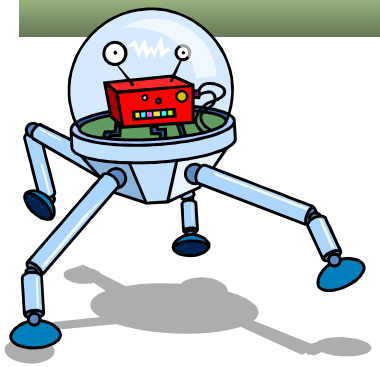
Planejamento automatizado



Representação

Leliane Nunes de Barros

Silvio do Lago Pereira

Com algumas modificações de Karina Valdivia Delgado

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 (2nd edition). Elsevier, 2006.

Principais questões no planejamento clássico

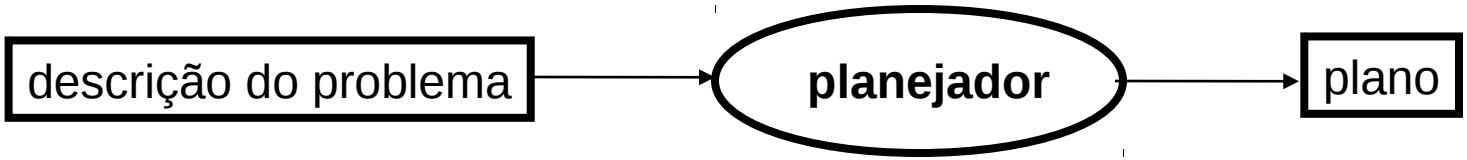
- Como representar os estados e as ações sem enumerar explicitamente S, A e γ.
- Como buscar uma solução de maneira eficiente: qual o espaço de busca? qual o algoritmo? qual heurística?

Principais questões no planejamento clássico

- Como representar os estados e as ações sem enumerar explicitamente S, A e γ.
- Como buscar uma solução de maneira eficiente: qual o espaço de busca? qual o algoritmo? qual heurística?

Representação de estados, ações e metas

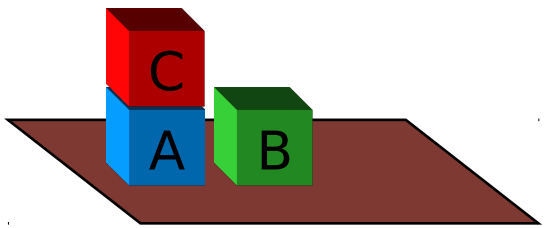
Como especificar um problema de planejamento?

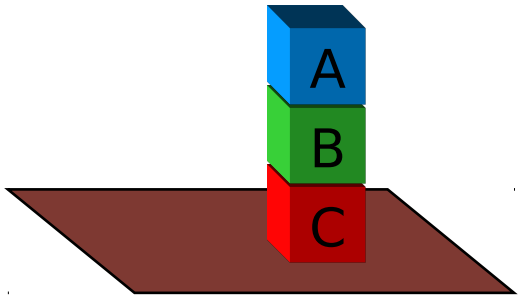


Como especificar um problema de planejamento?

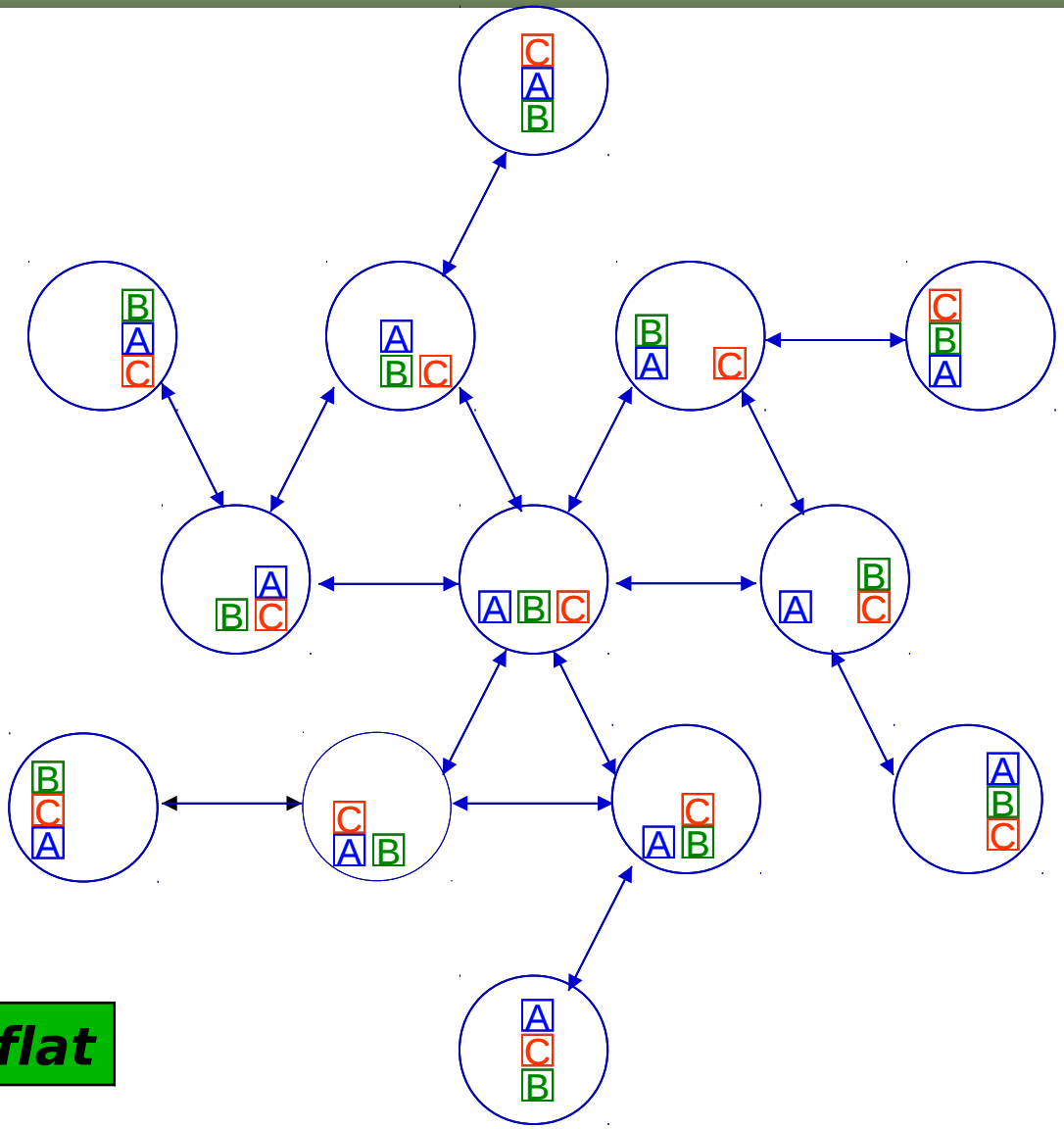
Exemplo: O Mundo dos Blocos

- Mesa infinitamente larga, número finito de blocos;
- 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;



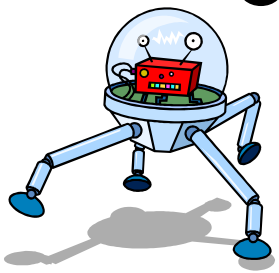


Representação *explícita*: todos os estados e transições possíveis



modelo *flat*

Representação *explícita*: todos os estados e transições possíveis

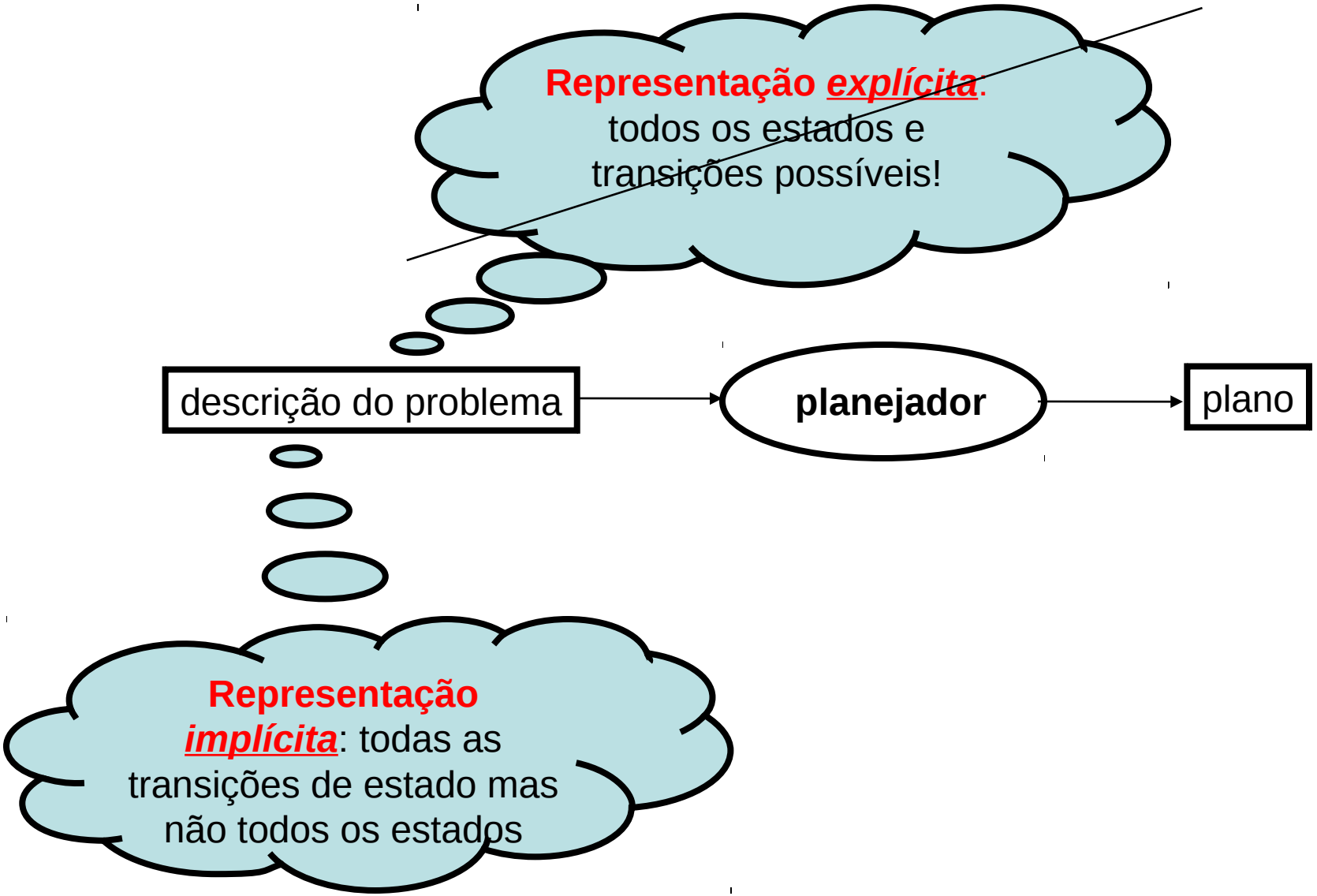


S_0

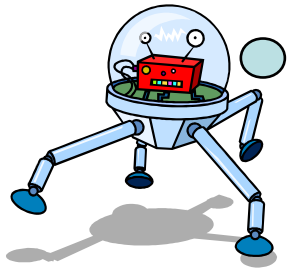
S_g

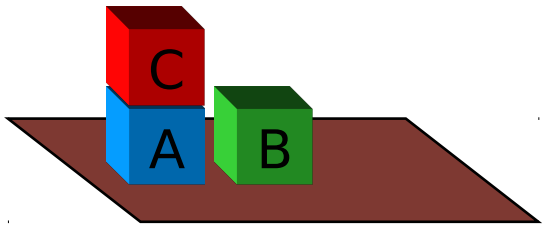
- Nos problemas de busca, os estados são considerados caixas pretas.
- Em planejamento precisamos raciocinar sobre os estados e as ações. Precisamos ter acesso à estrutura interna dos estados e ações.

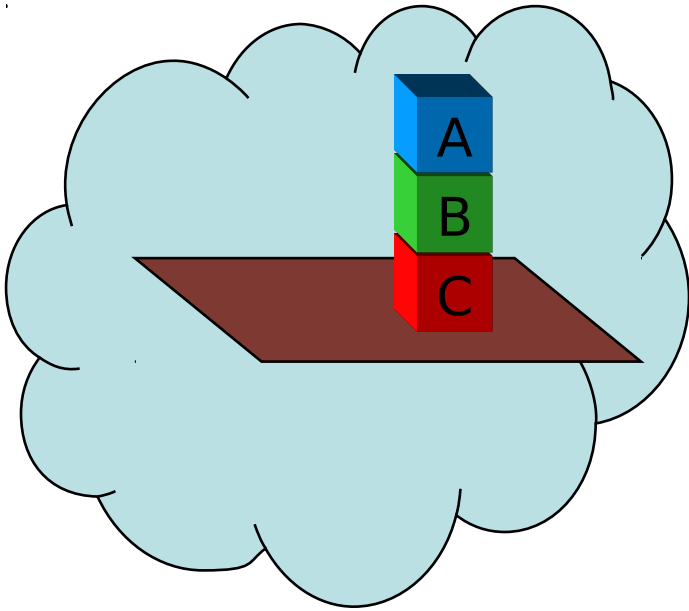
Como especificar um problema de planejamento?



Exemplo: O Mundo dos Blocos







As linguagens de planejamento

- O que seria uma “boa” linguagem?
- Suficientemente **expressiva** para descrever uma grande variedade de problemas.
- **Restritiva** o bastante para permitir que algoritmos eficientes operem sobre ela.
- Deve permitir a **decomposição** do problema em subproblemas.

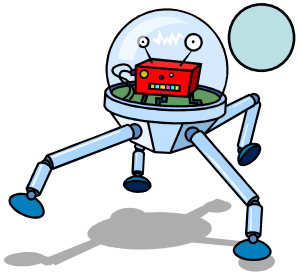
As linguagens de planejamento

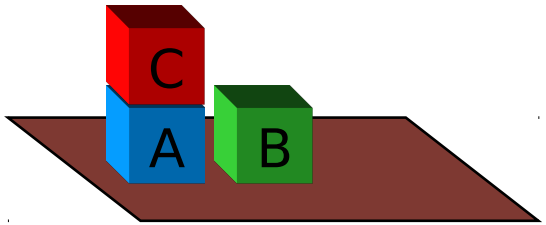
- **STRIPS**: Stanford Research Institute Problem Solver
- **ADL**: Action Description Language
- ...
- **PDDL**: Planning Domain Definition Language

Linguagem STRIPS

- Linguagem usada pelo planejador STRIPS. Ainda constitui a base de muitas das pesquisas feitas atualmente em Planejamento
- Linguagem formal para a especificação de problemas de planejamento
- Baseado em Lógica de Primeira Ordem (Lógica de predicados)
  - Objetos
  - Relações

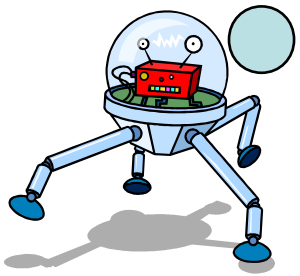
Exemplo: O Mundo dos Blocos

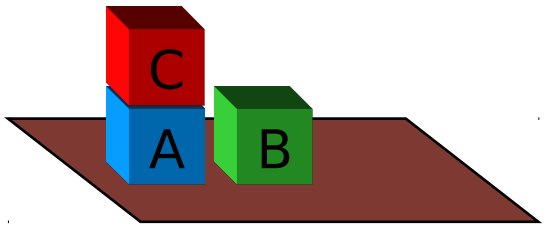




Objetos:

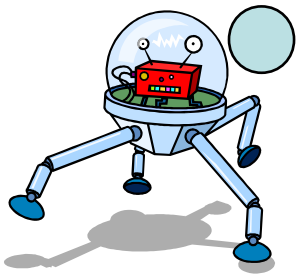
Exemplo: O Mundo dos Blocos

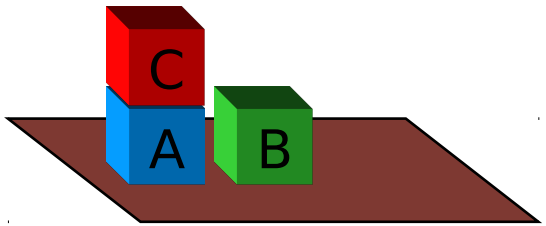




Objetos: blocos a,b,c,mesa

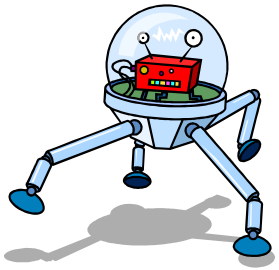
Exemplo: O Mundo dos Blocos

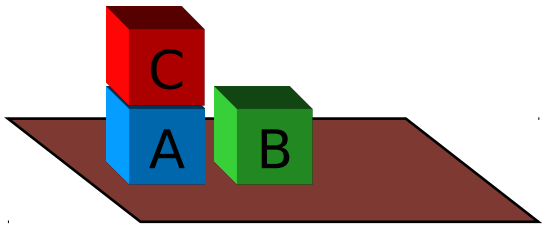




Propriedades:

Exemplo: O Mundo dos Blocos

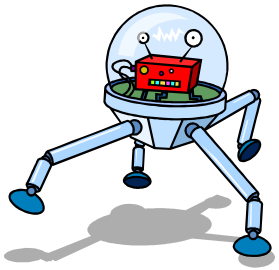


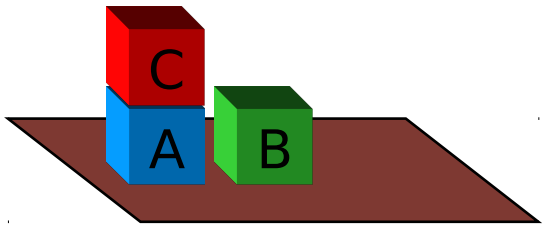


Propriedades:

a e b estão sobre a mesa
c está sobre a
o topo de c está livre
o topo de b está livre
o topo de a não está livre
c não está sobre b
c não está sobre a mesa
a não está sobre b
a não está sobre c
b não está sobre c
b não está sobre a
a mesa está sempre livre

Exemplo: O Mundo dos Blocos





Propriedades:
A e B estão sobre a mesa representado por:

sobre(a,mesa)
sobre(b,mesa)

...

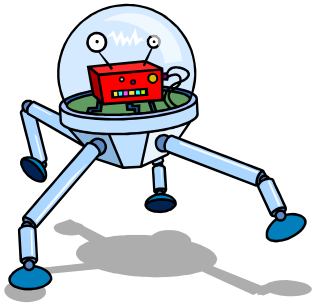
Linguagem para Descrição do Problema representação clássica

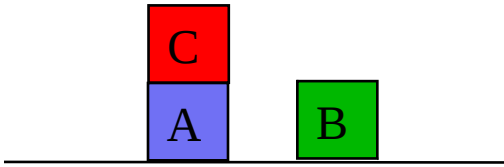
Baseada na Lógica de Predicados (LPO), livre de funções:

- conjunto finito de símbolos de predicados (ex.: sobre, limpo, move ...) e símbolos de objetos (variáveis ou constantes) (ex.: bloco X ou bloco a)
- um átomo é um predicado seguido de uma lista de objetos (ex.: sobre(X,Y), limpo(a)).

  * predicados definem relações entre objetos
- um átomo pode ser negativo ou positivo (¬limpo(Z) ou limpo(Z)).
  - um átomo positivo ou negativo é chamado de literal
  - Um literal constante não contém variável. Ex.: sobre(a,b).
- Expressões envolvem operadores clássicos da LPO, ex.: sobre(a,b) ∧ sobre(b,c).
- $\theta = \{X_1 / v_1, X_2 / v_2, \ldots, X_n / v_n\}$ representa uma substituição que quando aplicada a uma expressão troca a variável X_i pelo objeto v_i

Linguagem para Descrição do Problema
representação clássica



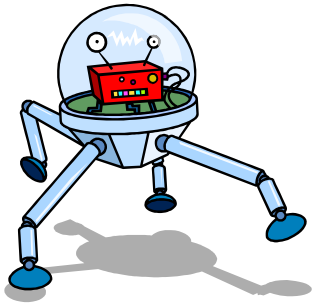


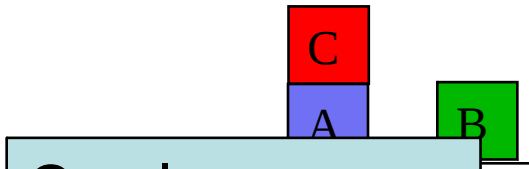
Estado Inicial:
sobre(b,mesa)
sobre(a,mesa)
sobre(c,a)
limpo(c)
limpo(b)

Representação de Estado:
um conjunto de átomos

- estados só representam as propriedades que são verdadeiras em uma configuração do sistema Σ
  ➔ ***suposição de mundo fechado***
- Note que com essa suposição é possível tratar a representação de estados como ***teoria de conjuntos***

Linguagem para Descrição do Problema
representação clássica





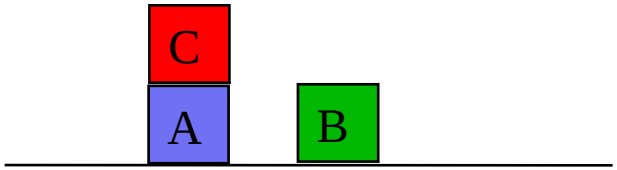
Qualquer condição não mencionada em um estado é considerada falsa.

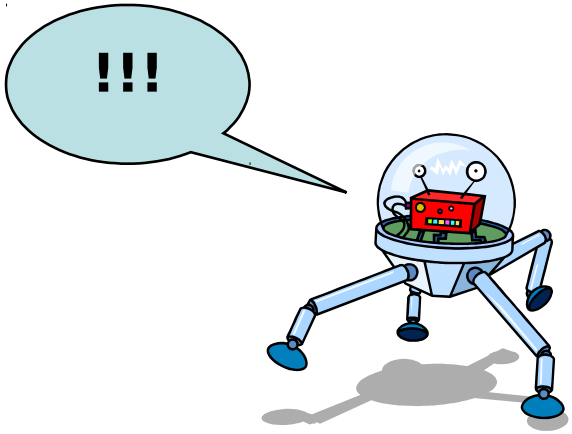
Representação de Estado:
um conjunto de átomos

- estados só representam as proposições verdadeiras em uma configuração do sistema Σ
  ➔ ***suposição de mundo fechado***
- Note que com essa suposição é possível tratar a representação de estados como ***teoria de conjuntos***

Linguagem para Descrição do Problema
representação clássica

Closed World Assumption: aquilo que não está descrito no estado é considerado falso



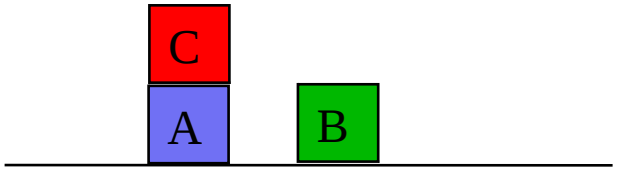


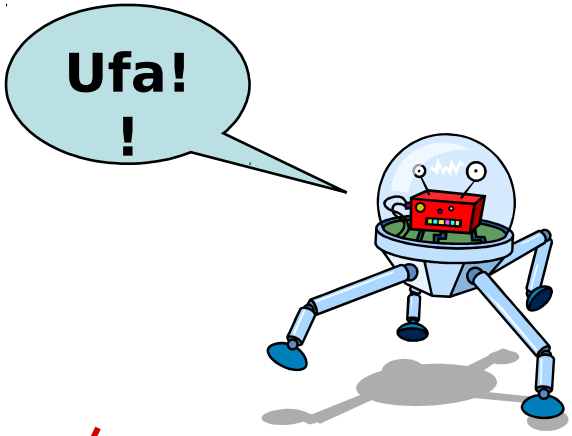
Estado Inicial:
sobre(b,mesa)
sobre(a,mesa)
sobre(c,a)
limpo(c)
limpo(b)

¬ *sobre(c,mesa)*
¬ *sobre(c,b)*
¬ *sobre(a,b)*
¬ *sobre(a,c)*
¬ *sobre(b,c)*
¬ *sobre(b,a)*
¬ *limpo(a)*

Linguagem para Descrição do Problema
representação clássica

Closed World Assumption: aquilo que não está descrito no estado é considerado falso





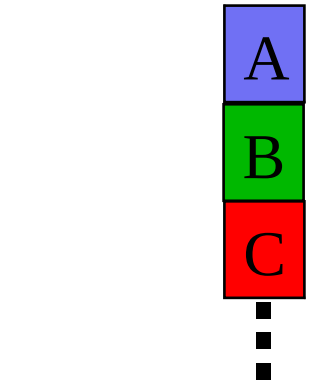
Estado Inicial:
sobre(b,mesa)
sobre(a,mesa)
sobre(c,a)
limpo(c)
limpo(b)

¬ *sobre(c,mesa)*
¬ *sobre(c,b)*
¬ *sobre(a,b)*
¬ *sobre(a,c)*
¬ *sobre(b,c)*
¬ *sobre(b,a)*
¬ *limpo(a)*

Na representação clássica de problemas de planejamento, um estado não descreve o que é falso, mas somente o que é verdadeiro

Descrição de meta

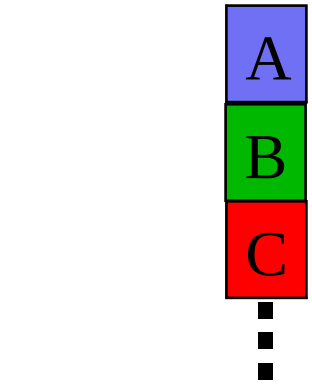
- A descrição da meta G não descreve um Estado Meta completo, mas somente as propriedades desejadas.
- Descrição da meta G: estados parcialmente especificados, representados como uma conjunção de literais positivos e possivelmente com variáveis.



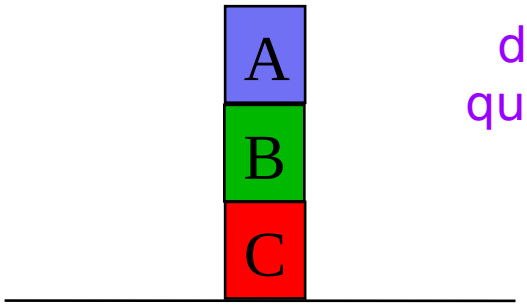
Meta G:
sobre(a,b) ∧ *sobre(b,c)*

Descrição de meta

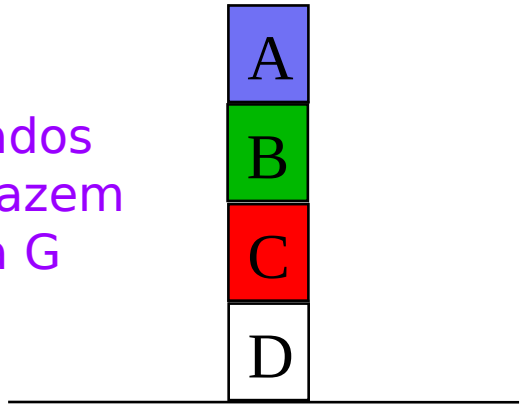
- A descrição da meta G não descreve um Estado Meta completo, mas somente as propriedades desejadas .
- Descrição da meta G: estados parcialmente especificados, representados como uma conjunção de literais positivos e possivelmente com variáveis.



Meta G:
sobre(a,b) ∧ sobre(b,c)



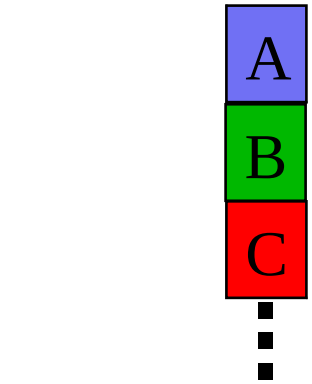
dois estados
que satisfazem
a meta G



Descrição de meta

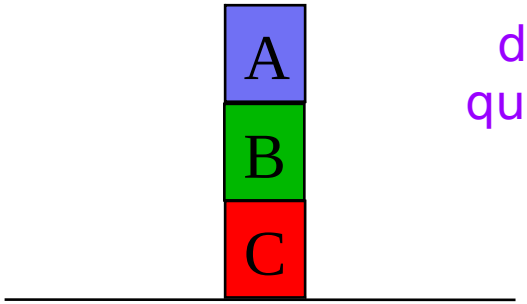
- A descrição da meta G não descreve um Estado Meta completo, propriedad
- Descrição parcialmer representa conjunção possivelme

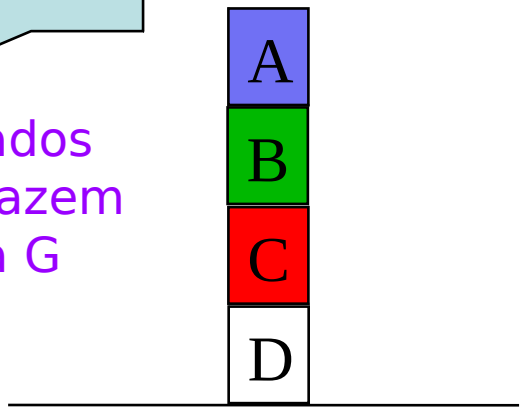
Um estado s satisfaz um objetivo G se s contém todos os literais em G (e possivelmente outros)



Meta G:
sobre(a,b) ∧ sobre(b,c)

dois estados que satisfazem a meta G





Descrição de ações

- Como descrever as “transições” ou ações?
- Vamos ver como operadores e ações são descritos pelo STRIPS

Descrição de operadores e ações no STRIPS

Operator: representação com variáveis de ações, dada pela tripla (name, precond, effects):

name: predicado lógico (nome da ação e lista de parâmetros) da forma $n\ (X_1,\ldots,X_k)$

precond: precondições que devem ser verdadeiras para ser possível usar/executar o operador

effects: lista de efeitos (conjunto de literais positivos e negativos) que serão verdadeiros e os que serão falsos, após a execução do operador

operator: (move (X,*Z*,*Y*)
;; move bloco X de cima do bloco Z, para cima do bloco Y

Descrição de operadores e ações no STRIPS

Operator: representação com variáveis de ações, dada pela tripla (name, precond, effects):

name: predicado lógico (nome da ação e lista de parâmetros) da forma $n\ (x_1,\ldots,x_k)$

precond: precondições que devem ser verdadeiras para ser possível usar/executar o operador

effects: lista de efeitos (conjunto de literais positivos e negativos) que serão verdadeiros e os que serão falsos, após a execução do operador

operator: (move (X,*Z*,*Y*)
;; move bloco x de cima do bloco z, para cima do bloco y
precond: {limpo(X), sobre(X,Z), limpo(Y)}
effects: {limpo(Z), sobre(X,Y), ¬ sobre(X,*Z*), ¬limpo(Y) })

Descrição de operadores e ações no STRIPS

Chamamos de ações as instâncias de um operador

operator: (move (X,*Z*,*Y*)
;; move bloco X de cima do bloco Z, para cima do bloco Y
precond: {limpo(X), sobre(X,Z), limpo(Y)}
effects: {limpo(Z), sobre(X,Y), ¬ sobre(X,*Z*), ¬limpo(Y)})

substituição = {X/c, Y/a, Z/b}

action: (move (c,*a*,*b*)
;; move bloco c de cima do bloco a, para cima do bloco b
precond: {limpo(c), sobre(c,a), limpo(b)}
effects: {limpo(a), sobre(c,b), ¬ sobre(c,*a*), ¬ limpo(b) })

Como as ações afetam os estados

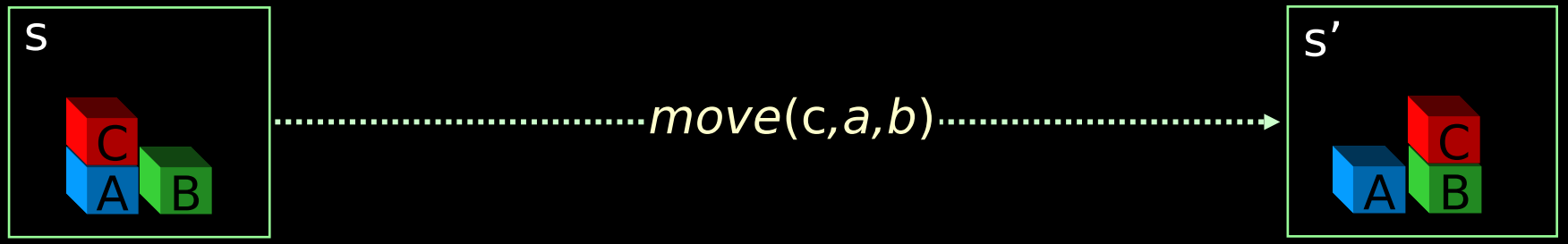
- A aplicabilidade de um operador envolve uma substituição para as variáveis na precondição.
- Uma ação é aplicável em todos os estados que satisfazem as suas precondições.

Semântica de Operadores

(na geração progressiva de estados sucessores)

operator: (*move*(X,Y,Z),
precond: { *limpo*(X), *limpo*(Z), *sobre*(X,Y) },
effects: { *limpo*(Y), *sobre*(X,Z), ¬*limpo*(Z), ¬*sobre*(X,Y) })

substituição = {X/c, Y/a, Z/b}



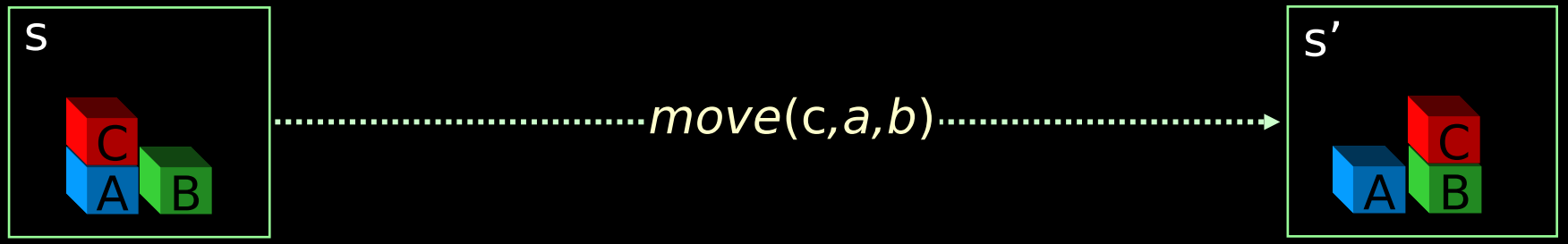
limpo(b)
limpo(c)
sobre (a,mesa)
Sobre(b,mesa)
sobre(c,a)

Semântica de Operadores

(na geração progressiva de estados sucessores)

action: (*move*(c,a,b),
precond: { *limpo*(c), *limpo*(b), *sobre*(c,a) },
effects: { *limpo*(a), *sobre*(c,b), ¬*limpo*(b), ¬*sobre*(c,a) })

s' = s + efeitos(positivos) – efeitos(negativos)



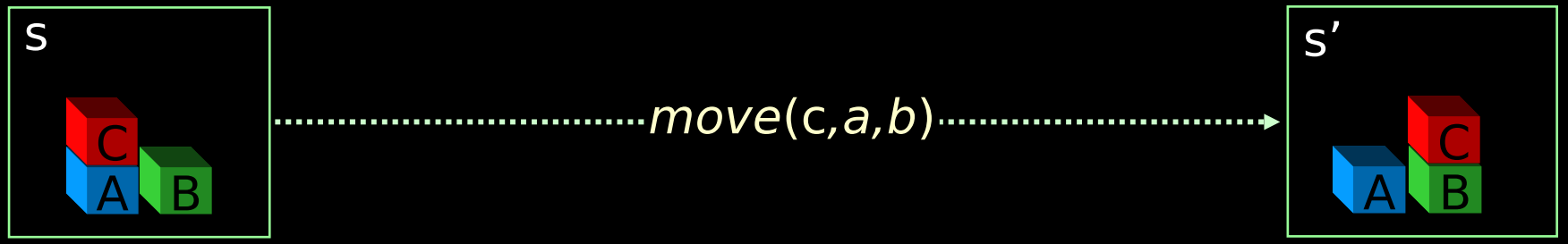
limpo(b)
limpo(c)
sobre (a,mesa)
Sobre(b,mesa)
sobre(c,a)

Semântica de Operadores

(na geração progressiva de estados sucessores)

action: (*move*(c,a,b),
precond: { *limpo*(c), *limpo*(b), *sobre*(c,a) },
effects: { *limpo*(a), *sobre*(c,b), ¬*limpo*(b), ¬*sobre*(c,a) })

s' = s + efeitos(positivos) – efeitos(negativos)



limpo(b)
limpo(c)
sobre (a,mesa)
Sobre(b,mesa)
sobre(c,a)

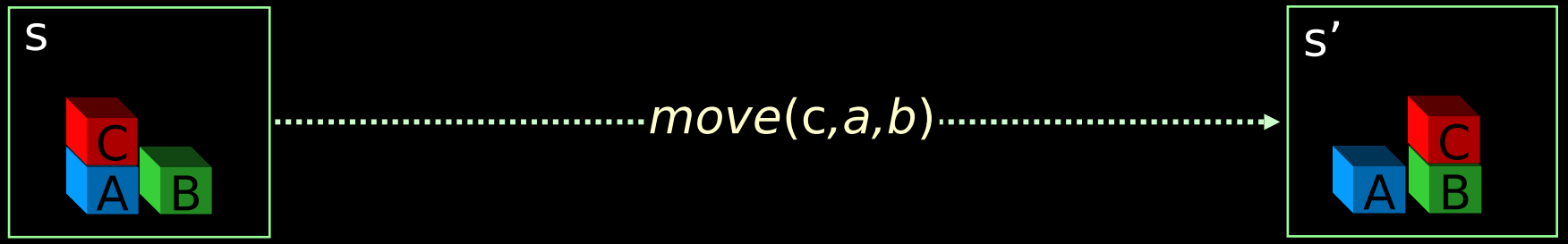
limpo(b)
limpo(c)
sobre(a,mesa)
sobre(b,mesa)
sobre(c,a)

Semântica de Operadores

(na geração progressiva de estados sucessores)

action: (*move*(c,a,b),
precond: { *limpo*(c), *limpo*(b), *sobre*(c,a) },
effects: { *limpo*(a), *sobre*(c,b), ¬*limpo*(b), ¬*sobre*(c,a) })

s' = s + efeitos(positivos) – efeitos(negativos)



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)

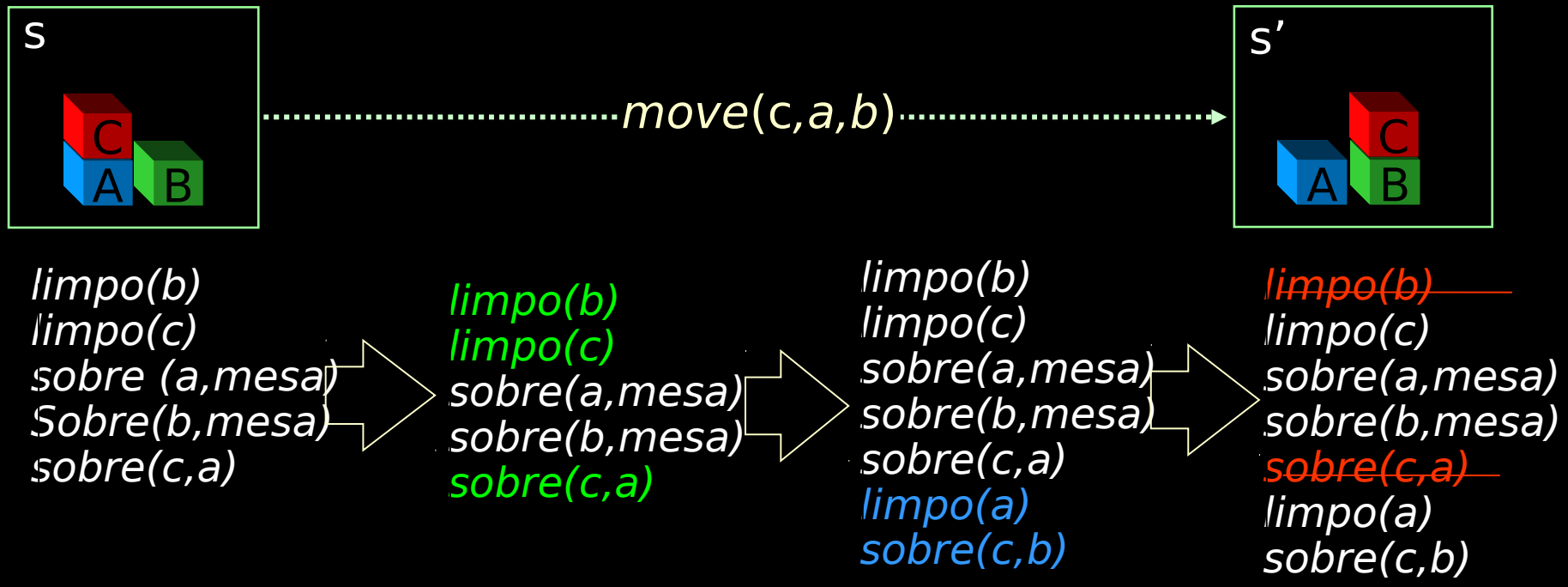
limpo(b)
limpo(c)
sobre(a,mesa)
sobre(b,mesa)
sobre(c,a)
limpo(a)
sobre(c,b)

Semântica de Operadores

(na geração progressiva de estados sucessores)

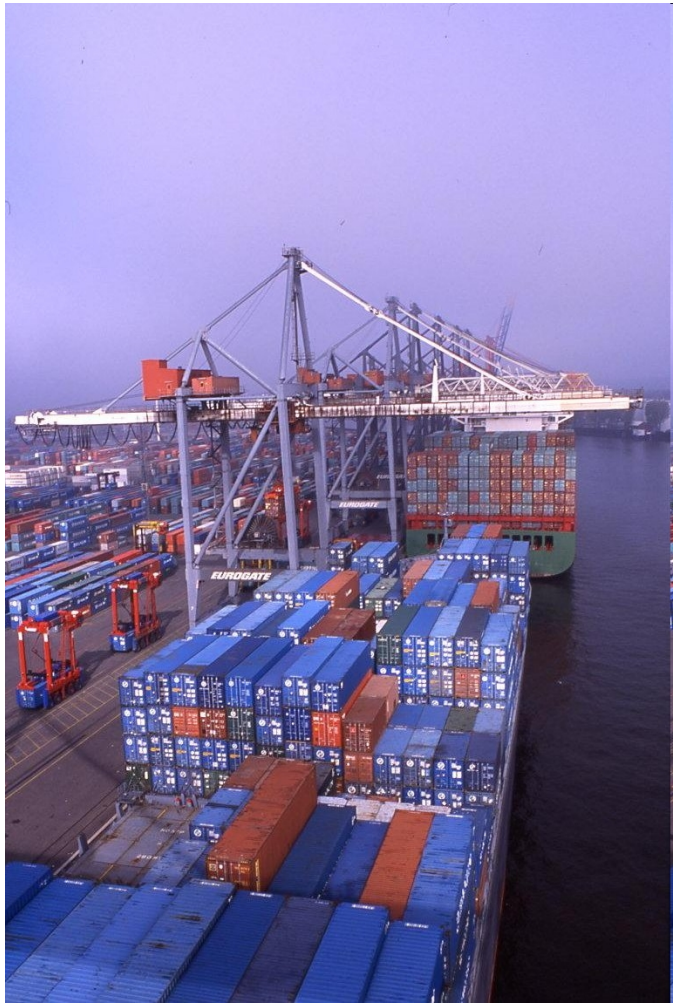
action: (*move*(c,a,b),
precond: { *limpo*(c), *limpo*(b), *sobre*(c,a) },
effects: { *limpo*(a), *sobre*(c,b), ¬*limpo*(b), ¬*sobre*(c,a) })

s' = s + efeitos(positivos) – efeitos(negativos)



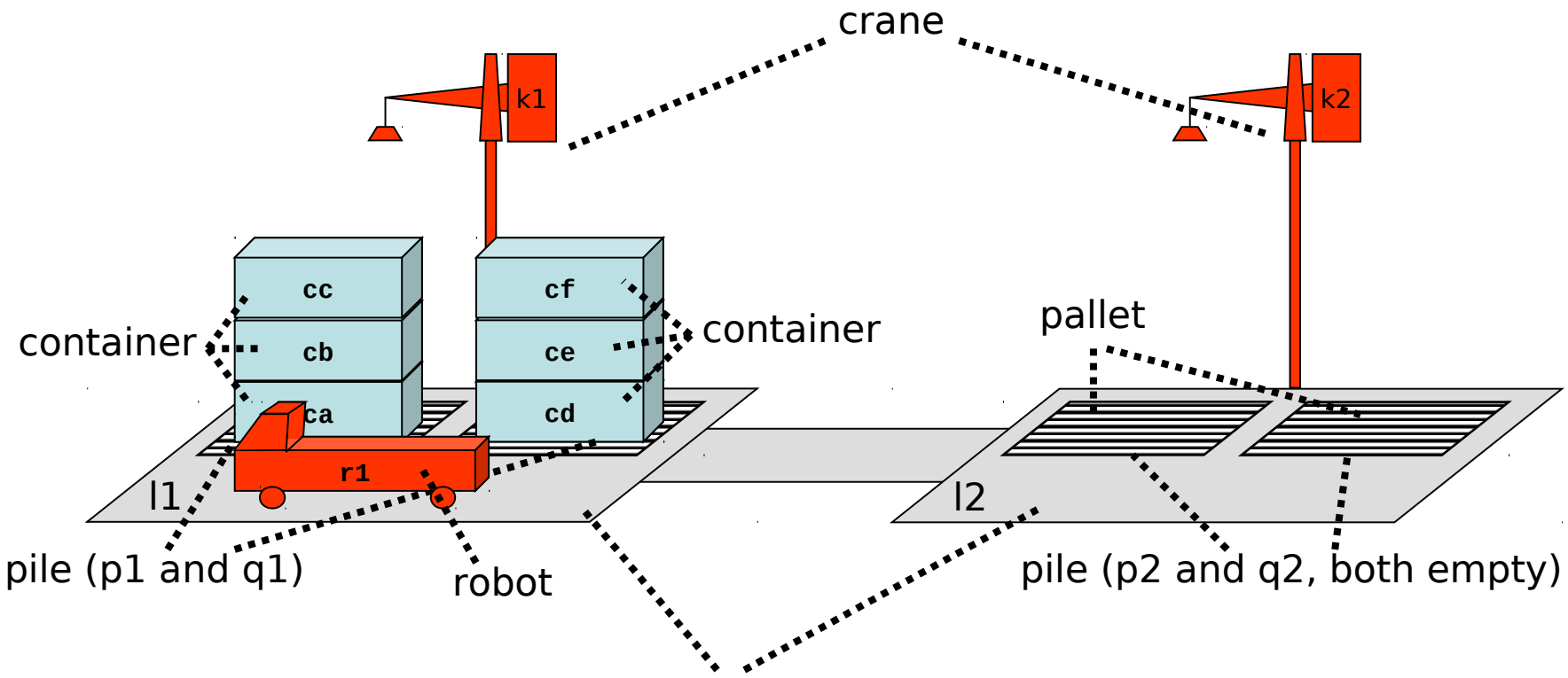
Domínio dos Robôs Portuários (Dock-Worker Robots -DWR)

- Porto, com vários locais (docas), navios ancorados , áreas de armazenamento de containers, e áreas de estacionamento para caminhões e trens.
- Guindastes (crane) para carregar e descarregar os navios, e carrinhos robô para mover os containers

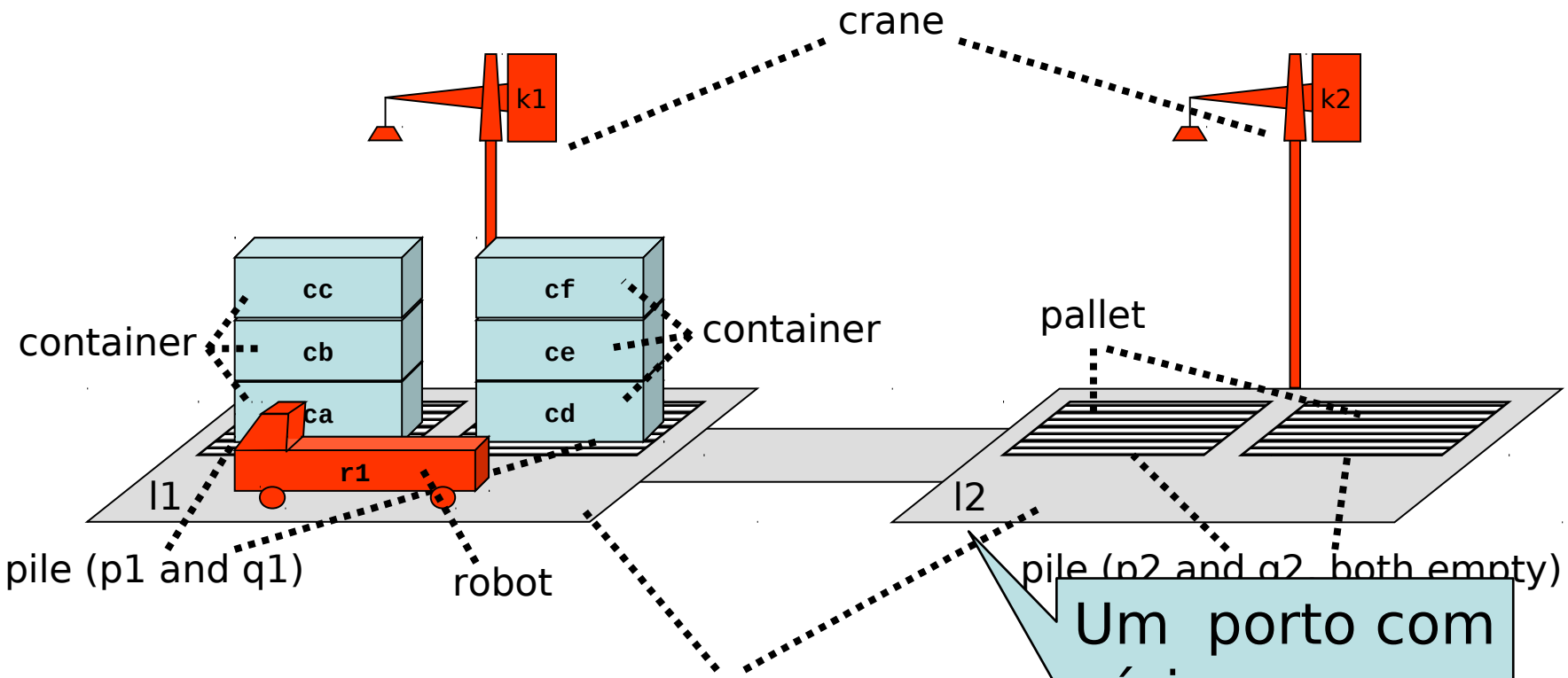


38

DWR: Objetos

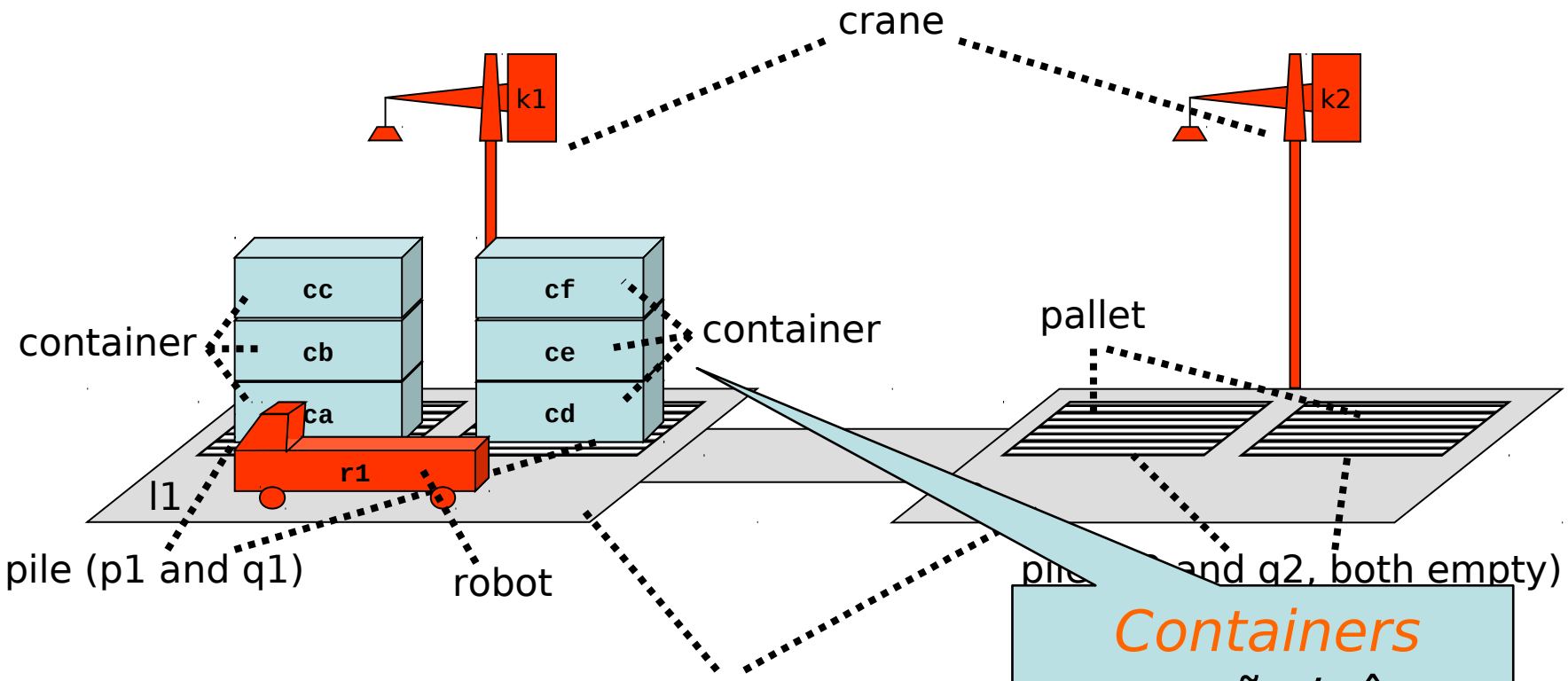

crane
k1
k2
cc
cb
ca
cf
ce
cd
container
container
pallet
r1
l1
l2
pile (p1 and q1)
robot
pile (p2 and q2, both empty)

DWR: Objetos



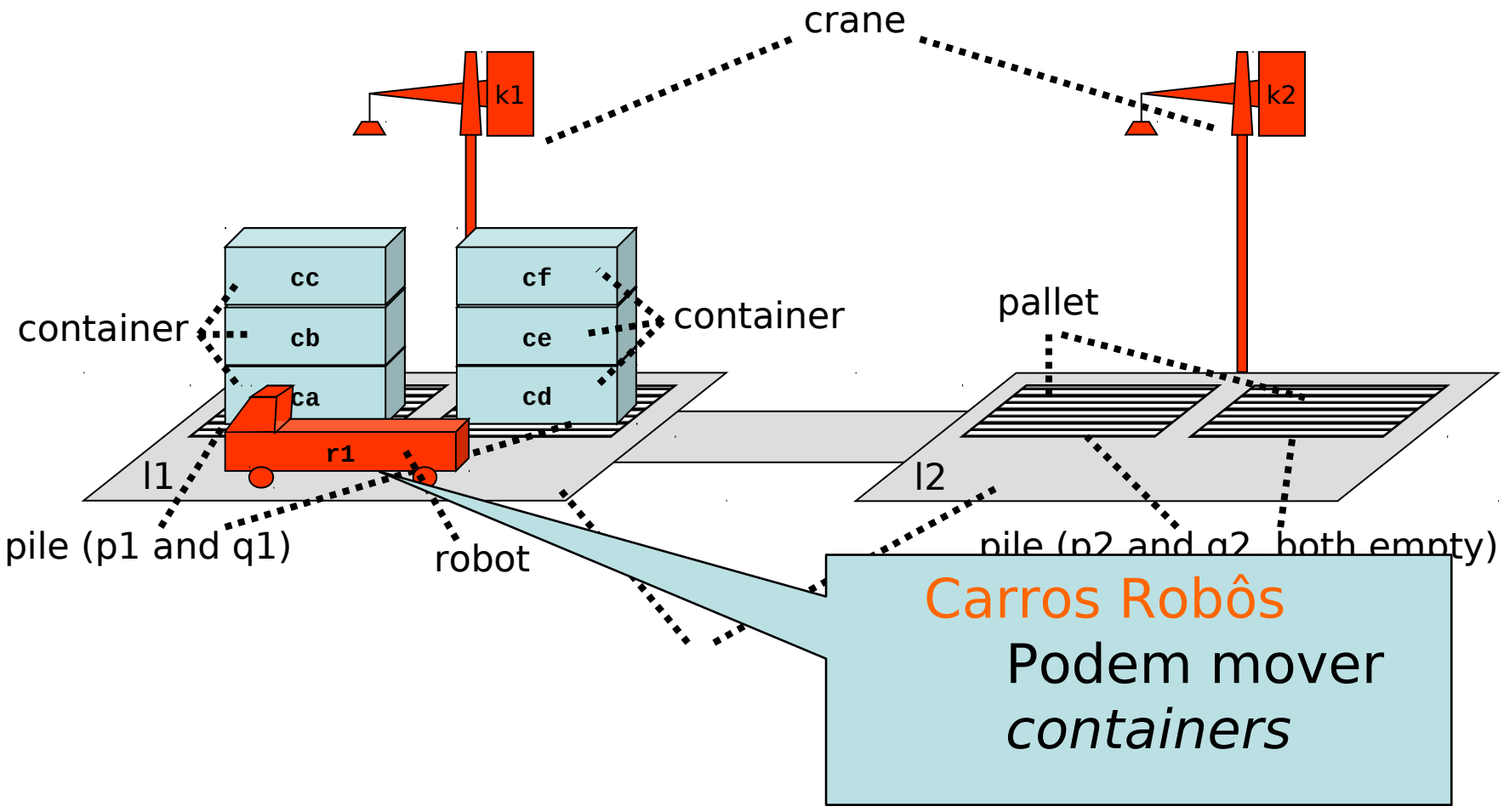
Um porto com várias localizações

DWR: Objetos

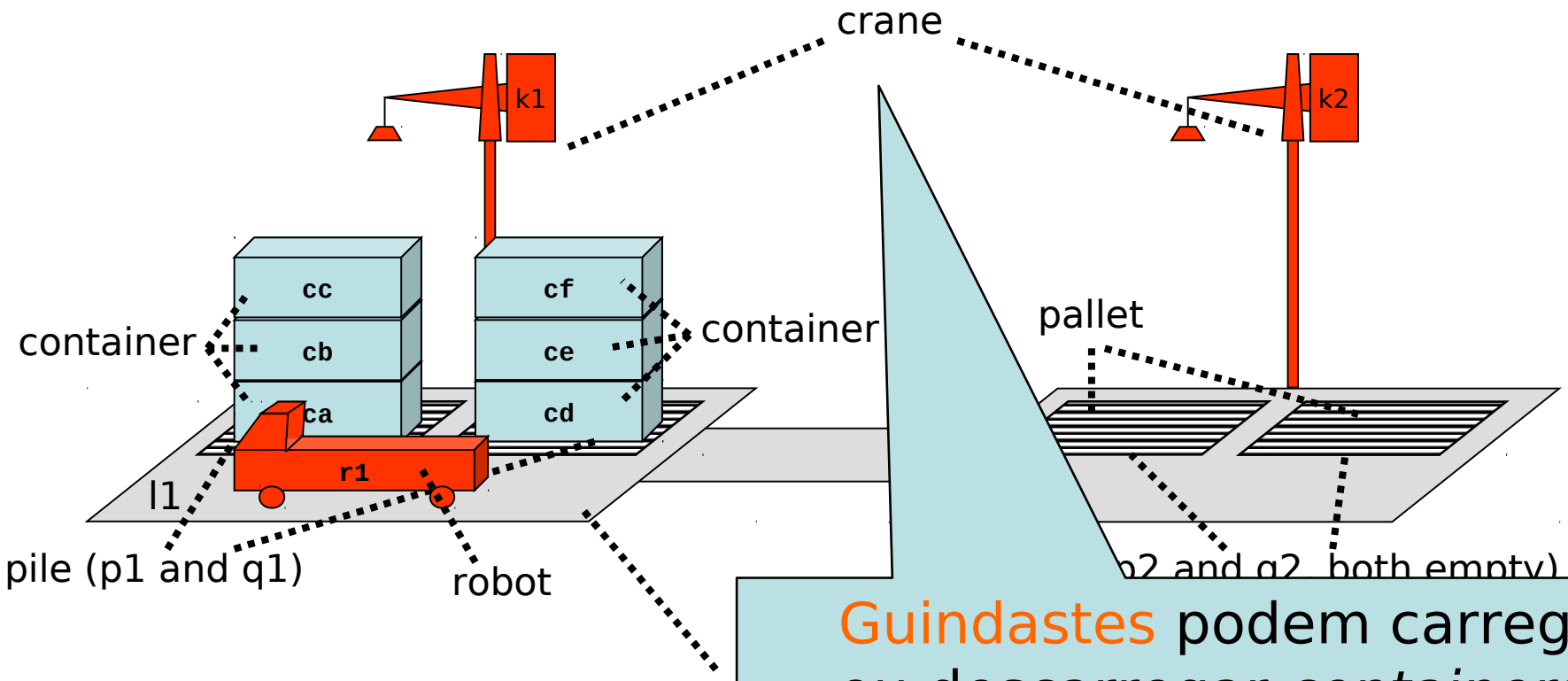


Containers
vão/vêm
de navios

DWR: Objetos


crane
k1
k2
cc
cb
ca
cf
ce
cd
container
container
pallet
r1
l1
l2
pile (p1 and q1)
robot
pile (p2 and q2 both empty)
Carros Robôs
Podem mover
containers

DWR: Objetos


crane
k1
k2
cc
cb
ca
cf
ce
cd
container
container
pallet
r1
l1
pile (p1 and q1)
robot
Guindastes podem carregar
ou descarregar *containers*

DWR: Objetos

Localizações: l1, l2, ...

Containers: c1, c2, ...

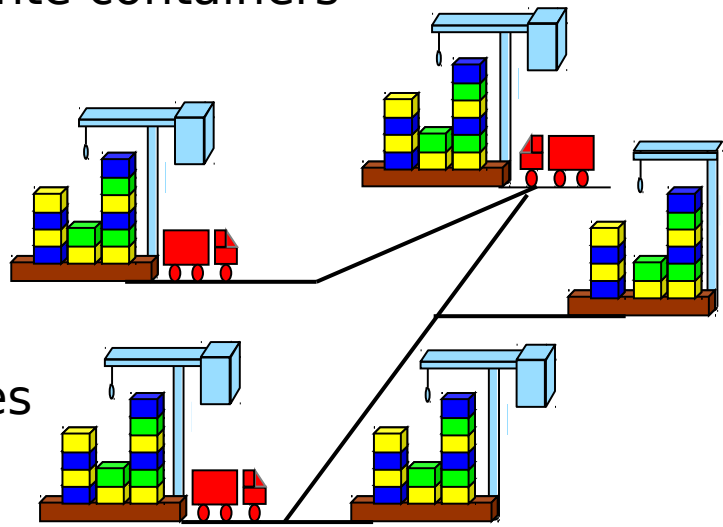
- Podem ser empilhados, carregados sobre os robôs, ou carregados pelos guindastes

Pile: p1, p2, ...

- Inclui um pallet no fundo com possivelmente containers empilhados nele

Pallets: pallet

- Plataforma fixa no fundo de cada pilha



Carros Robôs: r1, r2, ...

- Podem mover para localizações adjacentes
- carregam no máximo um container

Guindastes: k1, k2, ...

- cada um pertence a uma única localização
- carrega um container de uma pilha para um carro robô e vice-e-versa
- Se há uma pilha em uma localização então deve haver também um guindaste na mesma localização

DWR: predicados

- **Relações fixas:** é a mesma em todos os estados

  adjacent(L,*L'*) attached(P,*L*) belong(*K*,*L*)

- **Relações dinâmicas (fluentes):** diferem de um estado para outro

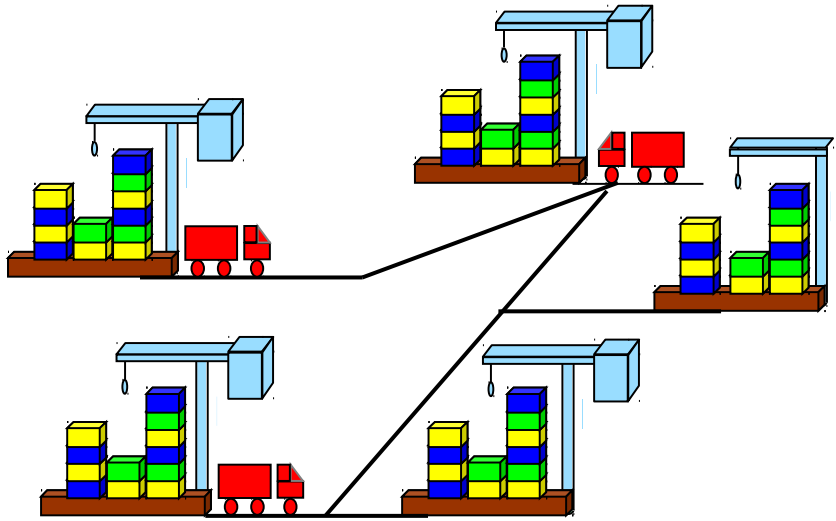
  occupied(L) at(R,*L*)

  loaded(R,C) unloaded(R)

  holding(K,C) empty(K)

  in(C,P) on(C,C')

  top(C,P)

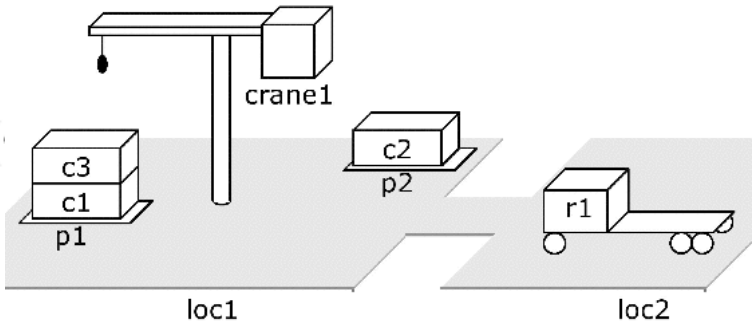


DWR: ações

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)

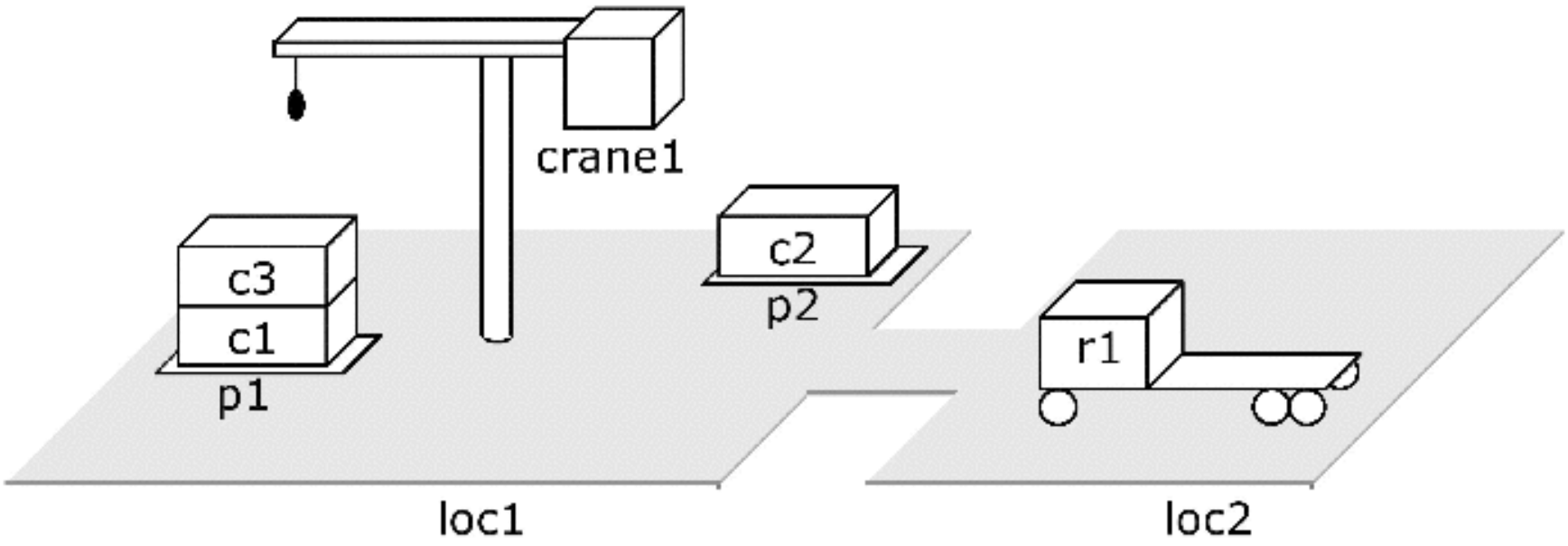
unload(k, l, c, r)
;; crane k at location l takes container c from robot r
precond: belong(k, l), at(r, l), loaded(r, c), empty(k)
effects: $\neg$ empty(k), holding(k, c), unloaded(r), $\neg$ loaded(



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)

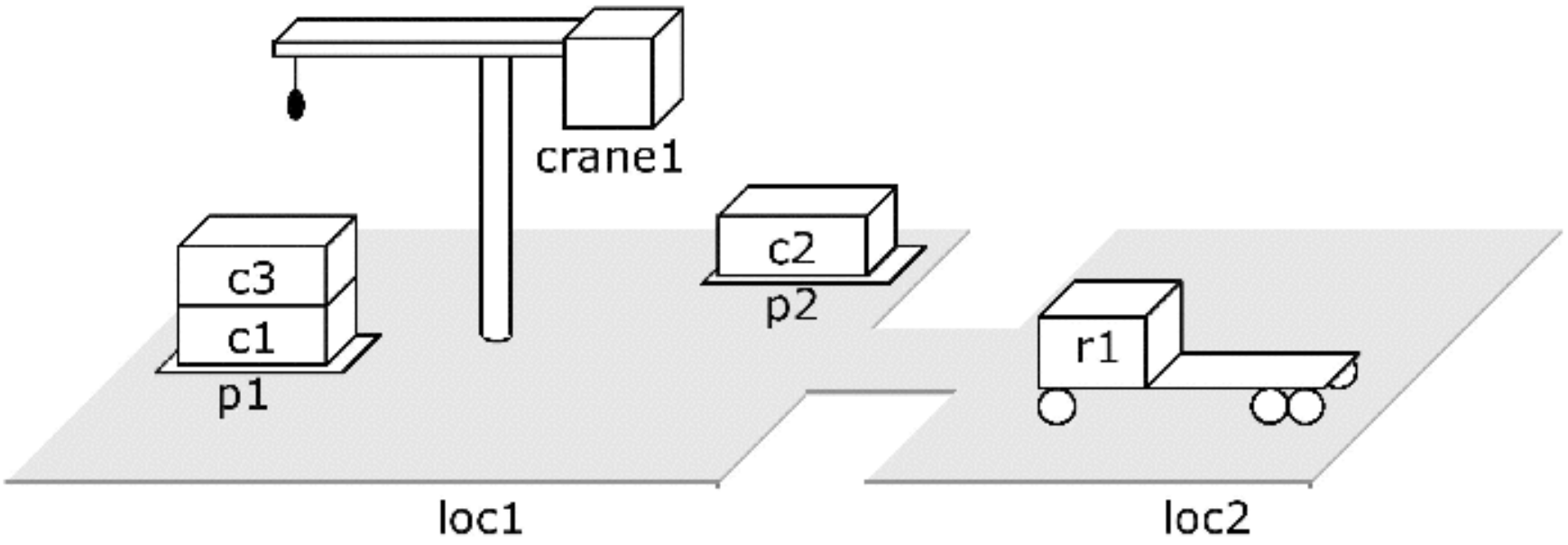
take(k, l, c, d, p)
;; crane k at location l takes c off of d in pile p
precond: belong(k, l), attached(p, l), empty(k), top(c, p), on(c, d)
effects: holding(k, c), $\neg$ empty(k), $\neg$ in(c, p), $\neg$ top(c, p), $\neg$ on(c, d), top(d, p)

Exemplo de um estado para o problema dos Robôs Portuários



{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)}.

Exemplo de um estado para o problema dos Robôs Portuários



top(c3,p1), on(c3,c1), on(c1,p
on(c2,pallet), belong(crane1,l
cent(loc2,loc1), at(r1,loc2), o

p1),
p2),
dja-

Quais ações podem ser aplicadas e quais os resultados da aplicação dessas ações?

Exemplo da aplicação de uma ação para o problema dos Robôs Portuários

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)

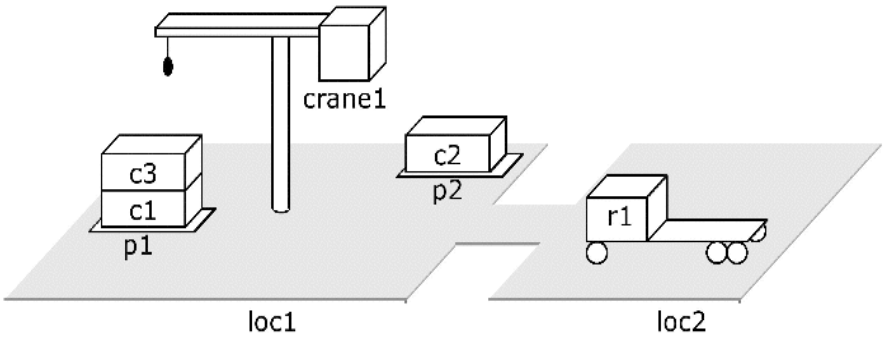


Figure 2.2: The DWR state s_1={attached(p1,loc1), in(c1,p1), in(c3,p1), top(c3,p1), on(c3,c1), on(c1,pallet), attached(p2,loc1), in(c2,p2), top(c2,p2), on(c2,pallet), belong(crane1,loc1), empty(crane1), adjacent(loc1,loc2), adjacent(loc2,loc1), at(r1,loc2), occupied(loc2), unloaded(r1)}.

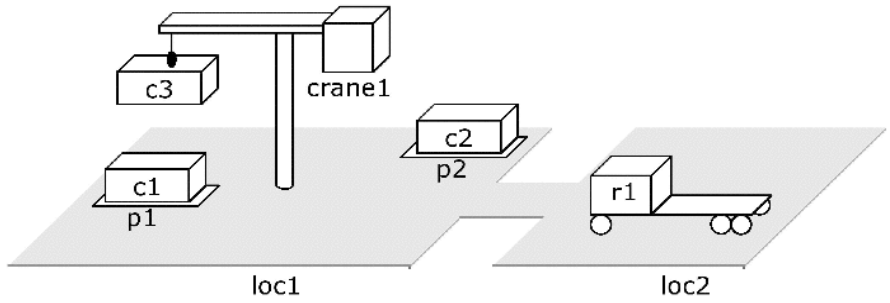


Figure 2.3: The DWR state s_5 = {attached(p1,loc1), in(c1,p1), top(c1,p1), on(c1,pallet), attached(p2,loc1), in(c2,p2), top(c2,p2), on(c2,pallet), belong(crane1,loc1), holding(crane1,c3), adjacent(loc1,loc2), adjacent(loc2,loc1), at(r1,loc2), occupied(loc2), unloaded(r1)}.