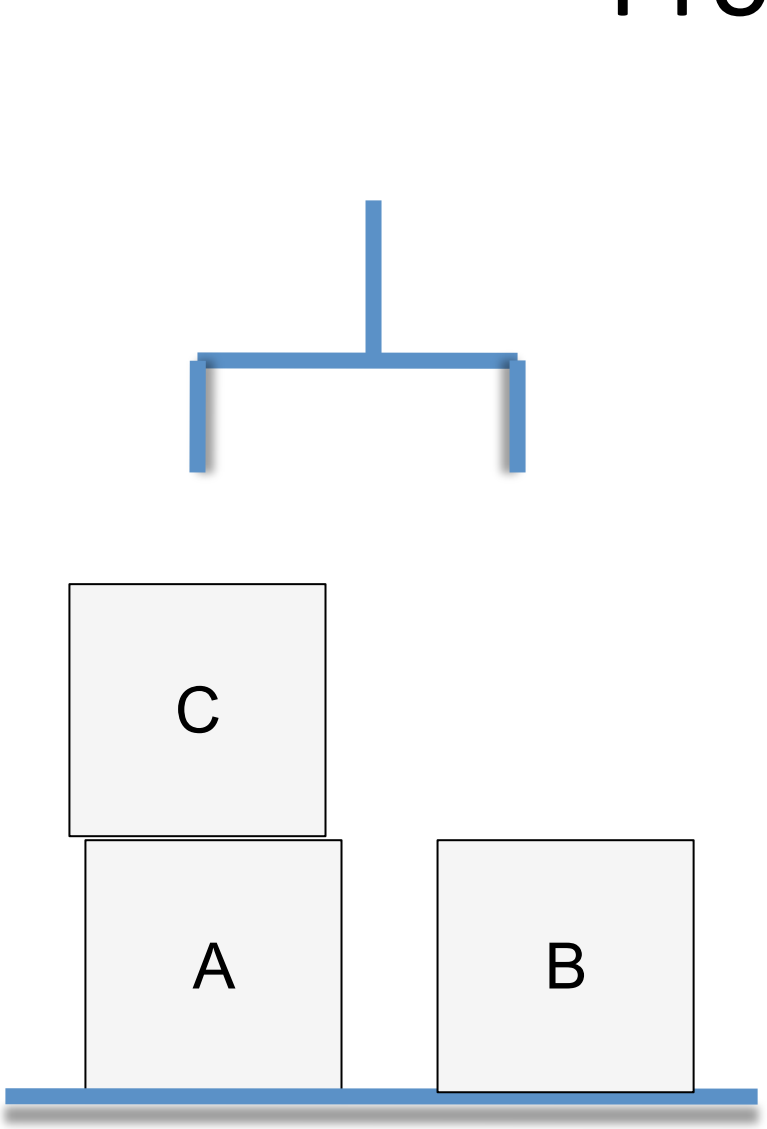
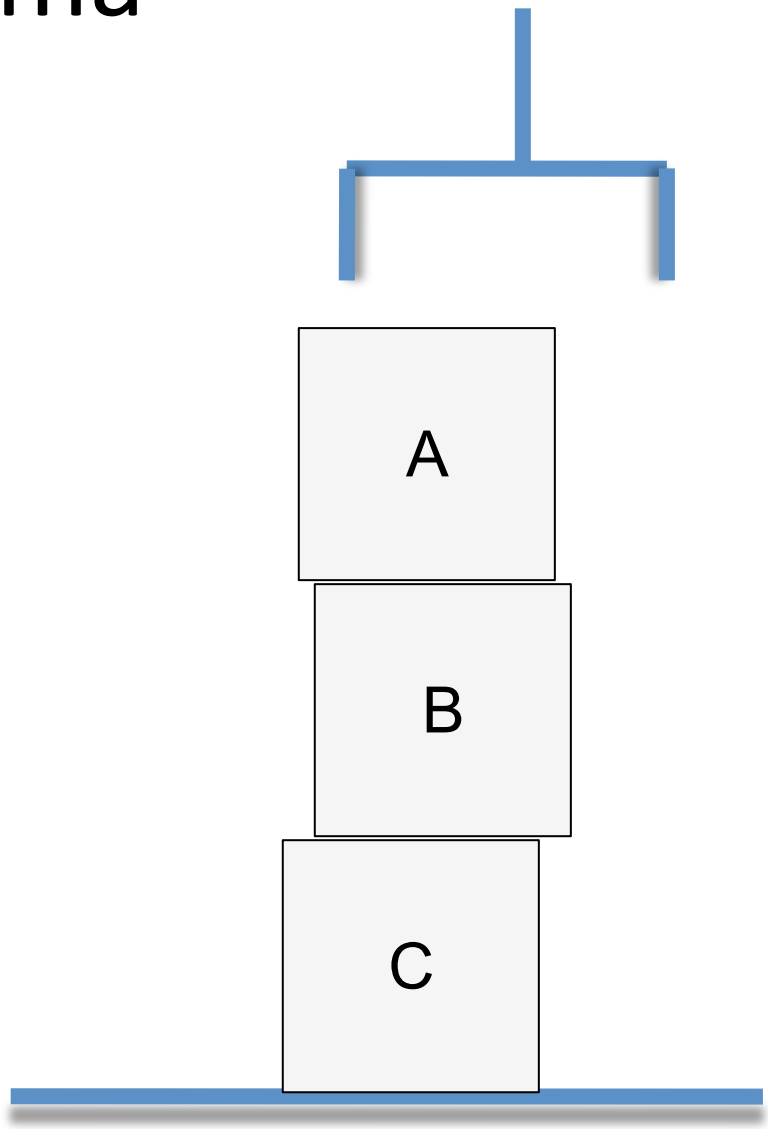
Curs 10 – Planificare și execuția planurilor

Problema

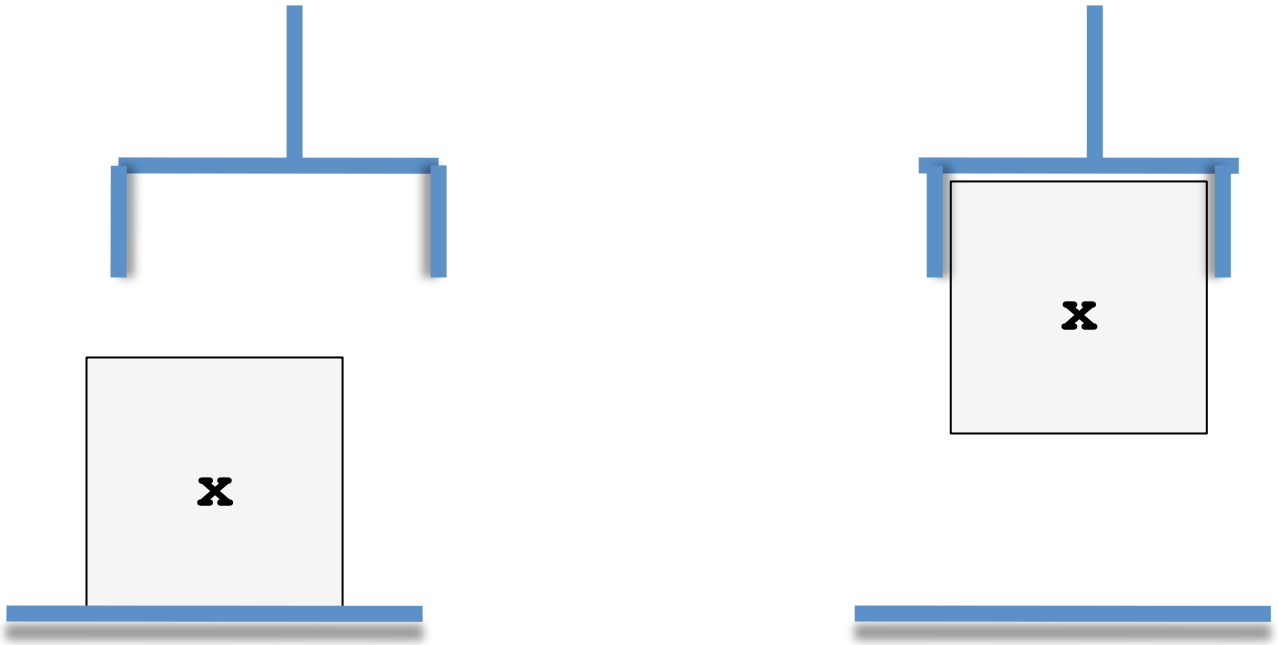


starea iniţială



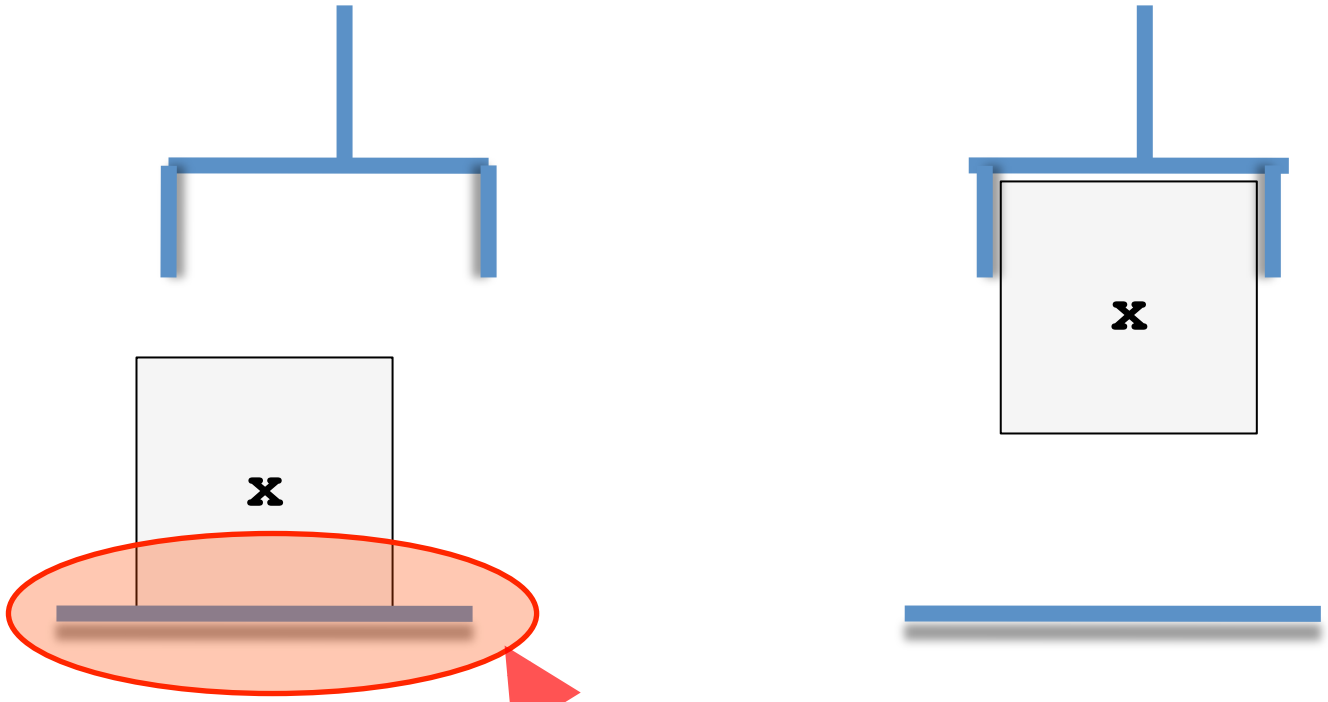
starea finală

Reguli STRIPS



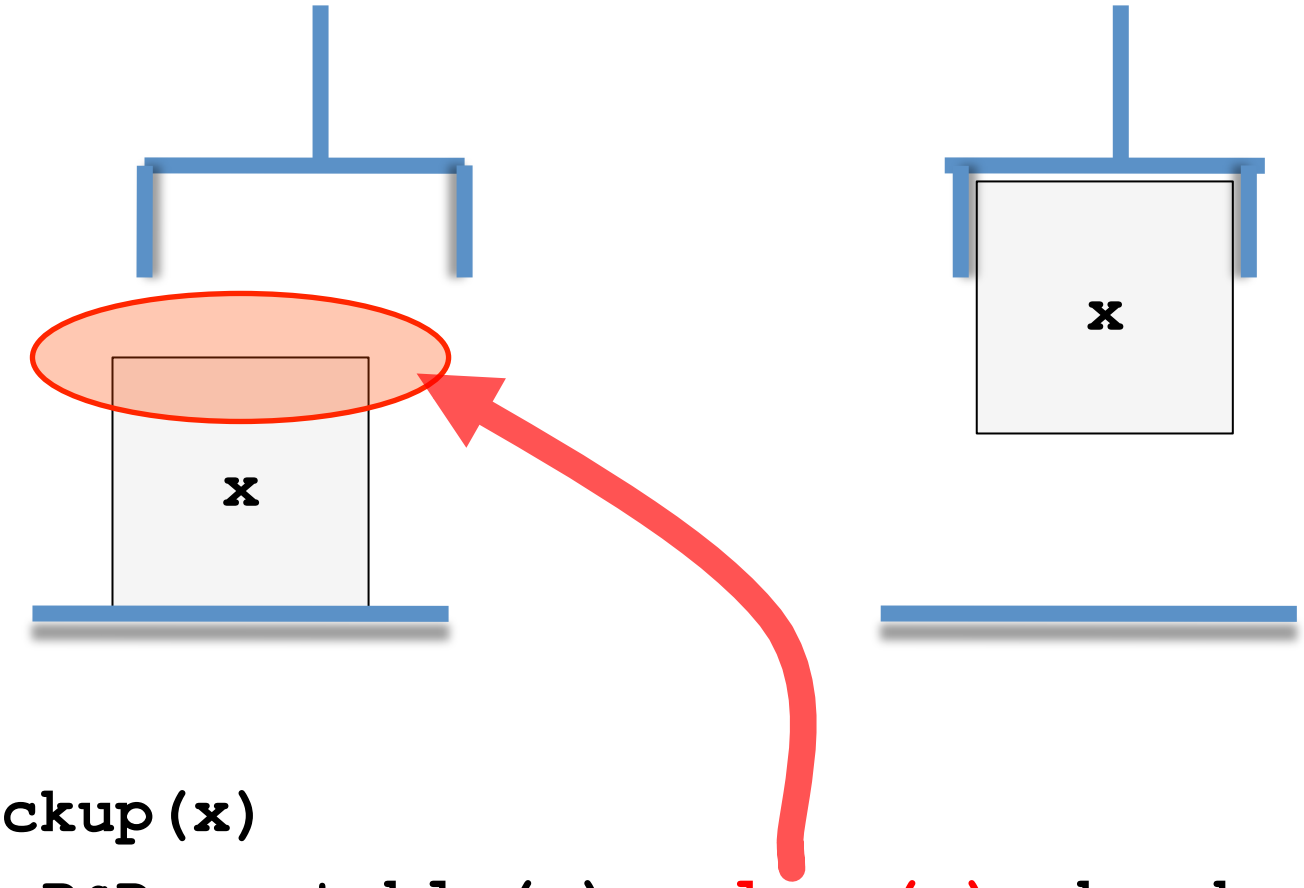

```
pickup(x)
    P&D: ontable(x), clear(x), handempty
    A: holding(x)
```

Reguli STRIPS



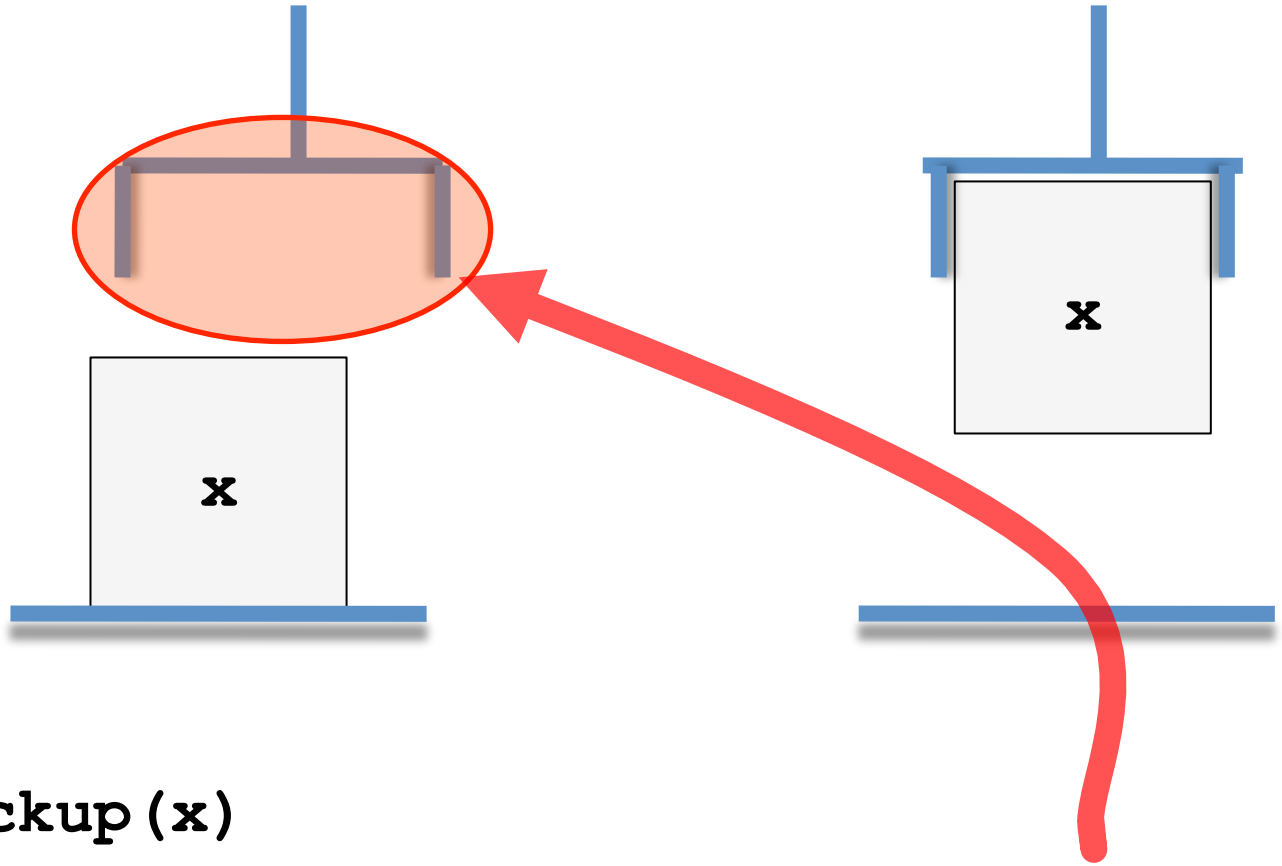

```
pickup(x)
    P&D: ontable(x), clear(x), handempty
    A: holding(x)
```

Reguli STRIPS



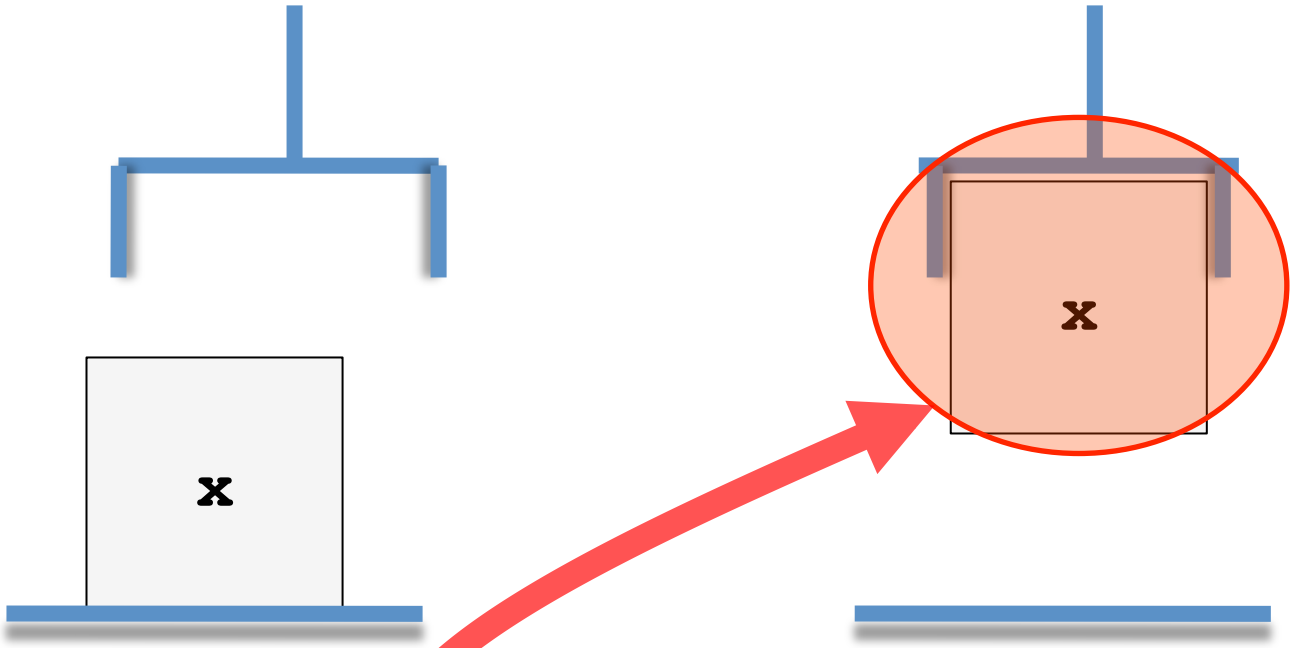

```
pickup(x)
    P&D: ontable(x), clear(x), handempty
    A: holding(x)
```

Reguli STRIPS



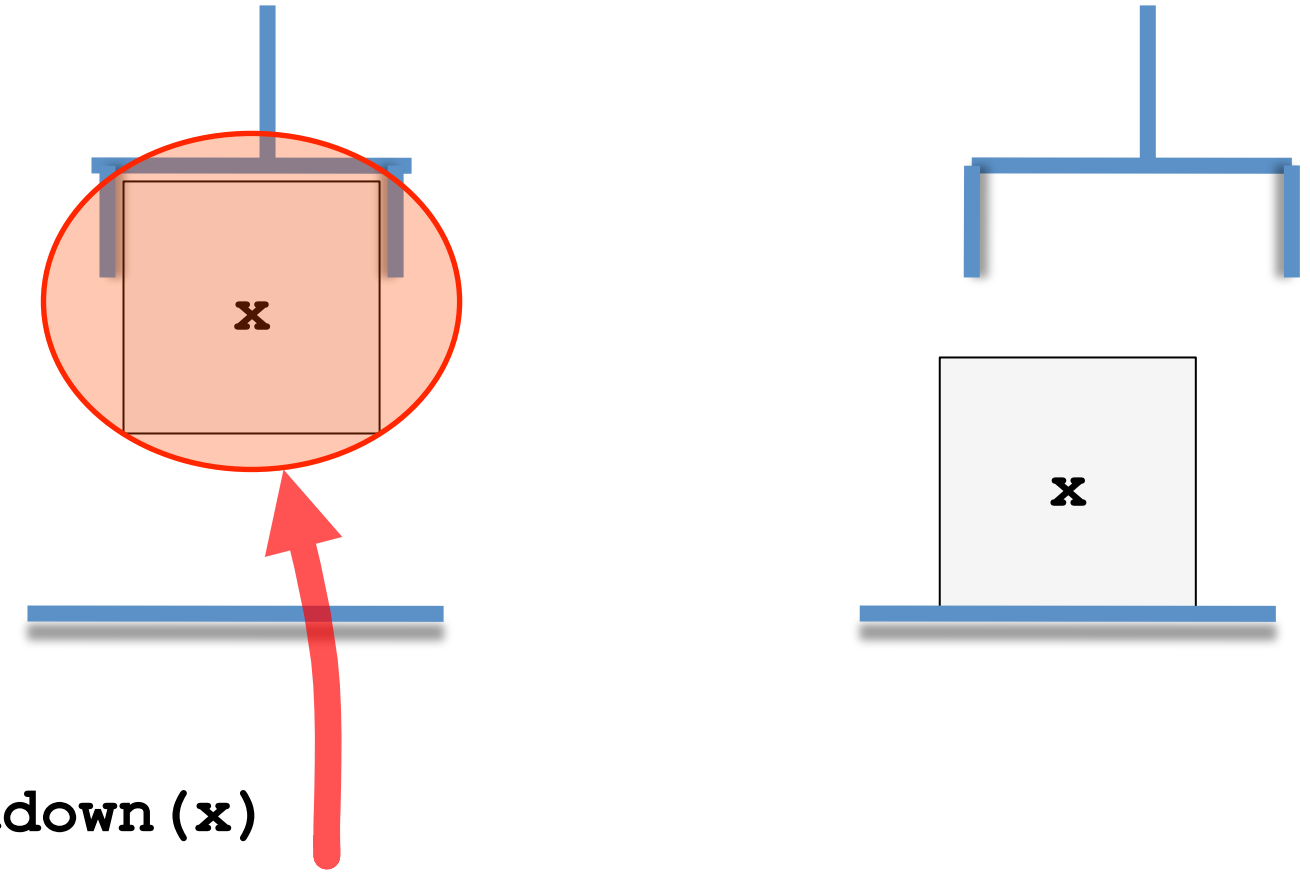

```
pickup(x)
    P&D: ontable(x), clear(x), handempty
    A: holding(x)
```

Reguli STRIPS



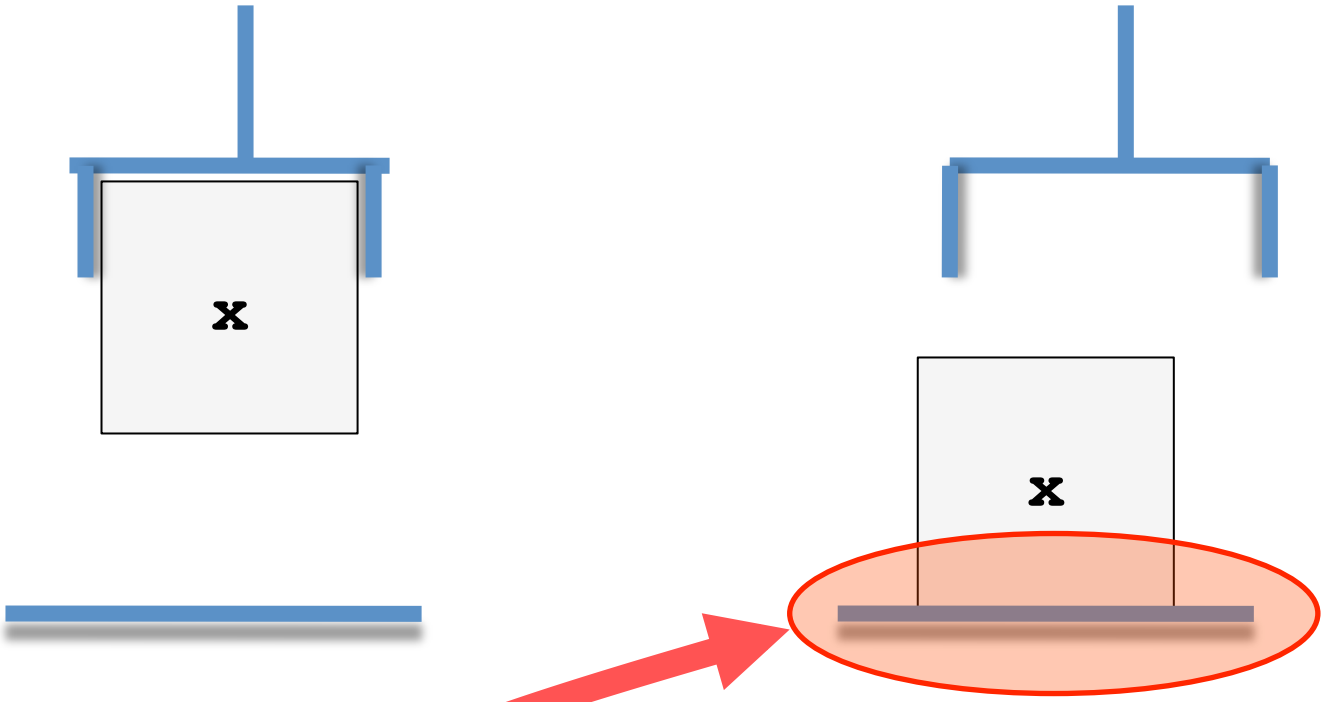

```
pickup(x)
    P&D: ontable(x), clear(x), handempty
    A: holding(x)
```

Reguli STRIPS



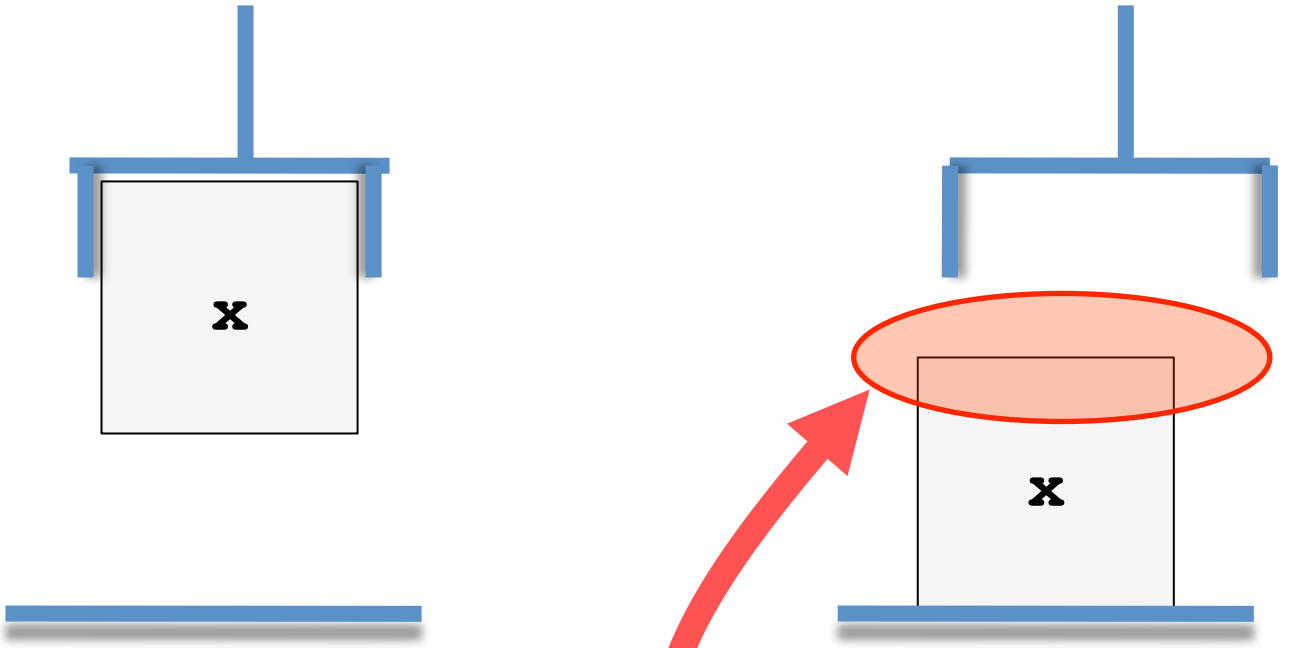

```
putdown(x)
    P&D: holding(x)
    A: ontable(x), clear(x), handempty
```

Reguli STRIPS



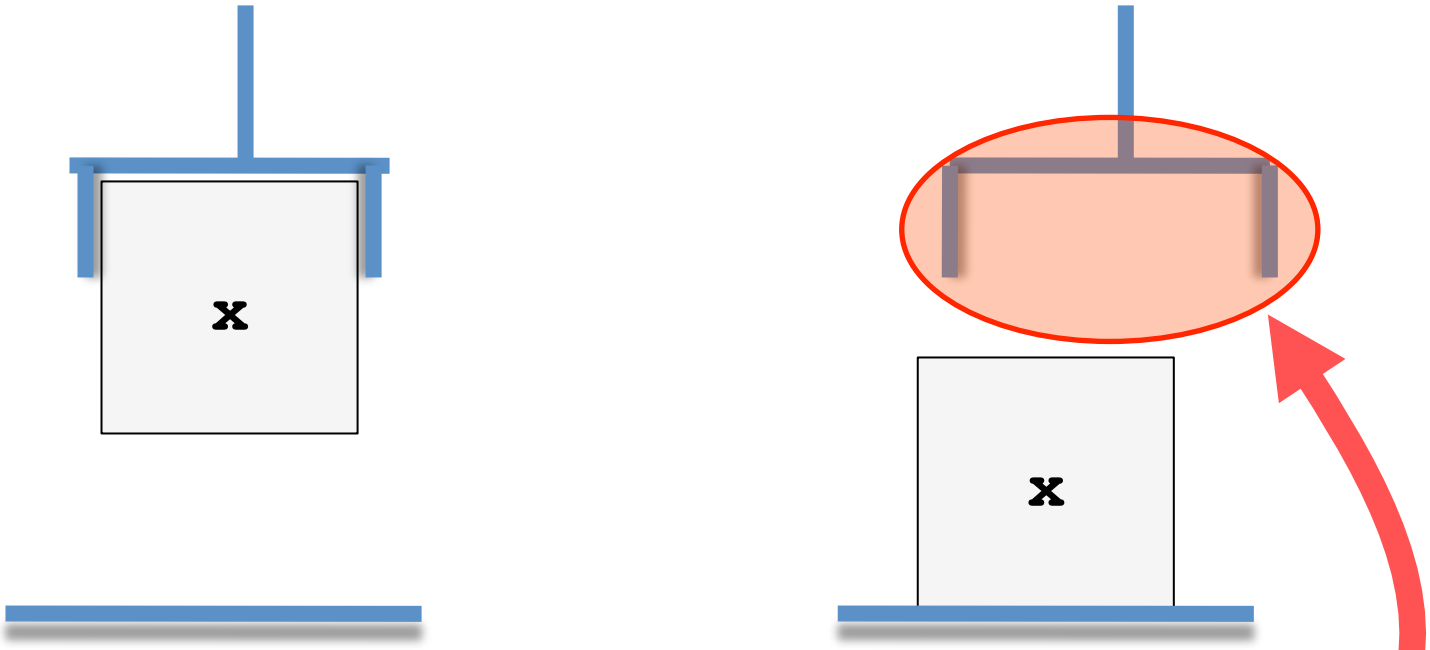

```
putdown(x)
    P&D: holding(x)
    A: ontable(x), clear(x), handempty
```

Reguli STRIPS



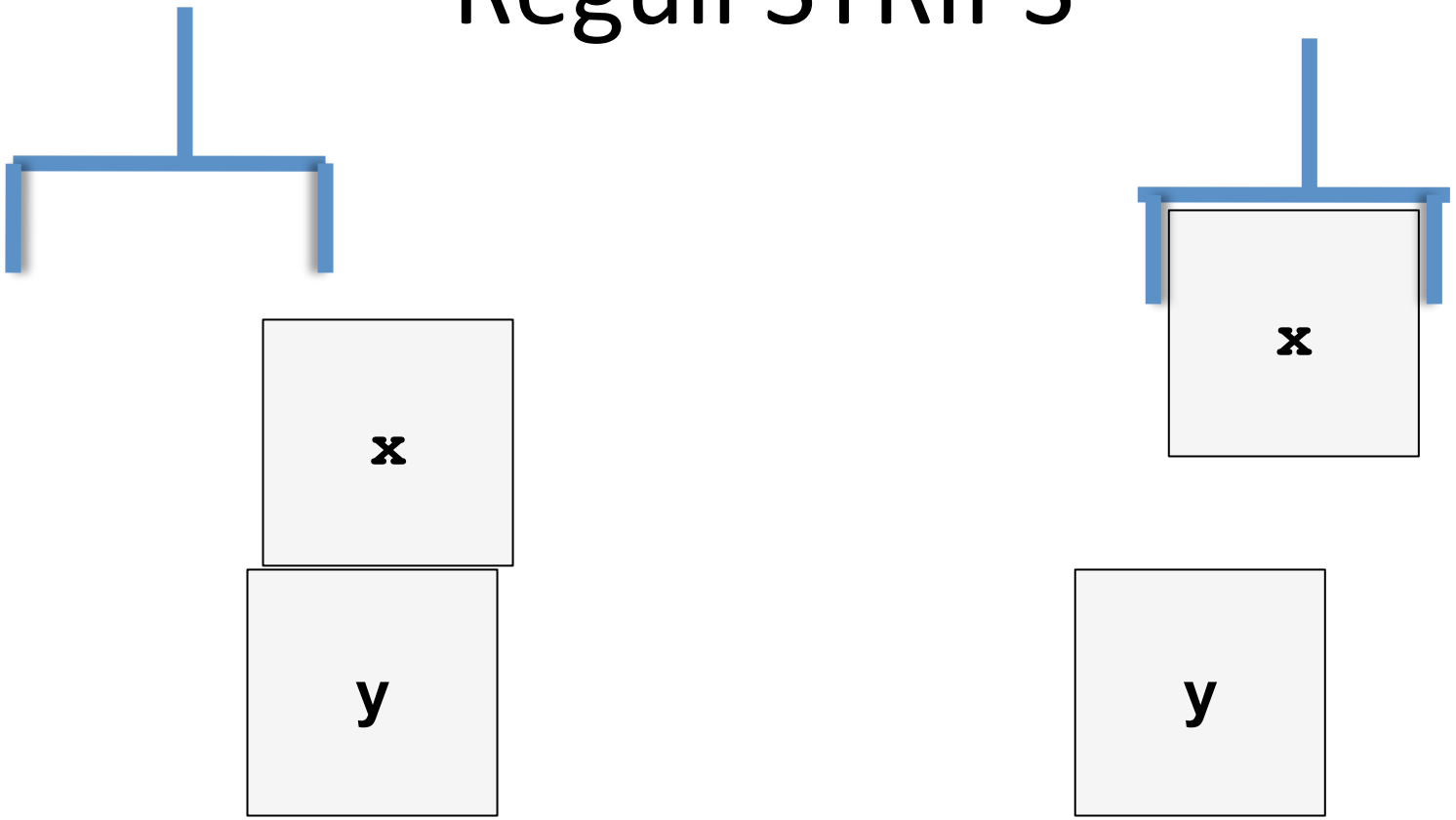

```
putdown(x)
    P&D: holding(x)
    A: ontable(x), clear(x), handempty
```

Reguli STRIPS



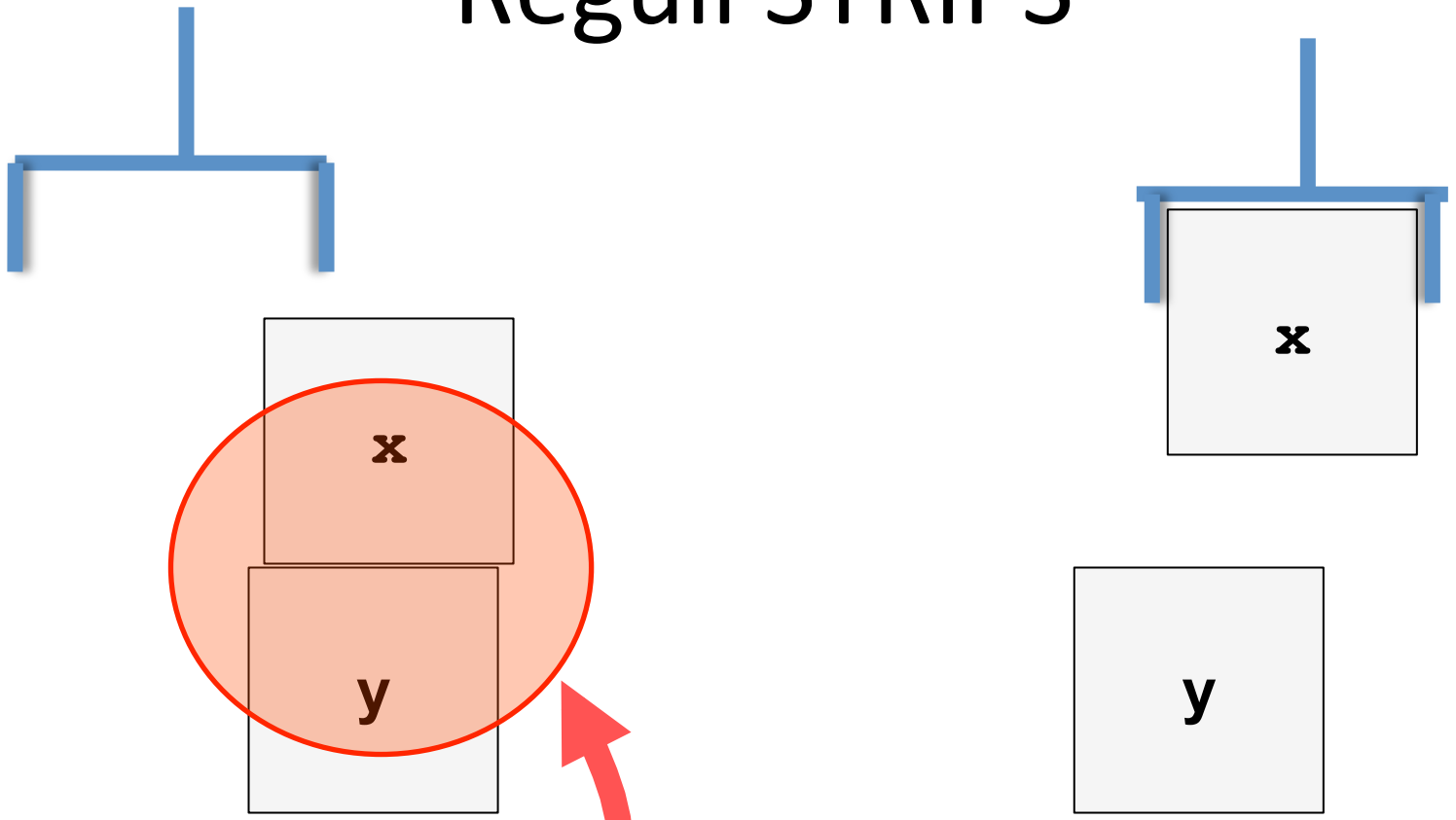

```
putdown(x)
    P&D: holding(x)
    A: ontable(x), clear(x), handempty
```

Reguli STRIPS




```
unstack(x,y)
    P&D: on(x,y), clear(x), handempty
    A: holding(x), clear(y)
```

Reguli STRIPS

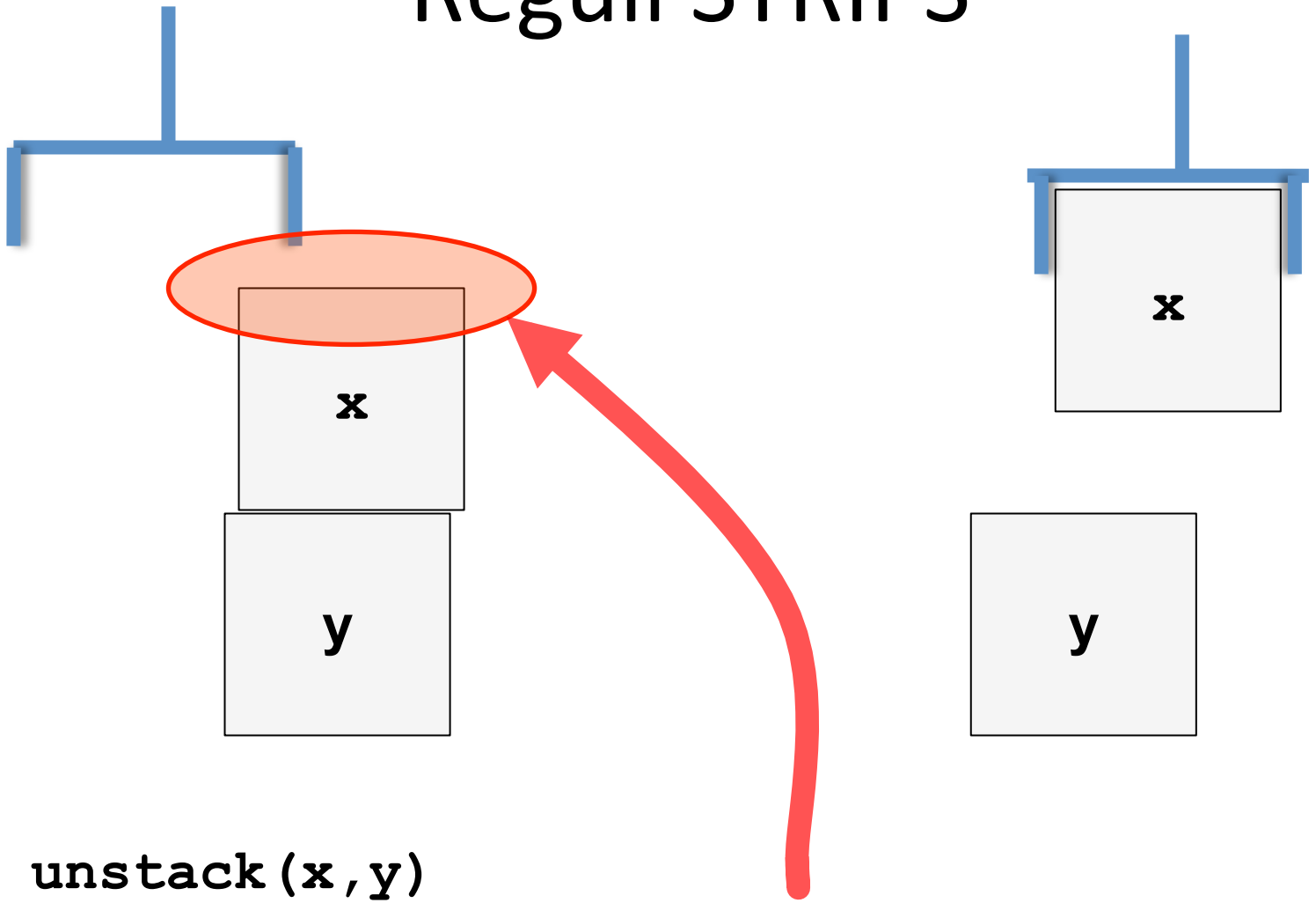


unstack(x,y)

P&D: on(x,y), clear(x), handempty

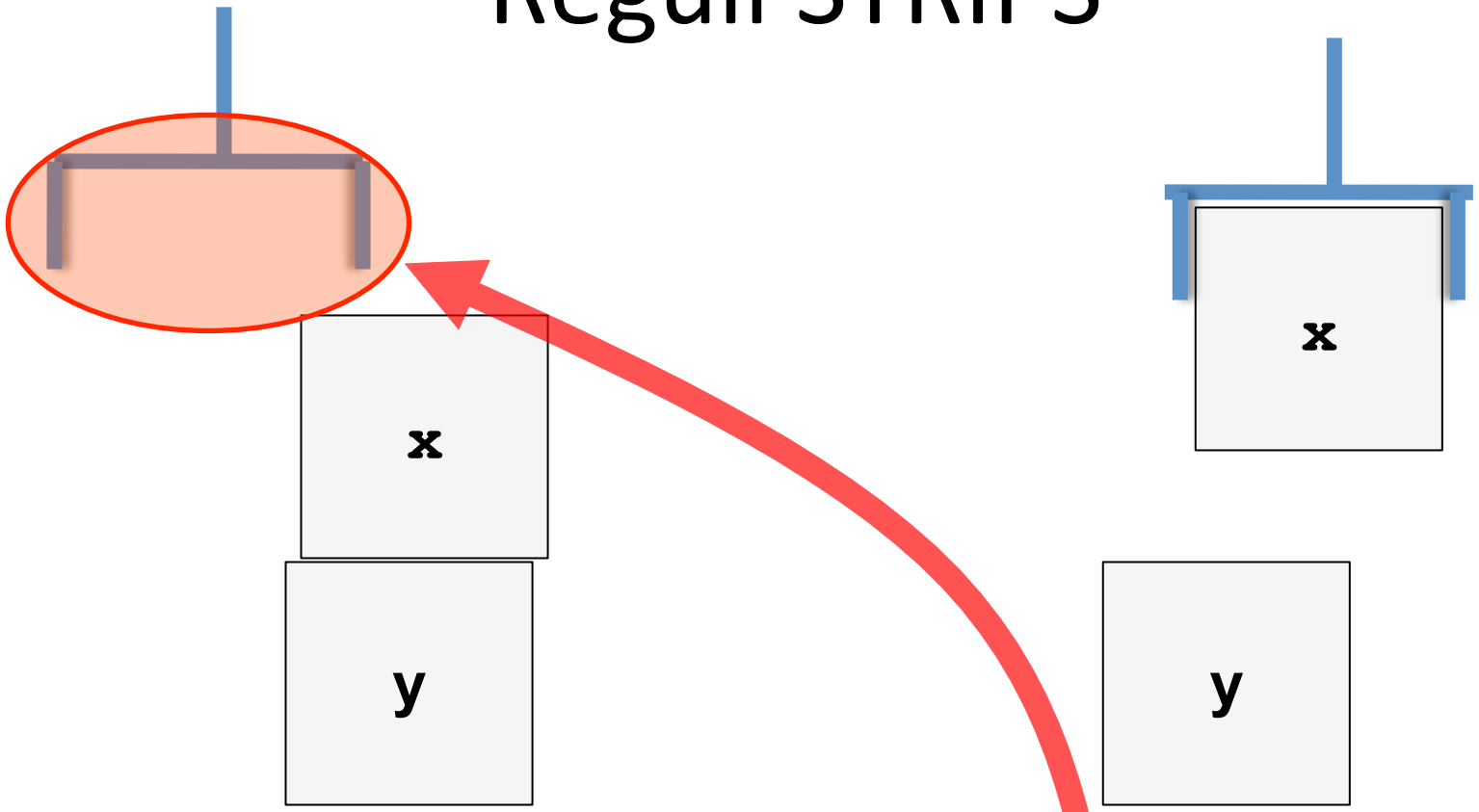
A: holding(x), clear(y)

Reguli STRIPS



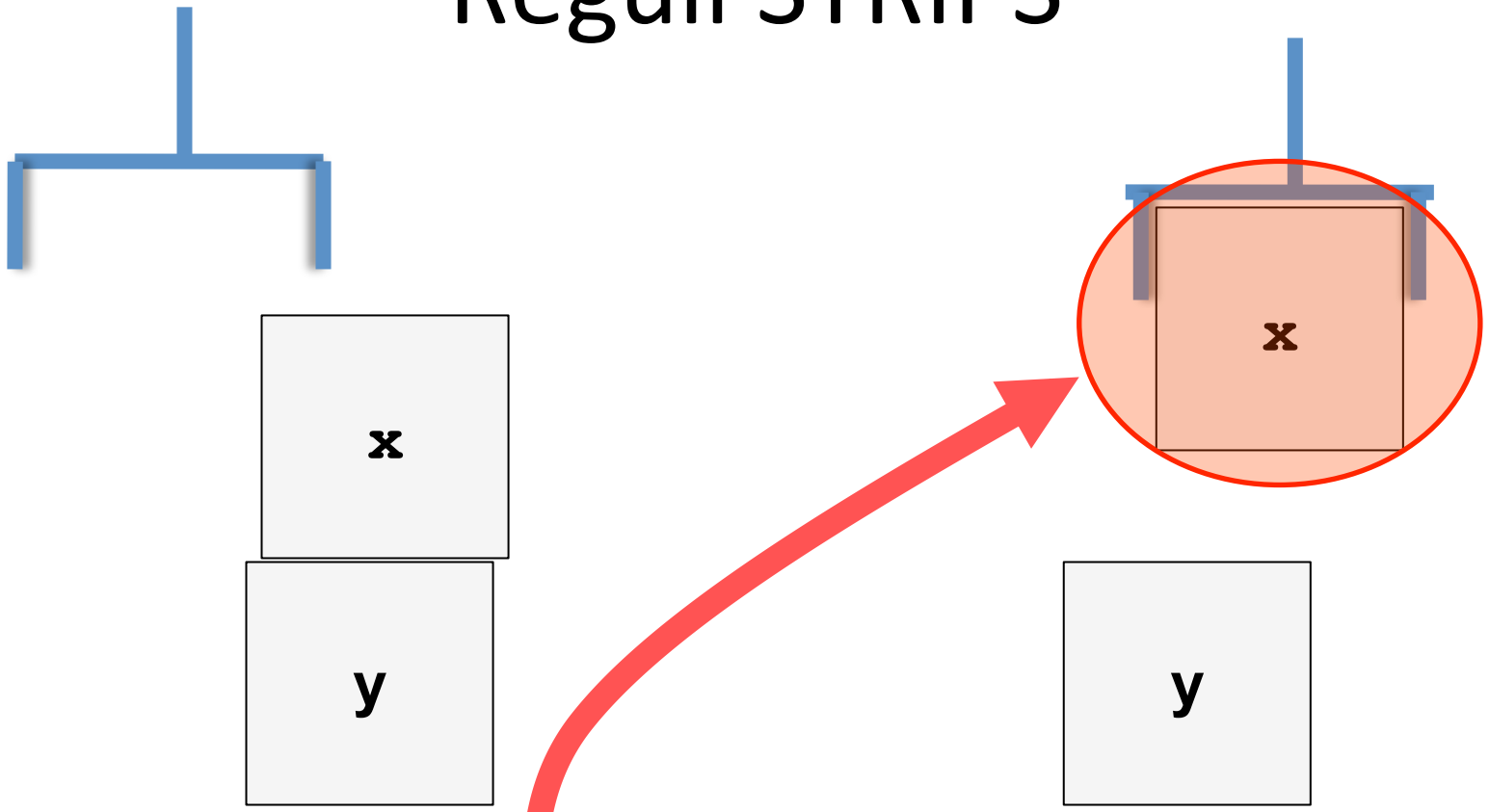

```
unstack(x,y)
    P&D: on(x,y), clear(x), handempty
    A: holding(x), clear(y)
```

Reguli STRIPS




```
unstack(x,y)
    P&D: on(x,y), clear(x), handempty
    A: holding(x), clear(y)
```

Reguli STRIPS

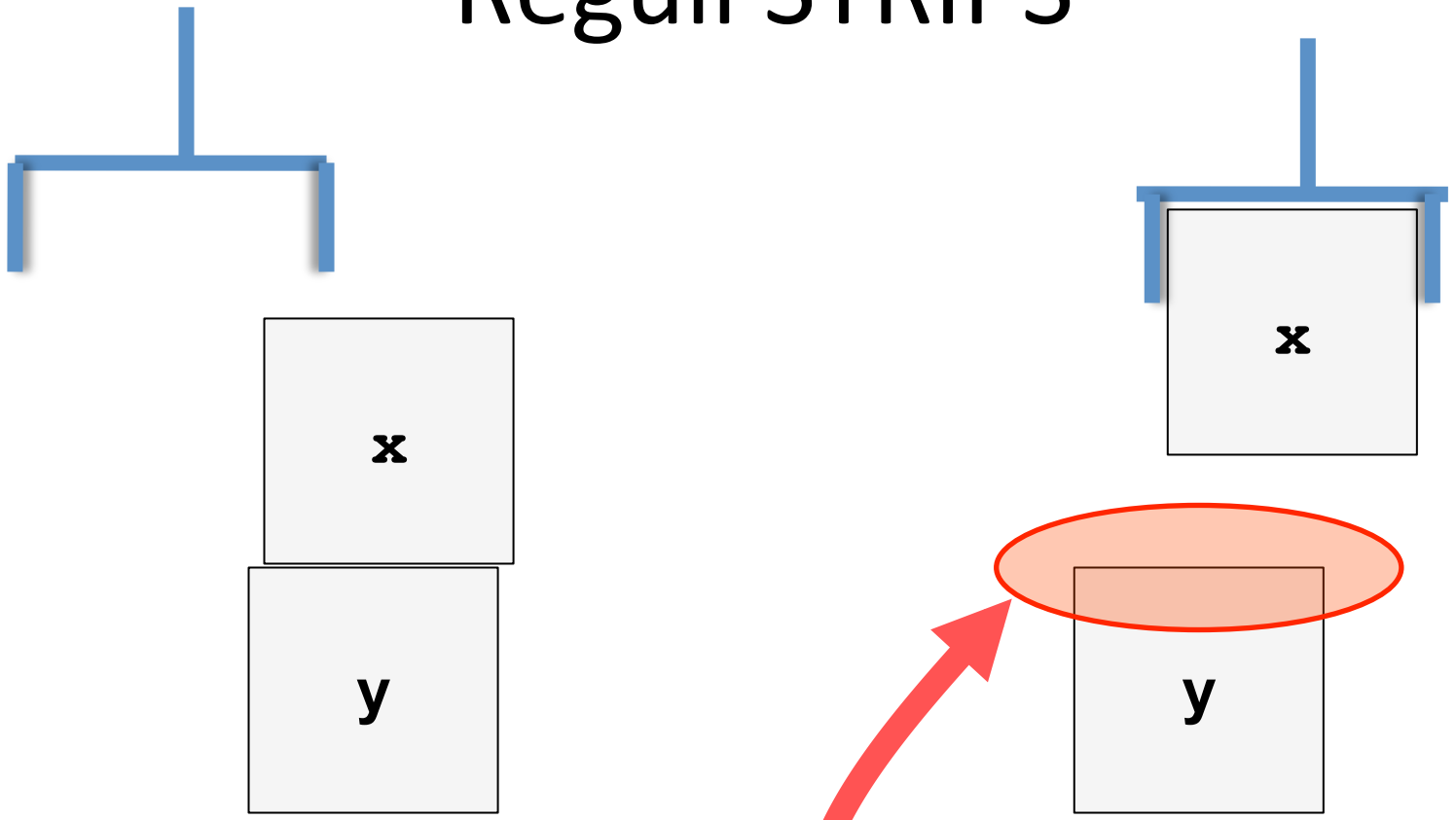


unstack(x,y)

P&D: on(x,y), clear(x), handempty

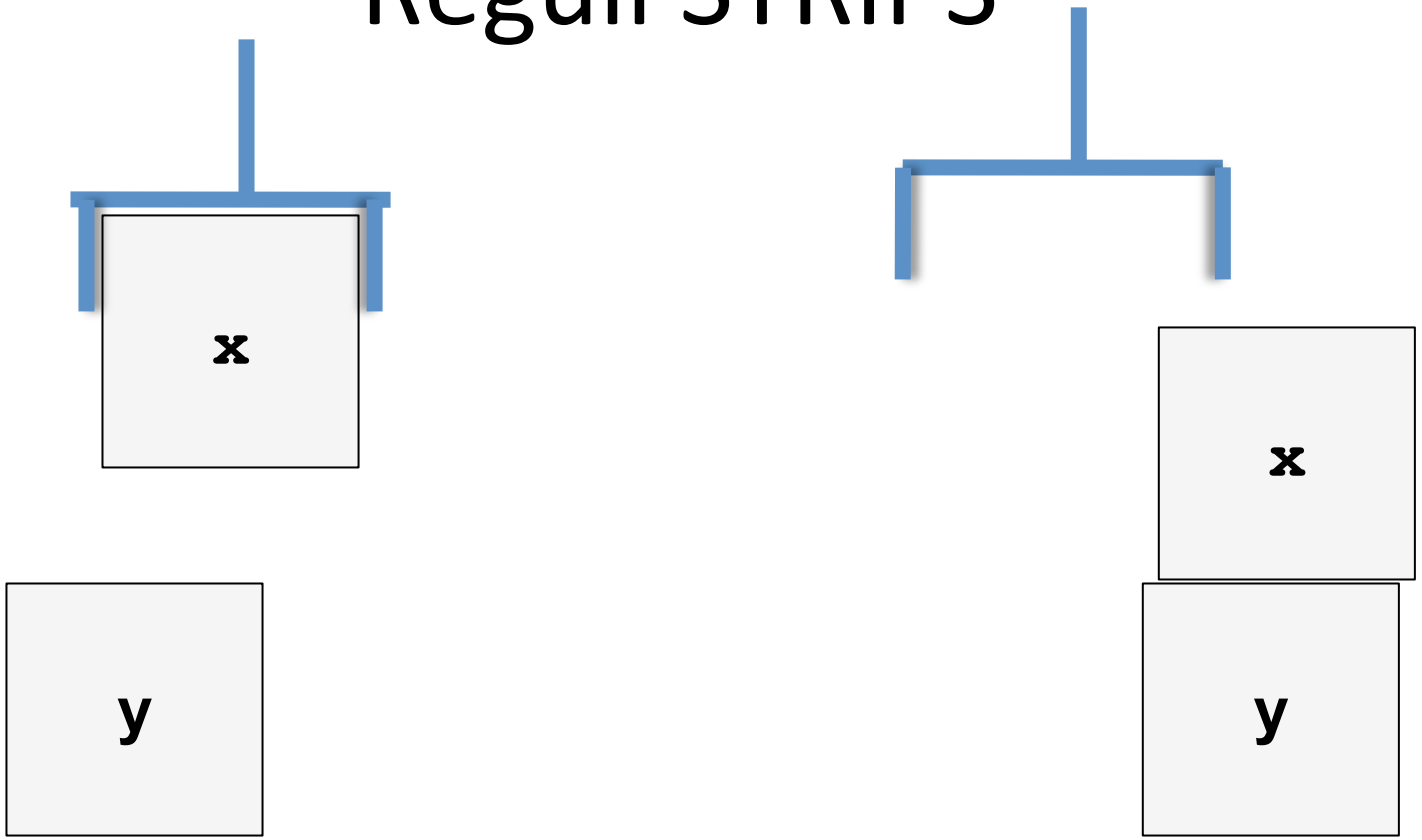
A: holding(x), clear(y)

Reguli STRIPS



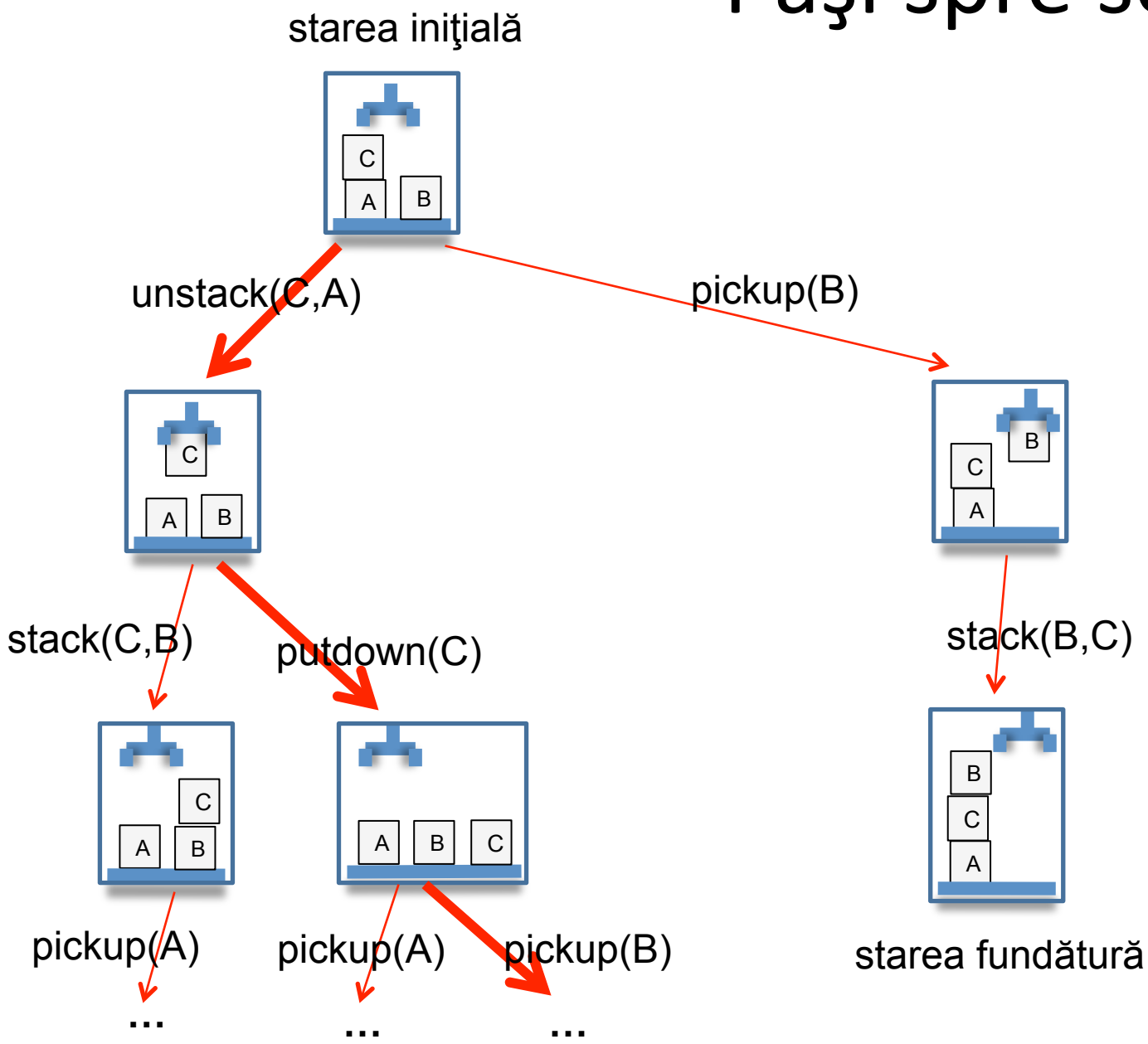

```
unstack(x,y)
    P&D: on(x,y), clear(x), handempty
    A: holding(x), clear(y)
```

Reguli STRIPS

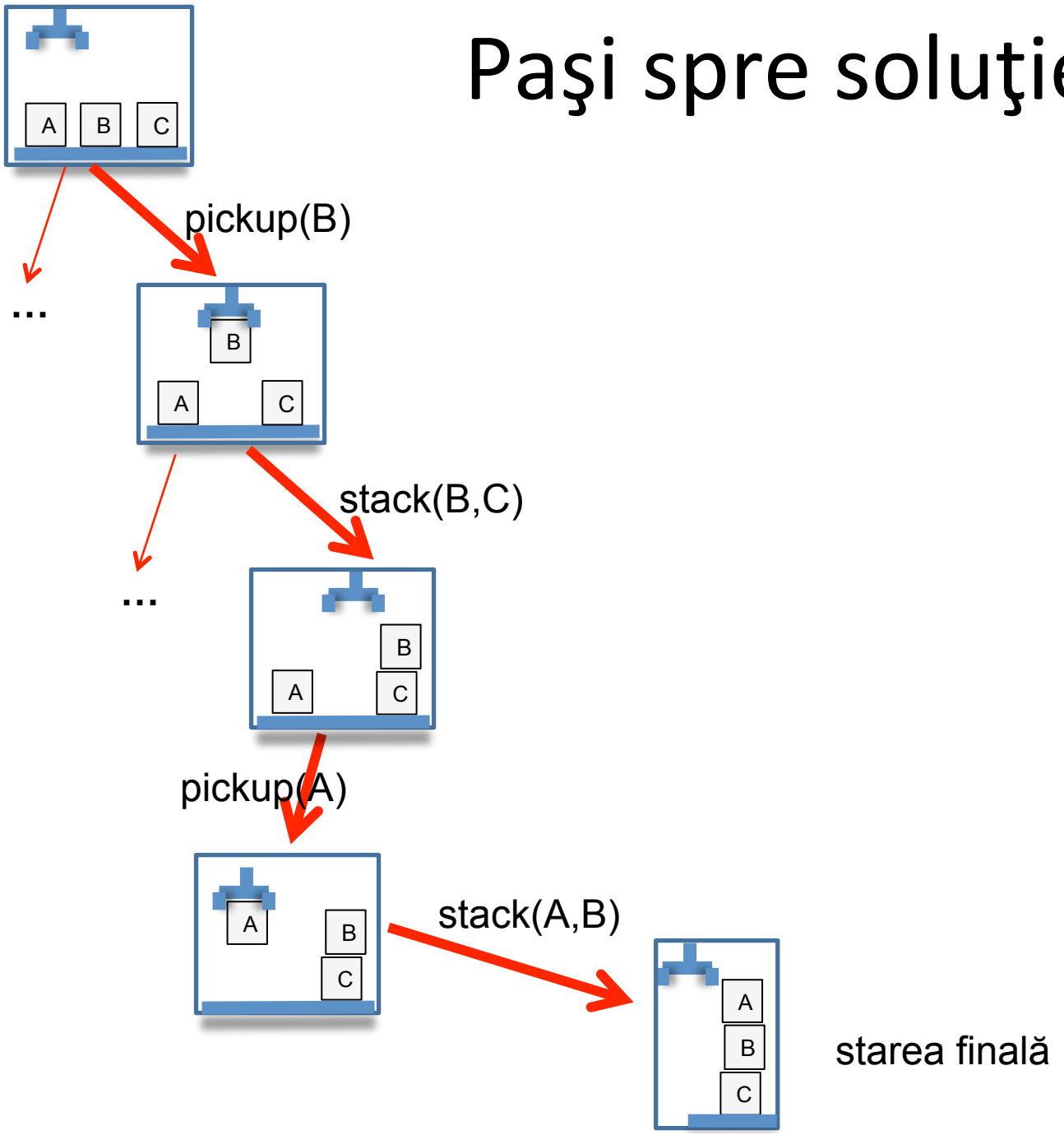



```
stack(x,y)
    P&D: holding(x), clear(y)
    A: on(x,y), handempty, clear(x)
```

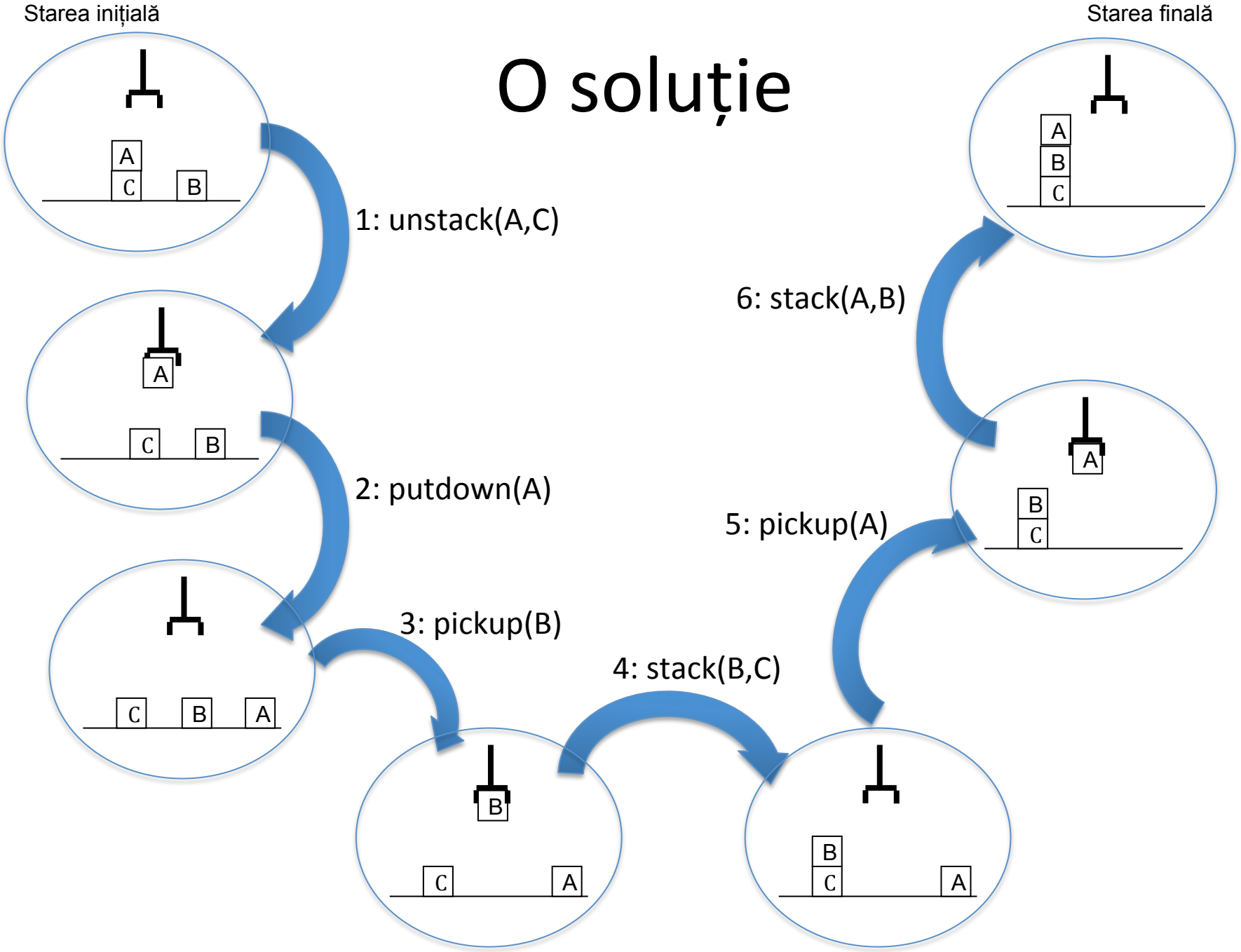
Paşi spre soluţie



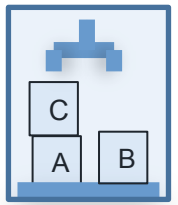
Paşi spre soluţie



O soluție


Starea inițială
Starea finală
1: unstack(A,C)
2: putdown(A)
3: pickup(B)
4: stack(B,C)
5: pickup(A)
6: stack(A,B)
A
C
B

Soluţia



starea iniţială

unstack(C,A)

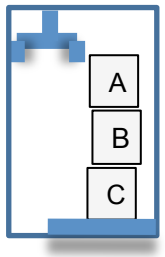
putdown(C)

pickup(B)

stack(B,C)

pickup(A)

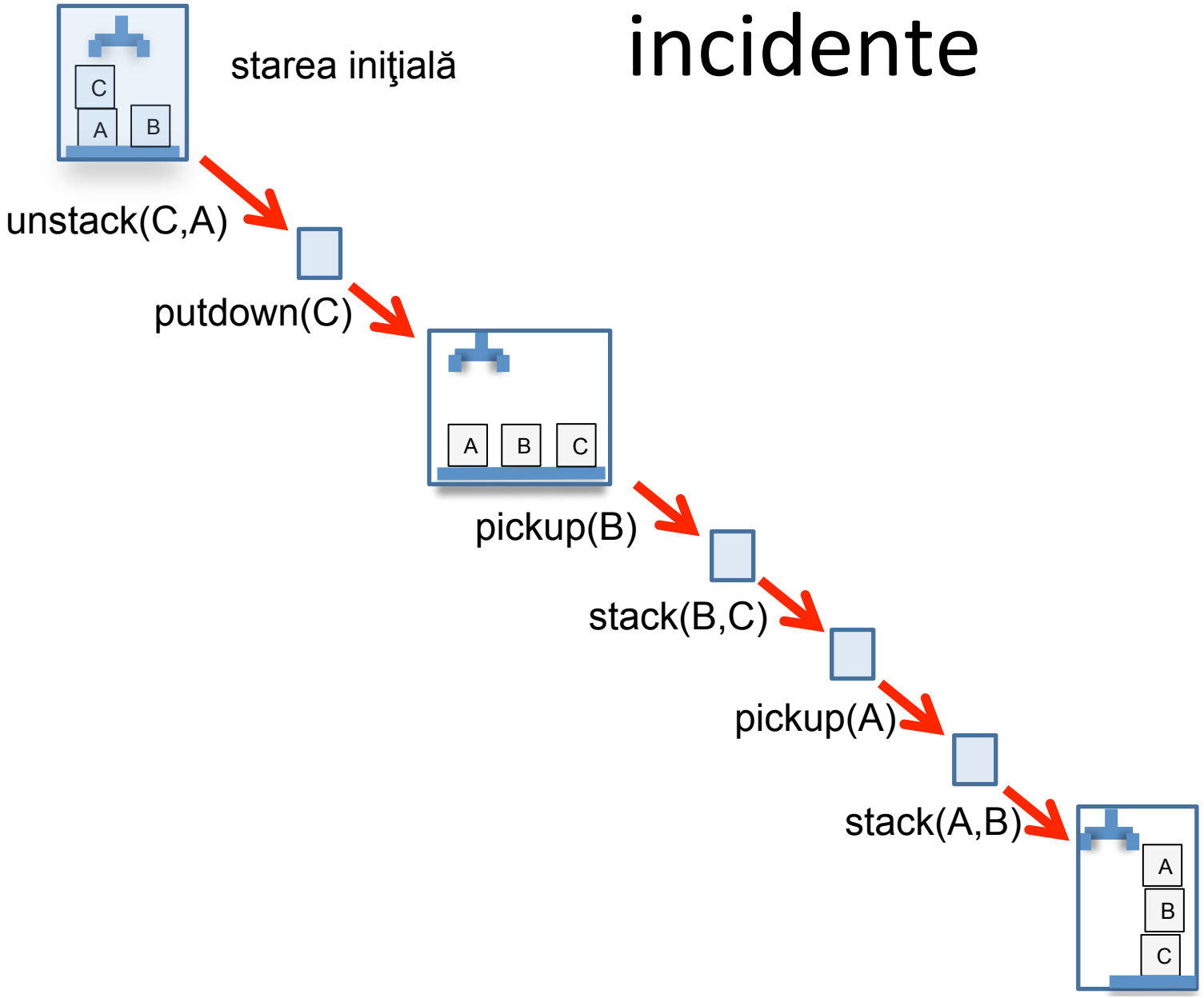
stack(A,B)



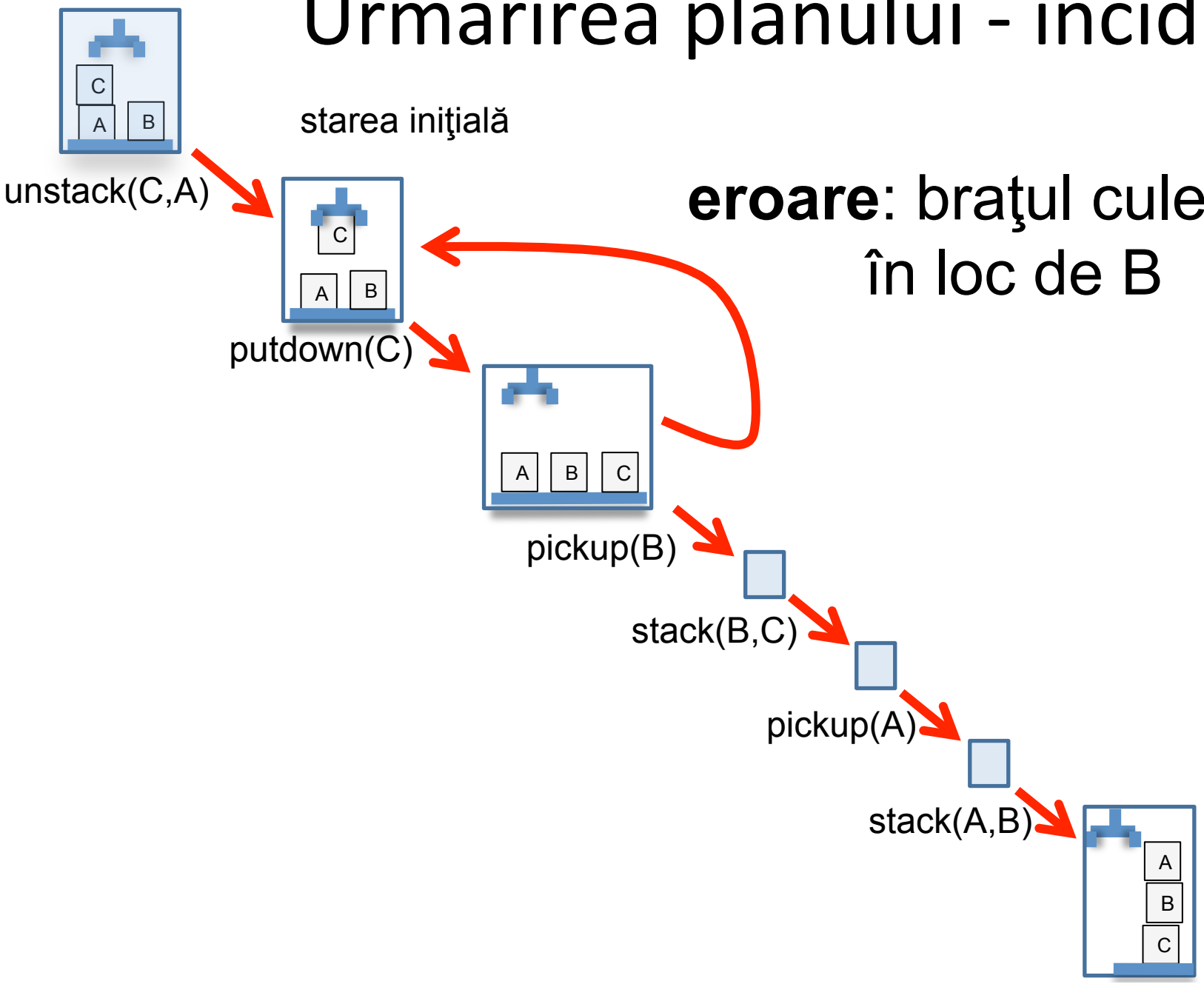
starea finală

$N = 6$

Urmărirea execuţiei planului - incidente

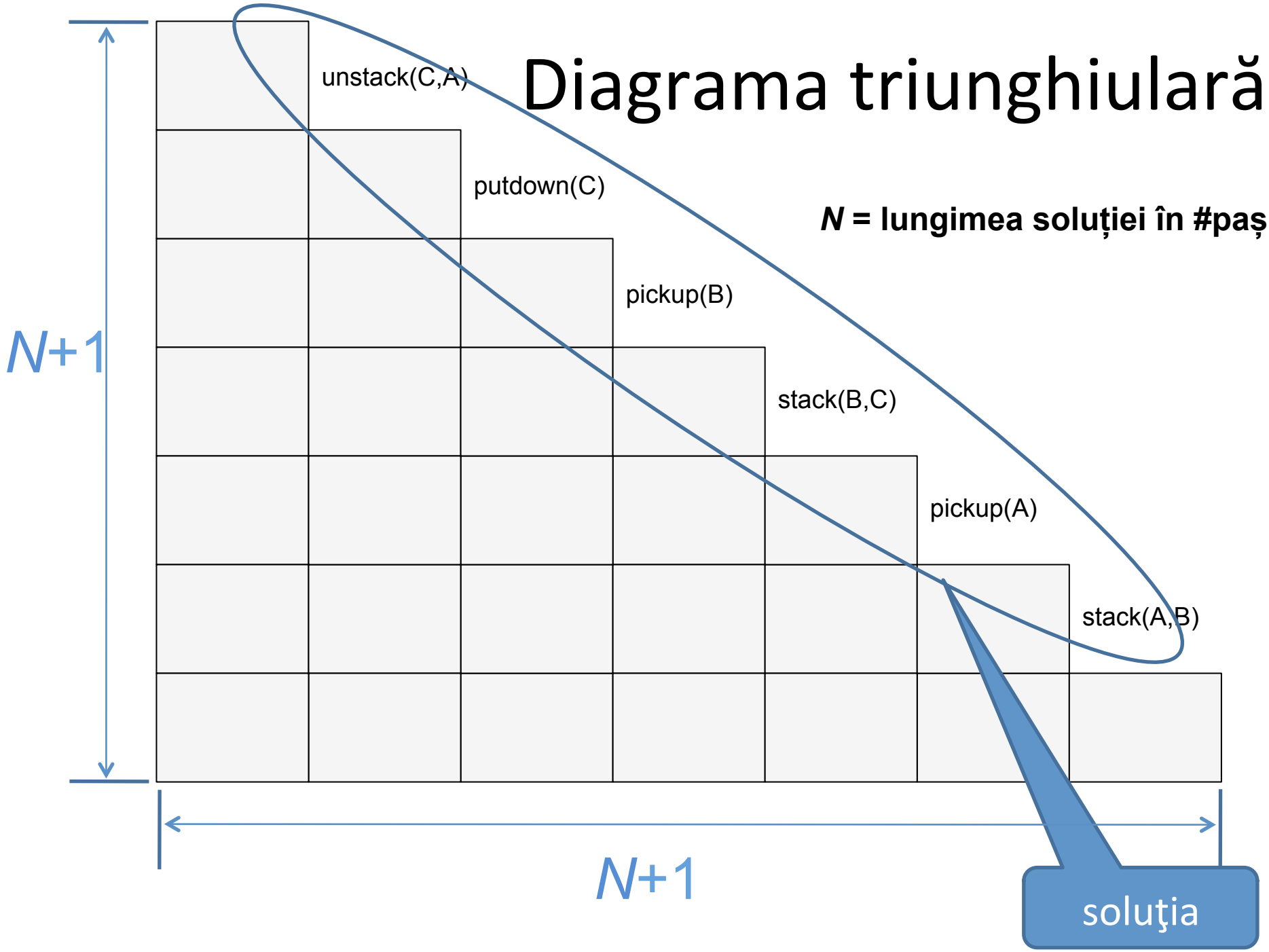


Urmărirea planului - incidente



eroare: braţul culege C în loc de B

Diagrama triunghiulară



***N* = lungimea soluției în #pași**

Diagrama triunghiulară

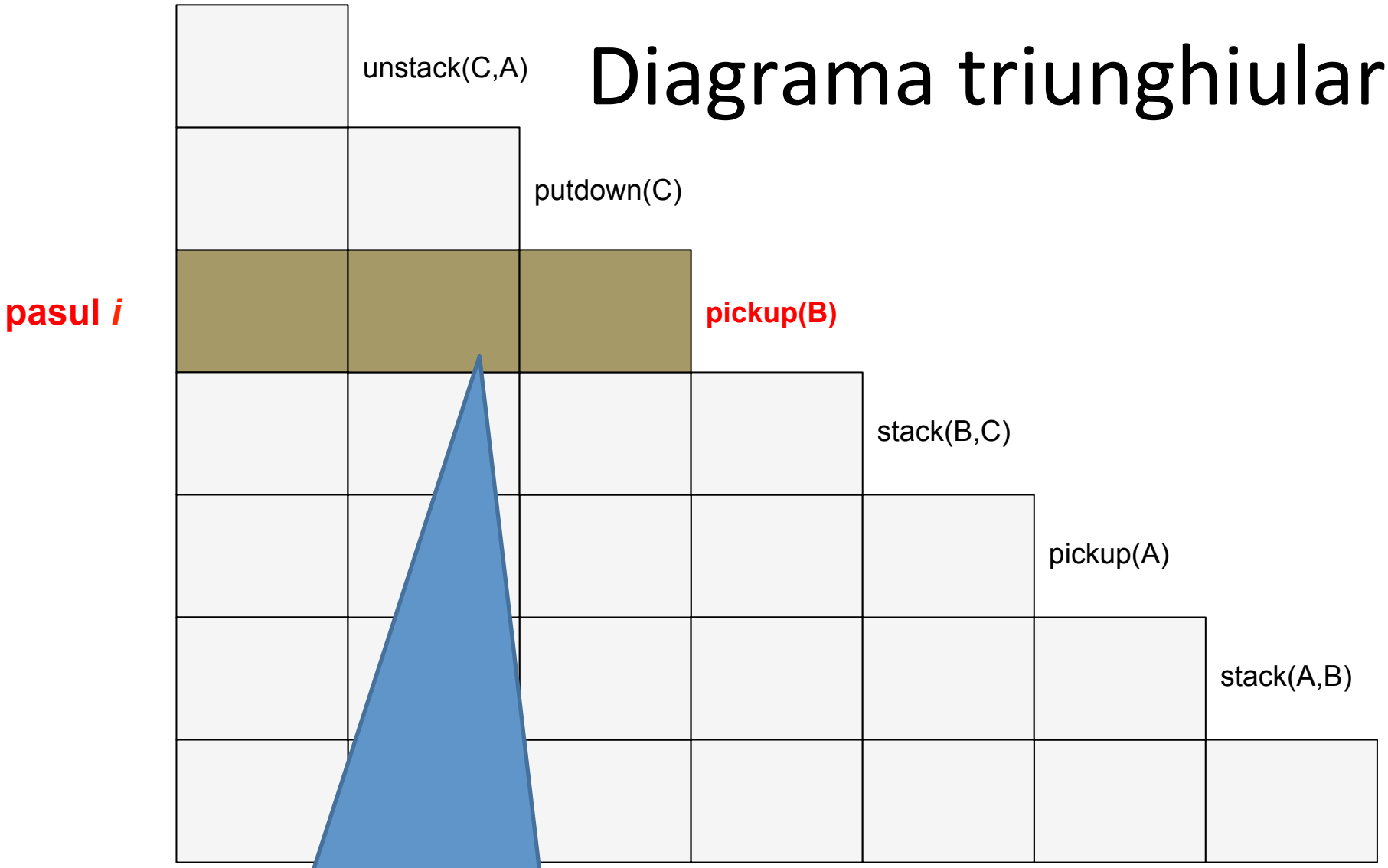
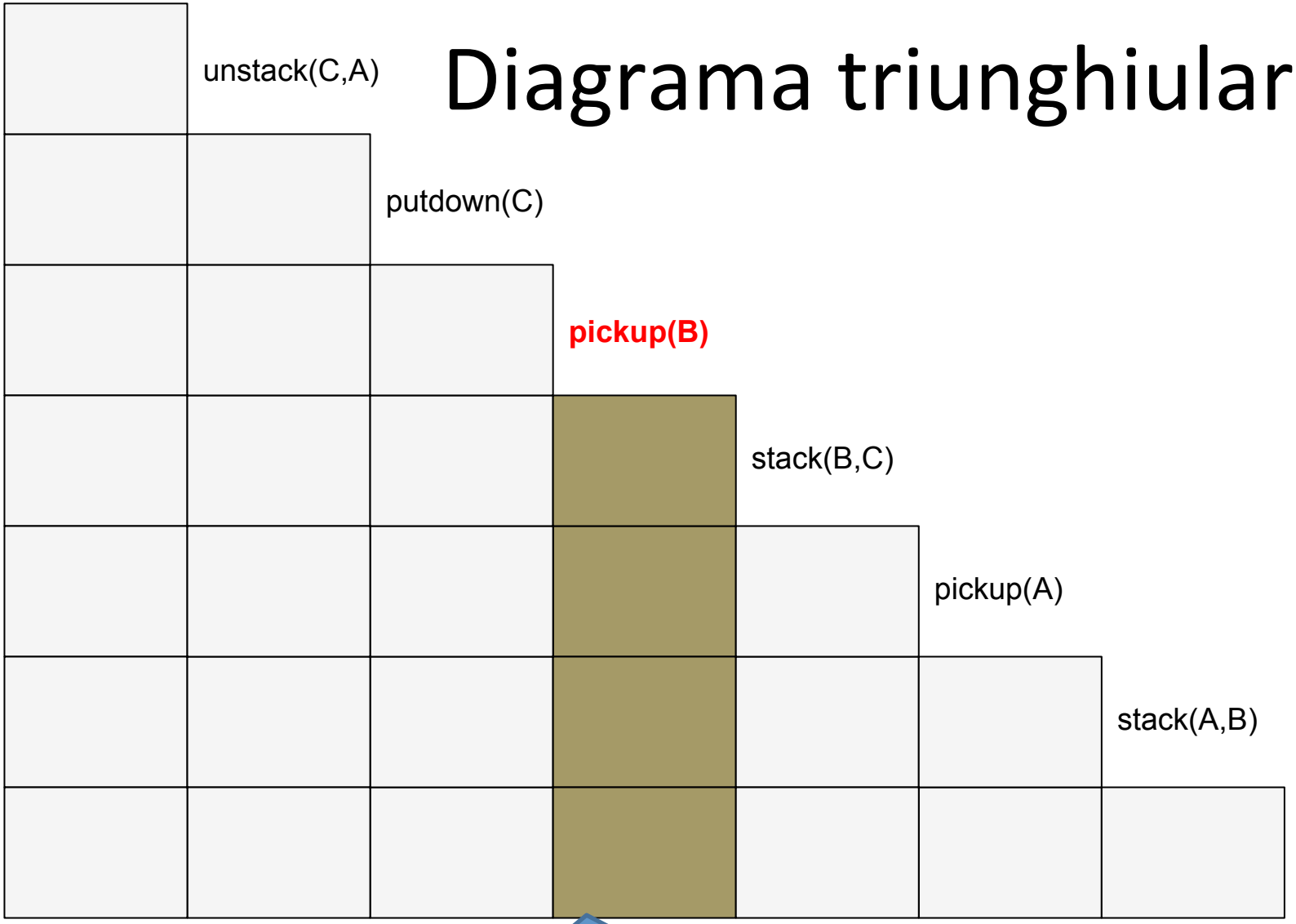


Diagrama triunghiulară



aceste casete concentrează toate predicatele din lista A a regulei pasului *i*

Diagrama triunghiulară

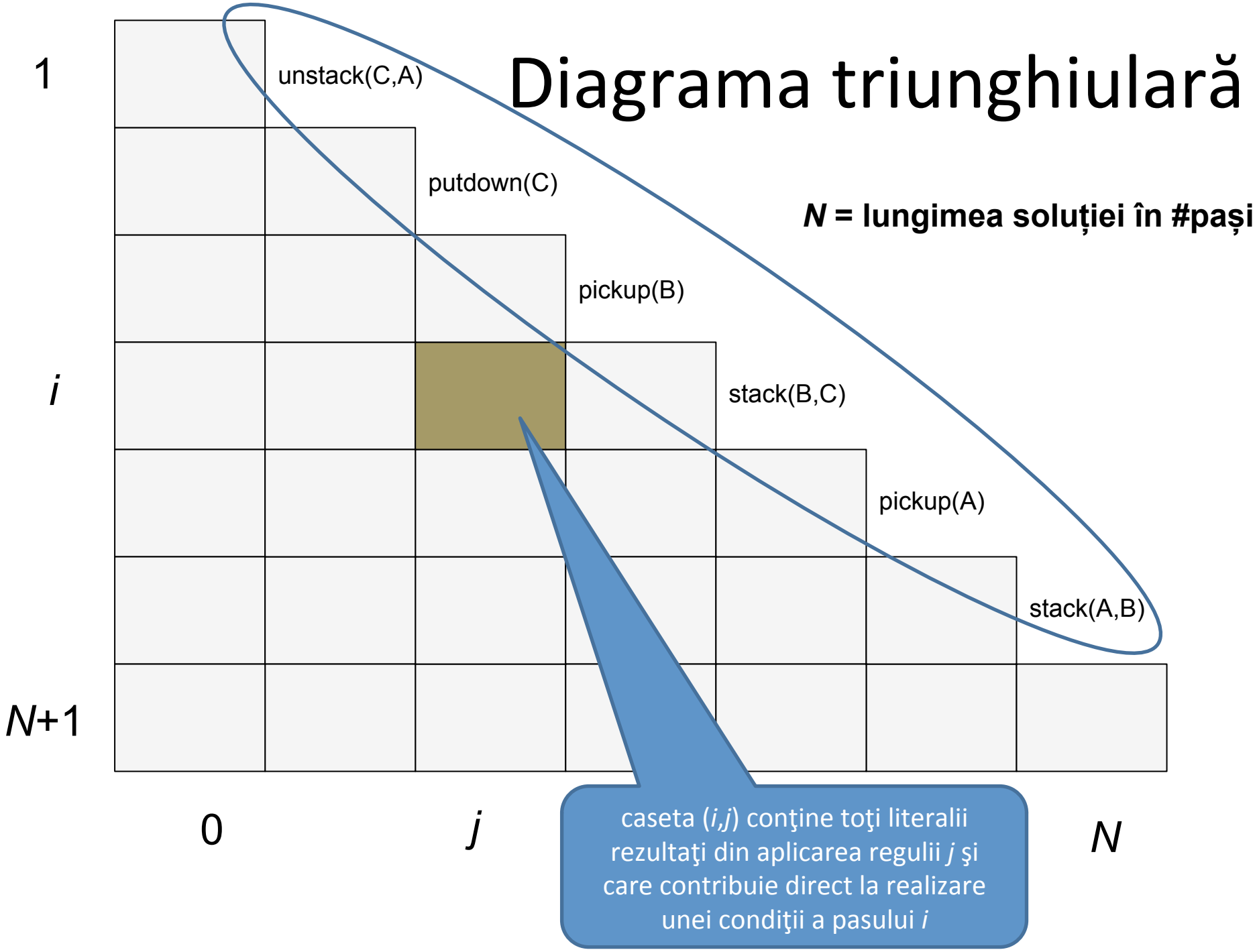


Diagrama triunghiulară

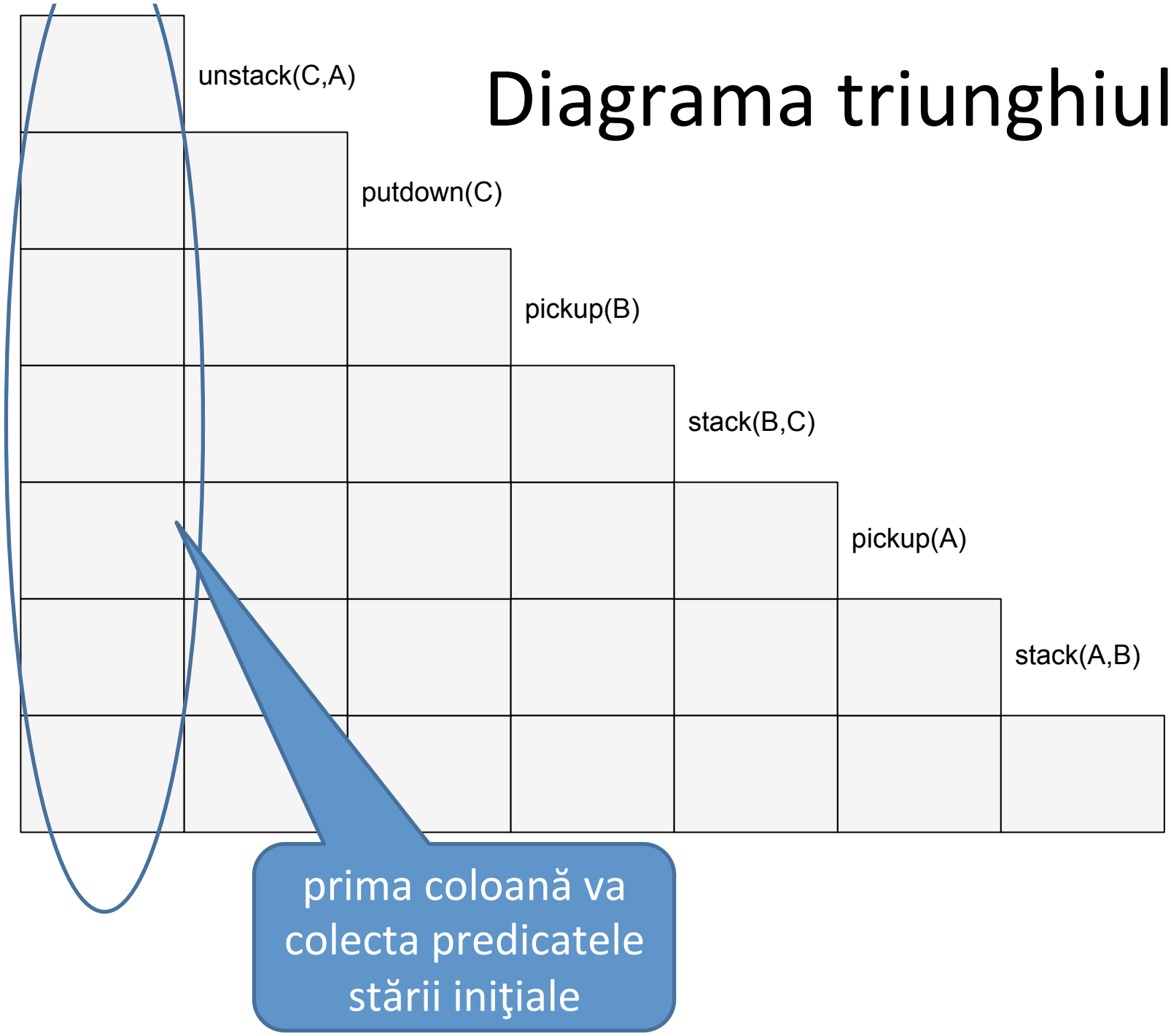


Diagrama triunghiulară

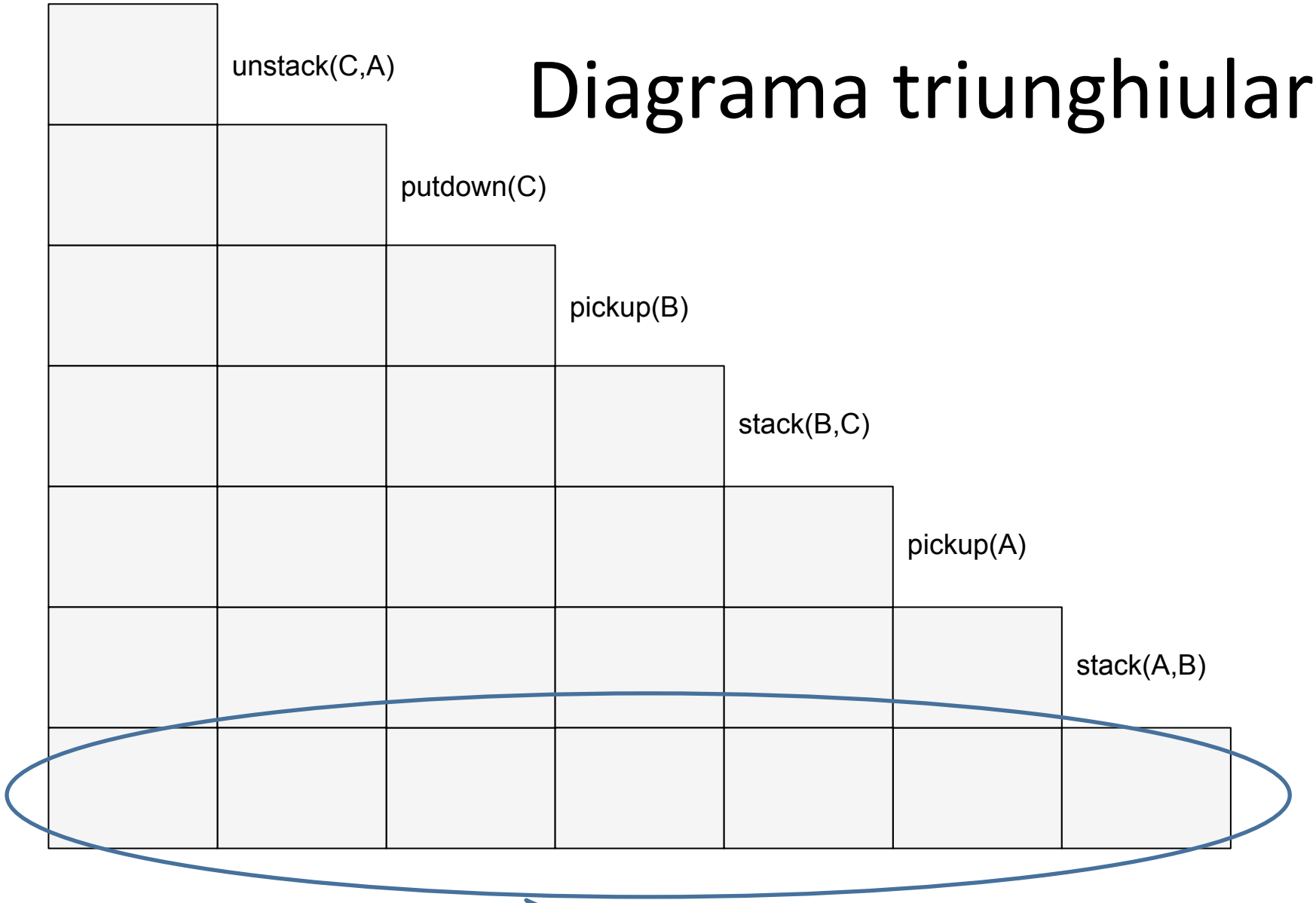


Diagrama triunghiulară: kernel de ordin *i*

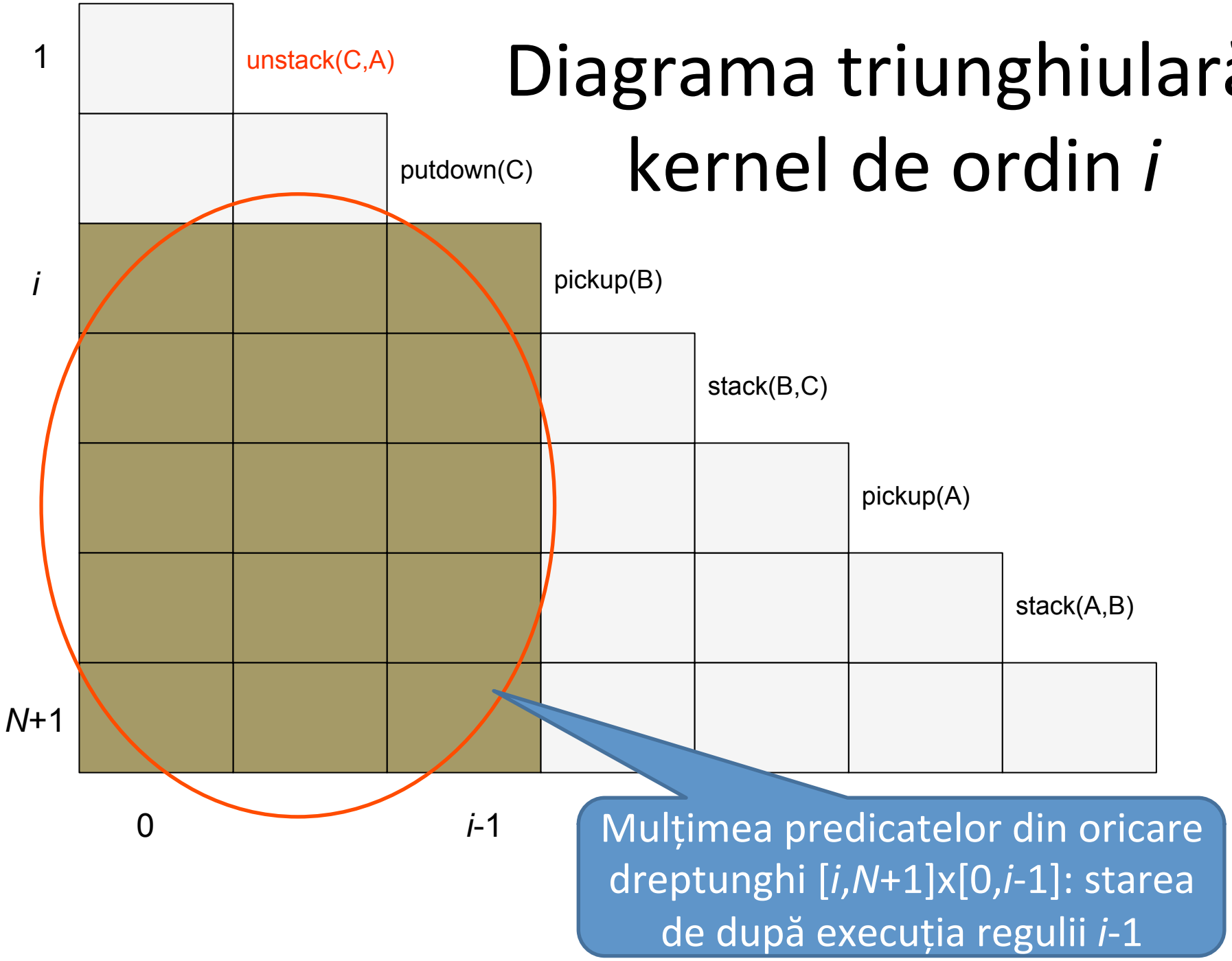


Diagrama triunghiulară: kernel

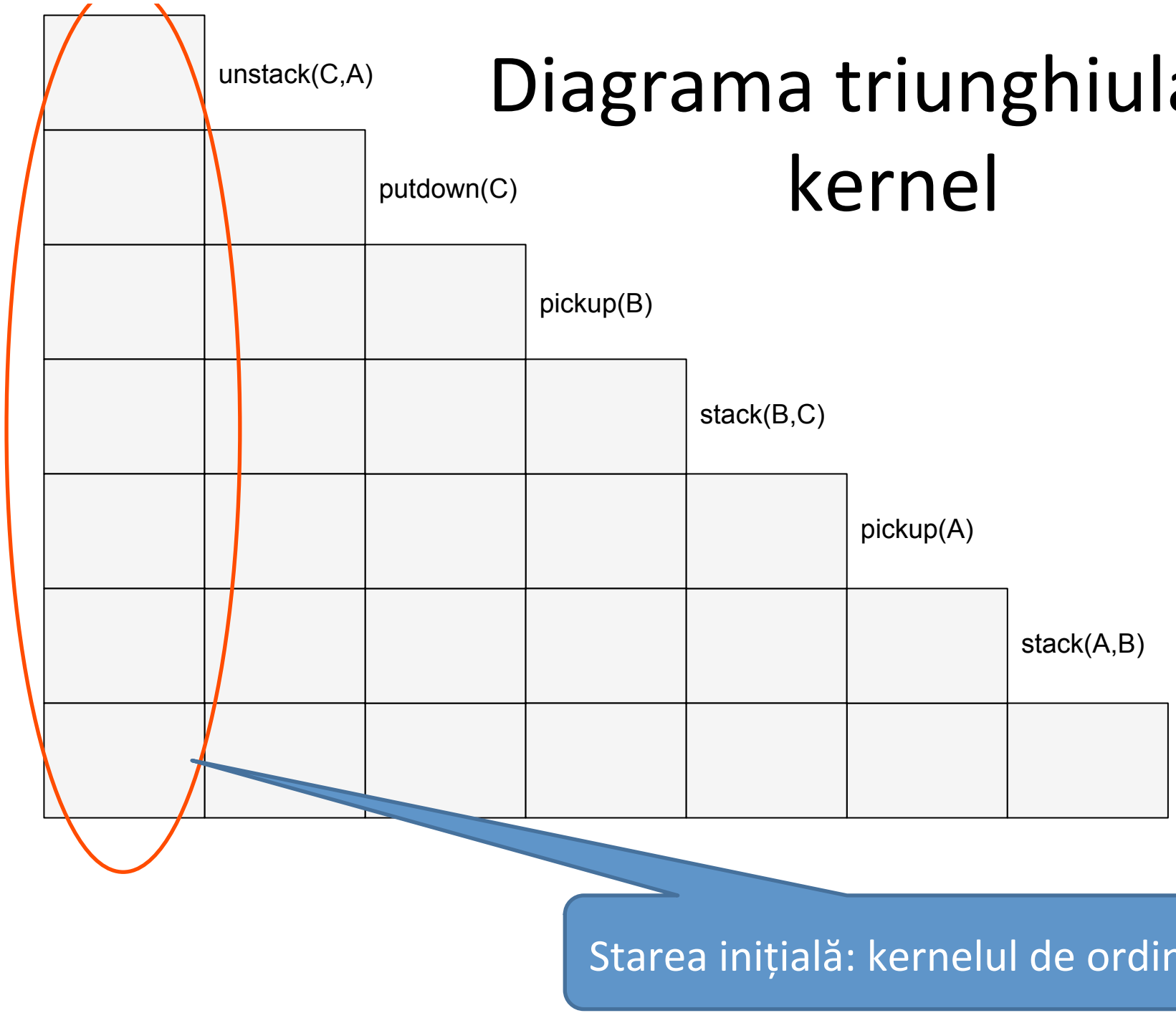


Diagrama triunghiulară: kernel

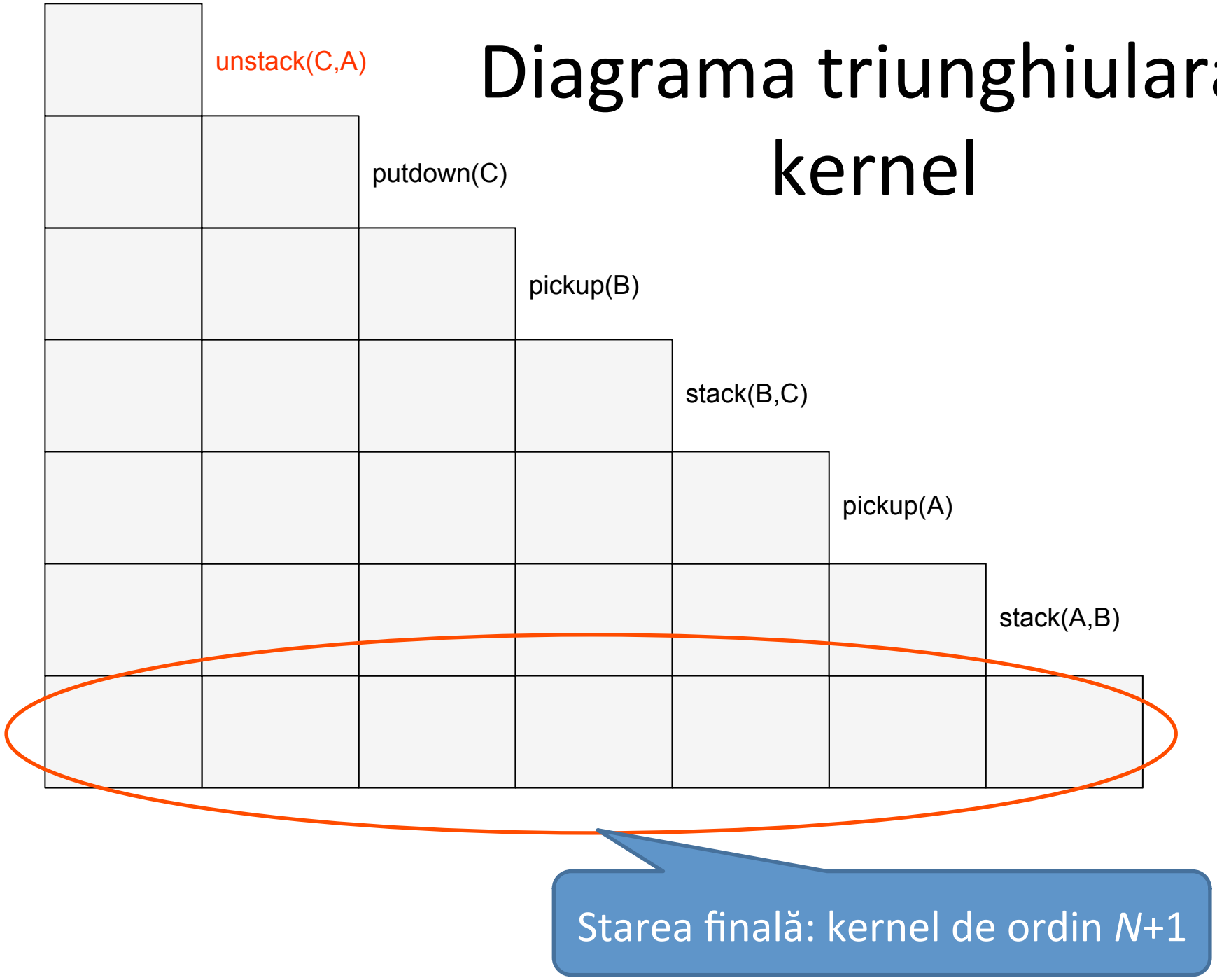
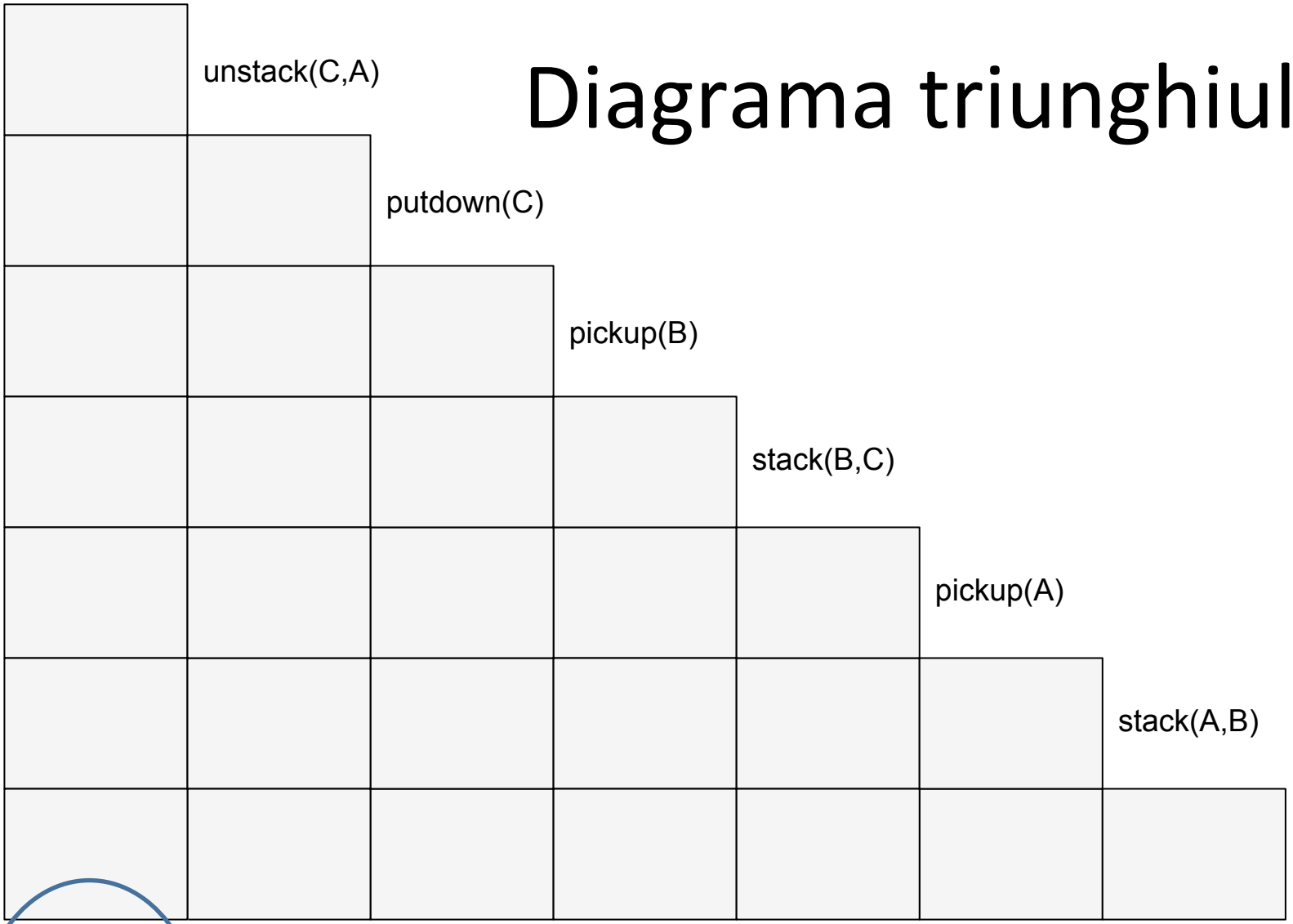
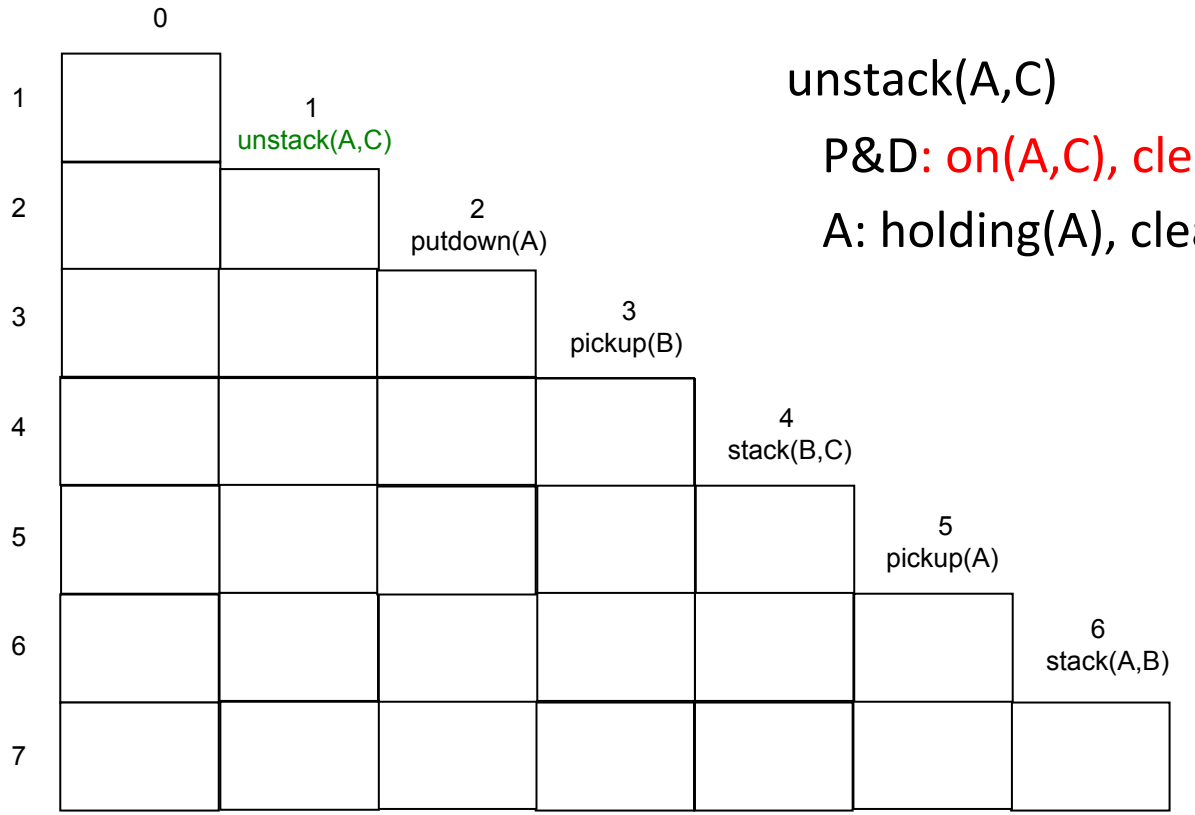


Diagrama triunghiulară



Execuția planului: pasul 1



unstack(A,C)

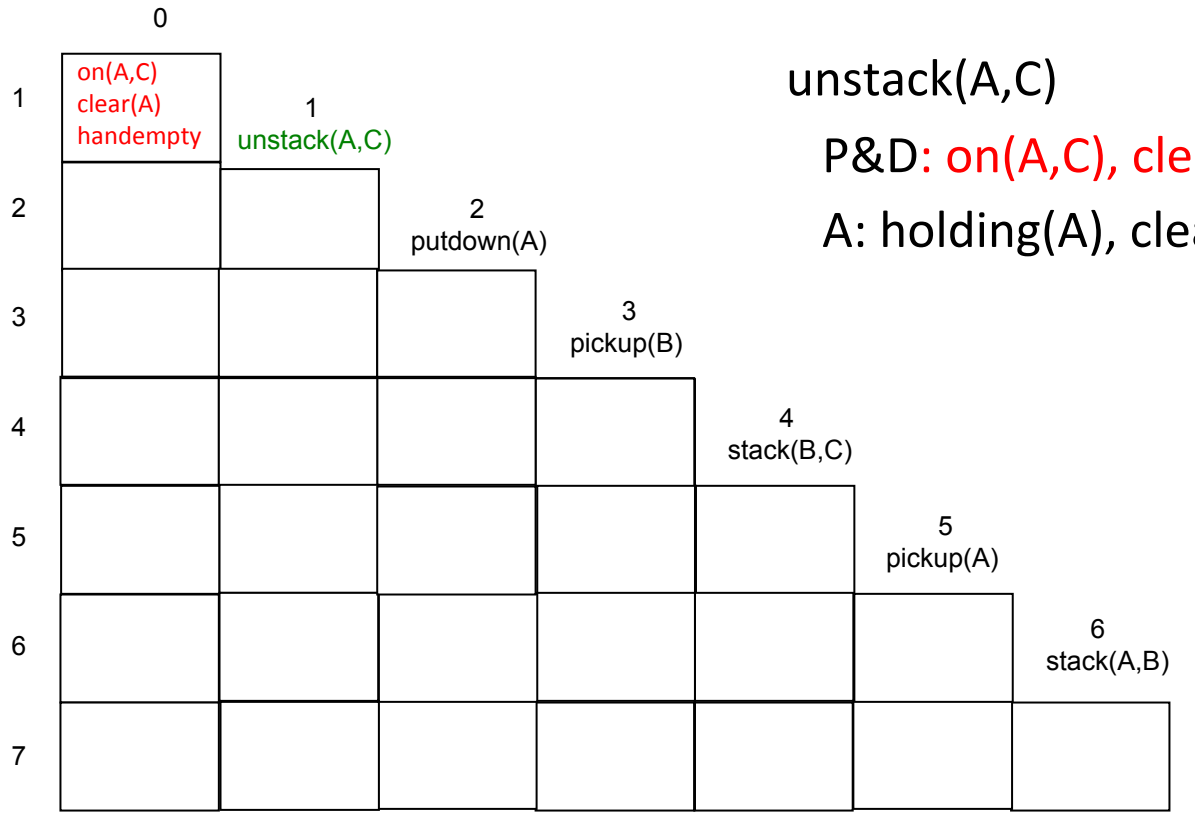
P&D: on(A,C), clear(A), handempty

A: holding(A), clear(C)

ontable(C),
ontable(B),
on(A,C),
clear(A)
clear(B)
handempty

dintre predicatele stării inițiale
caută-le pe cele necesare listei
P&D a regulii 1

Execuția planului: pasul 1



unstack(A,C)

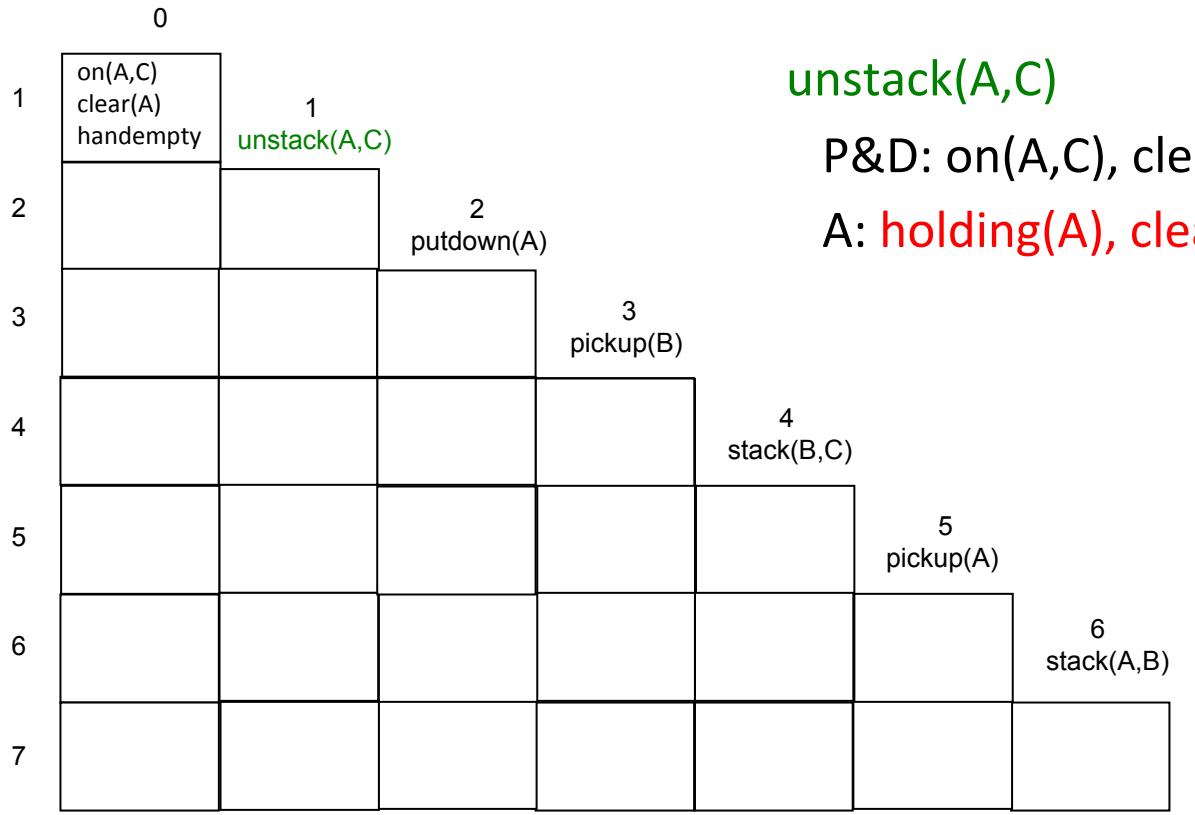
P&D: on(A,C), clear(A), handempty

A: holding(A), clear(C)

ontable(C)
ontable(B)
clear(B)

… și urcă-le în linia regulii 1

Execuția planului: pasul 1



unstack(A,C)

P&D: on(A,C), clear(A), handempty

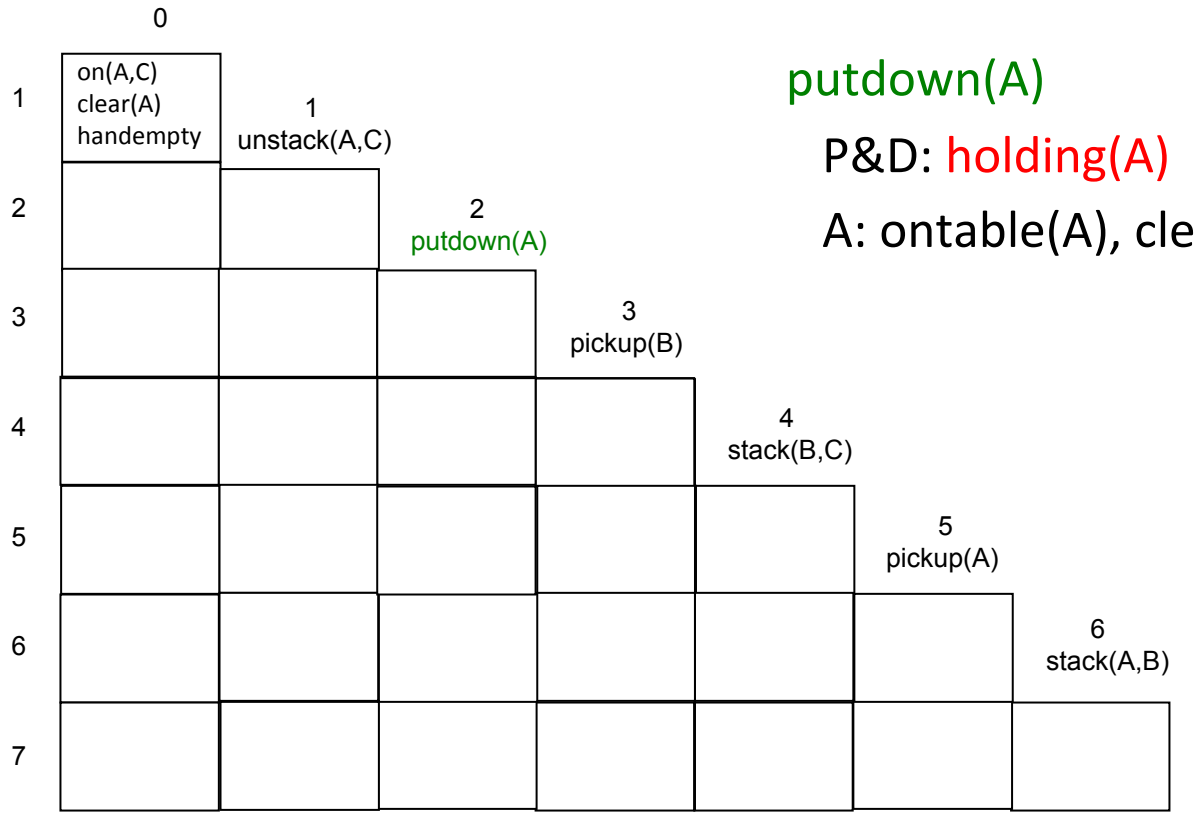
A: holding(A), clear(C)

ontable(C)
ontable(B)
clear(B)

holding(A)
clear(C)

colectează predicatele listei A
a regulii 1

Execuția planului: pasul 2



putdown(A)

P&D: holding(A)

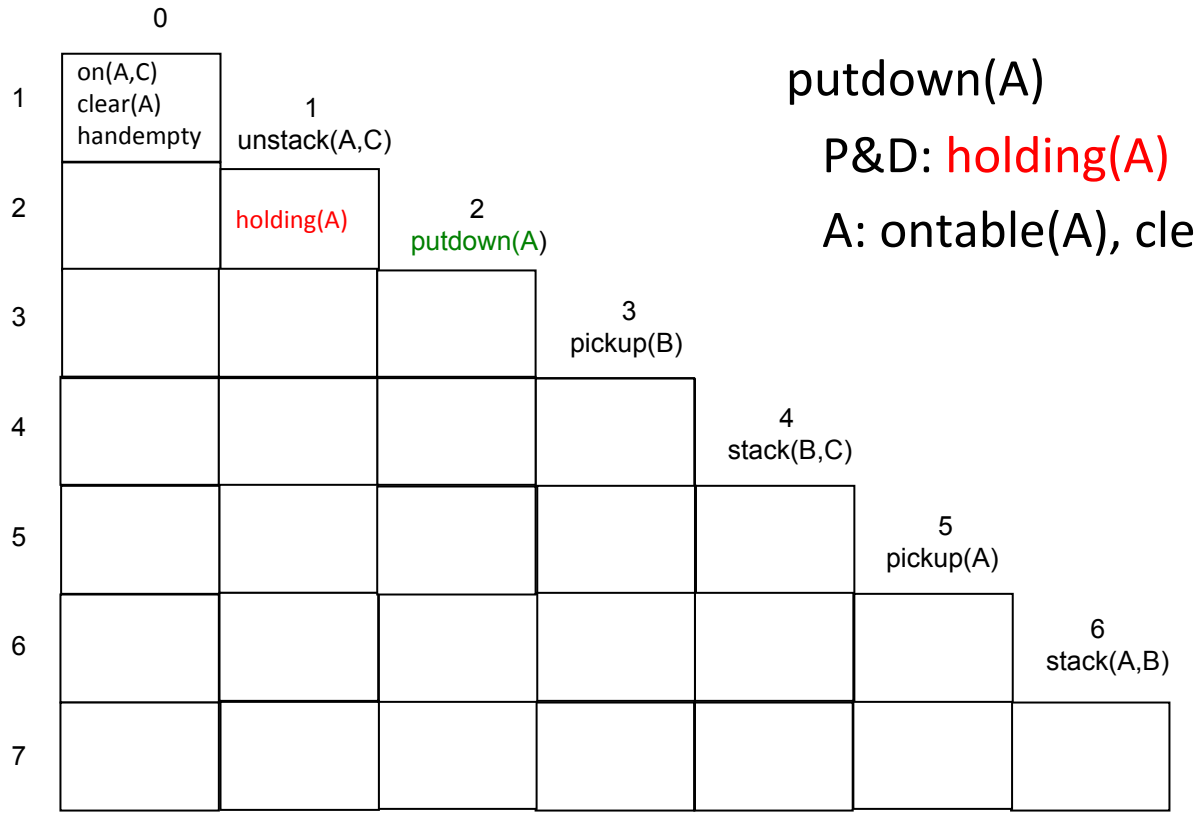
A: ontable(A), clear(A), handempty

ontable(C)
ontable(B)
clear(B)

holding(A)
clear(C)

caută predicatele listei P&D
a regulii 2

Execuția planului: pasul 2



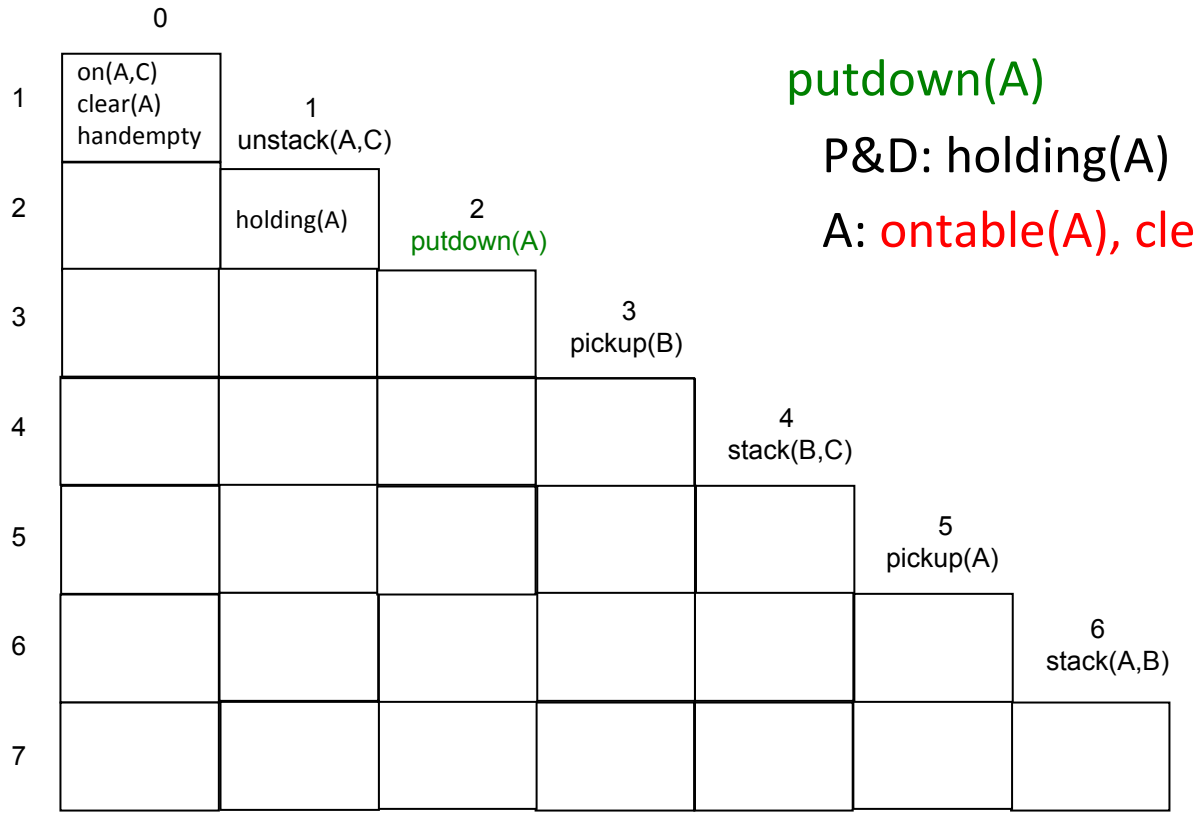
putdown(A)

P&D: holding(A)

A: ontable(A), clear(A), handempty

… și urcă-le în linia regulii 2

Execuția planului: pasul 2

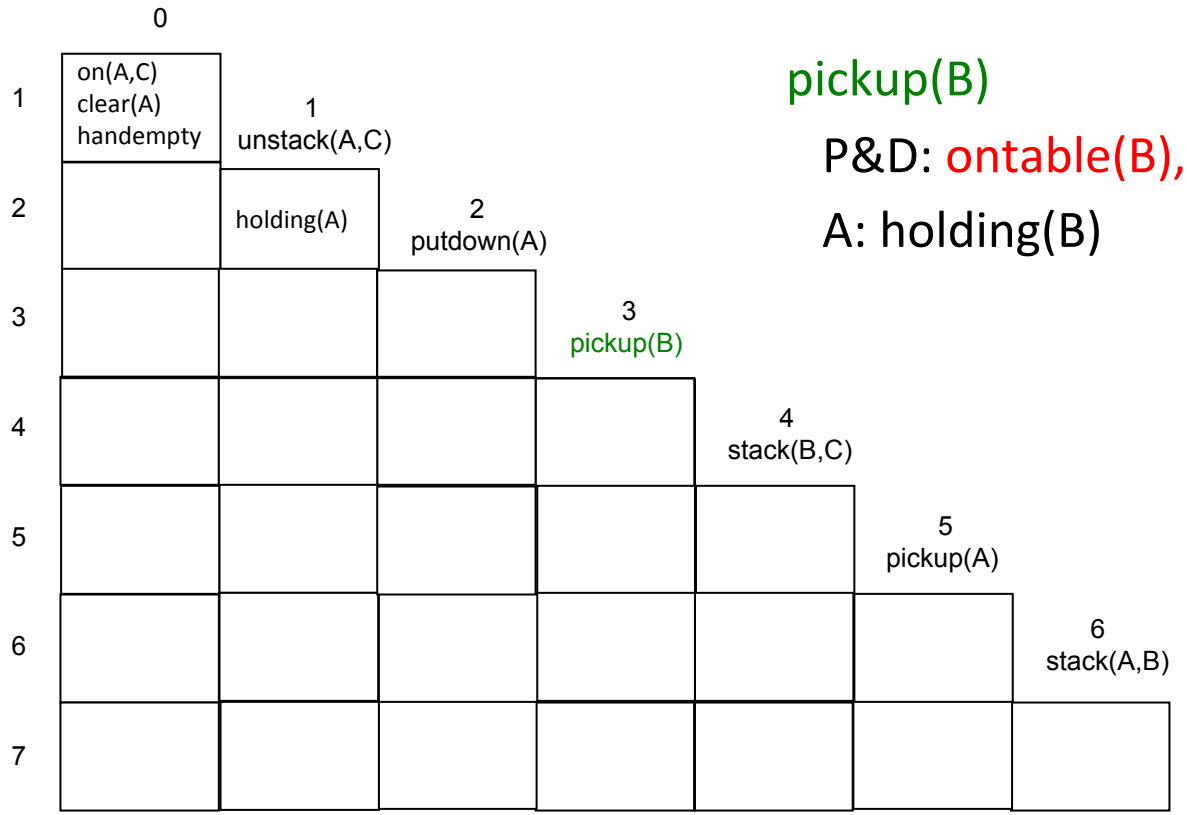


putdown(A)

P&D: holding(A)

A: ontable(A), clear(A), handempty

Execuția planului: pasul 3



pickup(B)

P&D: ontable(B), clear(B), handempty

A: holding(B)

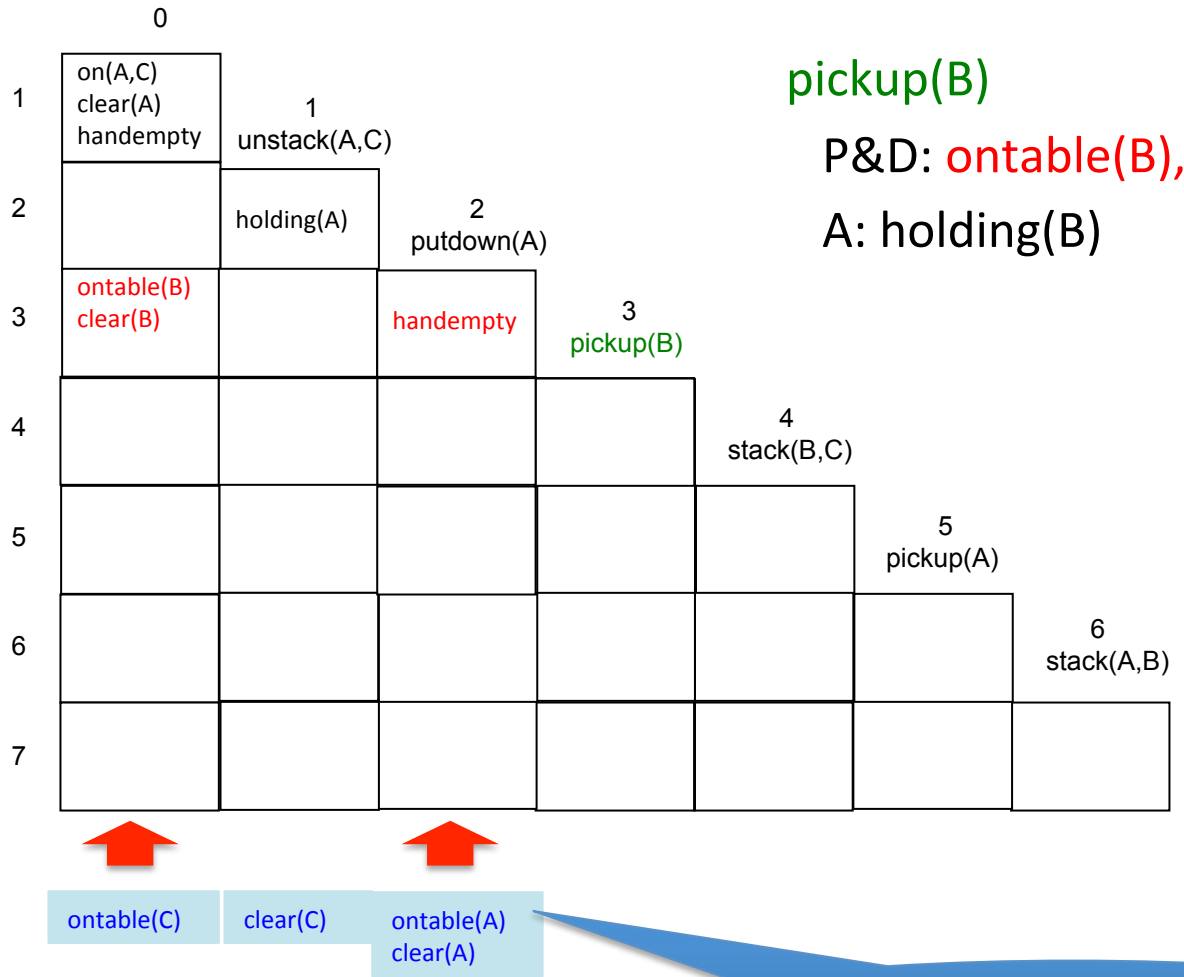
ontable(C)
ontable(B)
clear(B)

clear(C)

ontable(A)
clear(A)
handempty

caută predicatele listei P&D
a regulii 3

Execuția planului: pasul 3



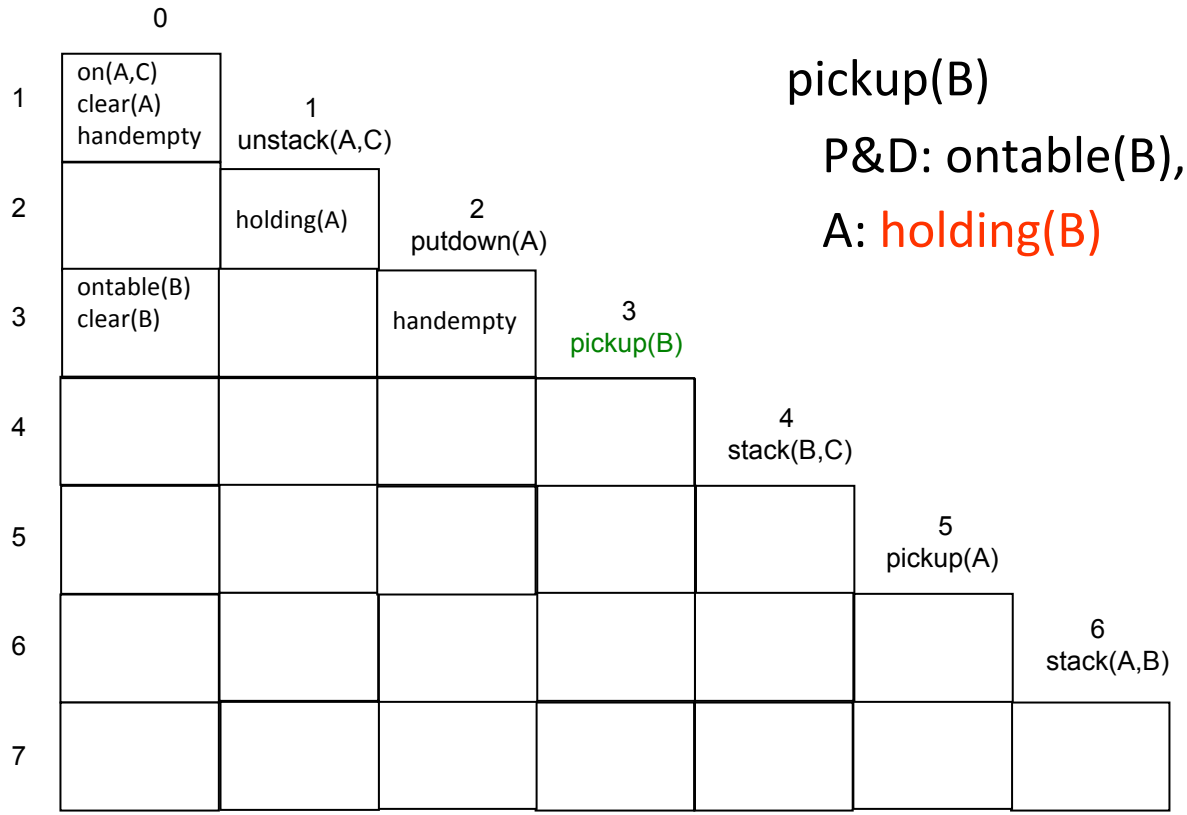
pickup(B)

P&D: ontable(B), clear(B), handempty

A: holding(B)

… și urcă-le în linia regulii 3

Execuția planului: pasul 3



pickup(B)

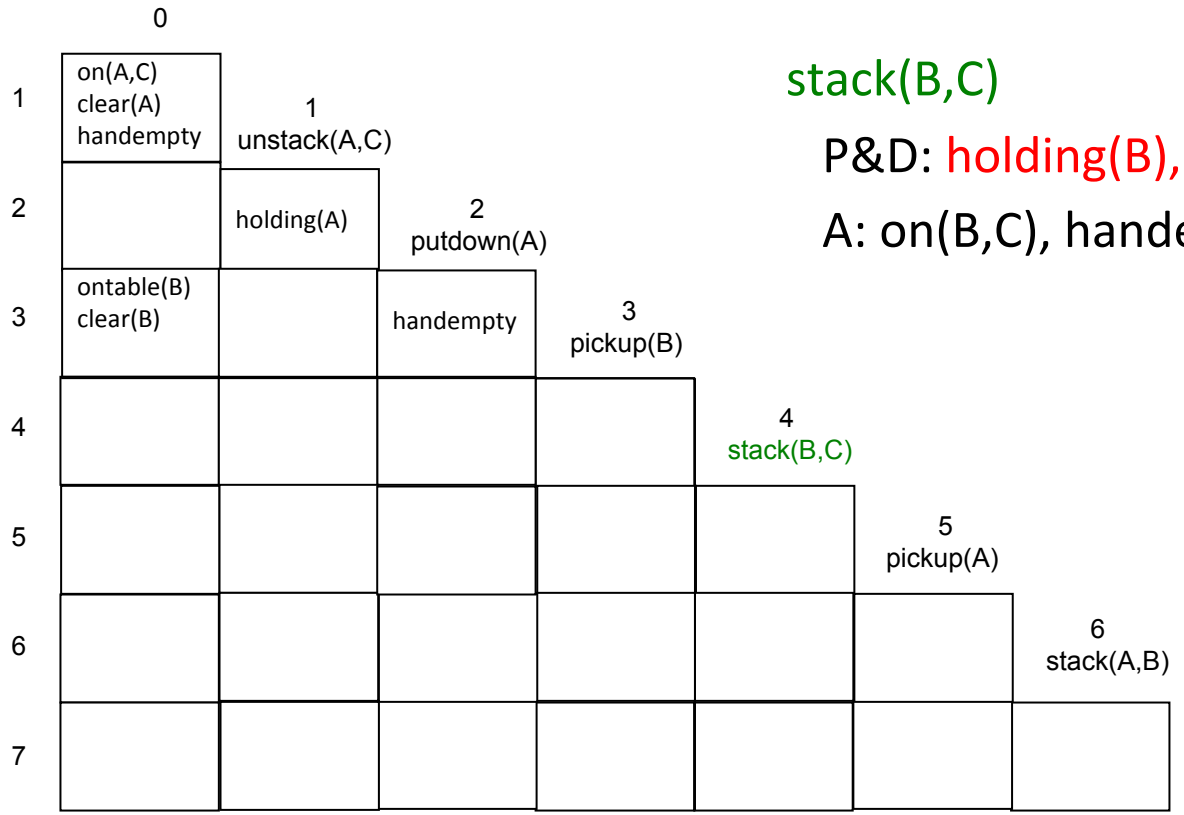
P&D: ontable(B), clear(B), handempty

A: holding(B)

ontable(C) | clear(C) | ontable(A) clear(A) | holding(B)

colectează predicatul listei A a regulii 3

Execuția planului: pasul 4



stack(B,C)

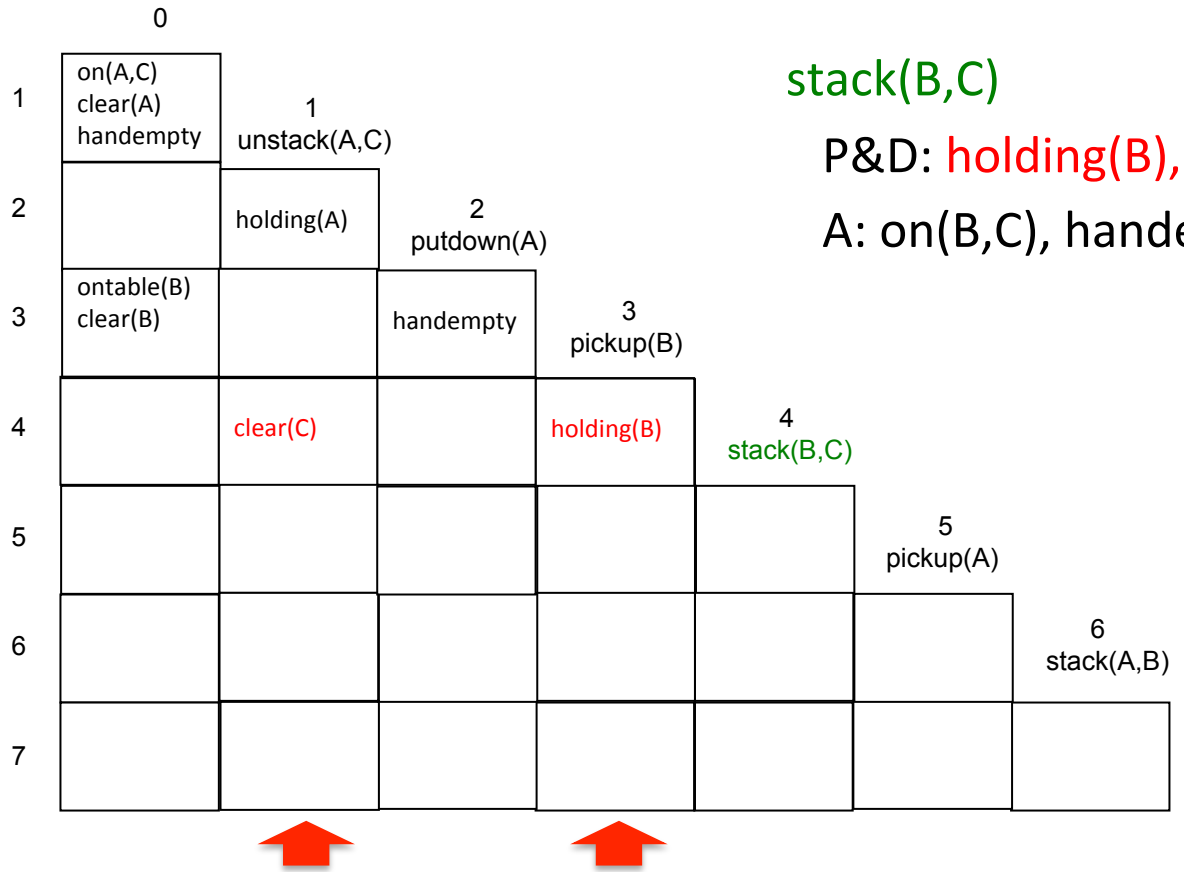
P&D: holding(B), clear(C)

A: on(B,C), handempty, clear(B)

ontable(C) clear(C) ontable(A) clear(A) holding(B)

caută predicatele listei P&D a regulii 4

Execuția planului: pasul 4



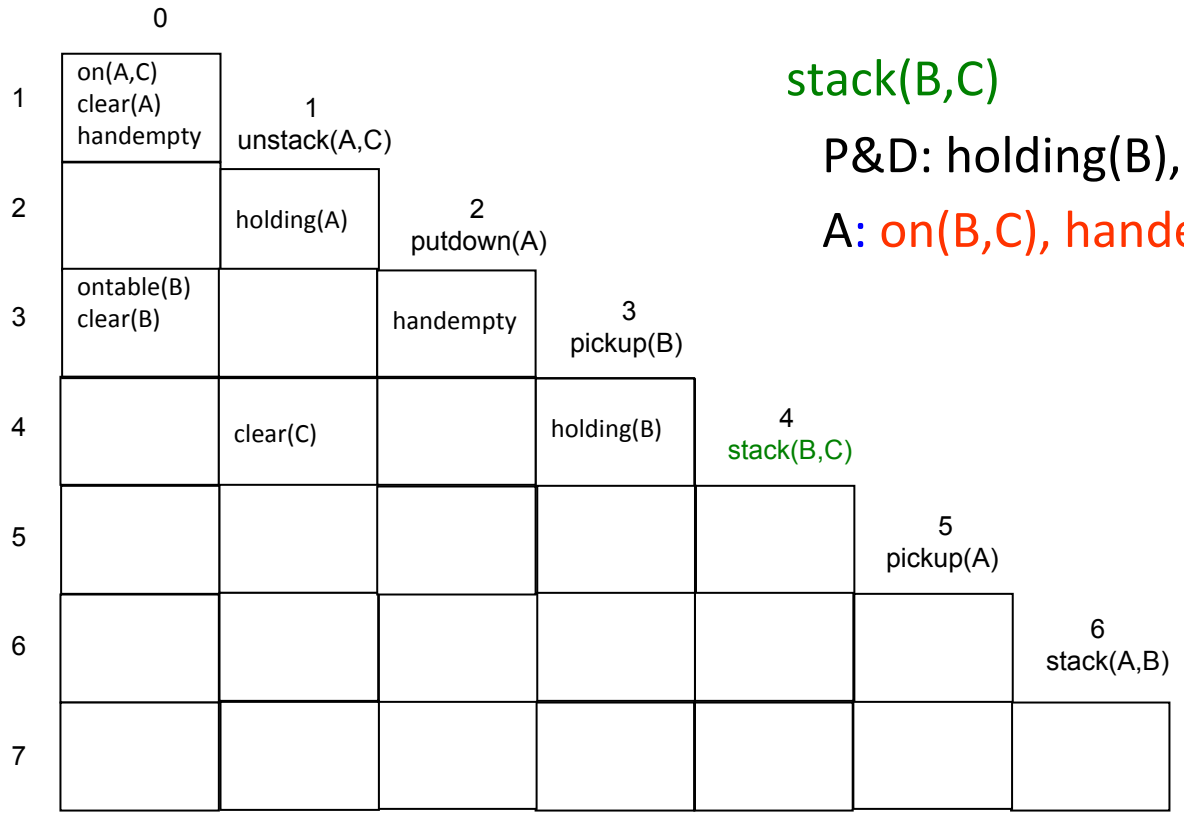
stack(B,C)

P&D: holding(B), clear(C)

A: on(B,C), handempty, clear(B)

… și urcă-le în linia regulii 4

Execuția planului: pasul 4



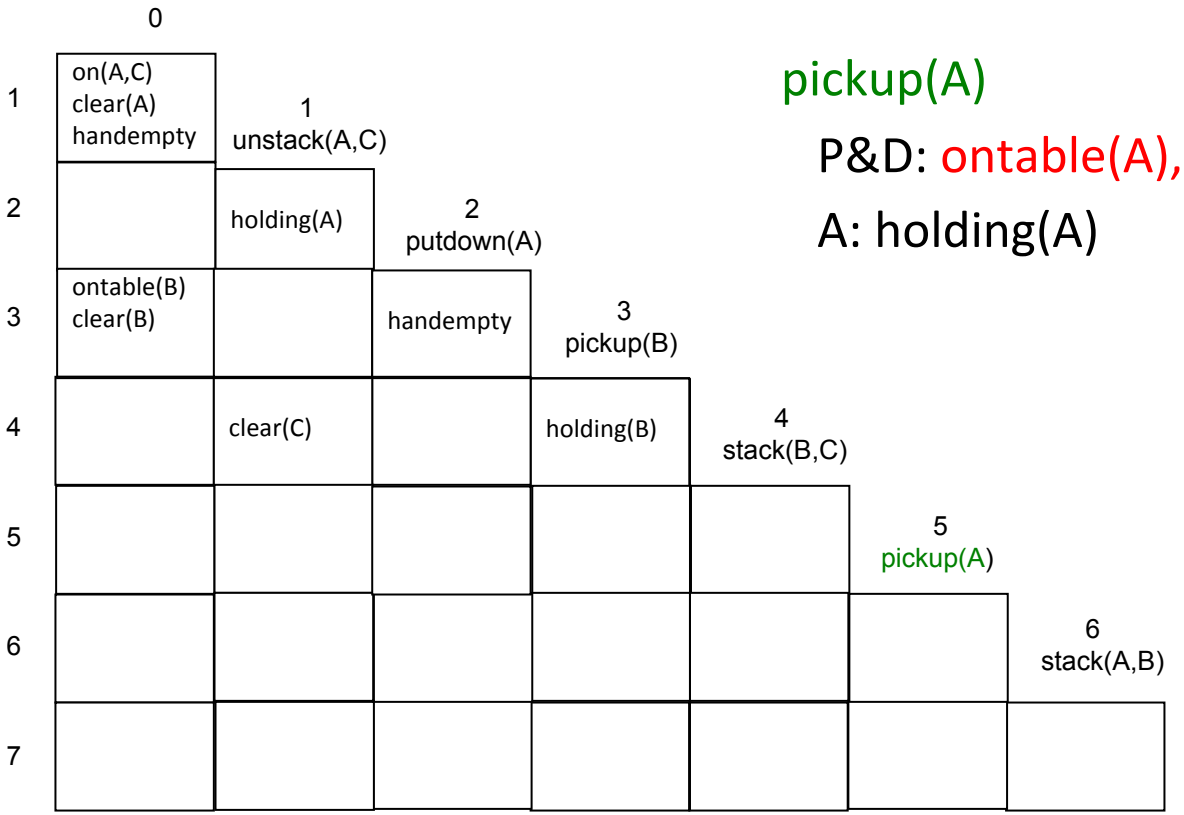
stack(B,C)

P&D: holding(B), clear(C)

A: on(B,C), handempty, clear(B)

colectează jos predicatele listei A a regulii 4

Execuția planului: pasul 5



pickup(A)

P&D: ontable(A), clear(A), handempty

A: holding(A)

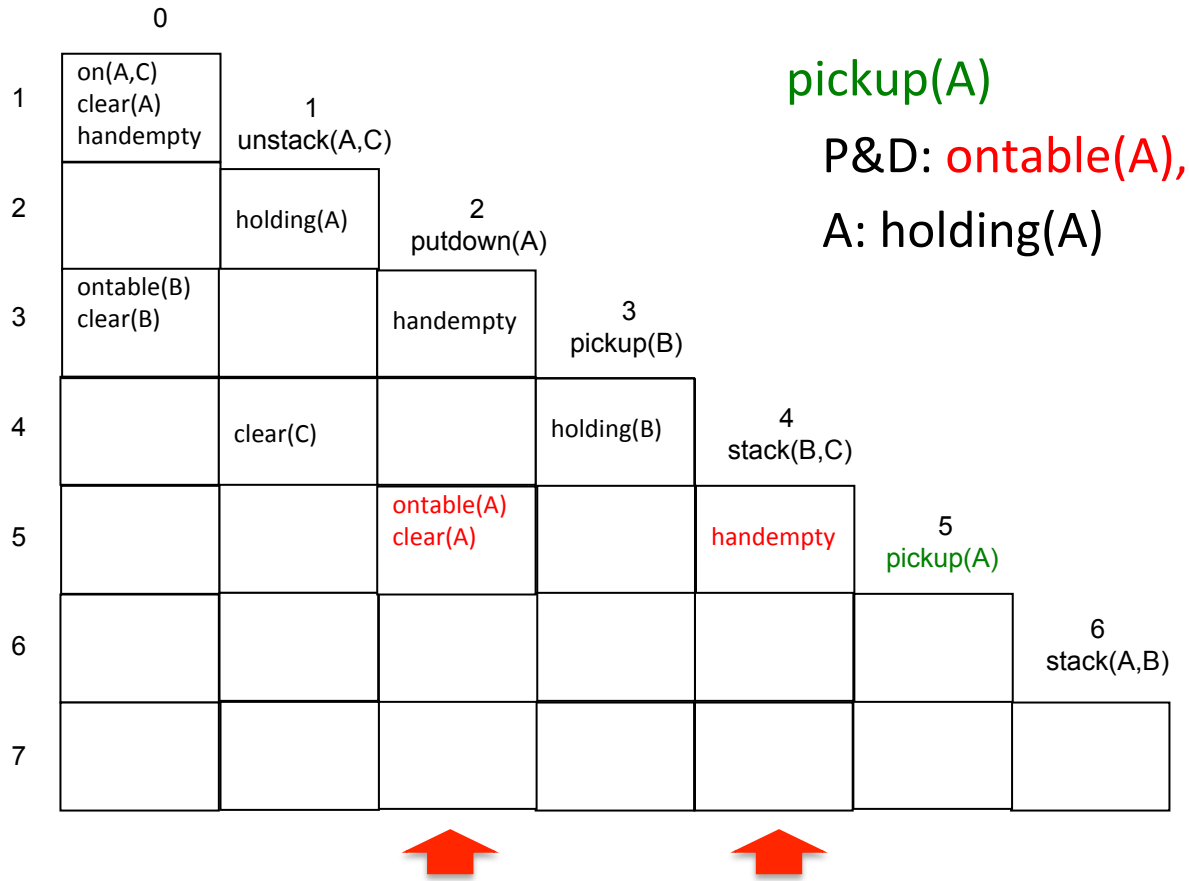
ontable(C)

ontable(A)
clear(A)

on(B,C)
handempty
clear(B)

caută predicatele listei
P&D a regulii 5

Execuția planului: pasul 5



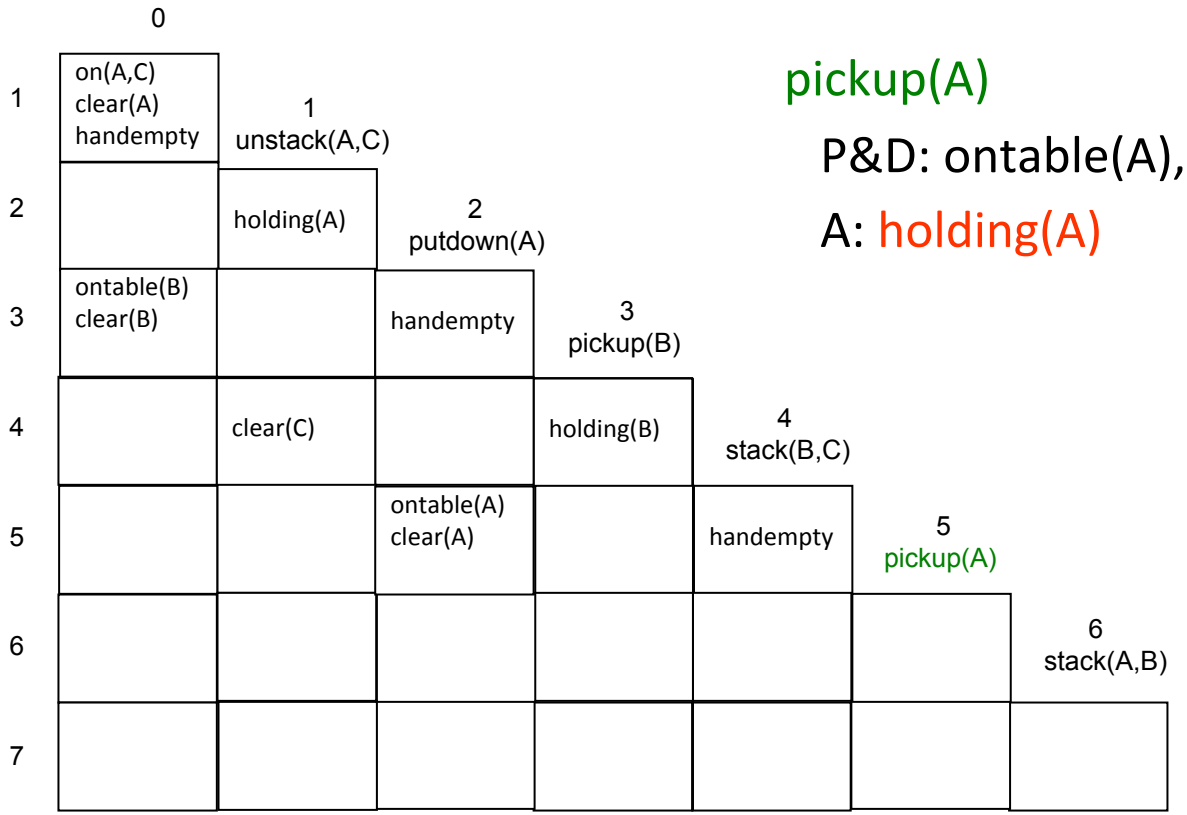
pickup(A)

P&D: ontable(A), clear(A), handempty

A: holding(A)

… și urcă-le în linia regulii 5

Execuția planului: pasul 5



pickup(A)

P&D: ontable(A), clear(A), handempty

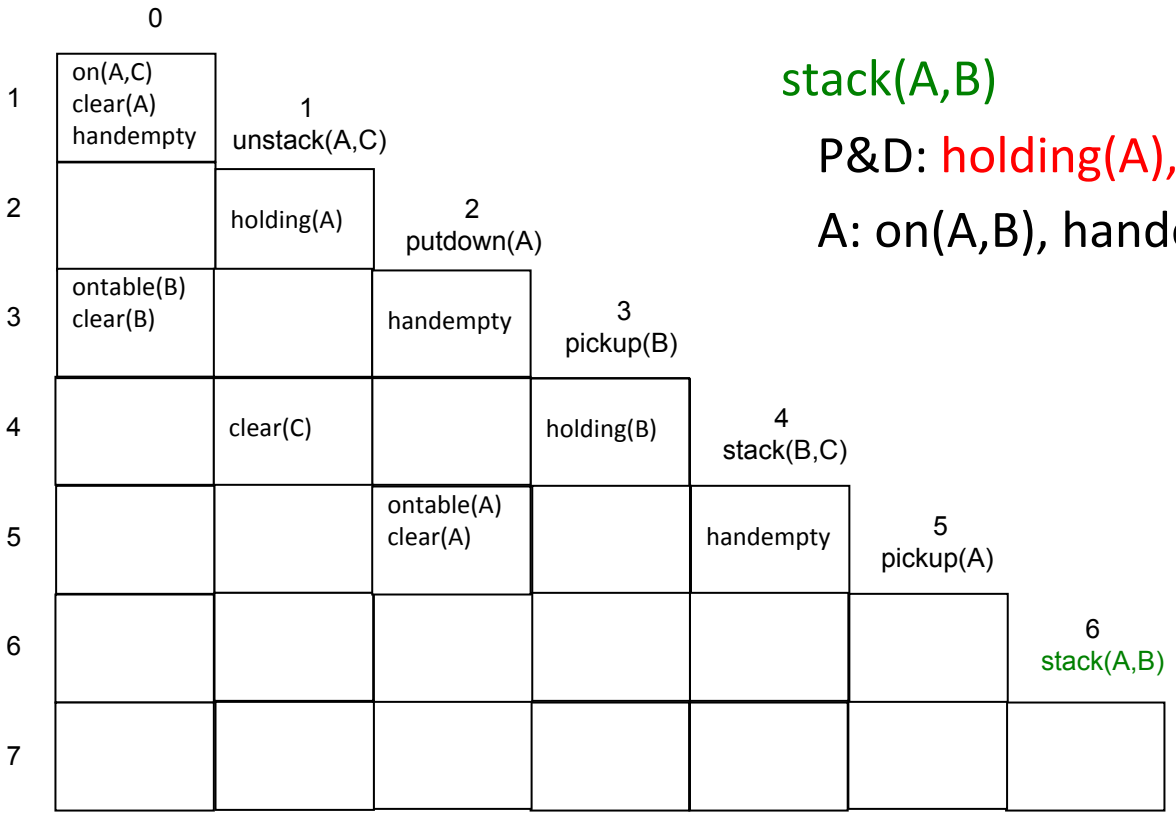
A: holding(A)

ontable(C)

on(B,C)
clear(B)
holding(A)

adună jos predicatele listei A a regulii 5

Execuția planului: pasul 6



stack(A,B)

P&D: holding(A), clear(B)

A: on(A,B), handempty, clear(A)

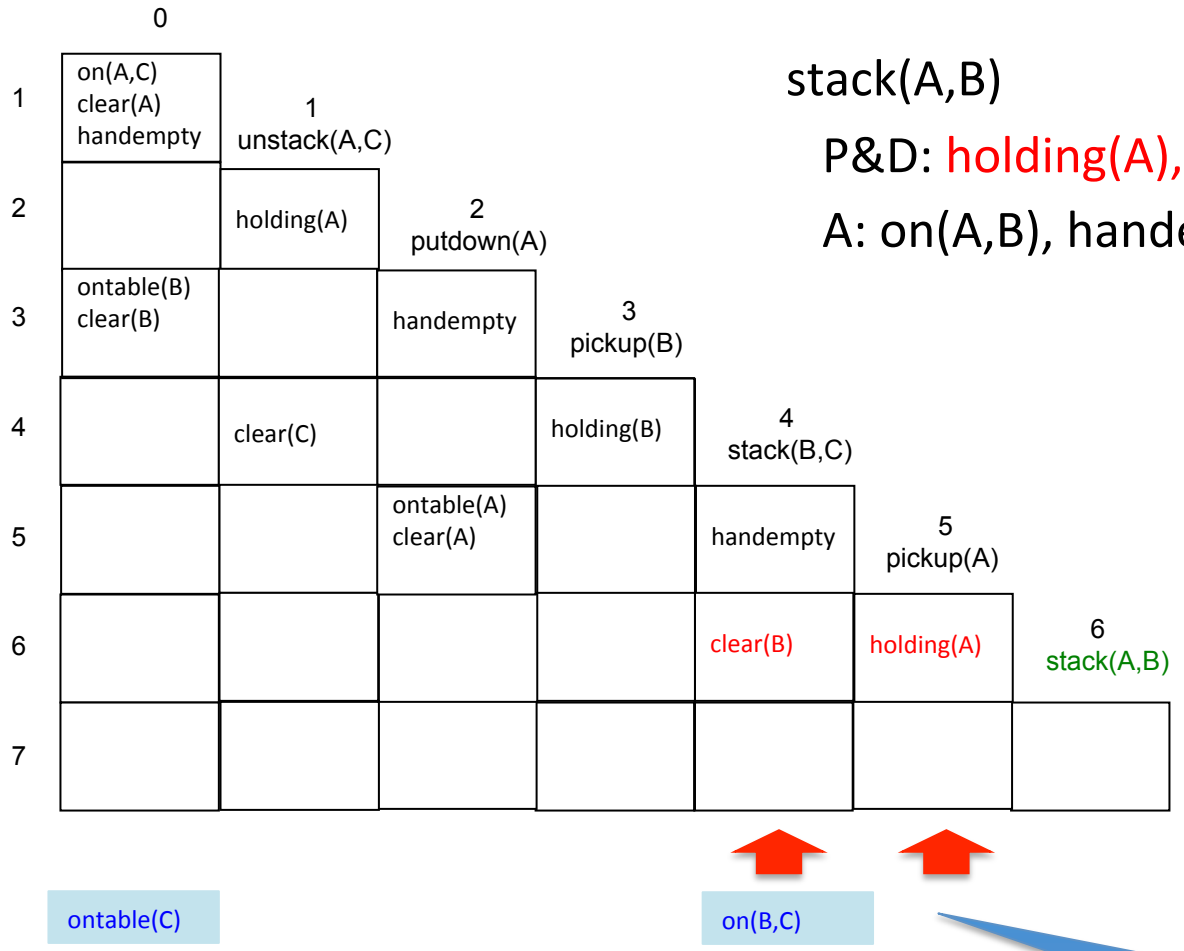
ontable(C)

on(B,C)
clear(B)

holding(A)

caută predicatele listei P&D a regulii 6

Matricea triunghiulară: pasul 6



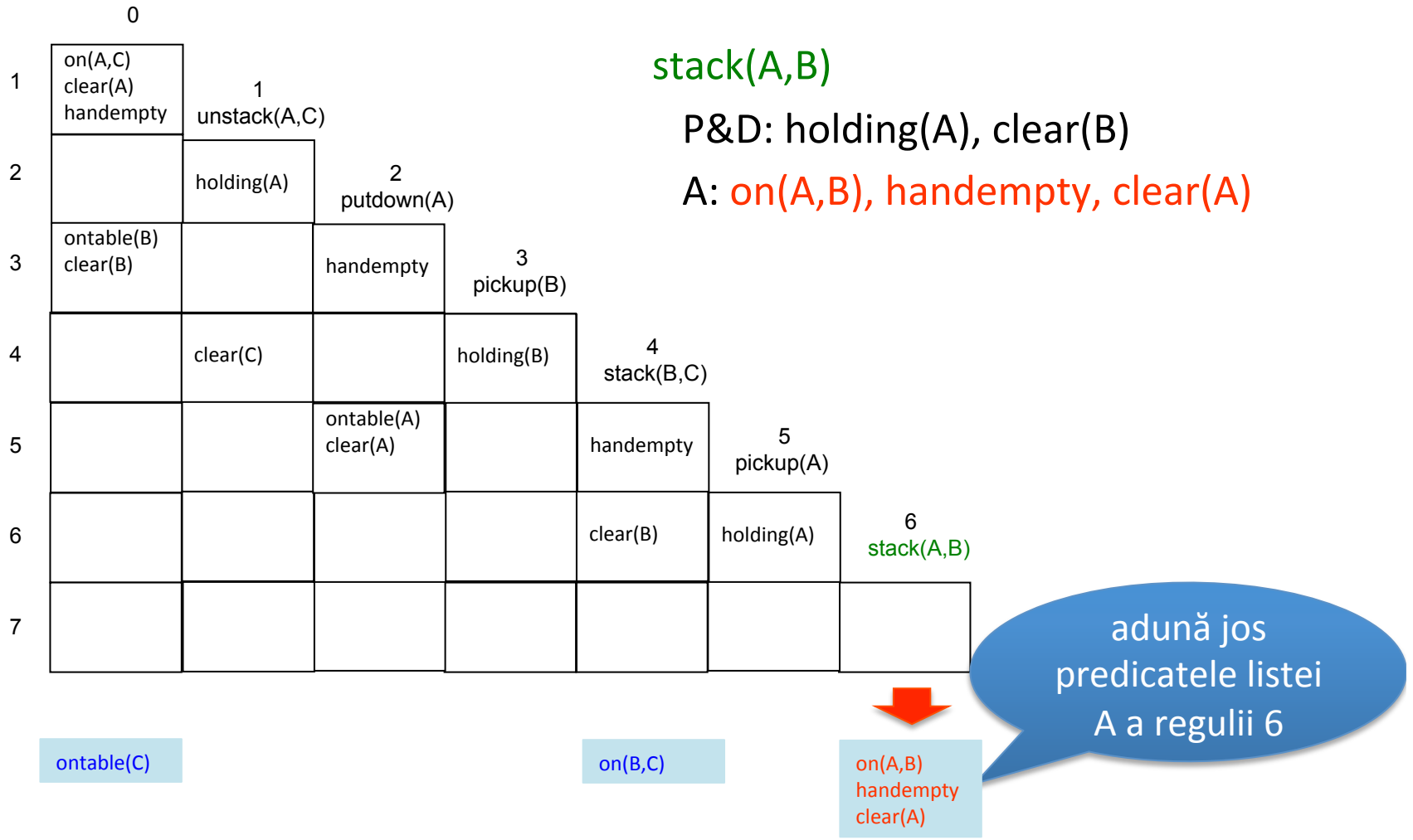
stack(A,B)

P&D: holding(A), clear(B)

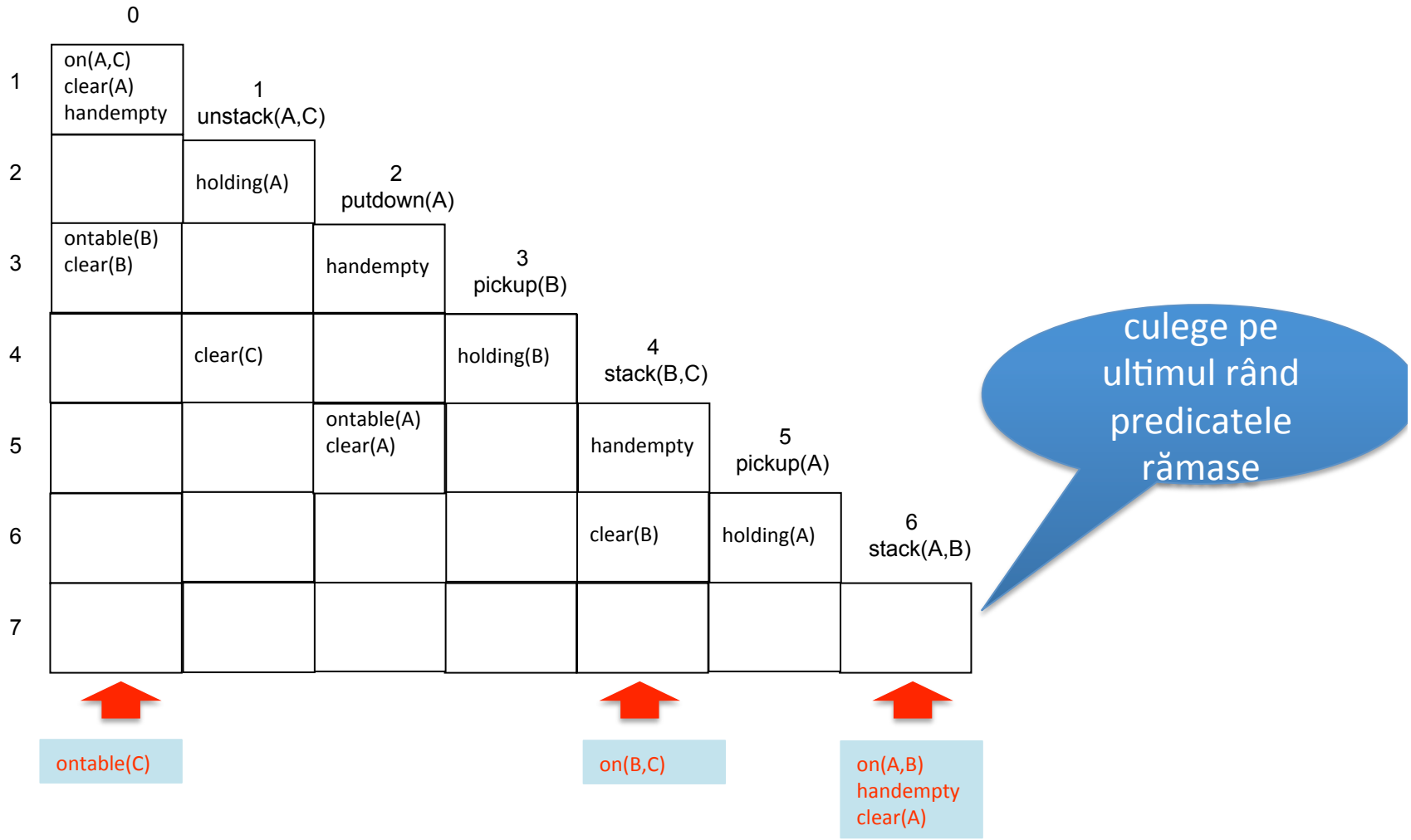
A: on(A,B), handempty, clear(A)

… și urcă-le în linia regulii 6

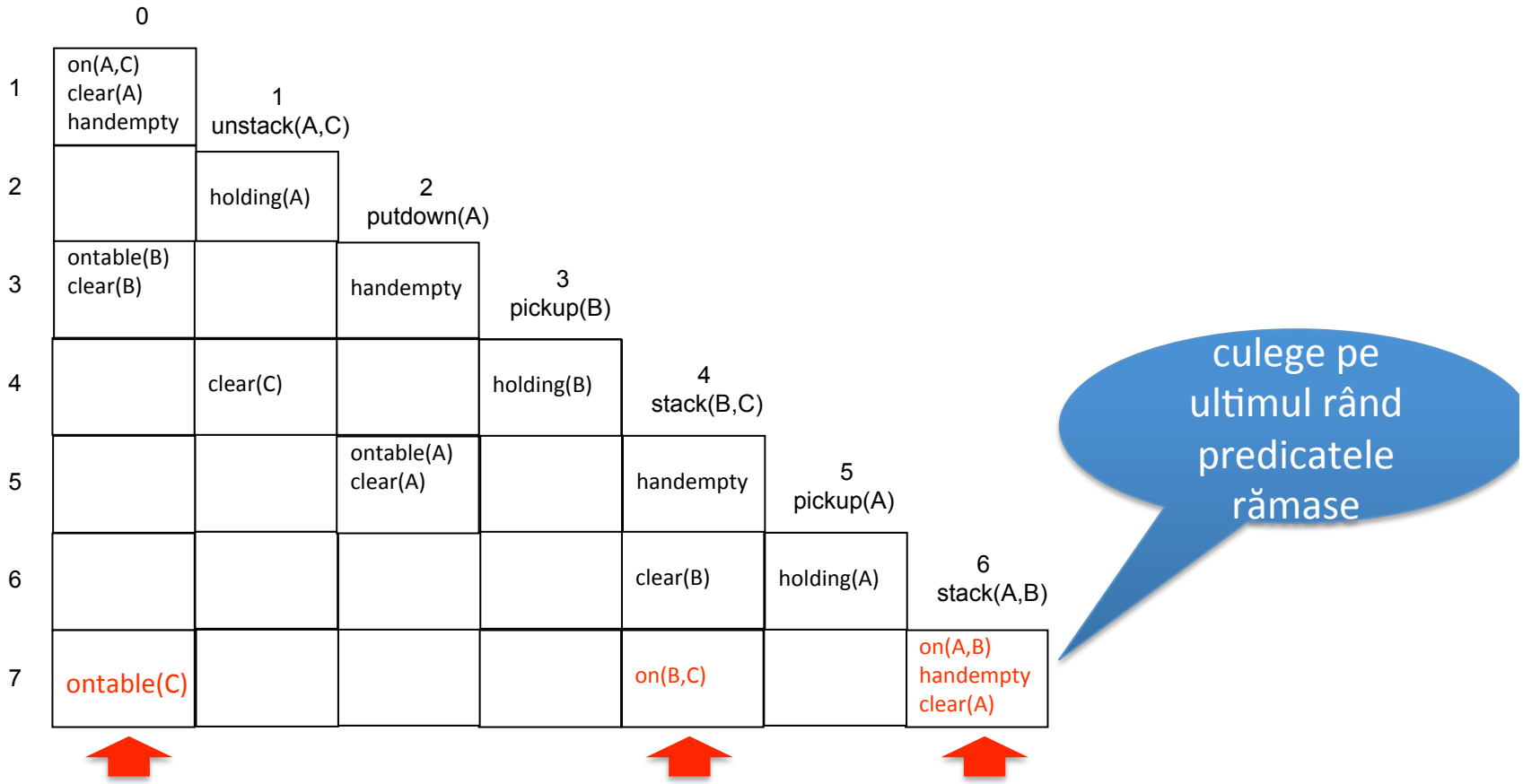
Execuția planului: pasul 6



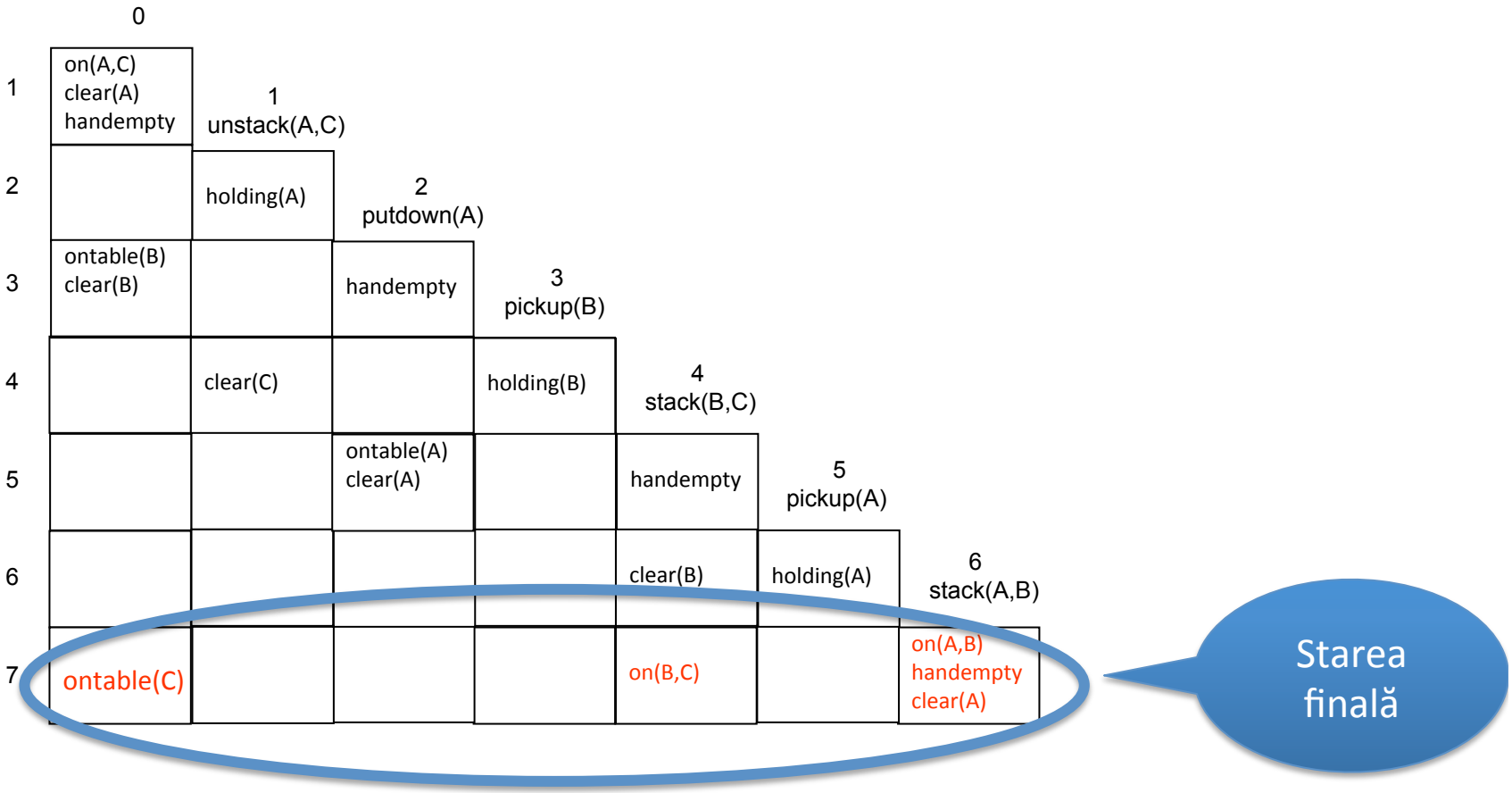
Execuția planului: finalizare



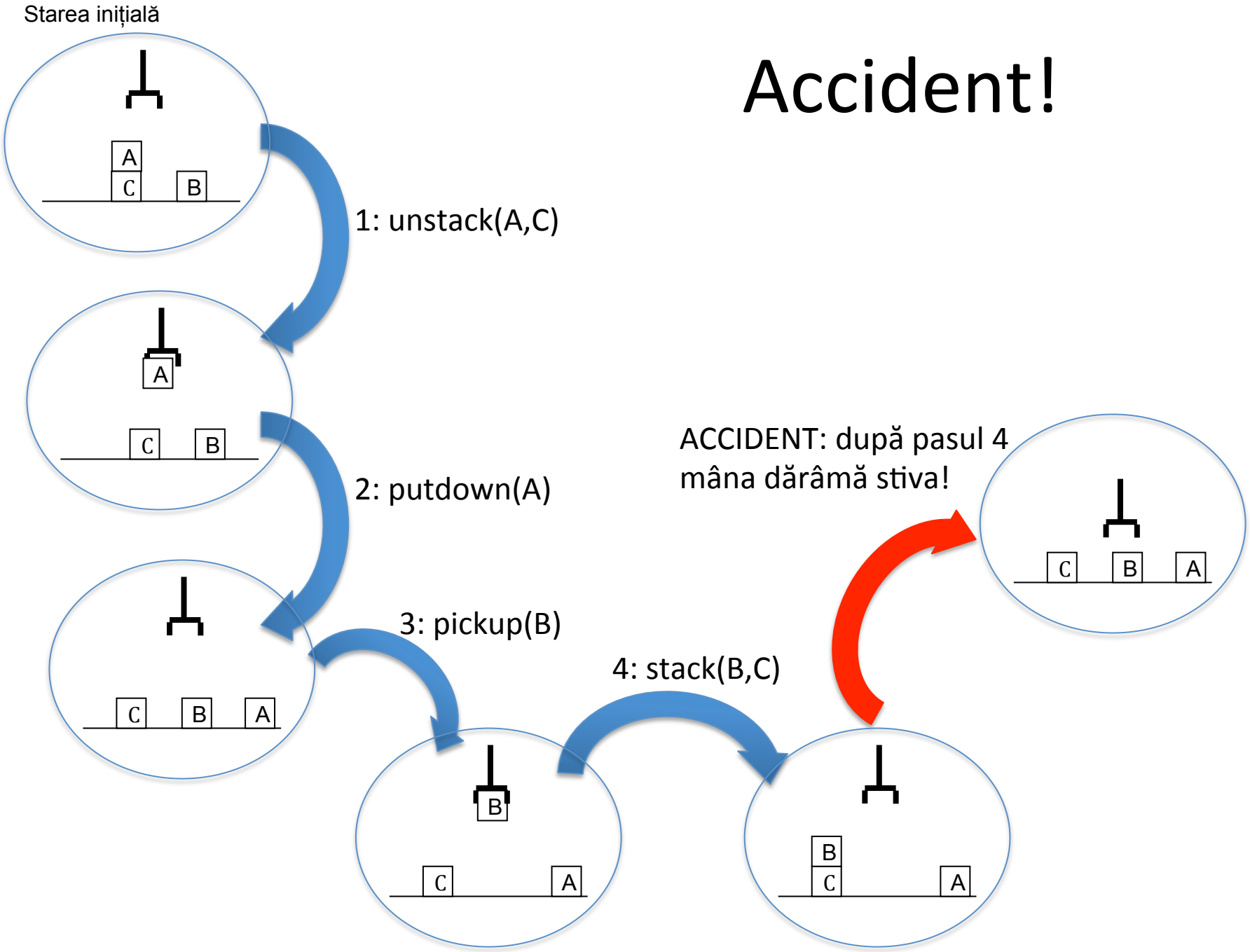
Execuția planului: finalizare

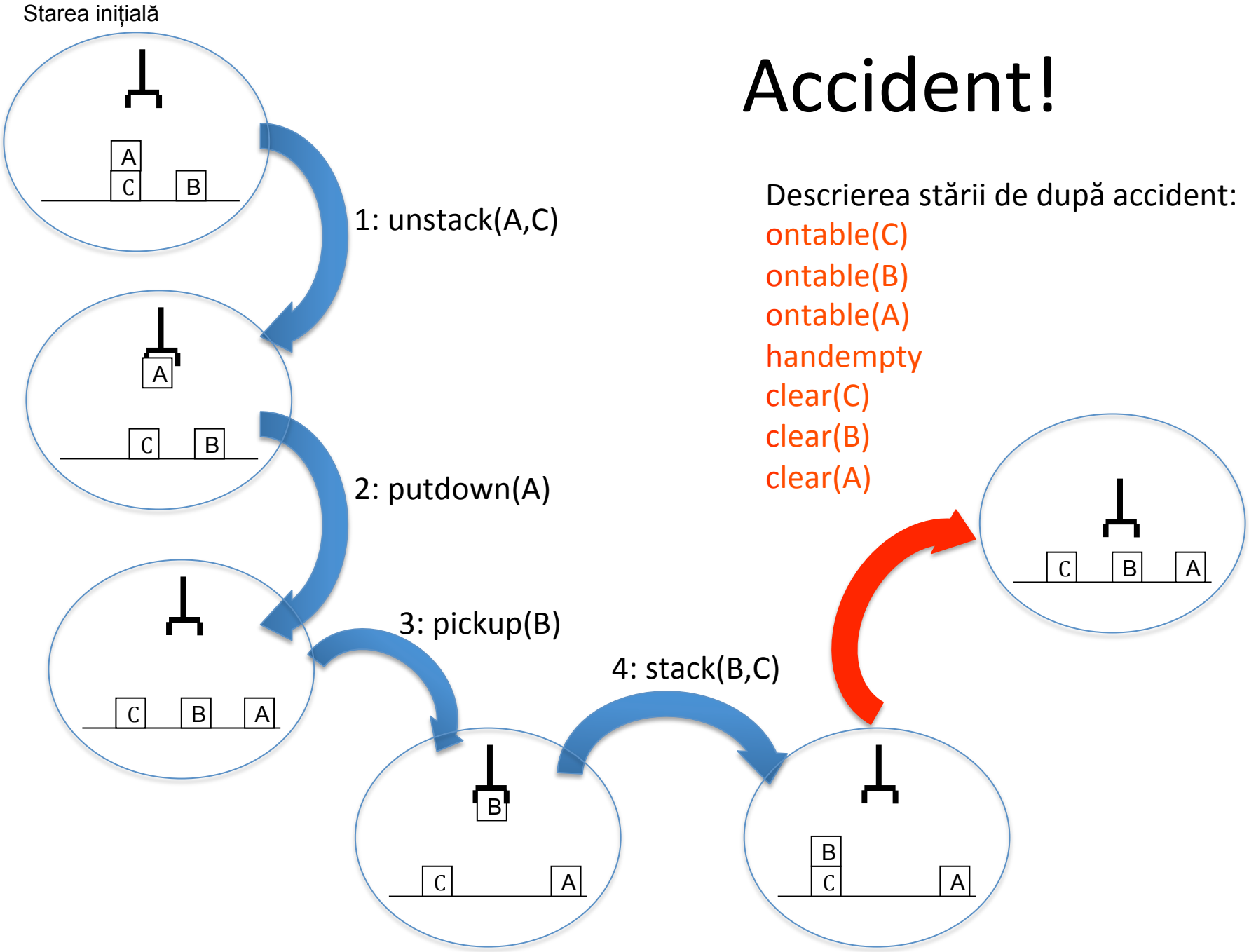


Execuția planului: finalizare



Accident!

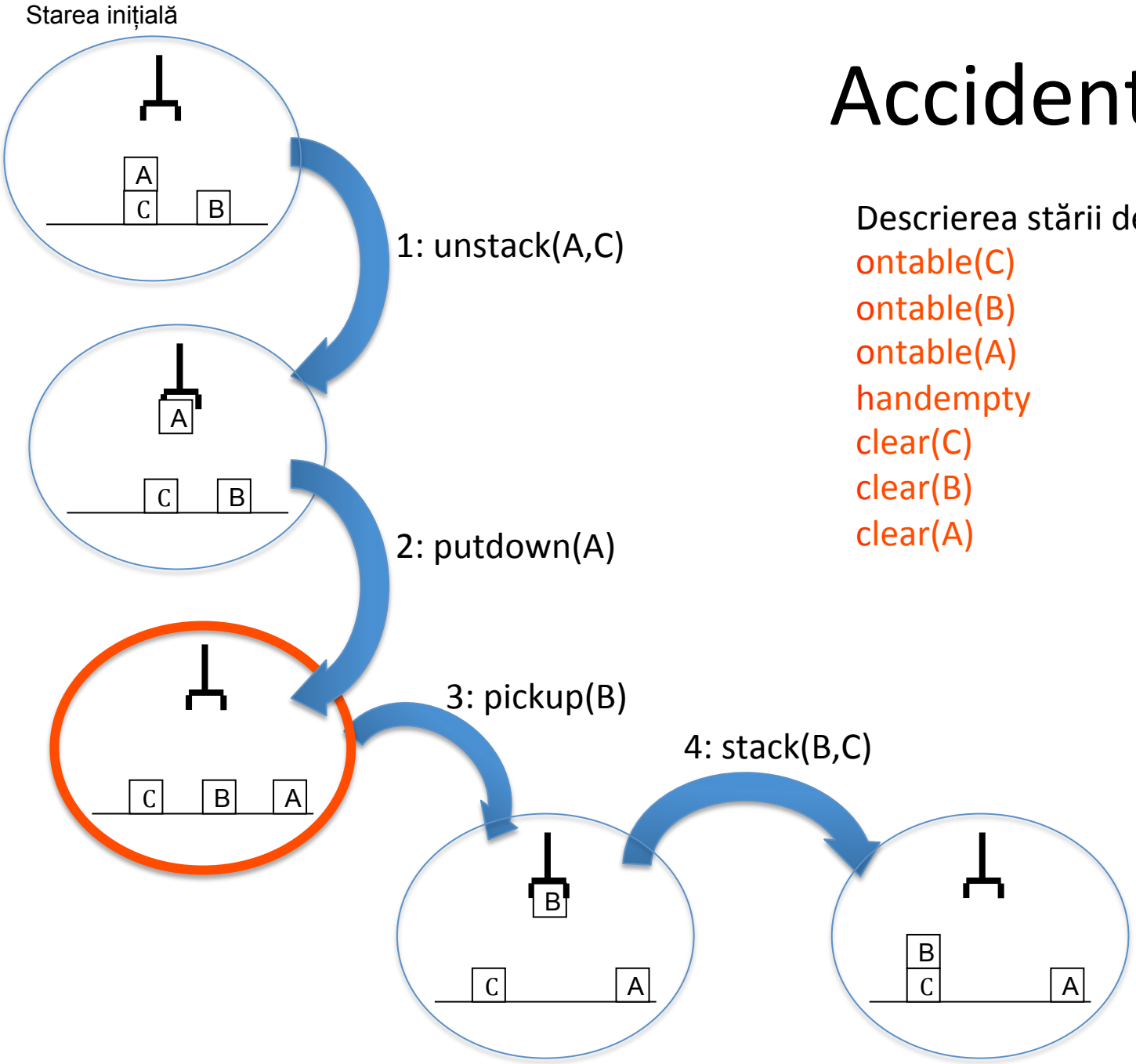

Starea inițială
A
C
B
1: unstack(A,C)
A
C
B
2: putdown(A)
C
B
A
3: pickup(B)
B
C
A
4: stack(B,C)
B
C
A
ACCIDENT: după pasul 4
mâna dărâmă stiva!
C
B
A



Accident!

Descrierea stării de după accident:

ontable(C)
ontable(B)
ontable(A)
handempty
clear(C)
clear(B)
clear(A)

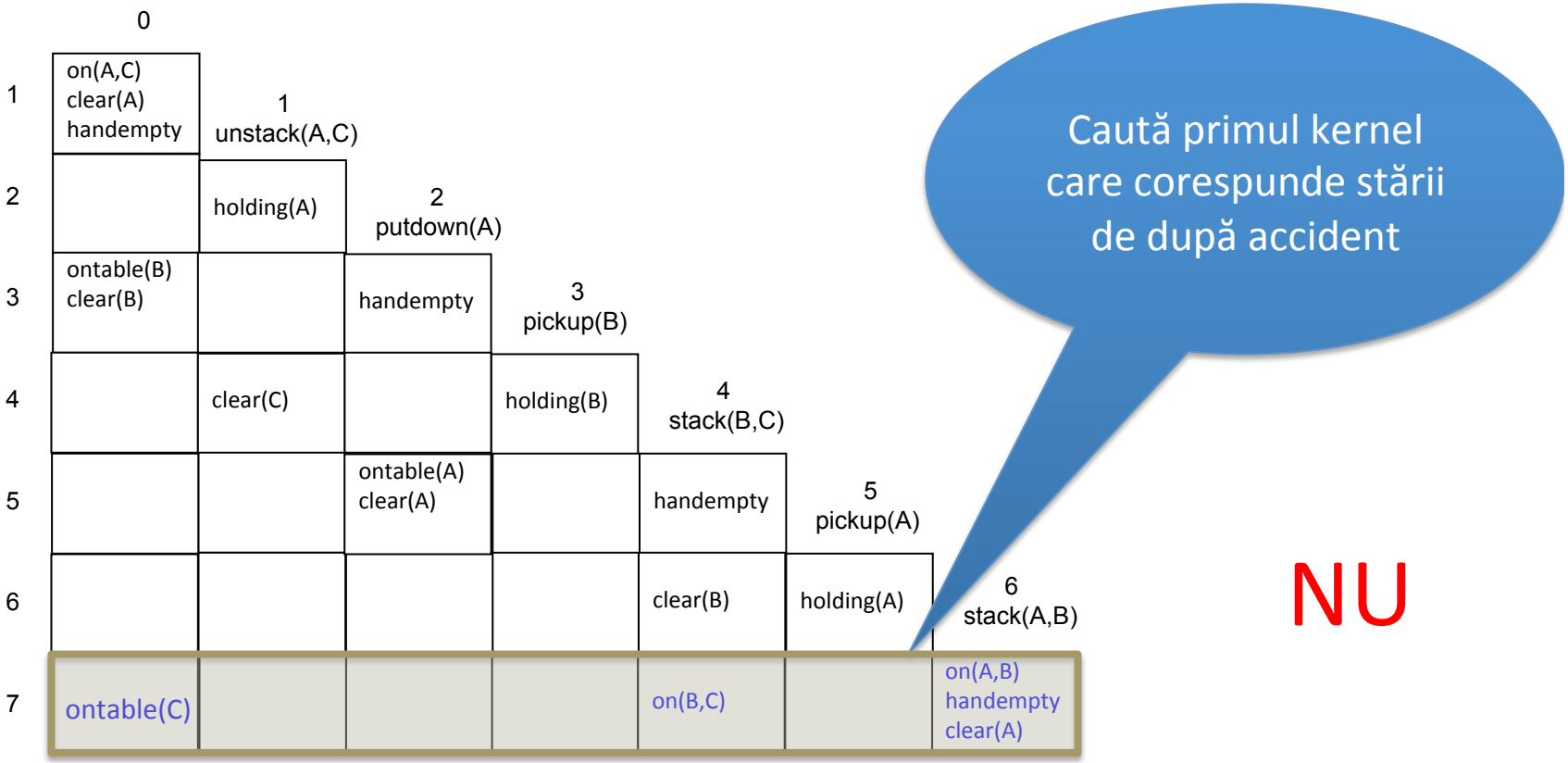


Accident!

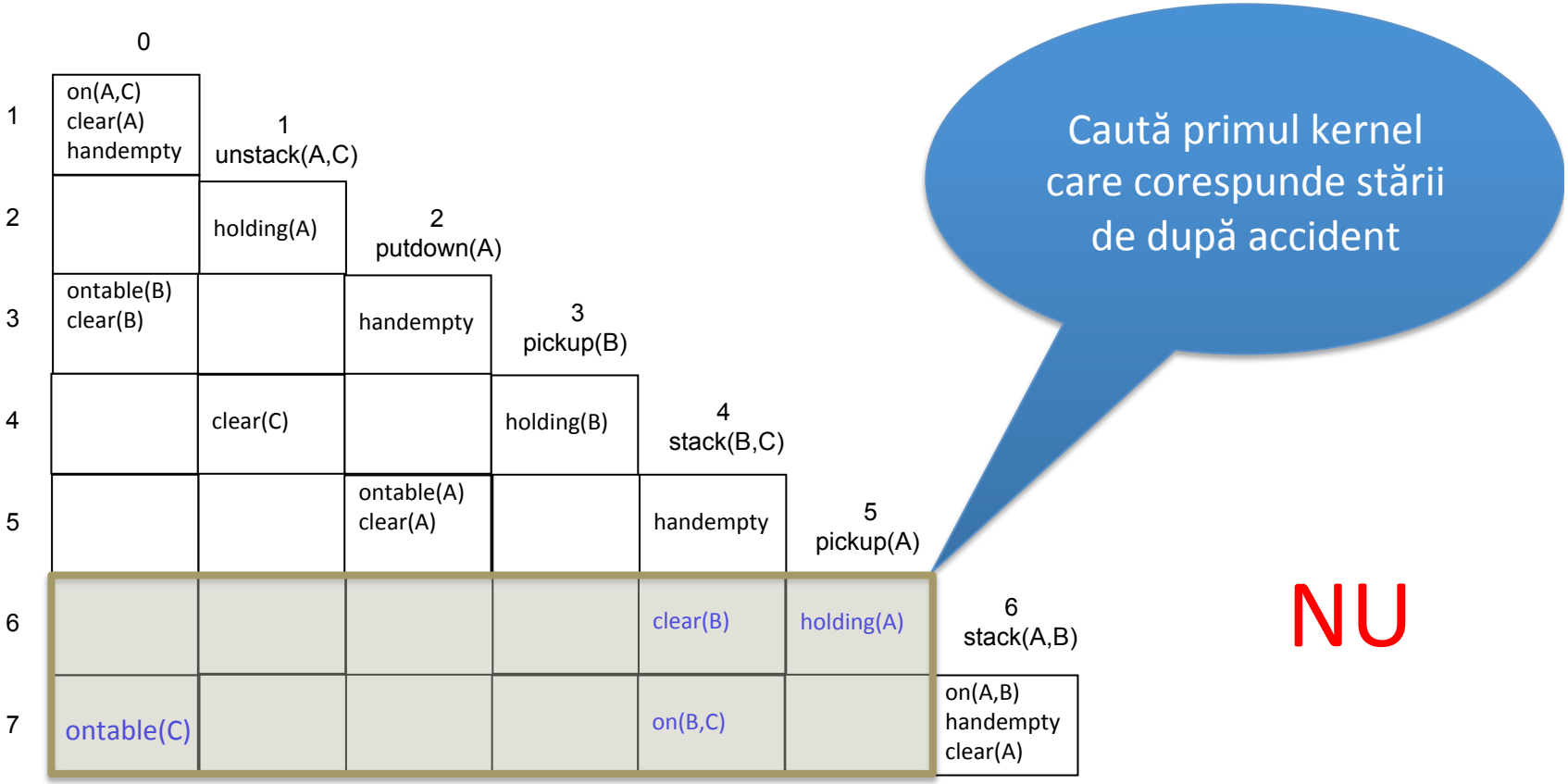
Descrierea stării de după accident:

ontable(C)
ontable(B)
ontable(A)
handempty
clear(C)
clear(B)
clear(A)

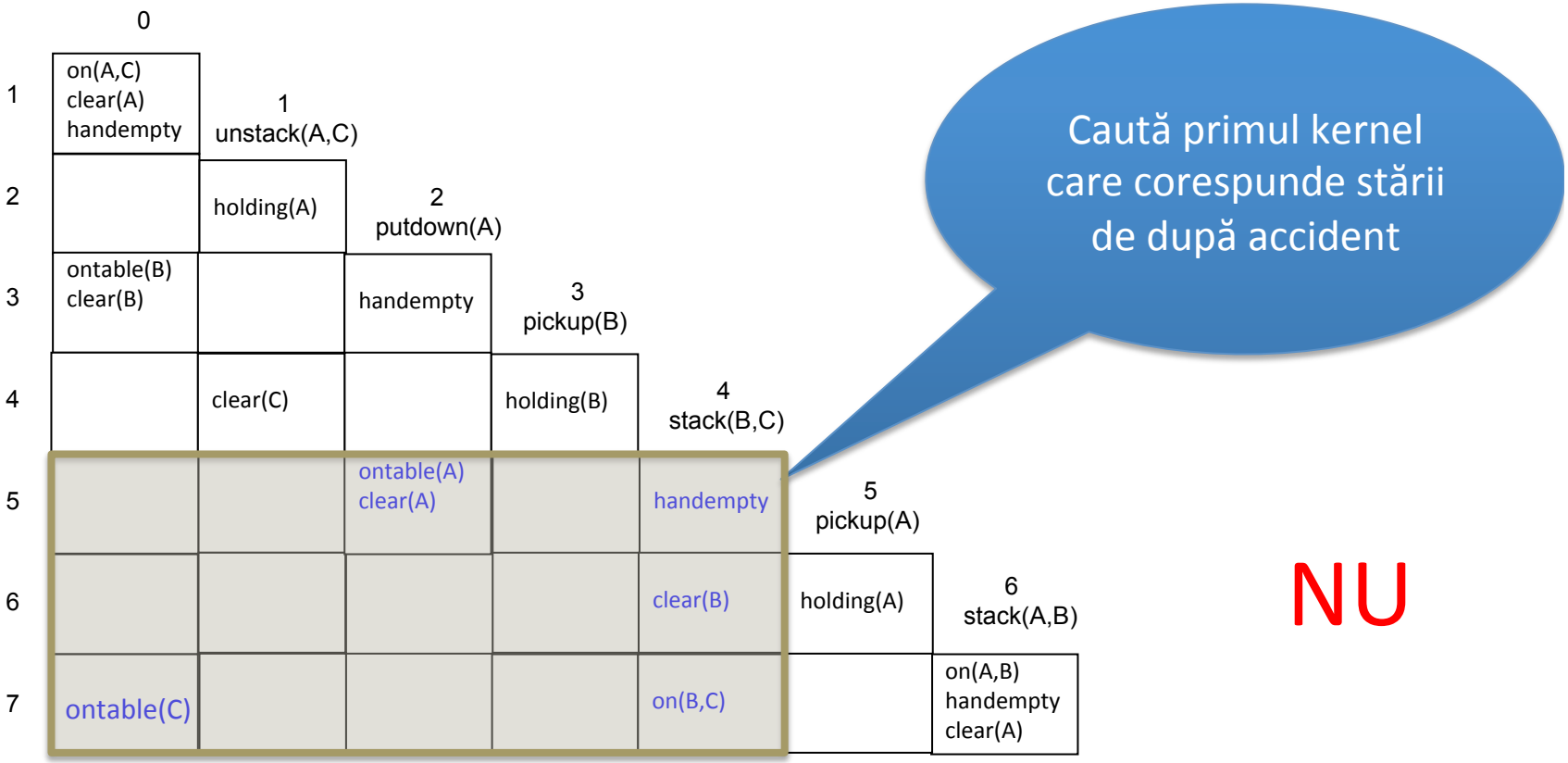
Kernel de rang 7



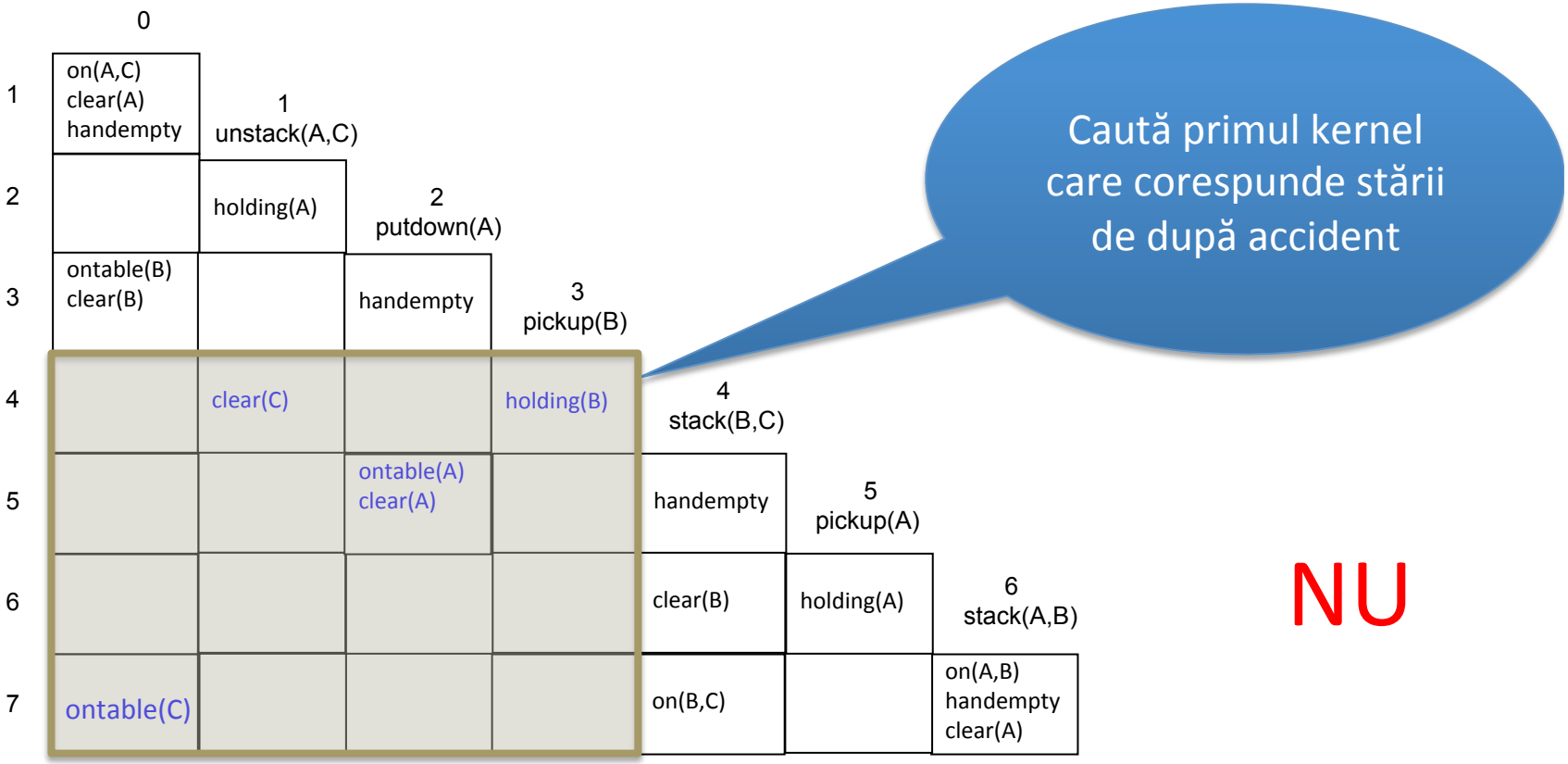
Kernel de rang 6



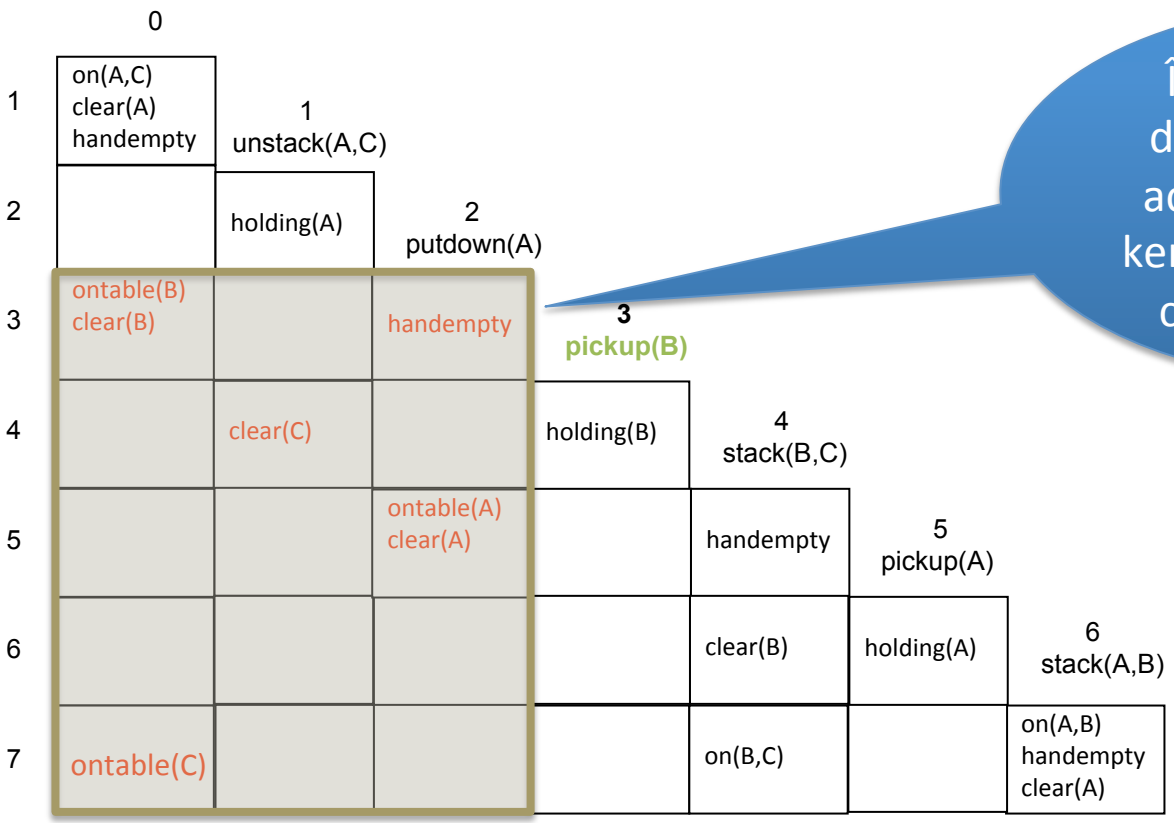
Kernel de rang 5



Kernel de rang 4

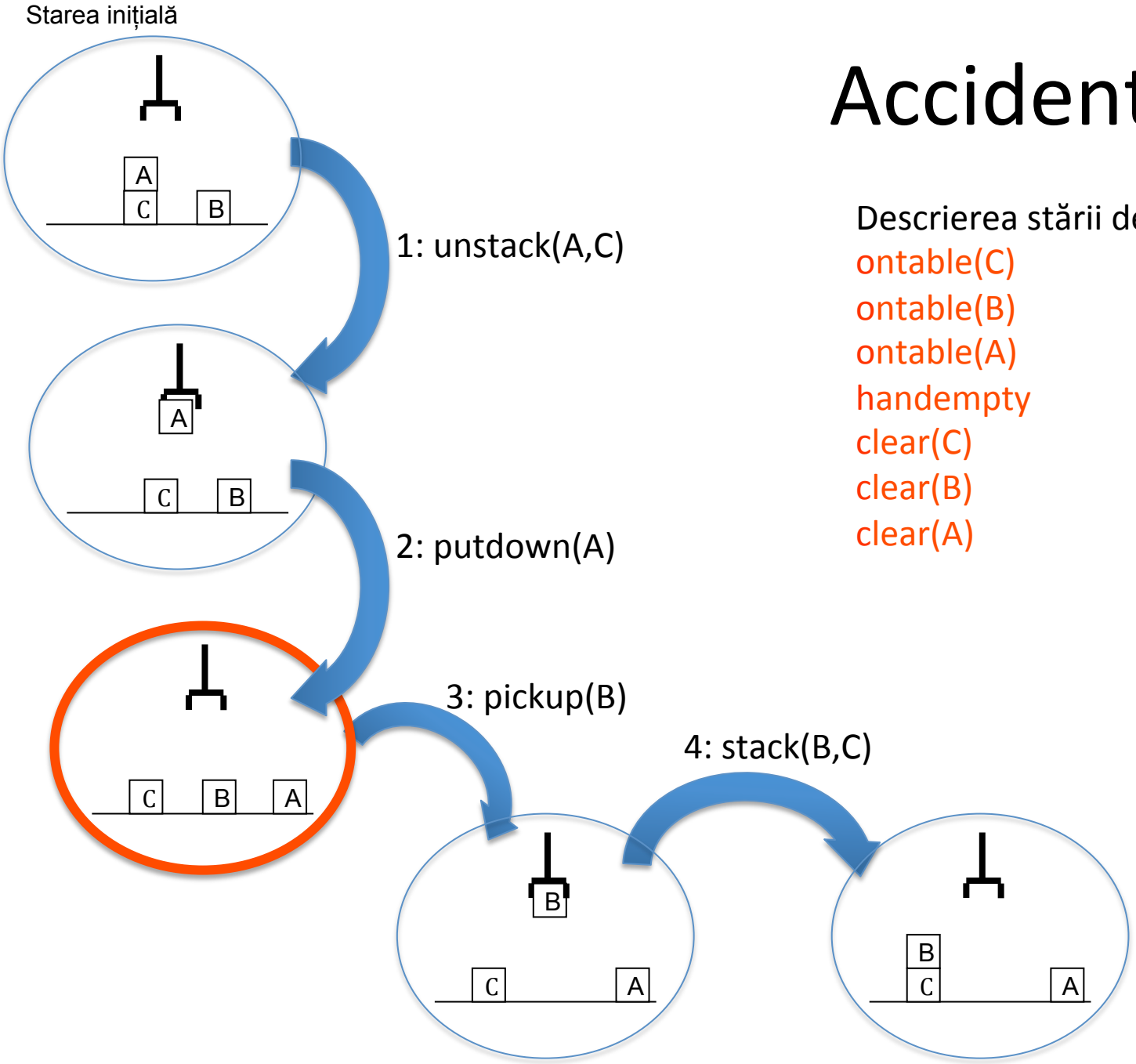


Kernel de rang 3



În caz de accident: dacă starea de după accident corespunde kernelului de rang *k* ➔ continuă cu pasul *k*

DA



Accident!

Descrierea stării de după accident:

ontable(C)
ontable(B)
ontable(A)
handempty
clear(C)
clear(B)
clear(A)

Concluzia:
care e conduita după un accident?

- Starea de după accident:
  - se regăsește în mulțimea de kerneluri => reia execuția de acolo!
  - nu se regăsește => construiește un nou plan cu această stare ca stare inițială! => execută planul!