

*Planejamento em
Inteligência Artificial*

Capítulo 2

Representação de Problemas em Planejamento Clássico

Leliane Nunes de Barros

MAC 5788 - IME/USP
segundo semestre de 2005

Revisão de Planejamento Clássico

- Planejamento clássico faz as 8 suposições restritivas:

A0: Finito

A1: Totalmente observável

A2: Determinístico

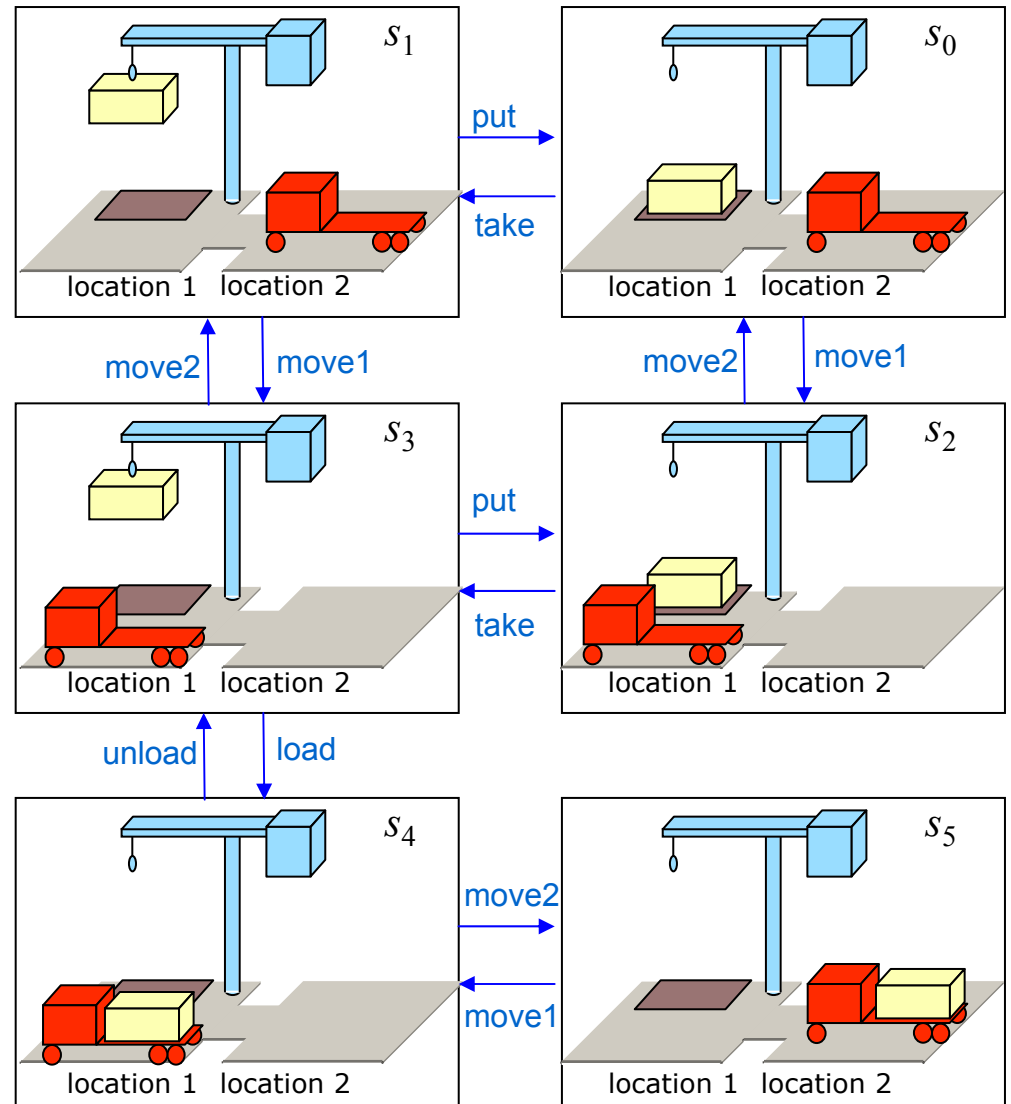
A3: Estático

A4: Satisfação de metas

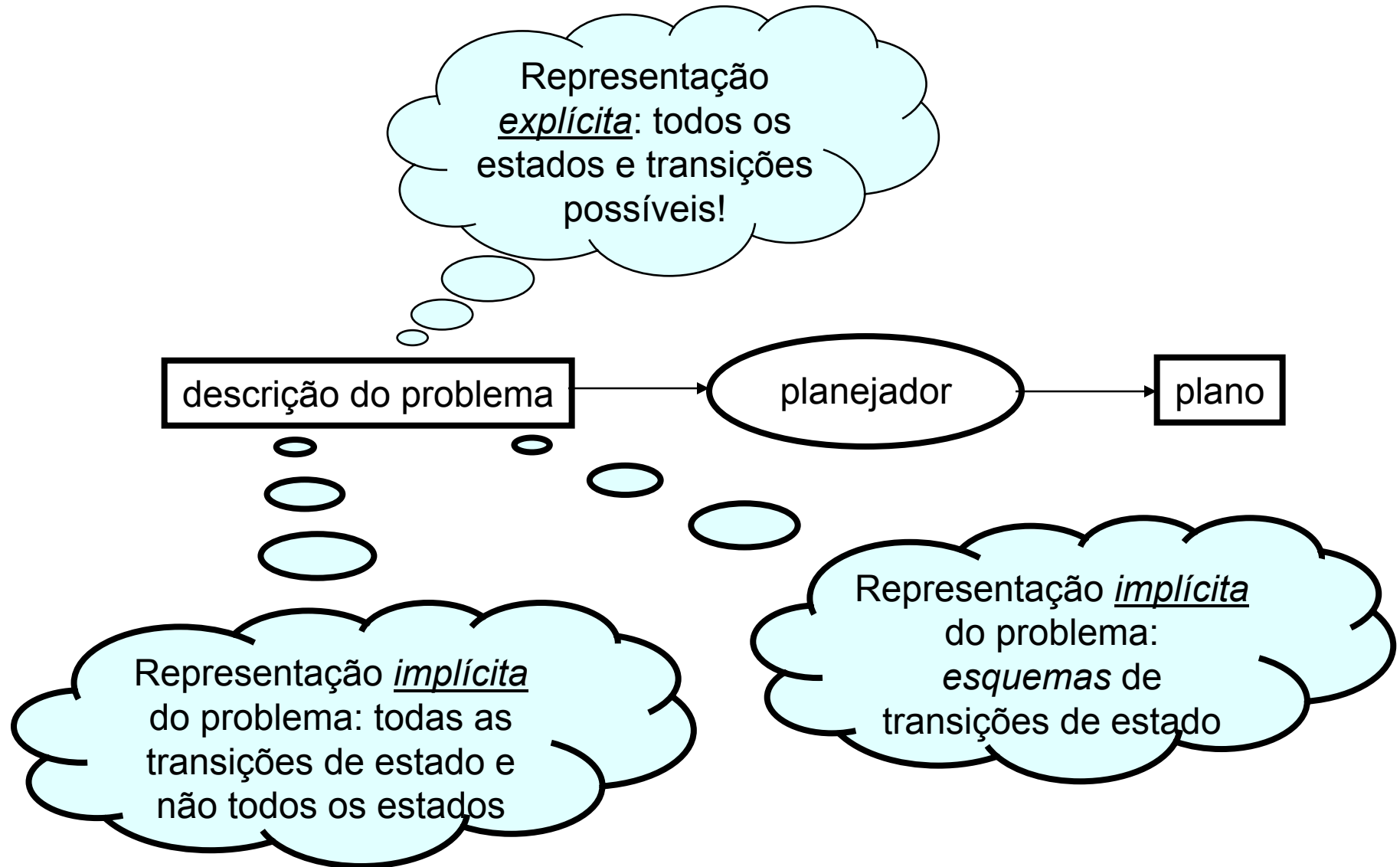
A5: Planos seqüenciais

A6: Tempo implícito

A7: Planejamento *off-line*



Descrição do Problema de Planejamento



Representações: Motivação

- Na maioria dos problemas, existem muitos estados para representá-los explicitamente como s_0, s_1, s_2, \dots
- Podemos representar cada estado como um conjunto de características. Por exemplo:
 - » um vetor de valores para um conjunto de variáveis
 - » um conjunto de (*ground*) átomos em alguma linguagem de primeira ordem L
- Podemos definir um conjunto de operadores que podem ser usados para computar as transições de estados
 - ◆ Não fornecendo todos os estados explicitamente:
 - » fornecendo apenas o estado inicial
 - » e usando o operador para gerar os outros estados, quando necessário

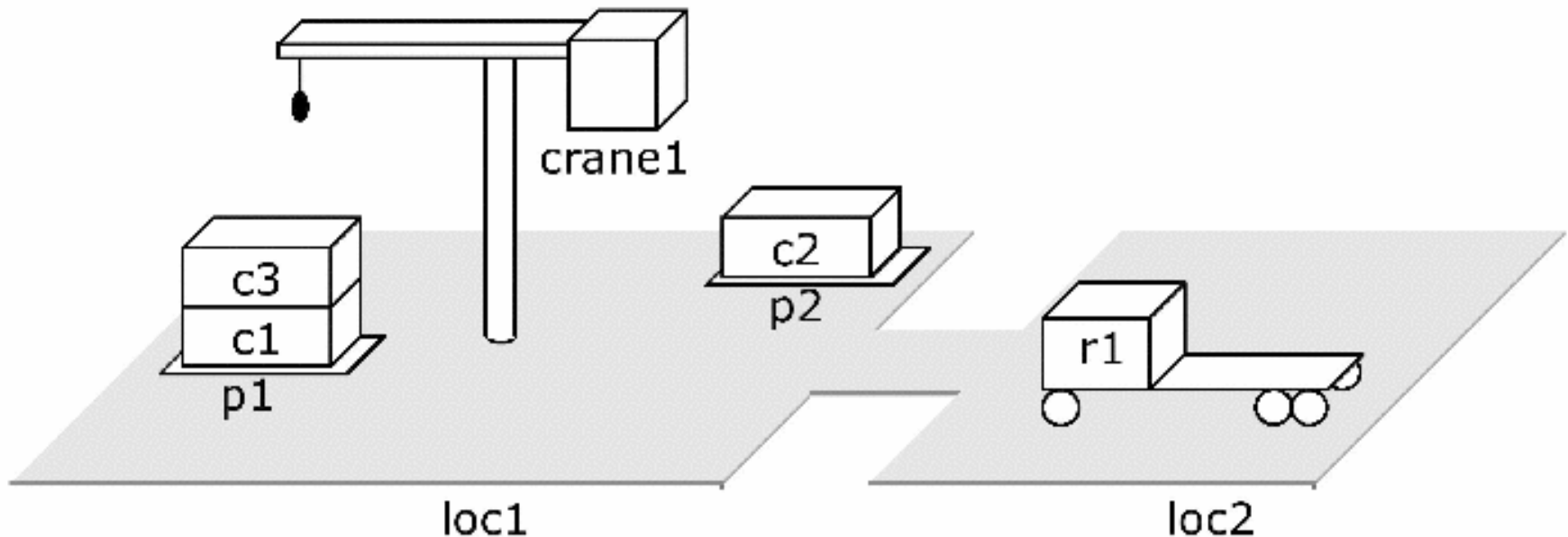
Tópicos dessa aula

- Representação de problemas de planejamento
 - ◆ Representação clássica
 - ◆ Representação de teoria de conjuntos
 - ◆ Representação de variáveis de estado
 - ◆ Exemplos: DWR e Mundo dos Blocos
 - ◆ Comparações

Representação Clássica

- Começamos com uma linguagem de primeira ordem, livre de funções com:
 - ◆ Finitamente muitos símbolos de predicados e símbolos constantes, mas sem símbolos funcionais
 - ◆ *Átomos*: símbolos predicados e termos (ctes ou vars) - e.g., $on(c1,c3)$, $on(c1,x)$
 - ◆ expressão (*ground*): não contém símbolos variáveis - e.g., $on(c1,c3)$
 - ◆ Expressão (*unground*): com pelo menos uma variável - e.g., $on(c1,x)$
 - ◆ *Substituição*: $\theta = \{x_1 \leftarrow v_1, x_2 \leftarrow v_2, \dots, x_n \leftarrow v_n\}$
 - » Cada x_i é um símbolo variável; cada v_i é um termo
 - ◆ *Instância* de e : resultado da aplicação de uma substituição θ a e
 - » Substituir variáveis de e simultaneamente, não sequencialmente
- *Estado*: um conjunto de (*ground*) átomos
 - ◆ Os estados representam as coisas que são verdadeiras em um dos estados do sistema Σ
 - ◆ Finitamente muitos (*ground*) átomos, portanto temos somente finitamente muitos estados possíveis

Exemplo de um estado



{attached(p1,loc1), in(c1,p1), in(c3,p1), top(c3,p1), on(c3,c1), on(c1,pallet), attached(p2,loc1), in(c2,p2), top(c2,p2), on(c2,pallet), belong(crane1,loc1), empty(crane1), adjacent(loc1,loc2), adjacent(loc2,loc1), at(r1,loc2), occupied(loc2), unloaded(r1)}.

Operadores

- *Operador* : uma tripla $o=(\text{nome}(o), \text{precond}(o), \text{efeitos}(o))$
 - ◆ $\text{nome}(o)$ é uma expressão sintática da forma $n(x_1, \dots, x_k)$
 - » n : símbolo de *operador* – deve ser único para cada operador
 - » x_1, \dots, x_k : símbolos variáveis (parâmetros)
 - deve incluir cada símbolo de variável em o
 - ◆ $\text{precond}(o)$: *precondições*
 - » literais que devem ser verdadeiras para ser possível usar/executar o operador
 - ◆ $\text{efeitos}(o)$: *efeitos*
 - » literais que o operador tornará verdadeiros

`take (k,l,c,d,p)`

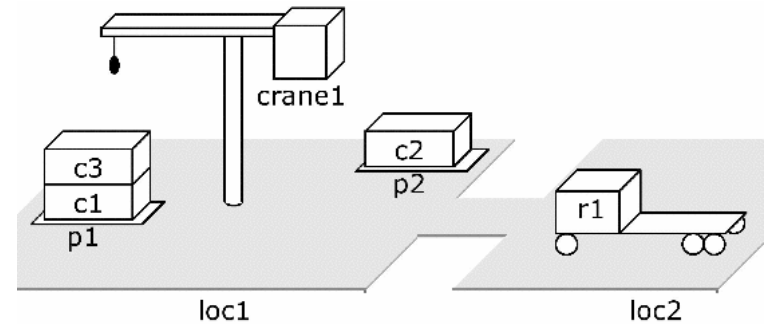
`:: guindaste k na localização l retira c de d na pilha p`

`precond: belong(k,l), attached(p,l), empty(k), top(c,p), on(c,d)`

`efeitos: holding(k,c), ¬ empty(k), ¬ in(c,p), ¬ top(c,p), ¬ on(c,d), top(d,p),`

Ações

- Ações: (*ground*) instância (através de substituições) de um operador



$\text{take}(k, l, c, d, p)$

;; crane k at location l takes c off of d in pile p

precond: $\text{belong}(k, l), \text{attached}(p, l), \text{empty}(k), \text{top}(c, p), \text{on}(c, d)$

effects: $\text{holding}(k, c), \neg \text{empty}(k), \neg \text{in}(c, p), \neg \text{top}(c, p), \neg \text{on}(c, d), \text{top}(d, p)$

$\text{take}(\text{crane1}, \text{loc1}, \text{c3}, \text{c1}, \text{p1})$

;; crane1 at location loc1 takes c3 off c1 in pile p1

precond: $\text{belong}(\text{crane1}, \text{loc1}), \text{attached}(\text{p1}, \text{loc1}),$
 $\text{empty}(\text{crane1}), \text{top}(\text{c3}, \text{p1}), \text{on}(\text{c3}, \text{c1})$

effects: $\text{holding}(\text{crane1}, \text{c3}), \neg \text{empty}(\text{crane1}), \neg \text{in}(\text{c3}, \text{p1}),$
 $\neg \text{top}(\text{c3}, \text{p1}), \neg \text{on}(\text{c3}, \text{c1}), \text{top}(\text{c1}, \text{p1})$

Notação

- Seja S um conjunto de literais. Então:
 - » $S^+ = \{\text{átomos que aparecem positivamente em } S\}$
 - » $S^- = \{\text{átomos que aparecem negativamente em } S\}$
- Mais especificamente, seja a um operador ou ação. Então
 - » $\text{precond}^+(a) = \{\text{átomos que aparecem positivamente em } a\}$
 - » $\text{precond}^-(a) = \{\text{átomos que aparecem negativamente em } a\}$
 - » $\text{efeitos}^+(a) = \{\text{átomos que aparecem positivamente em } a\}$
 - » $\text{efeitos}^-(a) = \{\text{átomos que aparecem negativamente em } a\}$

$\text{take}(k, l, c, d, p)$

;; crane k at location l takes c off of d in pile p

precond: $\text{belong}(k, l), \text{attached}(p, l), \text{empty}(k), \text{top}(c, p), \text{on}(c, d)$

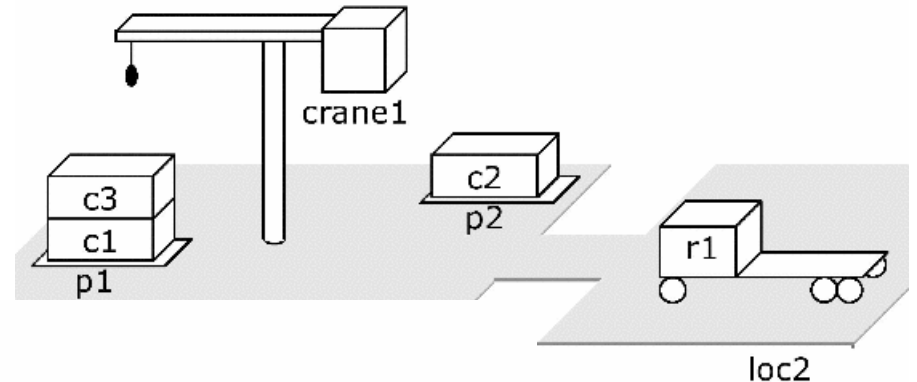
effects: $\text{holding}(k, c), \neg \text{empty}(k), \neg \text{in}(c, p), \neg \text{top}(c, p), \neg \text{on}(c, d), \text{top}(d, p)$

◆ $\text{efeitos}^+(\text{take}(k, l, c, d, p)) = \{\text{holding}(k, c), \text{top}(d, p)\}$

◆ $\text{efeitos}^-(\text{take}(k, l, c, d, p)) = \{\text{empty}(k), \text{in}(c, p), \text{top}(c, p), \text{on}(c, d)\}$

Aplicabilidade de ações

- Uma ação a é *aplicável* a um estado s se s satisfaz $\text{precond}(a)$,
 - ◆ i.e., se $\text{precond}^+(a) \subseteq s$ e $\text{precond}^-(a) \cap s = \emptyset$
- Exemplo de um estado e uma ação aplicável:



```
take(crane1,loc1,c3,c1,p1)
```

```
;; crane crane1 at location loc1 takes c3 off c1 in pile p1
```

```
precond: belong(crane1,loc1), attached(p1,loc1),  
          empty(crane1), top(c3,p1), on(c3,c1)
```

```
effects:  holding(crane1,c3), ¬empty(crane1), ¬in(c3,p1),  
          ¬top(c3,p1), ¬on(c3,c1), top(c1,p1)
```

Resultado da execução de uma ação

- Se a é aplicável a s , o resultado de sua execução é:

$$\gamma(s,a) = (s - \text{efeitos}^-(a)) \cup \text{efeitos}^+(a)$$

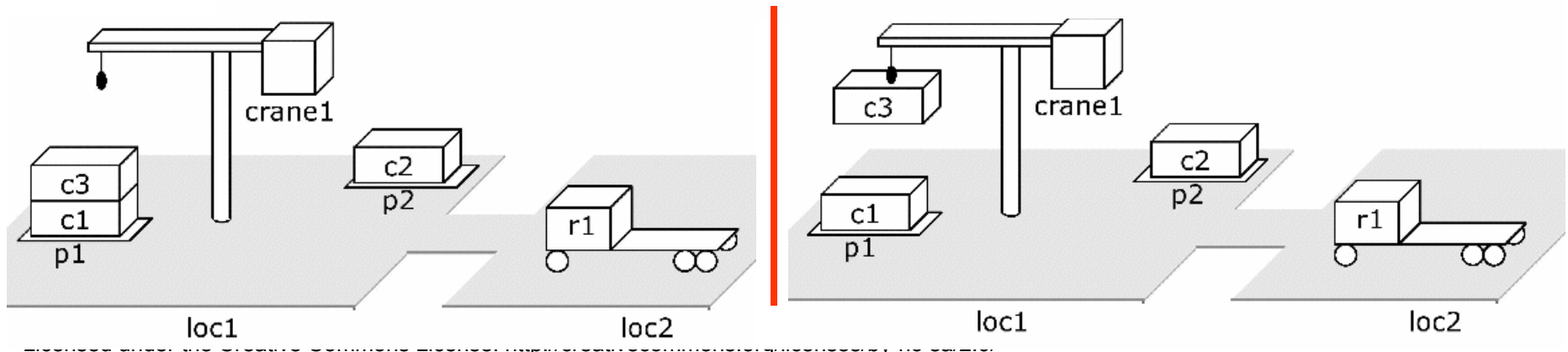
- ◆ Remover os efeitos negativos, e adicionar os efeitos positivos

`take(crane1,loc1,c3,c1,p1)`

`:: crane crane1 at location loc1 takes c3 off c1 in pile p1`

`precond: belong(crane1,loc1), attached(p1,loc1),
empty(crane1), top(c3,p1), on(c3,c1)`

`effects: holding(crane1,c3), \neg empty(crane1), \neg in(c3,p1),
 \neg top(c3,p1), \neg on(c3,c1), top(c1,p1)`



$move(r, l, m)$

:: robot r moves from location l to location m

precond: $adjacent(l, m), at(r, l), \neg occupied(m)$

effects: $at(r, m), occupied(m), \neg occupied(l), \neg at(r, l)$

$load(k, l, c, r)$

:: crane k at location l loads container c onto robot r

precond: $belong(k, l), holding(k, c), at(r, l), unloaded(r)$

effects: $empty(k), \neg holding(k, c), loaded(r, c), \neg unloaded(r)$

$unload(k, l, c, r)$

:: crane k at location l takes container c from robot r

precond: $belong(k, l), at(r, l), loaded(r, c), empty(k)$

effects: $\neg empty(k), holding(k, c), unloaded(r), \neg loaded$

$put(k, l, c, d, p)$

:: crane k at location l puts c onto d in pile p

precond: $belong(k, l), attached(p, l), holding(k, c), top(d, p)$

effects: $\neg holding(k, c), empty(k), in(c, p), top(c, p), on(c, d), \neg top(d, p)$

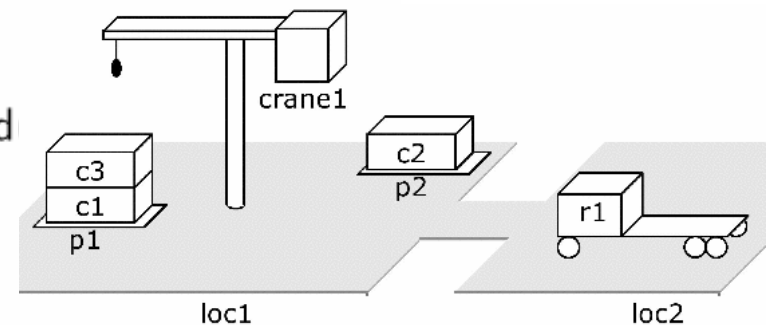
$take(k, l, c, d, p)$

:: crane k at location l takes c off of d in pile p

precond: $belong(k, l), attached(p, l), empty(k), top(c, p), on(c, d)$

effects: $holding(k, c), \neg empty(k), \neg in(c, p), \neg top(c, p), \neg on(c, d), top(d, p)$

- Domínio de planejamento:
linguagem + operadores
 - ◆ Corresponde a um conjunto de sistemas de estado-transição
 - ◆ Exemplo:
operadores para o domínio DWR



Problemas de Planejamento

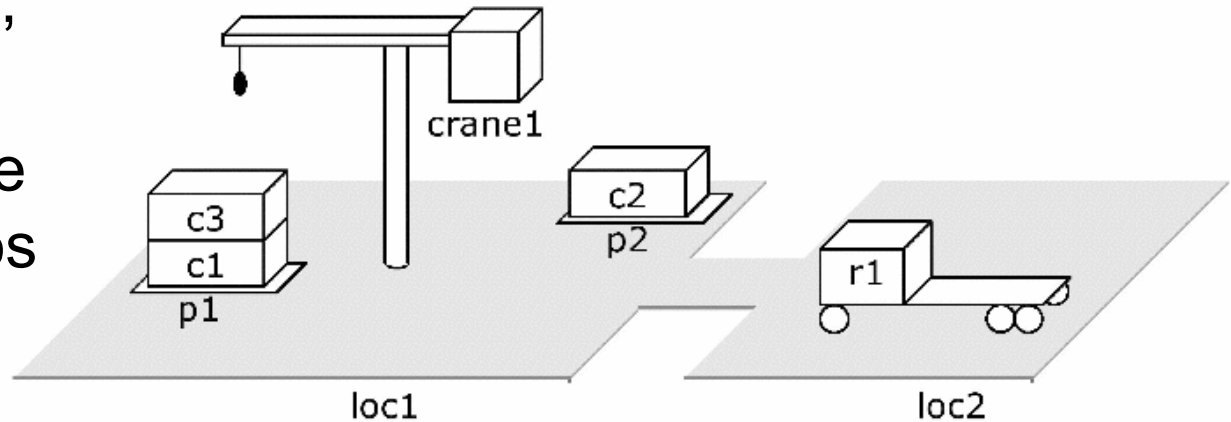
- Dado um domínio de planejamento (linguagem L , operadores O)
 - ◆ *Declaração de um problema de planejamento*: uma tripla $P=(O,s_0,g)$
 - » O é uma coleção de operadores
 - » s_0 é um estado (o estado inicial)
 - » g é um conjunto de literais (a fórmula meta)
 - ◆ O *problema de planejamento* é: $P = (\Sigma, s_0, S_g)$
 - » s_0 e S_g (como definido acima)
 - » $\Sigma = (S, A, \gamma)$ é um sistema de estado-transição
 - » $S = \{\text{conjuntos de todos } (ground) \text{ átomos em } L\}$
 - » $A = \{\text{todas as } (ground) \text{ instâncias dos operadores em } O\}$
 - » $\gamma = \text{a função de transição de estado determinada pelos operadores}$
- Chamaremos de “problema de planejamento” à declaração de um problema de planejamento

Planos e Soluções

- *Plano*: qualquer seqüência de ações $\sigma = \langle a_1, a_2, \dots, a_n \rangle$ tal que cada a_i é uma (*ground*) instância de um operador em O
- O plano é uma *solução para* $P=(O, s_0, g)$ se ele é executável e atinge g
 - ◆ i.e., se há estados s_0, s_1, \dots, s_n tal que
 - » $\gamma(s_0, a_1) = s_1$
 - » $\gamma(s_1, a_2) = s_2$
 - » ...
 - » $\gamma(s_{n-1}, a_n) = s_n$
 - » s_n satisfaz g

Exemplo

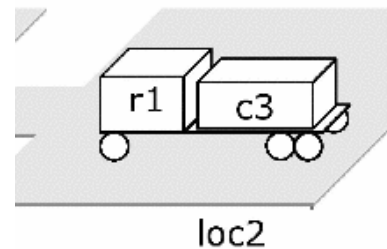
- Seja $P_1 = (O, s_1, g_1)$, onde
 - ◆ O é o conjunto de operadores dados anteriormente



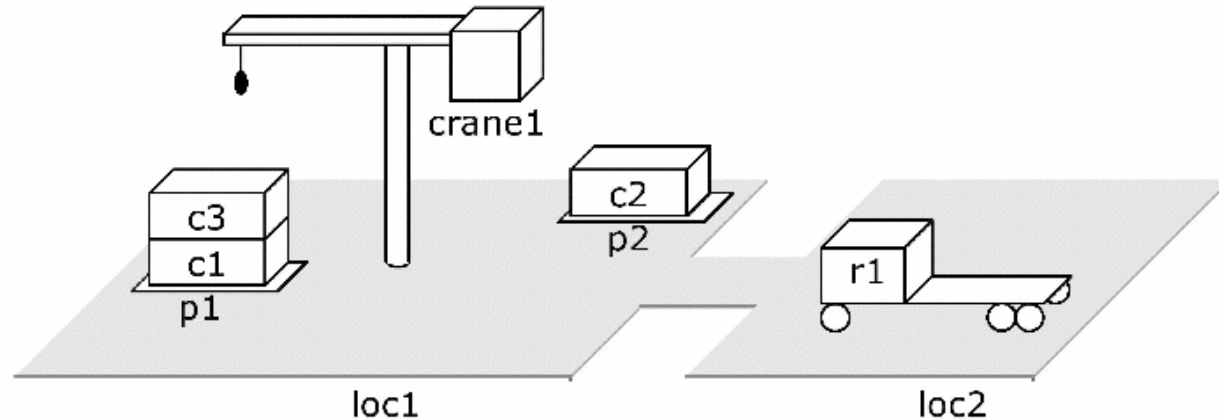
- ◆ s_1 é:

$\{attached(p1,loc1), in(c1,p1), in(c3,p1), top(c3,p1), on(c3,c1), on(c1,pallet), attached(p2,loc1), in(c2,p2), top(c2,p2), on(c2,pallet), belong(crane1,loc1), empty(crane1), adjacent(loc1,loc2), adjacent(loc2,loc1), at(r1,loc2), occupied(loc2), unloaded(r1)\}$.

- ◆ $g_1 = \{loaded(r1,c3), at(r1,loc2)\}$

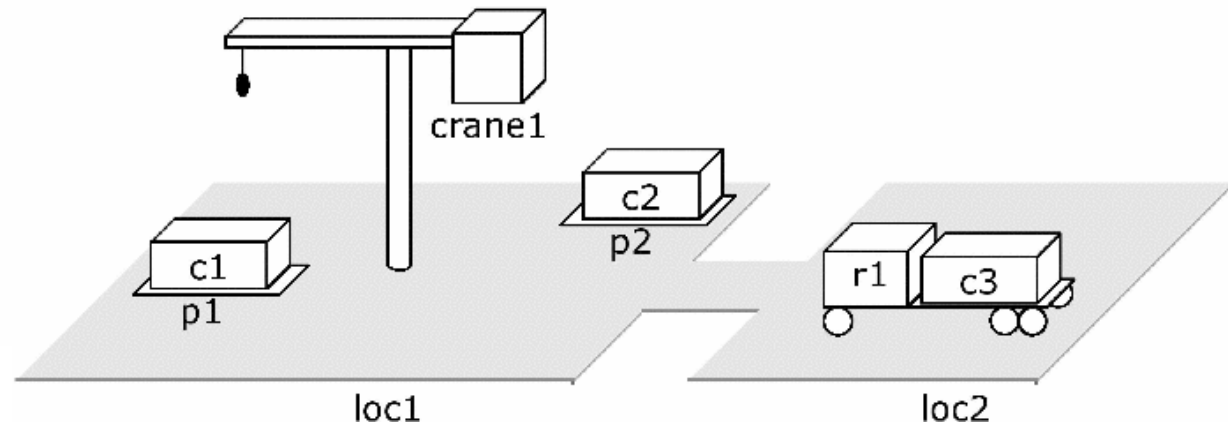


Exemplo (continuação)

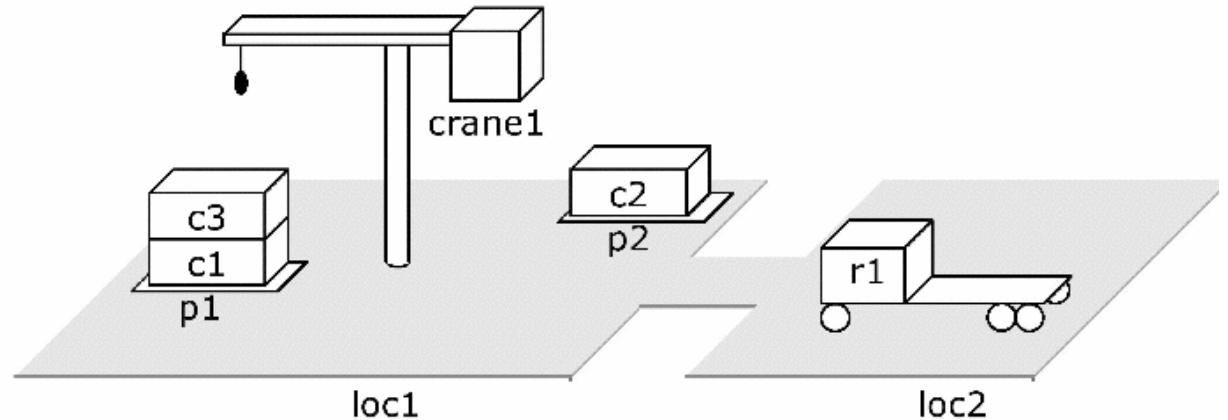


- Existem três soluções para P :
 - ◆ $\langle \text{take}(\text{crane1}, \text{loc1}, \text{c3}, \text{c1}, \text{p1}), \text{move}(\text{r1}, \text{loc2}, \text{loc1}), \text{move}(\text{r1}, \text{loc1}, \text{loc2}), \text{move}(\text{r1}, \text{loc2}, \text{loc1}), \text{load}(\text{crane1}, \text{loc1}, \text{c3}, \text{r1}), \text{move}(\text{r1}, \text{loc1}, \text{loc2}) \rangle$
 - ◆ $\langle \text{take}(\text{crane1}, \text{loc1}, \text{c3}, \text{c1}, \text{p1}), \text{move}(\text{r1}, \text{loc2}, \text{loc1}), \text{load}(\text{crane1}, \text{loc1}, \text{c3}, \text{r1}), \text{move}(\text{r1}, \text{loc1}, \text{loc2}) \rangle$
 - ◆ $\langle \text{move}(\text{r1}, \text{loc2}, \text{loc1}), \text{take}(\text{crane1}, \text{loc1}, \text{c3}, \text{c1}, \text{p1}), \text{load}(\text{crane1}, \text{loc1}, \text{c3}, \text{r1}), \text{move}(\text{r1}, \text{loc1}, \text{loc2}) \rangle$

- Cada uma delas produz o estado:

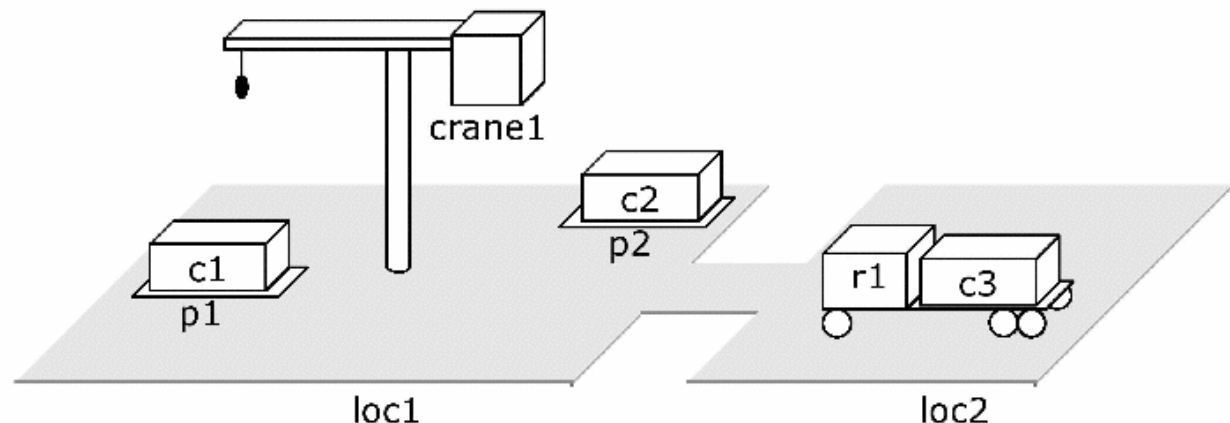


Exemplo (continuação)



- O primeiro é *redundante*: ações podem ser removidas e ainda teremos uma solução
 - ◆ $\langle \text{take}(\text{crane1}, \text{loc1}, \text{c3}, \text{c1}, \text{p1}), \text{move}(\text{r1}, \text{loc2}, \text{loc1}), \text{move}(\text{r1}, \text{loc1}, \text{loc2}), \text{move}(\text{r1}, \text{loc2}, \text{loc1}), \text{load}(\text{crane1}, \text{loc1}, \text{c3}, \text{r1}), \text{move}(\text{r1}, \text{loc1}, \text{loc2}) \rangle$
 - ◆ $\langle \text{take}(\text{crane1}, \text{loc1}, \text{c3}, \text{c1}, \text{p1}), \text{move}(\text{r1}, \text{loc2}, \text{loc1}), \text{load}(\text{crane1}, \text{loc1}, \text{c3}, \text{r1}), \text{move}(\text{r1}, \text{loc1}, \text{loc2}) \rangle$
 - ◆ $\langle \text{move}(\text{r1}, \text{loc2}, \text{loc1}), \text{take}(\text{crane1}, \text{loc1}, \text{c3}, \text{c1}, \text{p1}), \text{load}(\text{crane1}, \text{loc1}, \text{c3}, \text{r1}), \text{move}(\text{r1}, \text{loc1}, \text{loc2}) \rangle$

- O 2º e o 3º são *não-redundantes* e são planos mais curtos (*planos minimais*)

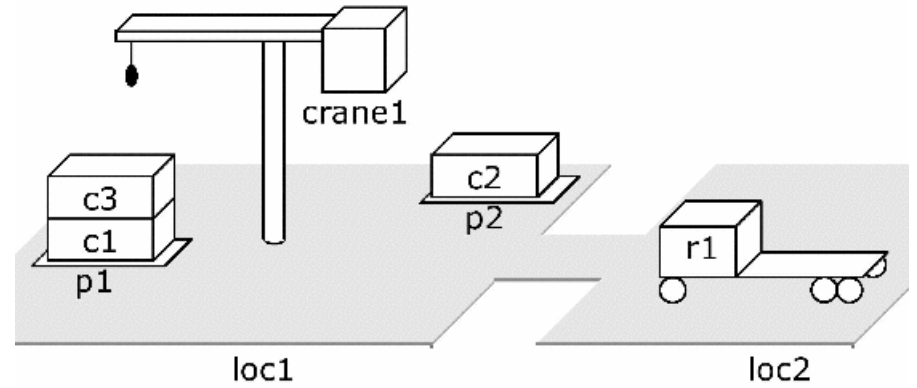


Relevância de ações

- Uma ação a é *relevante* para uma meta g se ela for aplicável no estado corrente e :

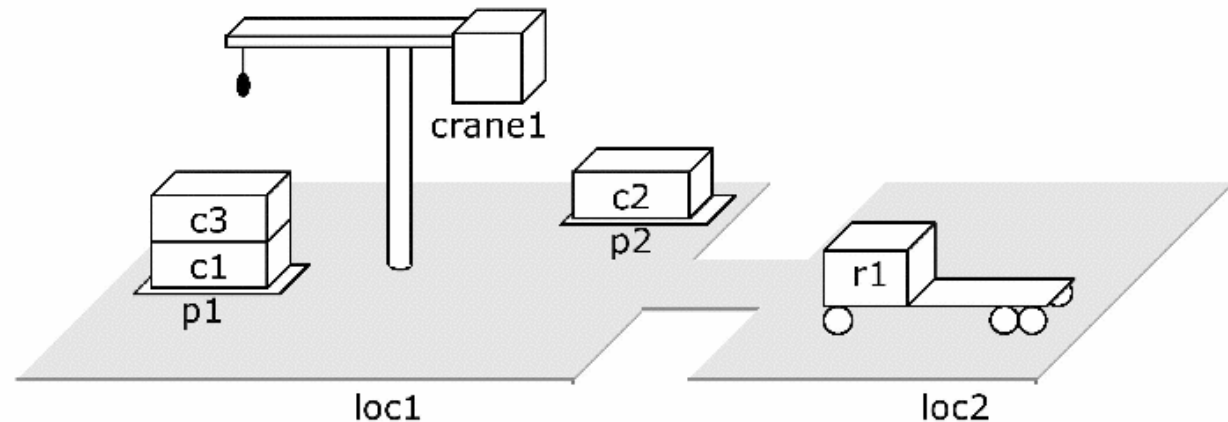
$$g \cap \text{efeitos}^+(a) \neq \emptyset \text{ e } g \cap \text{efeitos}^-(a) = \emptyset$$

- Exemplo de um estado e uma ação relevante:



Representação baseada em Teoria de Conjuntos

- Como a representação clássica, mas restrita à lógica proposicional



- Estados:

- ◆ Ao invés de uma coleção de *ground* átomos ...

{on(c1,pallet), on(c1,r1), on(c1,c2), ..., at(r1,l1), at(r1,l2), ...}

... usa uma coleção de proposições (variáveis booleanas):

{on-c1-pallet, on-c1-r1, on-c1-c2, ..., at-r1-l1, at-r1-l2, ...}

Representação baseada em Teoria de Conjuntos

- Ao invés de um operador como esse:

```
take(crane1,loc1,c3,c1,p1)
;; crane crane1 at location loc1 takes c3 off c1 in pile p1
precond: belong(crane1,loc1), attached(p1,loc1),
         empty(crane1), top(c3,p1), on(c3,c1)
effects:  holding(crane1,c3), ¬empty(crane1), ¬in(c3,p1),
         ¬top(c3,p1), ¬on(c3,c1), top(c1,p1)
```

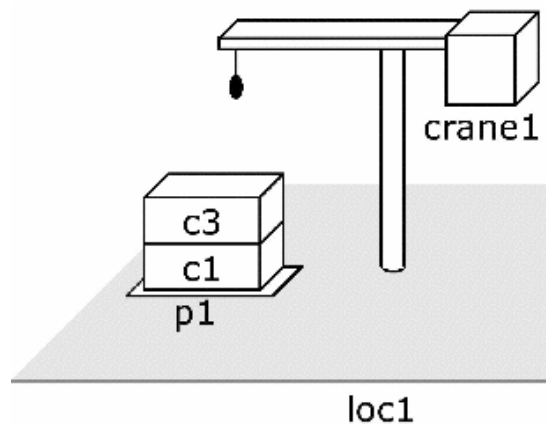
- ... existem várias ações como essa:

```
take-crane1-loc1-c3-c1-p1
precond: belong-crane1-loc1, attached-p1-loc1,
         empty-crane1, top-c3-p1, on-c3-c1
delete:  empty-crane1, in-c3-p1, top-c3-p1, on-c3-p1
add:     holding-crane1-c3, top-c1-p1
```

- Explosão exponencial
 - ◆ Se um operador clássico contém n átomos, cada um com aridade k , então ele corresponde a c^{nk} ações onde $c = |\{\text{símbolos constantes}\}|$

Representação de Variáveis de Estado

- Uma variável de estado é como um campo em uma estrutura de registros



{top(p1)=c3, cpos(c3)=c1,
cpos(c1)=pallet, ...}

- Representações clássica e de variáveis de estado consomem espaços similares
 - ◆ Cada uma pode ser traduzida para a outra em tempo polinomial de baixa ordem

load(c, r, l)

;; robot r loads container c at location l

precond: $rloc(r) = l, cpos(c) = l, rload(r) = nil$

effects: $rload(r) \leftarrow c, cpos(c) \leftarrow r$

unload(c, r, l)

;; robot r unloads container c at location l

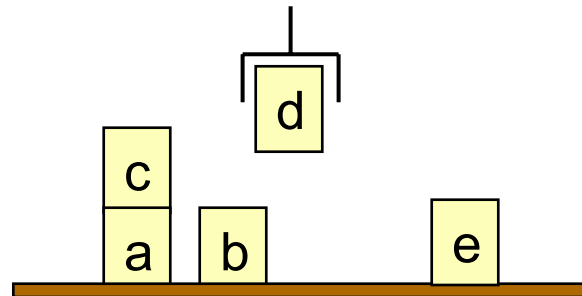
precond: $rloc(r) = l, rload(r) = c$

effects: $rload(r) \leftarrow nil, cpos(c) \leftarrow l$

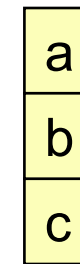
Exemplo: O Mundo dos Blocos

- Mesa infinitamente larga, número finito de blocos de criança
- Ignora a posição em que um bloco está sobre a mesa
- Um bloco pode estar sobre a mesa ou sobre um outro bloco
- Os blocos devem ser movidos de uma configuração para outra

◆ e.g.,



estado inicial



estado meta

- Pode ser expresso como um caso especial de DWR, porém sua formulação é mais simples
- Daremos as formulações: clássica, teoria de conjuntos e variáveis de estado, para o caso de existirem 5 blocos.

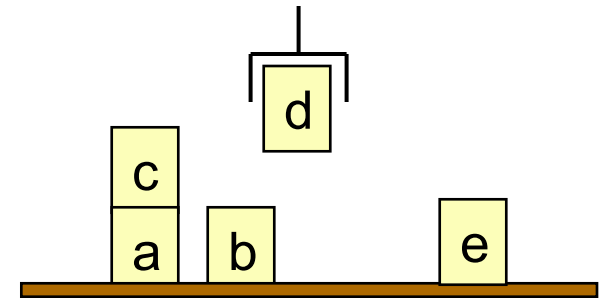
Representação Clássica: Símbolos

- Símbolos constantes:

- ◆ Os blocos: a, b, c, d, e

- Predicados:

- ◆ $\text{ontable}(x)$ - bloco x está sobre a mesa
- ◆ $\text{on}(x,y)$ - bloco x está sobre o bloco y
- ◆ $\text{clear}(x)$ - bloco x não tem nada sobre ele
- ◆ $\text{holding}(x)$ - a garra do robô está segurando o bloco x
- ◆ handempty - a garra do robô não está segurando nada



Operadores Clásicos

unstack(x,y)

Precond: $\text{on}(x,y)$, $\text{clear}(x)$, handempty

Effects: $\sim\text{on}(x,y)$, $\sim\text{clear}(x)$, $\sim\text{handempty}$,
 $\text{holding}(x)$, $\text{clear}(y)$

stack(x,y)

Precond: $\text{holding}(x)$, $\text{clear}(y)$

Effects: $\sim\text{holding}(x)$, $\sim\text{clear}(y)$,
 $\text{on}(x,y)$, $\text{clear}(x)$, handempty

pickup(x)

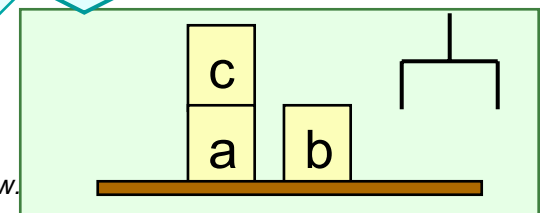
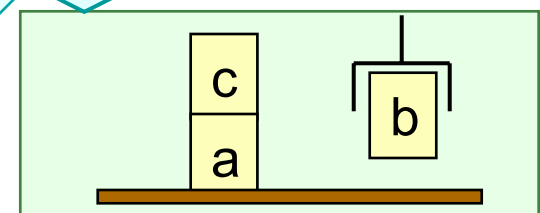
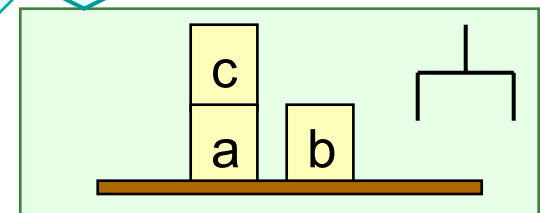
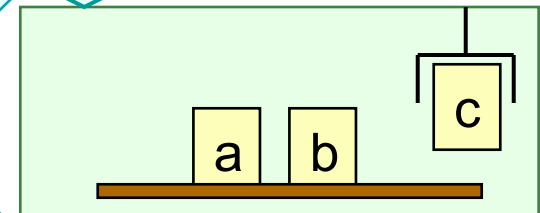
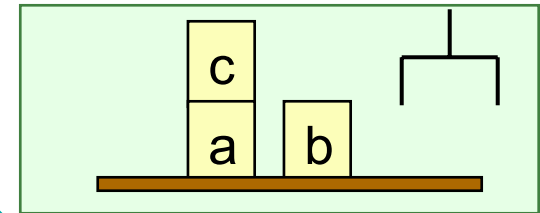
Precond: $\text{ontable}(x)$, $\text{clear}(x)$, handempty

Effects: $\sim\text{ontable}(x)$, $\sim\text{clear}(x)$,
 $\sim\text{handempty}$, $\text{holding}(x)$

putdown(x)

Precond: $\text{holding}(x)$

Effects: $\sim\text{holding}(x)$, $\text{ontable}(x)$,
 $\text{clear}(x)$, handempty



Representação baseada em Teoria de Conjuntos: Símbolos

- Para 5 blocos, há 36 proposições
- Aqui estão 5 delas:

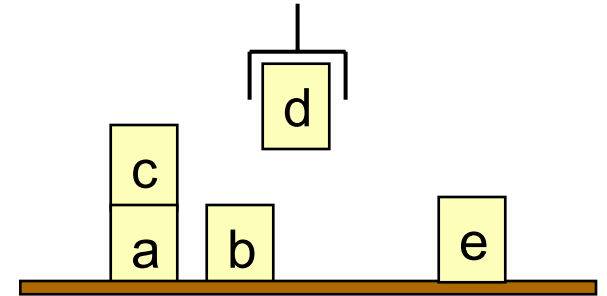
ontable-a - o bloco a está na mesa

on-c-a - o bloco c está sobre o bloco a

clear-c - o bloco c não possui nada sobre ele

holding-d - a garra do robô está segurando o bloco d

handempty - a garra do robô não está segurando nada



Ações de Teoria de Conjuntos

50 ações diferentes.

Aqui estão 4 delas:

unstack-c-a

Pre: on-c,a, clear-c, handempty

Del: on-c,a, clear-c, handempty

Add: holding-c, clear-a

stack-c-a

Pre: holding-c, clear-a

Del: holding-c, \sim clear-a

Add: on-c-a, clear-c, handempty

pickup-c

Pre: ontable-c, clear-c, handempty

Del: ontable-c, clear-c, handempty

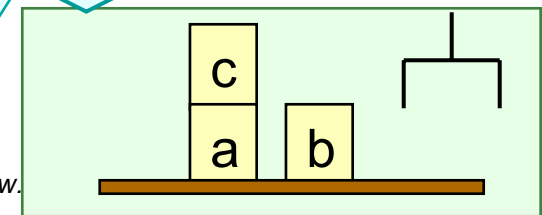
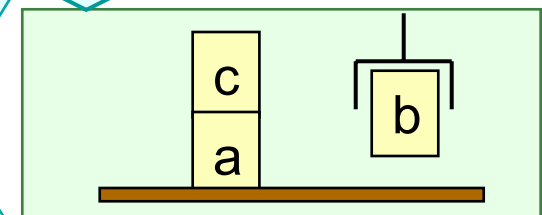
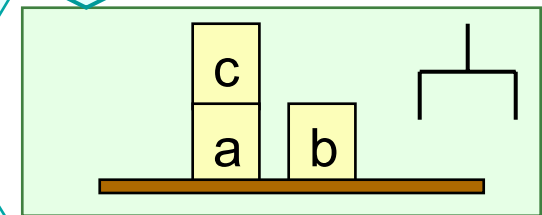
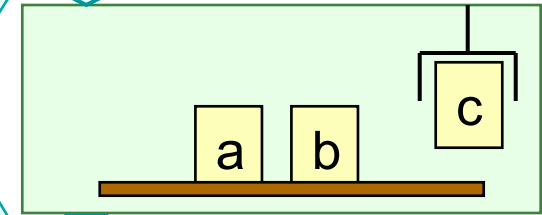
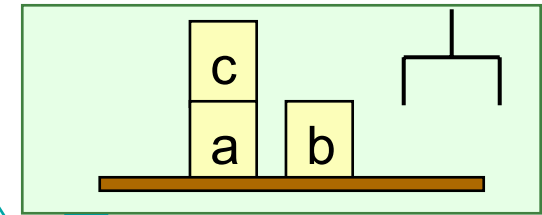
Add: holding-c

putdown-c

Pre: holding-c

Del: holding-c

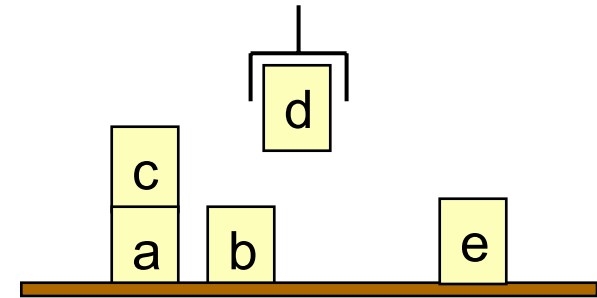
Add: ontable-c, clear-c, handempty



Representação de variáveis de estado: Símbolos

- Símbolos constantes:

a, b, c, d, e do tipo bloco
0, 1, table, nil de outros tipos



- Variáveis de estado:

$\text{pos}(x) = y$ se bloco x está sobre o bloco y

$\text{pos}(x) = \text{table}$ se bloco x está sobre a mesa

$\text{pos}(x) = \text{nil}$ se bloco x está na garra do robô

$\text{clear}(x) = 1$ se bloco x não tem nada sobre ele

$\text{clear}(x) = 0$ se bloco x está na garra ou se tem outro bloco sobre ele

$\text{holding} = x$ se a garra do robô está segurando o bloco x

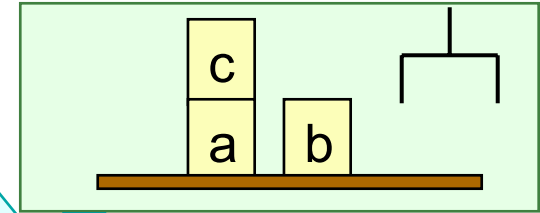
$\text{holding} = \text{nil}$ a garra do robô não está segurando nada

Operadores de Variáveis de Estado

unstack(x : block, y : block)

Precond: $\text{pos}(x)=y$, $\text{clear}(y)=0$, $\text{clear}(x)=1$, $\text{holding}=\text{nil}$

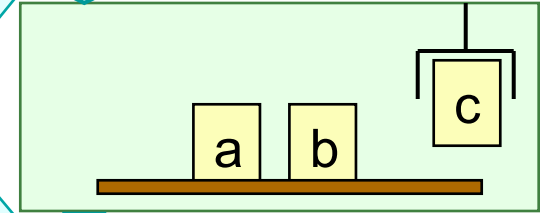
Effects: $\text{pos}(x)=\text{nil}$, $\text{clear}(x)=0$, $\text{holding}=x$, $\text{clear}(y)=1$



stack(x : block, y : block)

Precond: $\text{holding}=x$, $\text{clear}(x)=0$, $\text{clear}(y)=1$

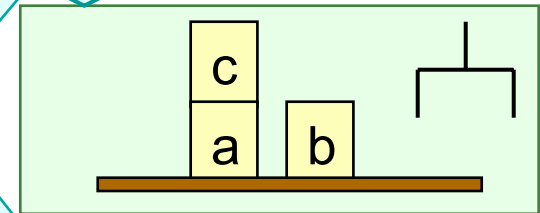
Effects: $\text{holding}=\text{nil}$, $\text{clear}(y)=0$, $\text{pos}(x)=y$, $\text{clear}(x)=1$



pickup(x : block)

Precond: $\text{pos}(x)=\text{table}$, $\text{clear}(x)=1$, $\text{holding}=\text{nil}$

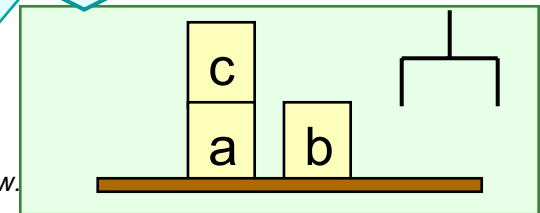
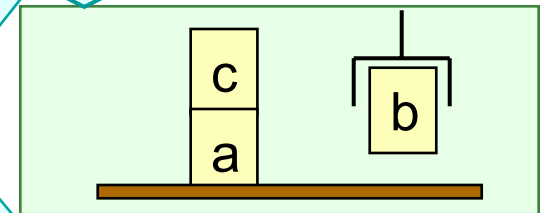
Effects: $\text{pos}(x)=\text{nil}$, $\text{clear}(x)=0$, $\text{holding}=x$



putdown(x : block)

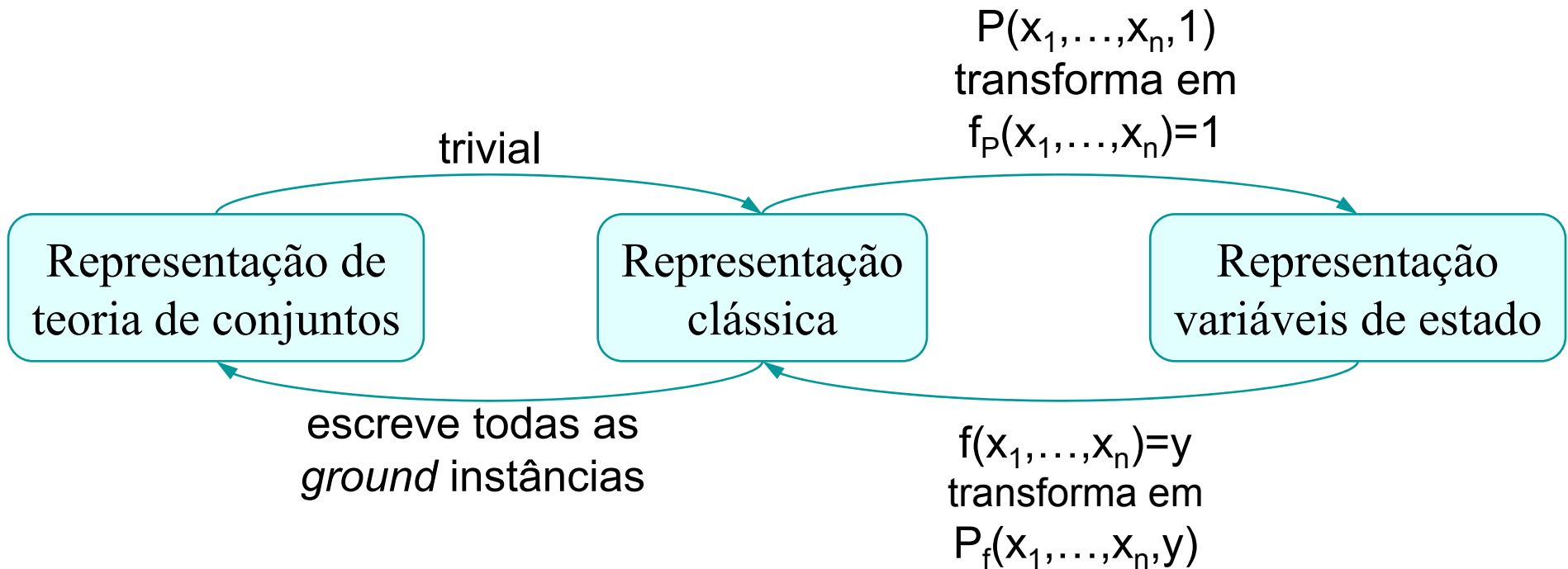
Precond: $\text{holding}=x$

Effects: $\text{holding}=\text{nil}$, $\text{pos}(x)=\text{table}$, $\text{clear}(x)=1$



Poder de expressão

- Qualquer problema que pode ser representado em uma representação pode ser representado nas outras duas
- Conversão em tempo e espaço linear, exceto para:
 - ◆ Conversão para teoria de conjuntos das outras duas representações: pode causar uma explosão combinatória



Comparação

- Representação clássica
 - ◆ A mais popular para planejamento clássico, parte por razões históricas
- Representação de teoria de conjuntos
 - ◆ Consome muito mais espaço do que a representação clássica
 - ◆ Útil em algoritmos que manipulam diretamente *ground* átomos
 - » e.g., grafos de planejamento (Capítulo 6), satisfazibilidade (Capítulo 7)
 - ◆ Útil também para certos tipos de estudos teóricos
- Representação de variável de estado
 - ◆ Menos natural para os lógicos, mais natural para os engenheiros
 - ◆ Útil em problemas de planejamento não-clássicos como uma maneira de tratar números, funções e tempo