

planejamento com sensoriamento
planejamento com recursos
planejamento temporal
...

⟹ Modelos diferentes

Um problema simples

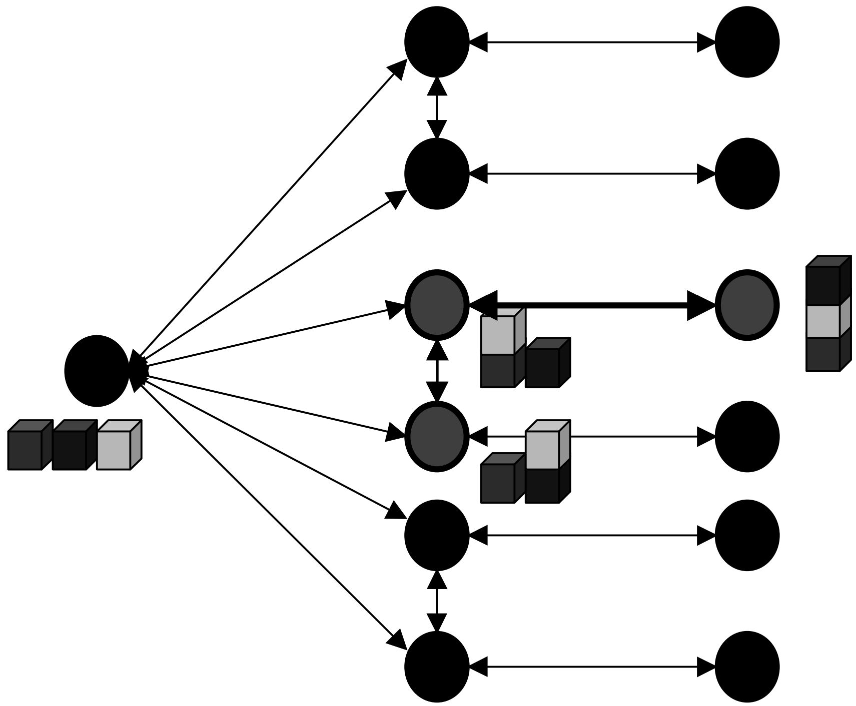
- **Domínio** = Estados (posições dos blocos) + Ações (*move*)
- **s_0:** (“*red on table*”, “*blue on table*”, “*green on blue*”)
- **S_G:** (“*red on table*”, “*green on blue*”, “*blue on green*”)



- **Plano:** (“*move green on red*” 〉 “*move blue on green*”)
- **Problema de Planejamento:**

  Dado um domínio (estados e ações), um estado inicial e o estado meta, o problema de planejamento é encontrar um plano de ações que leva do estado inicial a estado meta

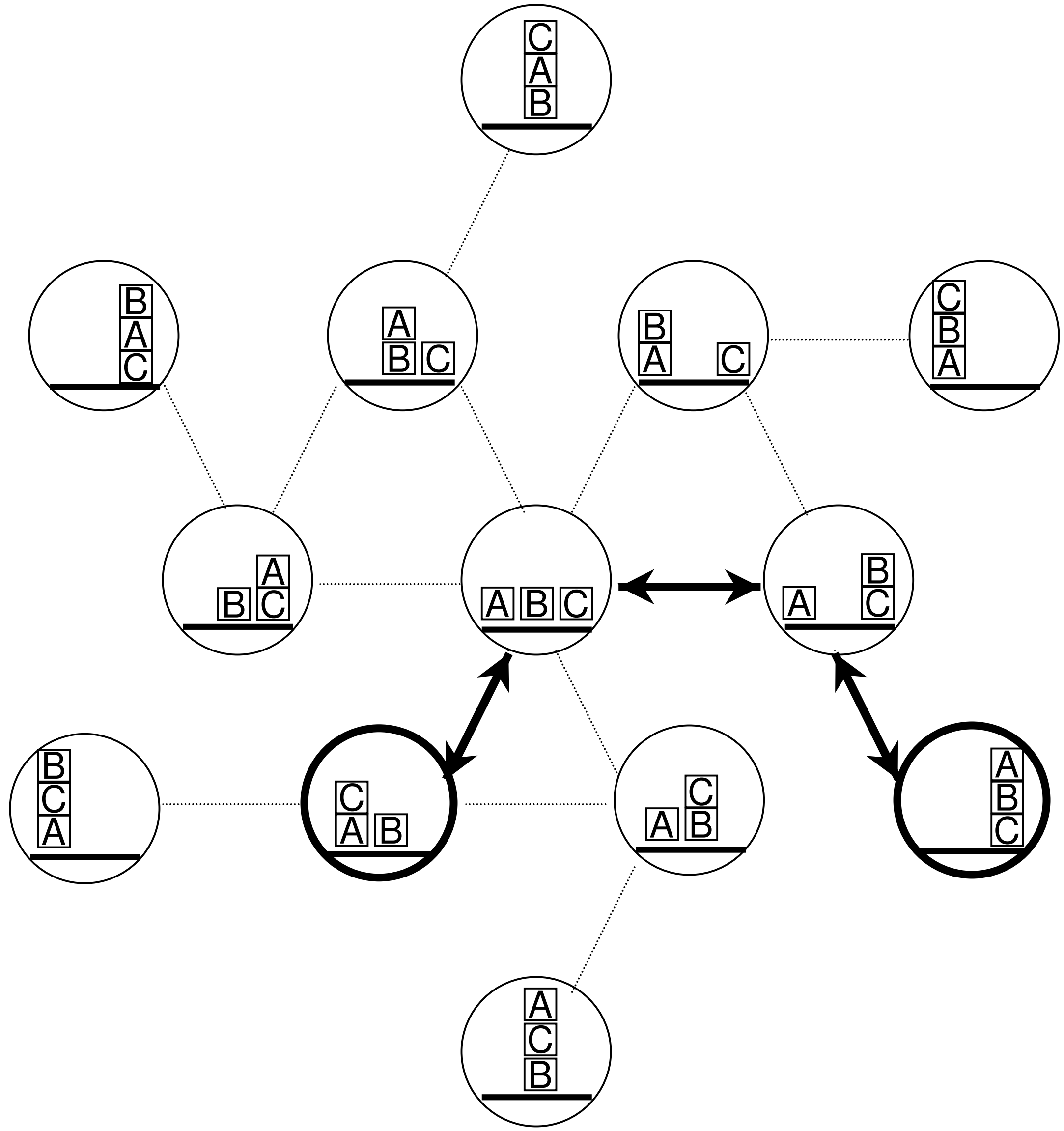
Um problema simples



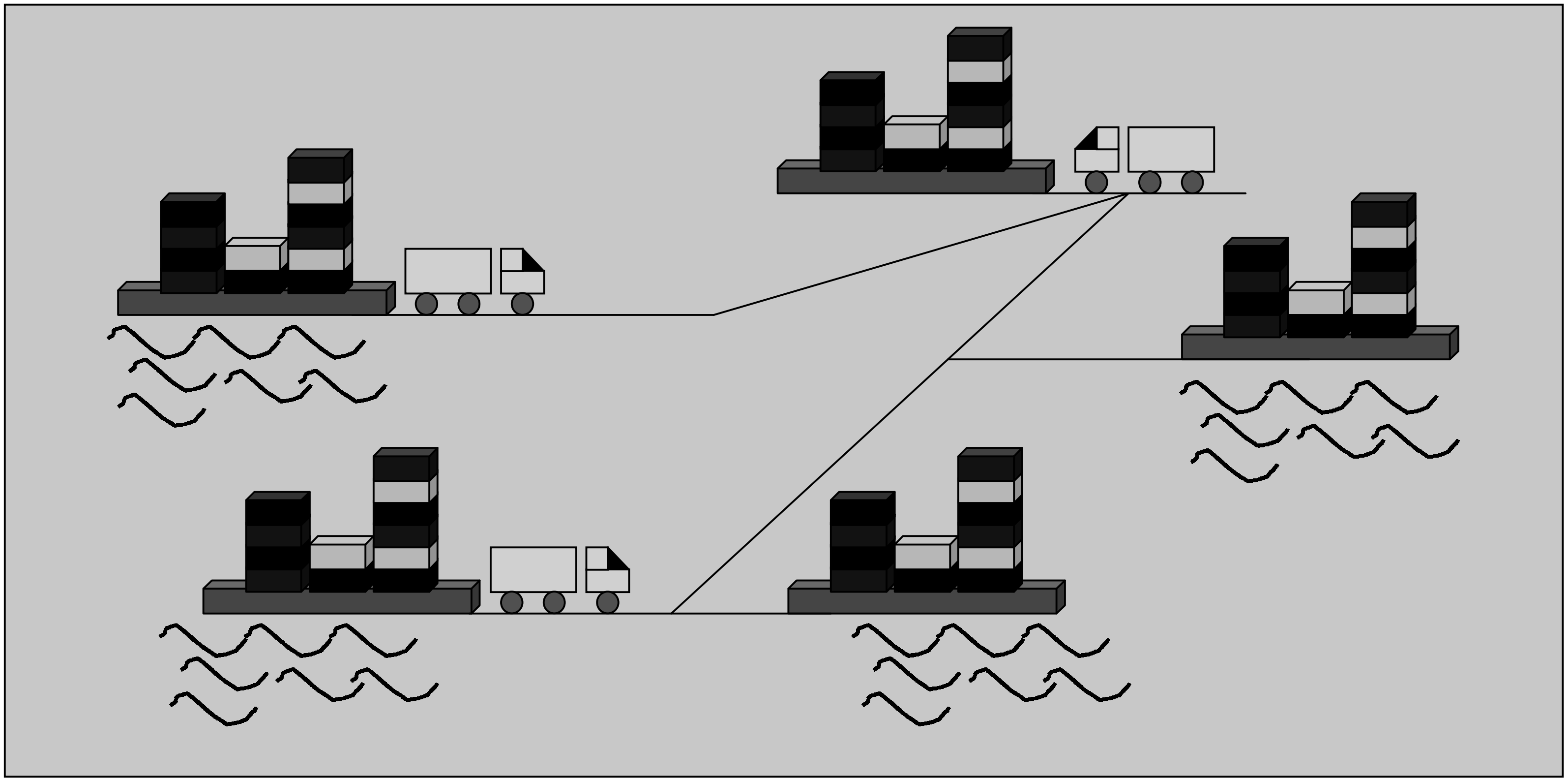
- Domínio = Espaço de Busca
- Problema:

  encontrar um caminho que ligue o estado inicial e o estado meta

Espaço de estados



Um problema difícil



5 localidades, 3 pilhas, 100 containers $\rightarrow 10^{277}$ estados

Outras áreas e modelos

- Modelos para PO, controle, combinatória, ...
  - programação linear
  - programação inteira
  - SAT
  - CSP ...

áreas que começam
a fazer distinção entre
modelos e linguagens

- Planejamento em IA difere de 2 maneiras:
  - trata uma classe diferente de modelos
  - modelos são representados implicitamente na linguagem

Modelos Matemáticos em Planejamento

- Modelo de Estados: tipo mais simples
- Modelo de Estados Probabilístico e Não-determinístico: dinâmica mais complexa
- Modelo de Estados Probabilístico e Não-determinístico com informação parcial do mundo: percepção mais complexa (realimentação)
- modelos ainda mais complexos envolvem aspectos de escalonamento: recursos, tempo, ações durativas, concorrência, etc

Modelos Lógicos em Planejamento

Modelos lógicos definem mais do que o escopo de planejamento mas também a linguagem e o tipo de raciocínio

- Cálculo de Situações com raciocínio dedutivo (originalmente proposto em 63)
- Cálculo de Eventos com raciocínio abdutivo
- outras lógicas

A grande maioria das lógicos de ações não "deliberam" planos (planejam) mas só permitem o raciocínio sobre ações e planos (dados).

Ingredientes de Planejamento

~~Modelos~~

Linguagens

Algoritmos

Linguagens de Planejamento

Linguagens de planejamento servem para:

– descrever problemas
– descrever estados
– descrever ações

Ocupam dois papéis:

– especificação: descrição precisa do problema
– computação: revelam informações heurísticas

Linguagem dos Modelos Lógicos

- Adota-se uma linguagem lógica para descrever estados, ações e problemas
- Vantagem: pode-se utilizar um mecanismo de inferência lógica como algoritmo

A melhor maneira de se explicar o comportamento inteligente é interpretando-o como produto de um raciocínio correto sobre uma representação correta ==>

principal motivação da Robótica Cognitiva

Cálculo de situações

- Ontologia: situações, fluentes e ações
- Linguagem
  - s_0
  - $do(\alpha, \sigma)$
  - $poss(\alpha, \sigma)$
  - $holds(\phi, \sigma)$

Exemplos

- fluentes:

  $clear(X)$

  $ontable\ (X)$

  $on(X,Y)$

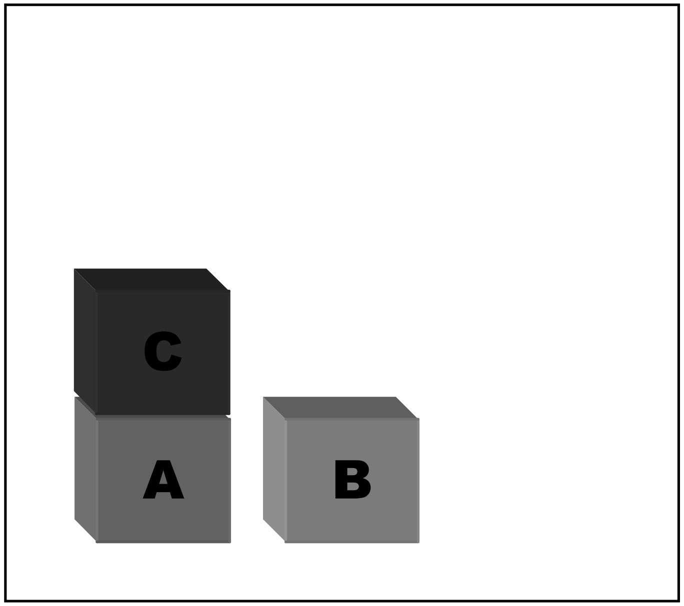
- ações:

  $stack\ (X,Y)$

  $unstack\ (X,Y)$

  $move\ (X,Y,Z)$

s_0



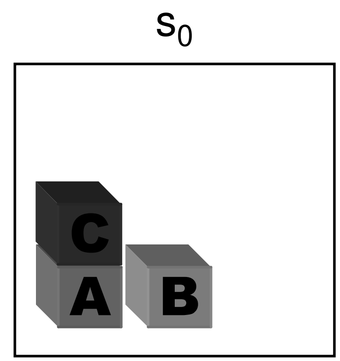
Especificação lógica do domínio

- axiomas de observação:

$holds\ (clear\,(c), s_0)$

$holds\ (on\,(c,a), s_0)$

...



- axiomas de efeito:

$holds\ (clear\,(Y),\ do\,(move\ (X,Y,Z),S\,))$

$holds\ (on(X\,,Z),\ do\,(move\ (X,Y,Z),S\,))$

...

- axiomas de precondições:

$poss\,(move\ (X,Y,Z),S\,) \leftarrow$

$\quad holds\ (clear\,(X),S\,) \wedge holds\ (clear\,(Z),S\,) \wedge holds\ (on\,(X,Y\,),S\,)$

…

- axioma de persistência (*frame axiom*):

holds(F,do(A,S)) :- poss(A,S), holds(F,S), not affects(A,F).

O problema da persistência

s_0 $\quad\quad\quad\quad\quad\quad\quad\quad$ $s_1 := do(move\,(c,a,b),s_0)$

$holds(\;clear(b)\,,\,s_0)$
$holds(\;clear(c)\,,\,s_0)$
$holds(\;ontable(a)\;,\,s_0)$
$holds(\;ontable(b)\;,\,s_0)$
$holds(\;on(c,a)\,,\,s_0)$

axiomas de efeito

$holds(\;clear(a)\,,\,s_1)$
$holds(\;on(c,b)\,,\,s_1)$

$holds(\;clear(c)\,,\,s_1)$
$holds(\;ontable(a)\;,\,s_1)$
$holds(\;ontable(b)\;,\,s_1)$

axiomas de persistência

Planejamento dedutivo

Dados:

- $\mathcal{A}$: axiomatização do domínio
- I : situação inicial
- $\mathcal{G}$: meta de planejamento

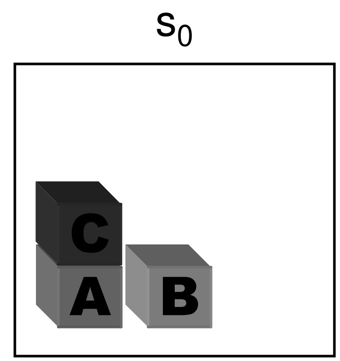
O planejamento consiste em provar que

$$\mathcal{A} \wedge I \models \exists S\,[exec(S) \wedge \mathcal{G}(S)],$$

sendo executabilidade definida indutivamente por:

$$exec(s_0)$$
$$exec(do(A,S)) \leftarrow poss(A,S) \wedge exec(S)$$

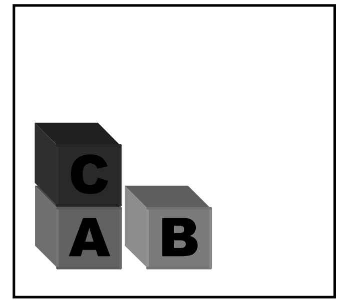
Um planejador em PROLOG




```
holds(clear(b),s0).
holds(clear(c),s0).
holds(ontable(a),s0).
holds(ontable(b),s0).
holds(on(c,a),s0).

holds(on(X,Y),do(stack(X,Y),S)).
holds(clear(Y),do(unstack(X,Y),S)).
holds(ontable(X),do(unstack(X,Y),S)).

poss(stack(X,Y),S) :- holds(ontable(X),S), holds(clear(X),S), holds(clear(Y),S), X\=Y.
poss(unstack(X,Y),S) :- holds(clear(X),S), holds(on(X,Y),S).

holds(F,do(A,S)) :- poss(A,S), holds(F,S),  not affects(A,F).

affects(stack(X,Y),clear(Y)).
affects(stack(X,Y),ontable(X)).
affects(unstack(X,Y),on(X,Y)).

exec(s0).
exec(do(A,S)) :- poss(A,S), exec(S).

plan(s0).
plan(do(A,S)) :- plan(S).
```

Consultando o planejador

s_0



?- plan(S), exec(S), holds(on(a,c),S).

S = do(stack(a,c),do(unstack(c,a),s0))
yes

?- plan(S), exec(S), holds(on(a,b),S), holds(on(b,c),S).

S = do(stack(a,b),do(stack(b,c),do(unstack(c,a),s0)))
yes

?- holds(F, do(stack(a,b),do(stack(b,c),do(unstack(c,a),s0)))).

F = on(a,b) ;

F = on(b,c) ;

F = clear(a) ;

F = ontable(c) ;
no

A linguagem Strips (Fikes e Nilsson, 1971)

- Strips (*Stanford Research Institute Problem-Solving*) é a mais antiga, simples e usadas linguagem de planejamento
- evolução: Strips $\rightarrow$ ADL $\rightarrow$ PDDL2.1 $\rightarrow$ PDDL+
- Um problema em Strips é uma tupla $\langle A, O, I, G \rangle$
  - A é o conjunto de todos os átomos (variáveis booleanas como descritores de estados),
  - O é um conjunto de todos os operadores (ações proposicionais), e
  - $I \subseteq A$ representa a situação inicial
  - $G \subseteq A$ representa as situações meta

A linguagem Strips

- Os operadores $o \in O$ são representados por três listas:
  - lista Pre, $Pre \subseteq A$
  - lista Add, $Add(o) \subseteq A$
  - lista Del, $Del \subseteq A$

Intuitivamente: $Pre(o)$ especifica os átomos que devem ser verdadeiros para o ser aplicável, $Add(o)$ especifica os átomos que passam a ser verdadeiros após a execução de o e $Del(o)$ especifica os átomos que passam a ser falsos após a execução de o

Strips: da linguagem aos modelos

Um problema Strips, $P = \langle A, O, I, G \rangle$ determina um modelo de estados $S(P)$ em que:

- os estados $s \in S$ são coleções de átomos;
- o estado inicial s_0 é I
- os estados meta $s \in S_G$ são aqueles em que $G \subseteq s$
- as ações $a \in A(s)$ são operadores $o \in O$ tal que $Prec(o) \subseteq s$
- a função de transição *f* mapeia estados *s* em estados $s' = f(s, a)$, tal que $s' = s - Del(a) + Add(a)$ para $a \in A(s)$
- os custos das ações são iguais a 1
- a solução (ótima) de um problema de planejamento é a solução (ótima) do Modelo de Estados $S(P)$

STRIPS: sintaxe e semântica

$oper(act: move(c,a,b),$
$\quad pre: \{ clear(c), clear(b), on(c,a) \},$
$\quad add: \{ clear(a), on(c,b) \},$
$\quad del: \{ clear(b), on(c,a) \})$

clear(b)	*clear(b)*	*clear(b)*	~~*clear(b)*~~
clear(c)	*clear(c)*	*clear(c)*	*clear(c)*
ontable(a)	*ontable(a)*	*ontable(a)*	*ontable(a)*
ontable(b)	*ontable(b)*	*ontable(b)*	*ontable(b)*
on(c,a)	*on(c,a)*	*on(c,a)*	~~*on(c,a)*~~
		clear(a)	*clear(a)*
		on(c,b)	*on(c,b)*

PDDL+

- Inclui: tipos, funções, variáveis numéricas, ações durativas, funções de otimização ==> planejamento/escalonamento
- AIPS 2002 Planning Competition http://www.dur.ac.uk/d.p.long/competition.htm

Ingredientes de Planejamento

~~Modelos~~

~~Linguagens~~

Algoritmos

Algoritmos para resolver Modelos de Estado

- Problemas de planejamento podem ser resolvidos por algoritmos de busca no espaço de estados $S(P)$ ==> técnica considerada ineficiente até recentemente …
- Espaço de busca:
  - espaço de progressão (*forward planning*)
  - espaço de regressão (*backward planning*)
- Algoritmos de busca exaustiva
  - Os estados meta não são levados em conta: *DFS, BFS, Custo Uniforme, ID*

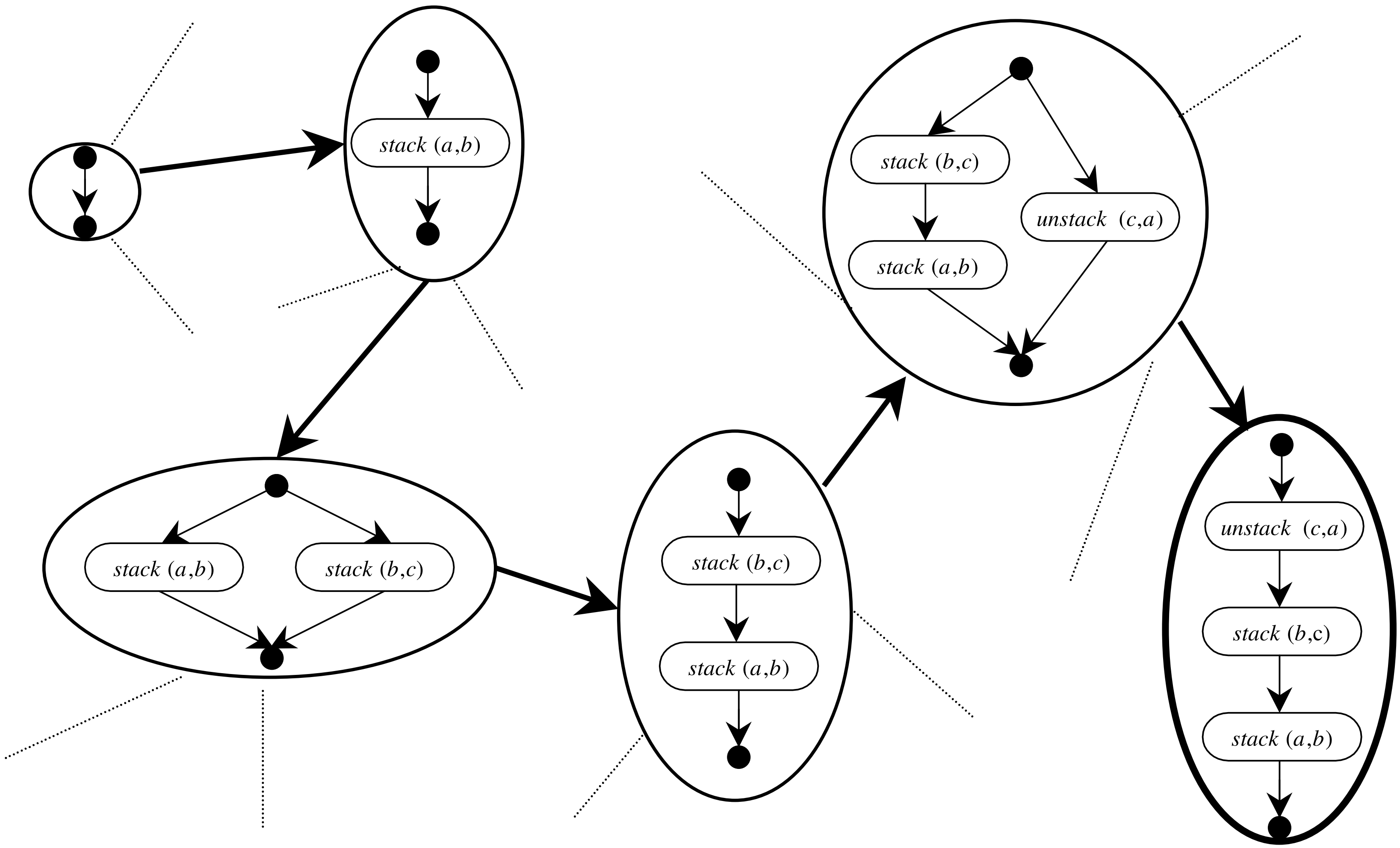
Algoritmos para resolver Modelos de Estado

- Algoritmos de busca informada
  - Para controlar o processo, a busca usa uma função $h(s)$ que estima a distância (custo) do estado s ao estado S_G *: A*, IDA*, Best First Search, Hill Climbing, Branch & Bound*

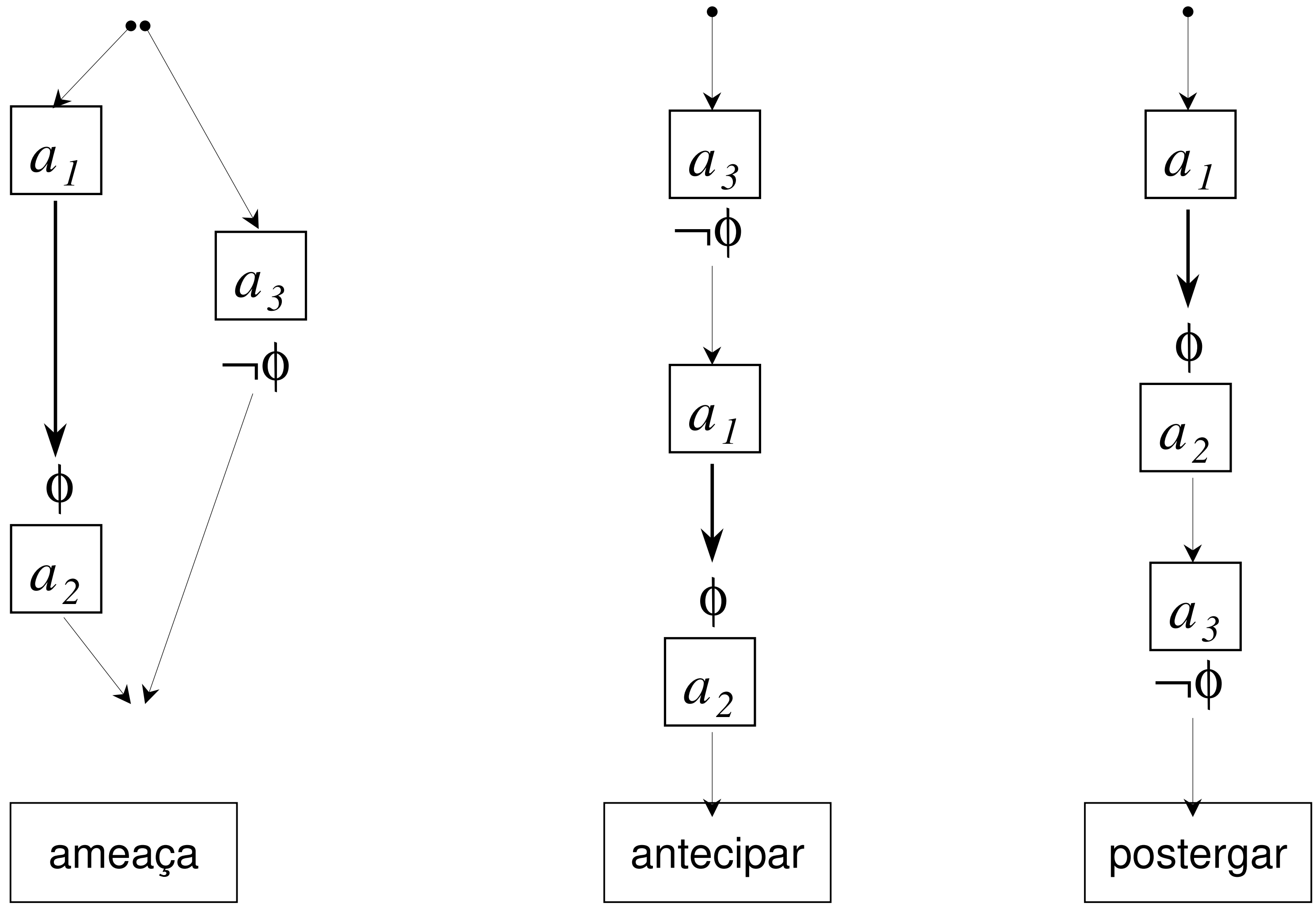
Planejadores conhecidos

- Algoritmo Strips (71): planejamento totalmente ordenado:
  - Análise means-ends (50): *adicione ações no plano que sejam aplicáveis e atinjam sub-metas de G*
  - Dificuldade: em geral, problemas interessantes não existem ações que sejam aplicáveis e relevantes para satisfazer a meta, ao mesmo tempo.
  - Solução Strips (71): construir estados da busca mais complicados mantendo uma hierarquia de sub-metas (pilha de sub-metas) e planos incompletos ==> **Busca num Espaço de Planos**

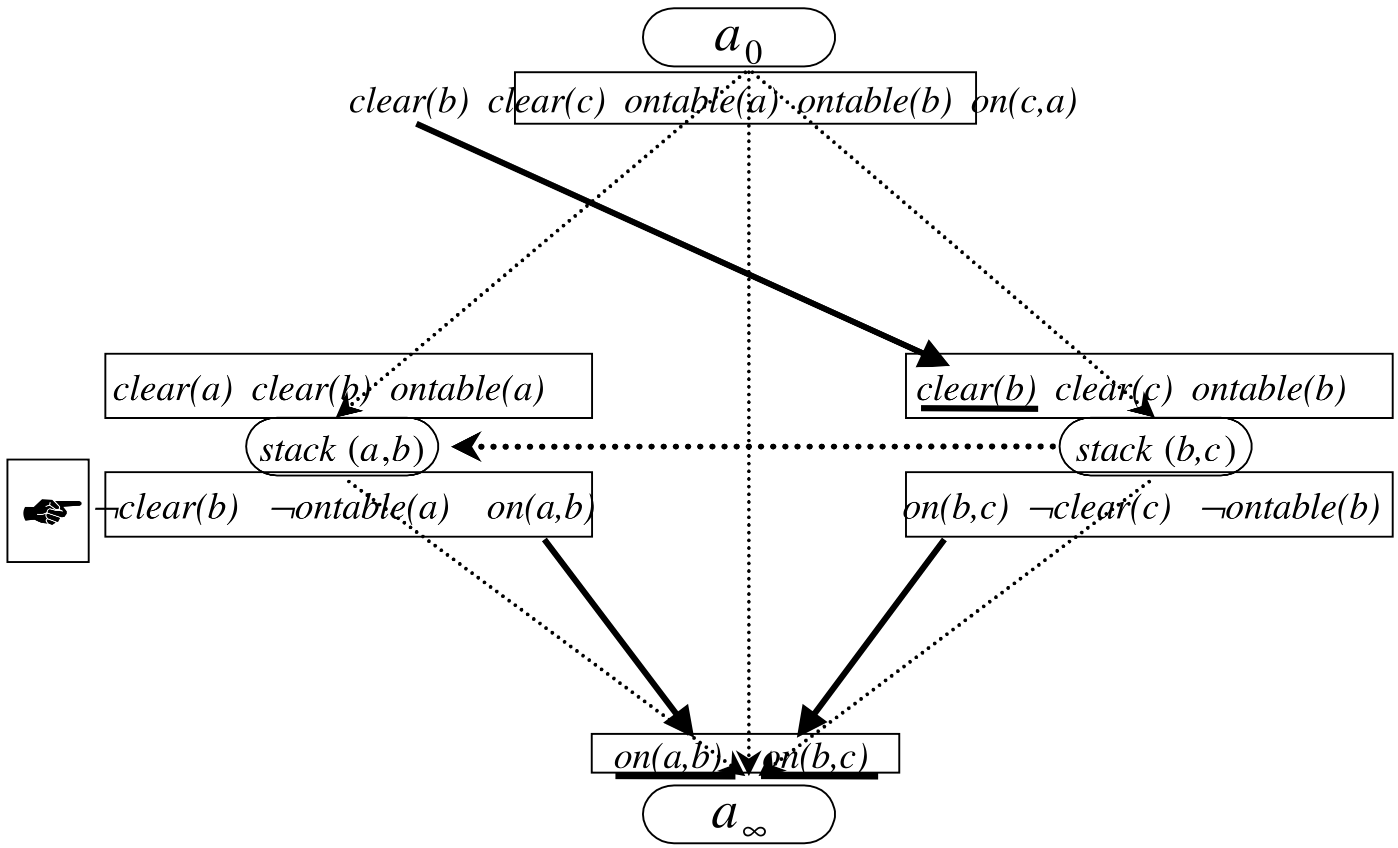
Busca no espaço de planos



Ordem parcial, vínculos e ameaças



Ordem parcial, vínculos e ameaças



[redacted]

Algoritmo POP

function POP(*initial, goal, operators)* **returns** *plan*

plan $\leftarrow$ Make-Minimal-Plan(*initial, goal*)

loop do

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

S_{need} , c $\leftarrow$ 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 algorithm (cont.)

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 algorithm (cont.)

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 $S_j < S_{threat}$ to Orderings(plan)

if not consistent(*plan*) **then fail**

end

POP é correto, completo e sistemático (busca sem repetição)

Algoritmos Originais

- Graphplan (1995 - …)
  – constrói um grafo de planejamento contendo todos os planos possíveis de um determinado comprimento para então extrair um plano por meio de uma busca regressiva no grafo
- SatPlan (1996 - …) mapeia problemas de planejamentp em um problema SAT e usa um SATsolver eficiente* .
- Blackbox: = Satplan + Graphplan
- Planejamento como Busca Local com uso de Heurísticas … algoritmos de maior sucesso nas últimas competições de planejamento

Planejamento como busca heurística

- UNPOP [Drew McDermott, 96] *Greedy Regression-Match Graphs*
- HSP [Geffner, 97] *Heuristic Search Planner*
- FF [Hoffmann, 2000] *Fast Forward*
- ...

Planejamento Heurístico

- Heurística h(s) é computada resolvendo um problema relaxado
- Exemplo: Heurística da distância de Manhattan
- Em planejamento: comprimento do plano solução de um problema relaxado em que são removidas todas as listas de eliminação das ações
- Heurística informativa e admissível … mas ainda intratável

HSP

- Forward planning
- algoritmos de busca: Hill-Climbing com uso de uma função heurística derivada de uma *descrição de alto-nível de ações*

Planejamento Heurístico: HSP

Função heurística aditiva

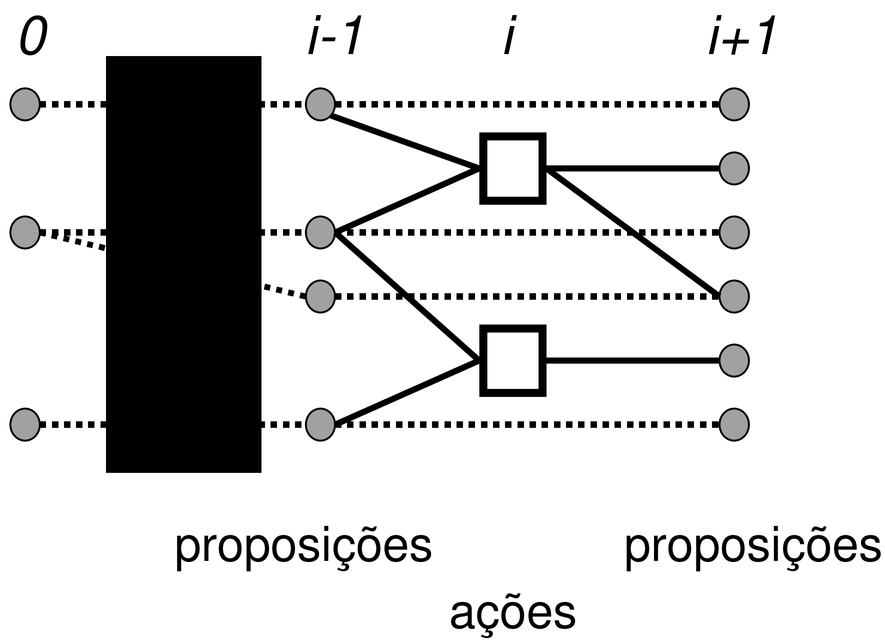
- heurística $h(s)$ é uma estimativa do número de passos necessários para resolver o problema relaxado, ou seja, ações sem listas de eliminação.
- s: estado, p: proposição,
- $c_s(p)$: estimativa do custo para atingir p apartir de s
- $c_s(Prec(op))$: custo estimado para se atingir as precondições da ação op de s

$$c_s(p) = \begin{cases} 0 & \textit{se } p \in s \\ \min_{op \in O(p)} \left[1 + c_s(Prec(op))\right] & \textit{caso contrário} \end{cases}$$

Heurística do HSP (não-admissível): $h(s) = \sum_{p \in G} c_s(p)$

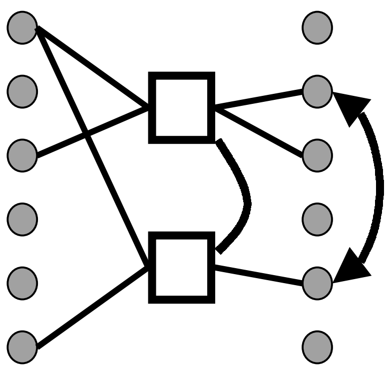
Grafo de Planejamento

- Planning Graph **[Blum&Furst 1997]**
  - expansão do grafo

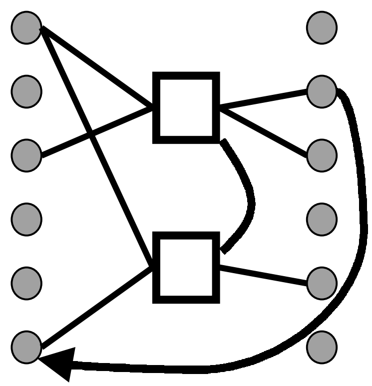


Grafo de Planejamento

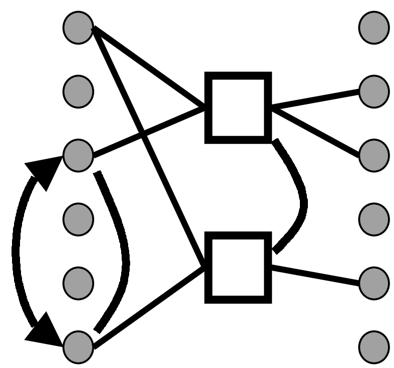
- Grafo de Planejamento
  - Relações de Mutex



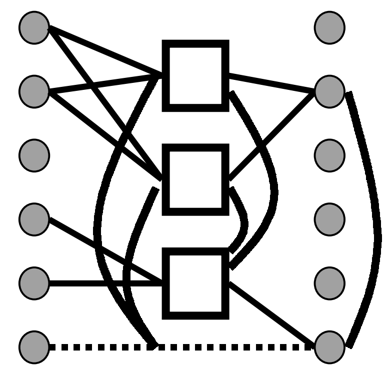
Efeitos Inconsistentes



Interferência



Necessidades conflitantes

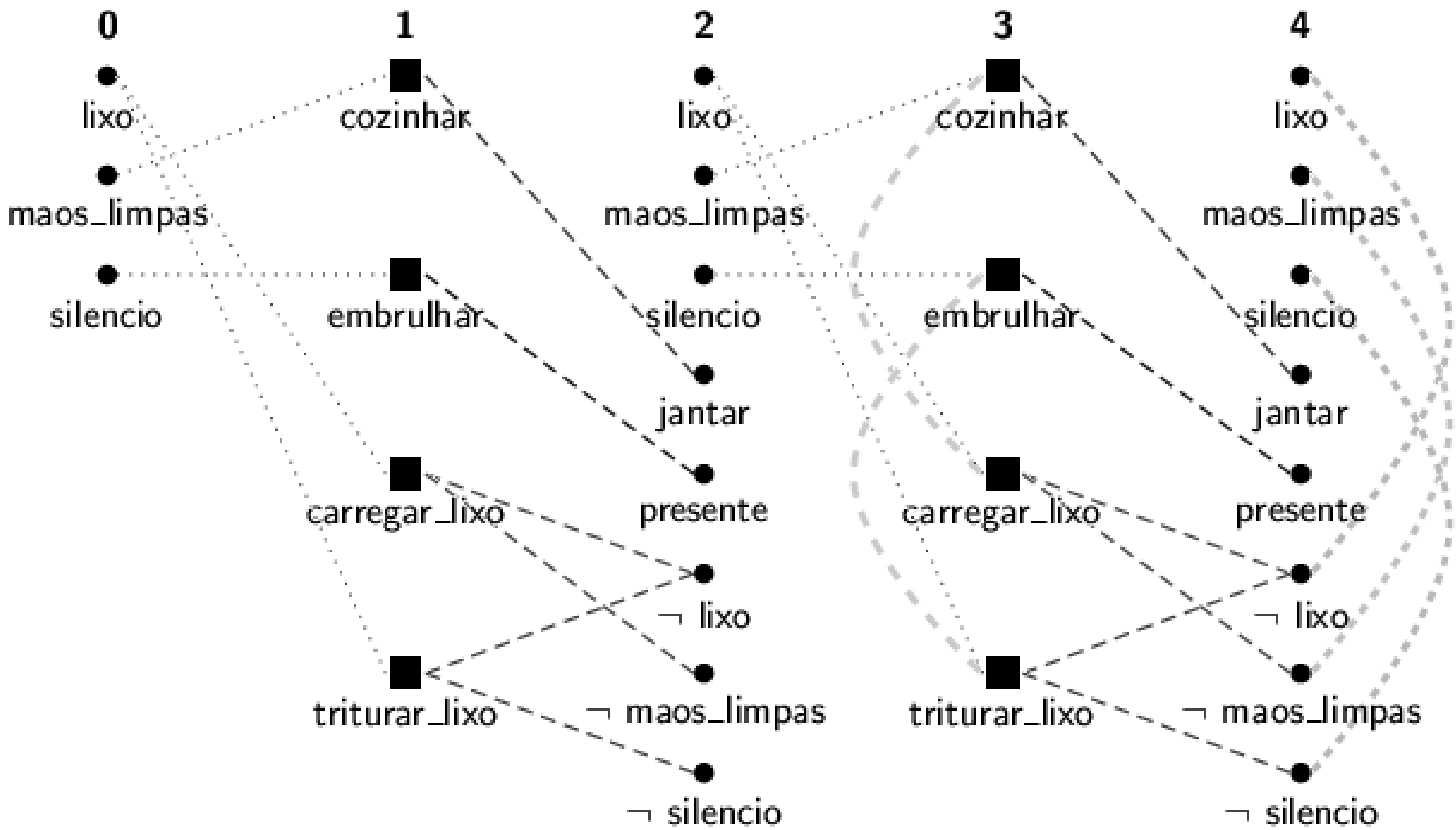


Suporte Inconsistente

Domínio Jantar

cozinhar	:prec (mãos_limpas)	
	:efeito (jantar)	
embrulhar	:prec (silêncio)	
	:efeito (presente)	
carregar_lixo		:prec (lixo)
		:efeito (¬lixo, ¬mãos_limpas)
triturar_lixo		:prec (lixo)
		:efeito (¬lixo, ¬silêncio)

Grafo de Planejamento



Planejamento e Seleção de Ações

- Existem três abordagens em IA para a seleção de ação ou controle:
  1. Programação: especificação de controle pelo projetista
  2. Planejamento: especificação do problema pelo projetista; o controle é derivado automaticamente;
  3. Aprendizado: o controle é “aprendido” através da experiência
- As três abordagens possuem seus pontos fortes e fracos
- são consideradas não exclusivas e em geral, se complementam.