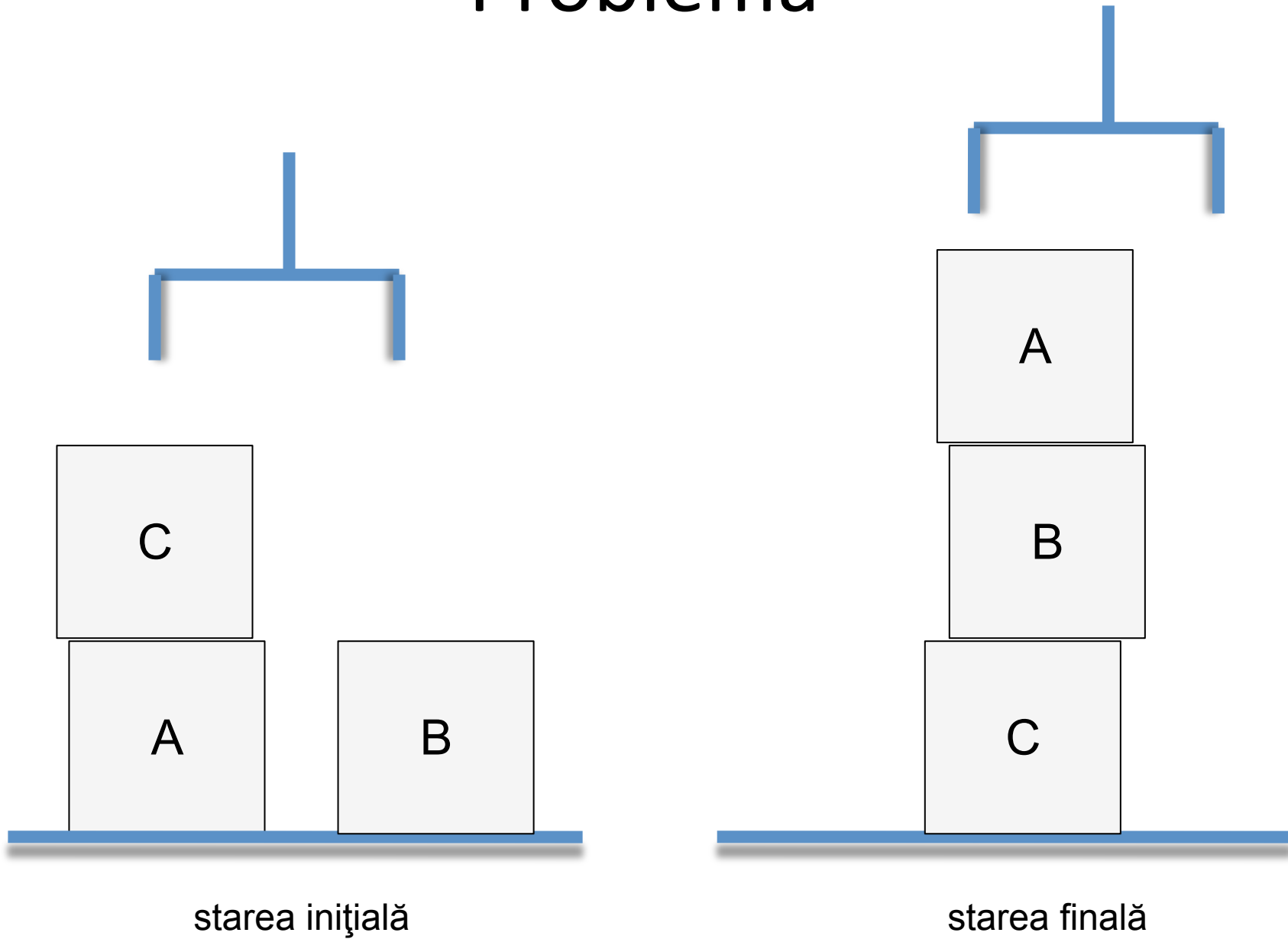
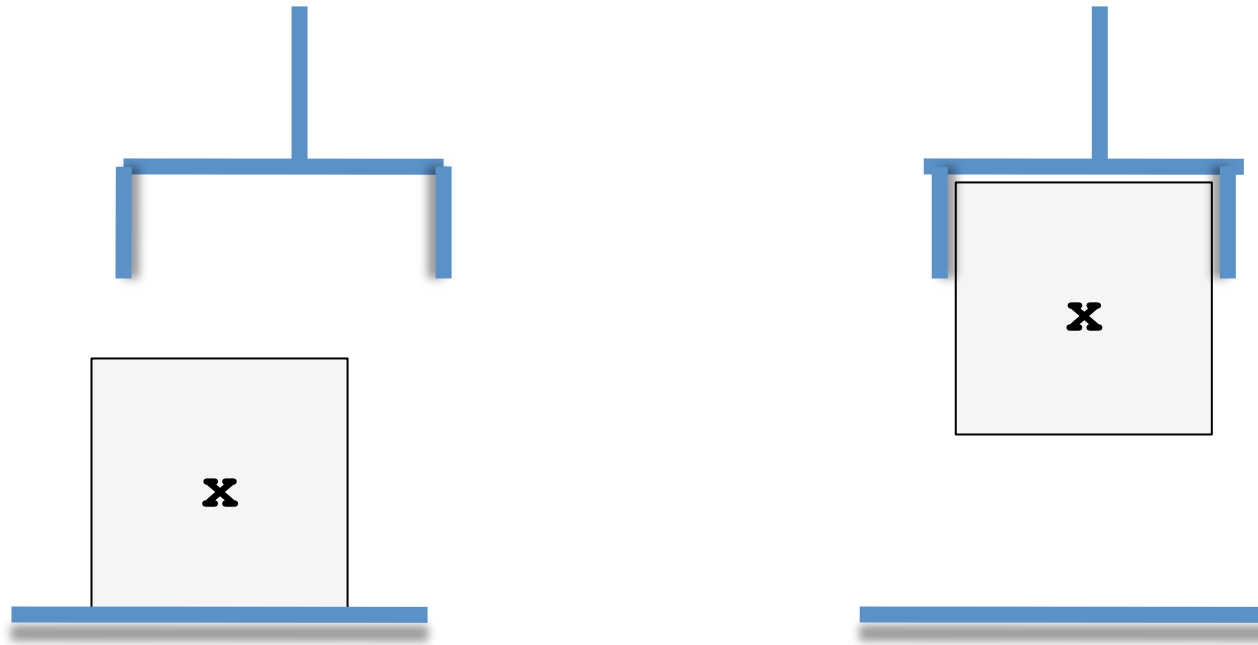


# Curs 12 – Planificare și execuția planurilor

# Problema



# Reguli STRIPS

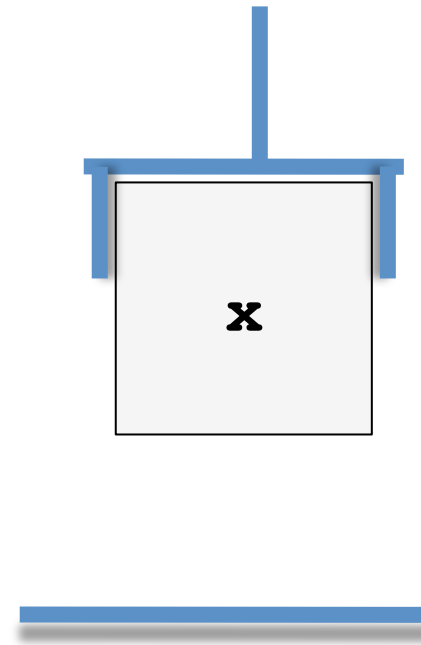
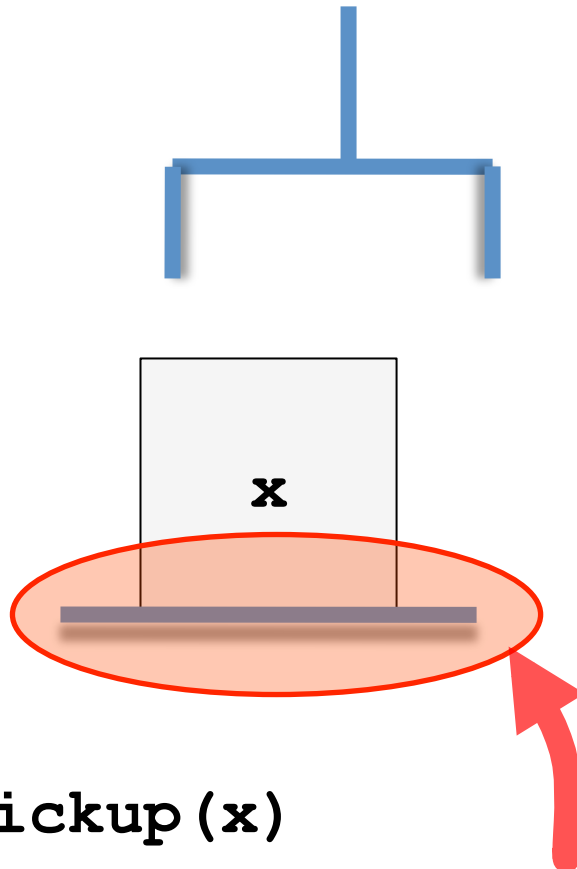


`pickup(x)`

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

A: `holding(x)`

# Reguli STRIPS

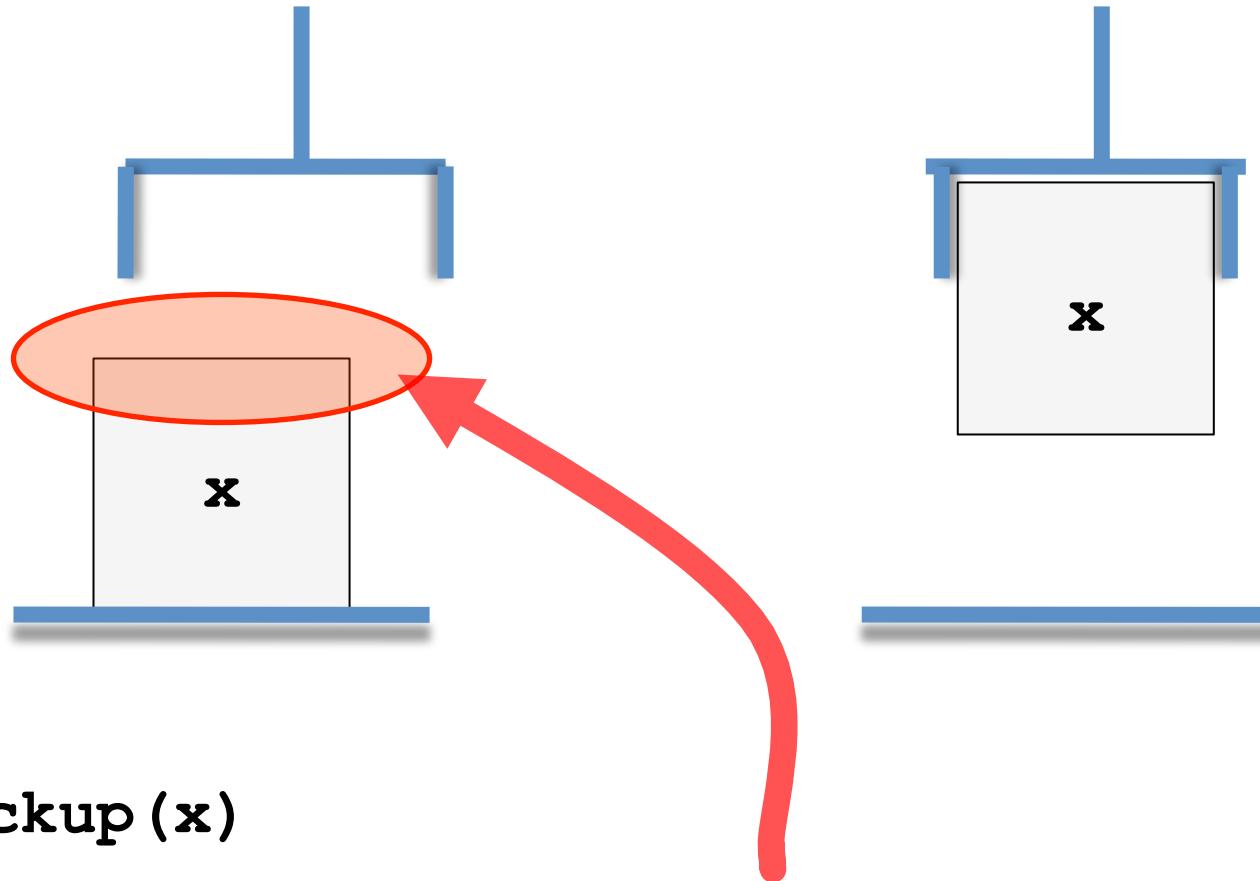


`pickup(x)`

P&D: **`ontable(x)`** , `clear(x)` , `handempty`

A: `holding(x)`

# Reguli STRIPS

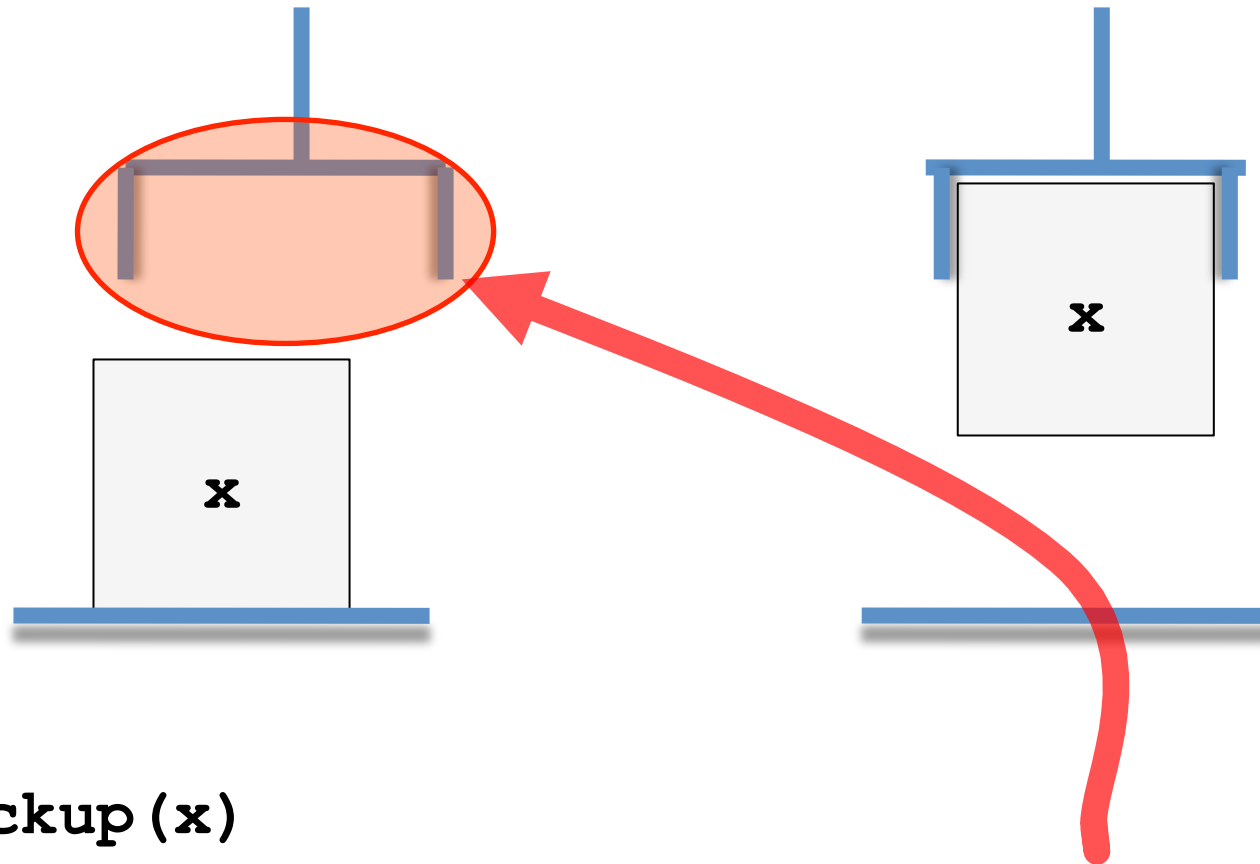


`pickup(x)`

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

A: `holding(x)`

# Reguli STRIPS

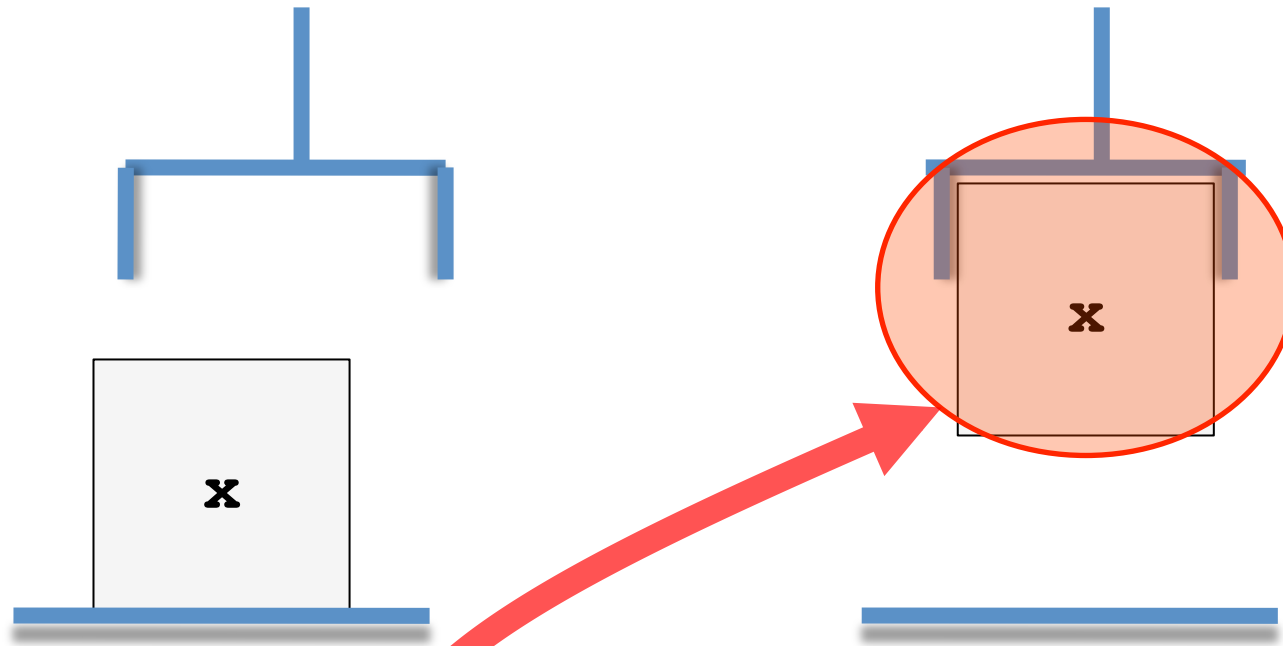


`pickup(x)`

P&D: `ontable(x)` , `clear(x)` , **`handempty`**

A: `holding(x)`

# Reguli STRIPS

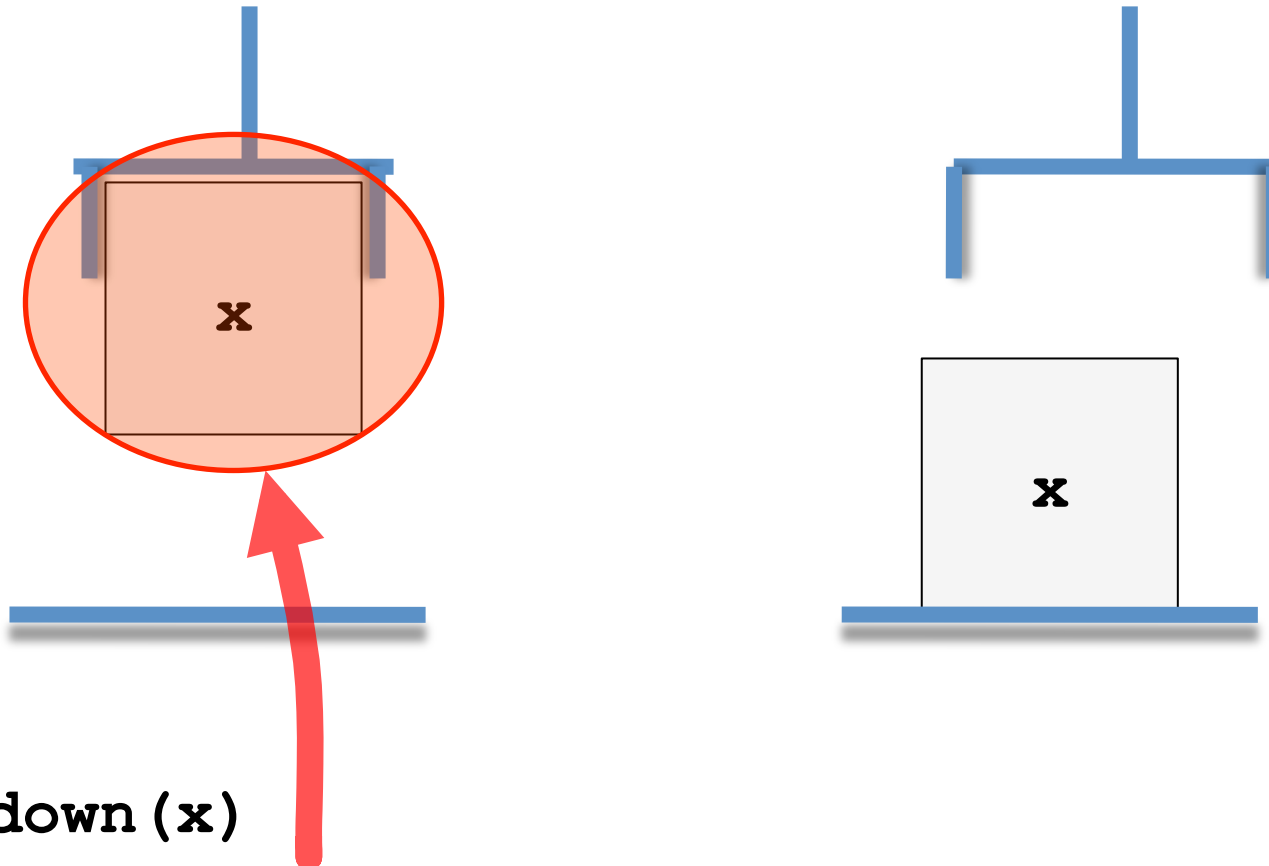


`pickup(x)`

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

A: **`holding(x)`**

# Reguli STRIPS



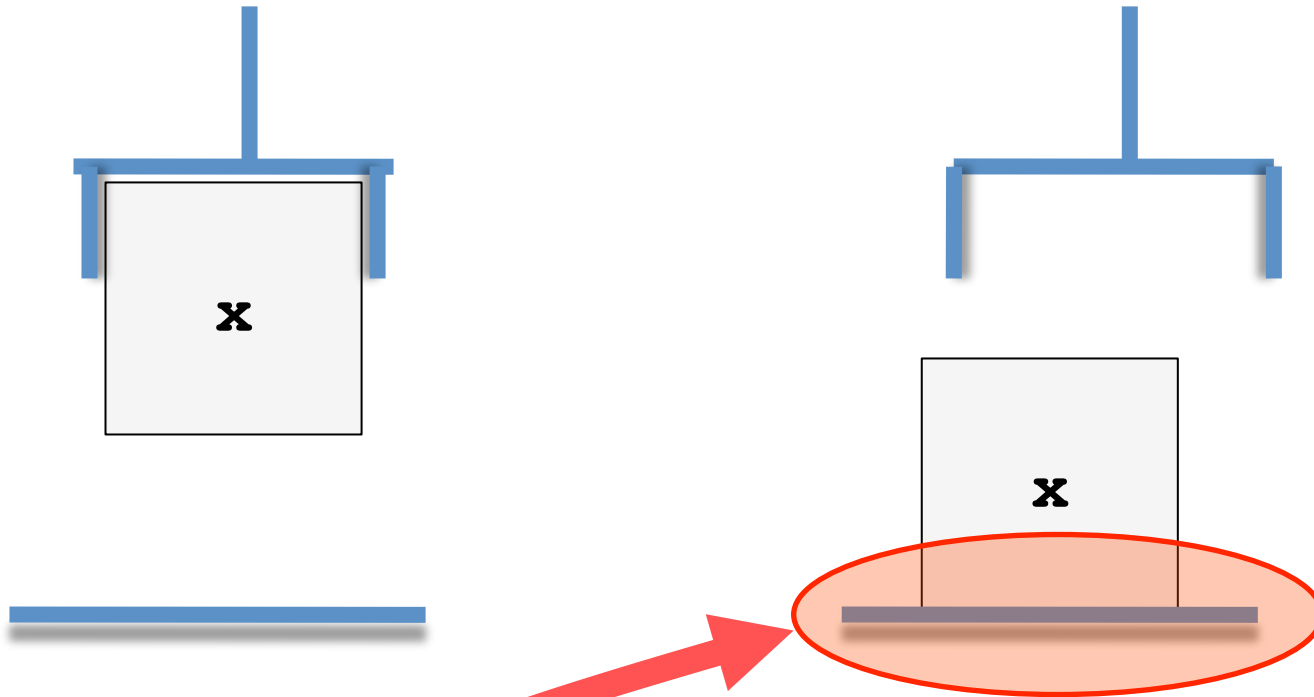
`putdown(x)`

P&D: **holding(x)**

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



# Reguli STRIPS

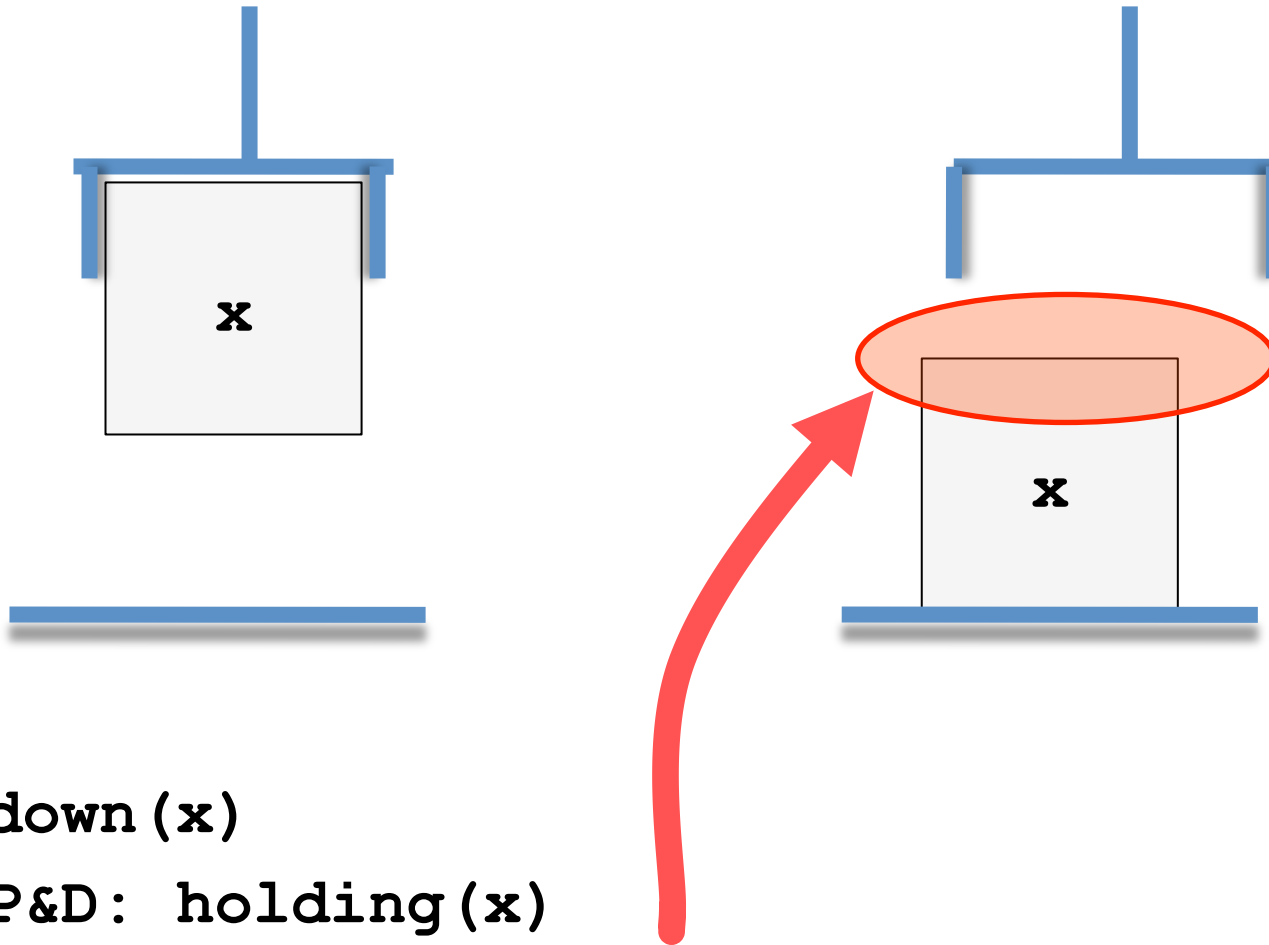


`putdown(x)`

P&D: `holding(x)`

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

# Reguli STRIPS

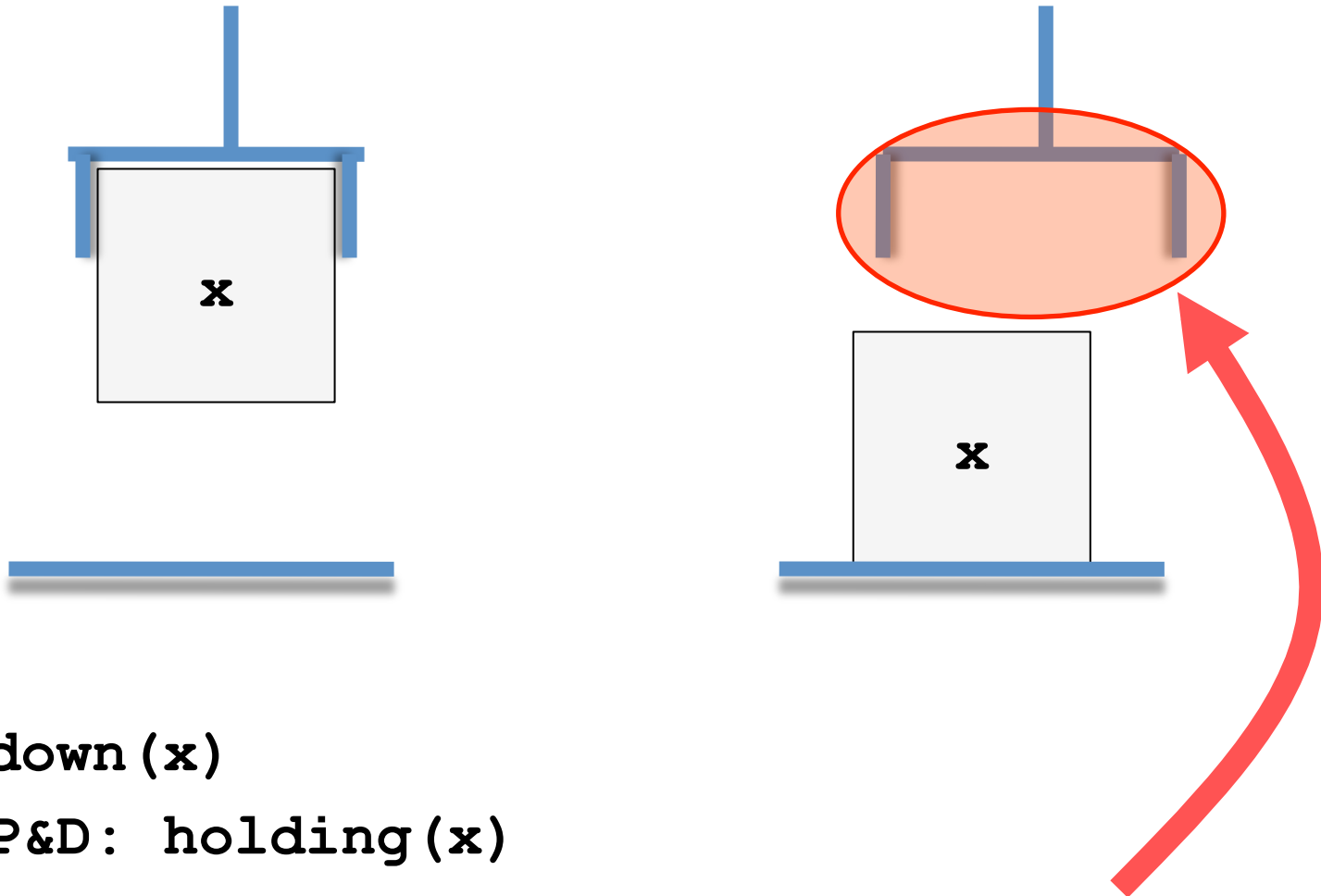


`putdown(x)`

P&D: `holding(x)`

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

# Reguli STRIPS

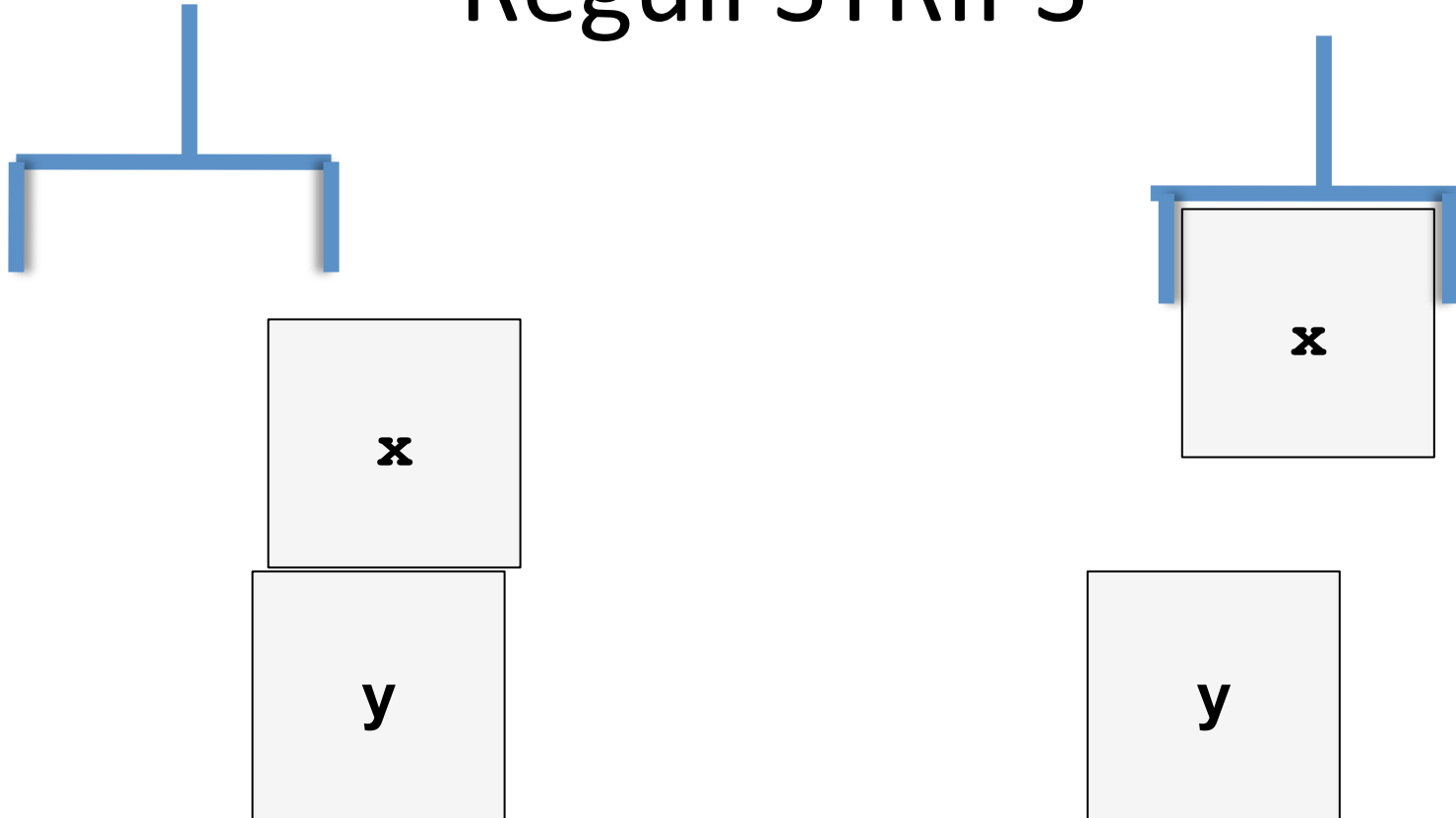


`putdown(x)`

P&D: `holding(x)`

A: `ontable(x)`, `clear(x)`, **`handempty`**

# Reguli STRIPS

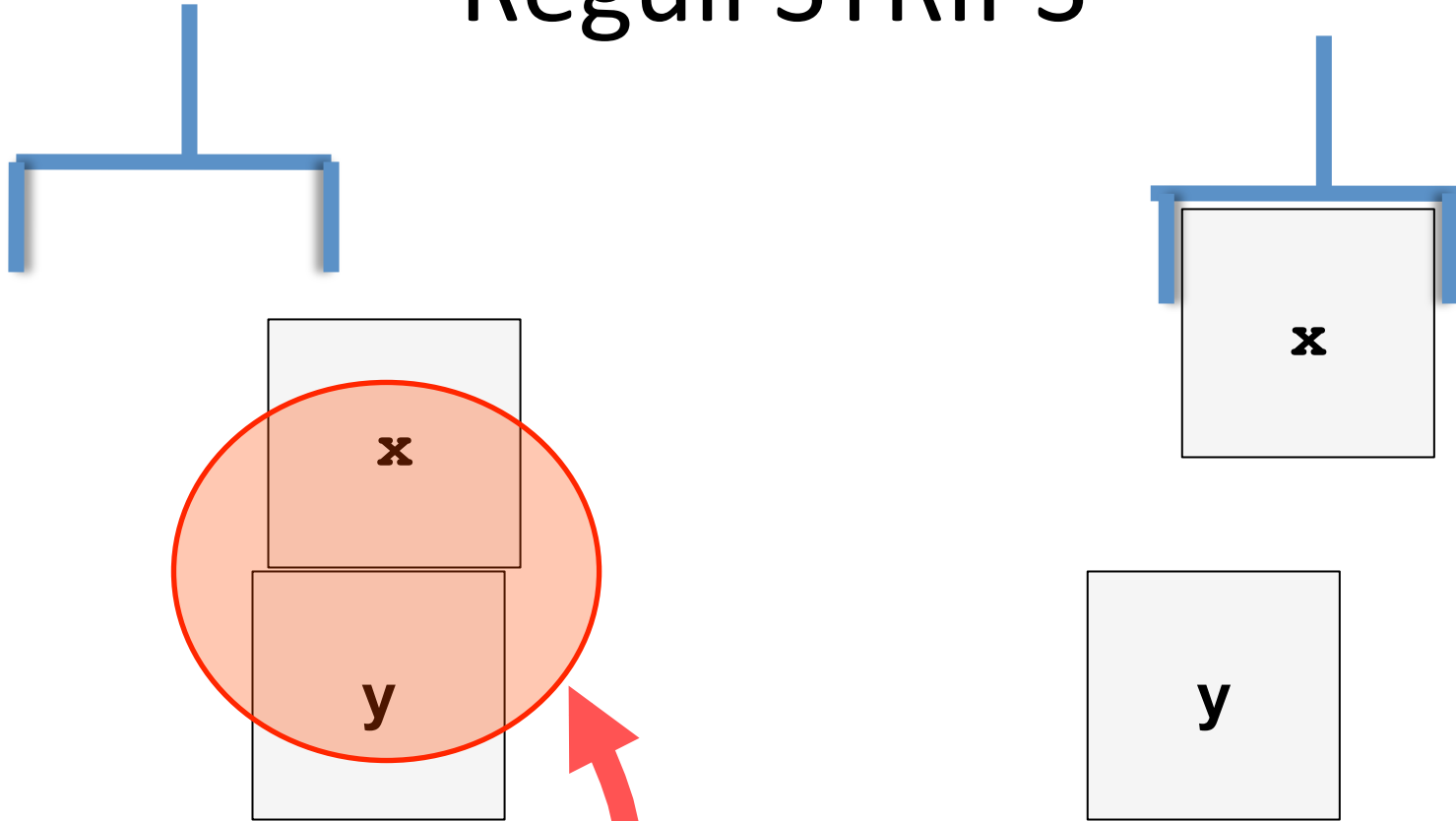


`unstack(x,y)`

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

A: `holding(x)` , `clear(y)`

# Reguli STRIPS

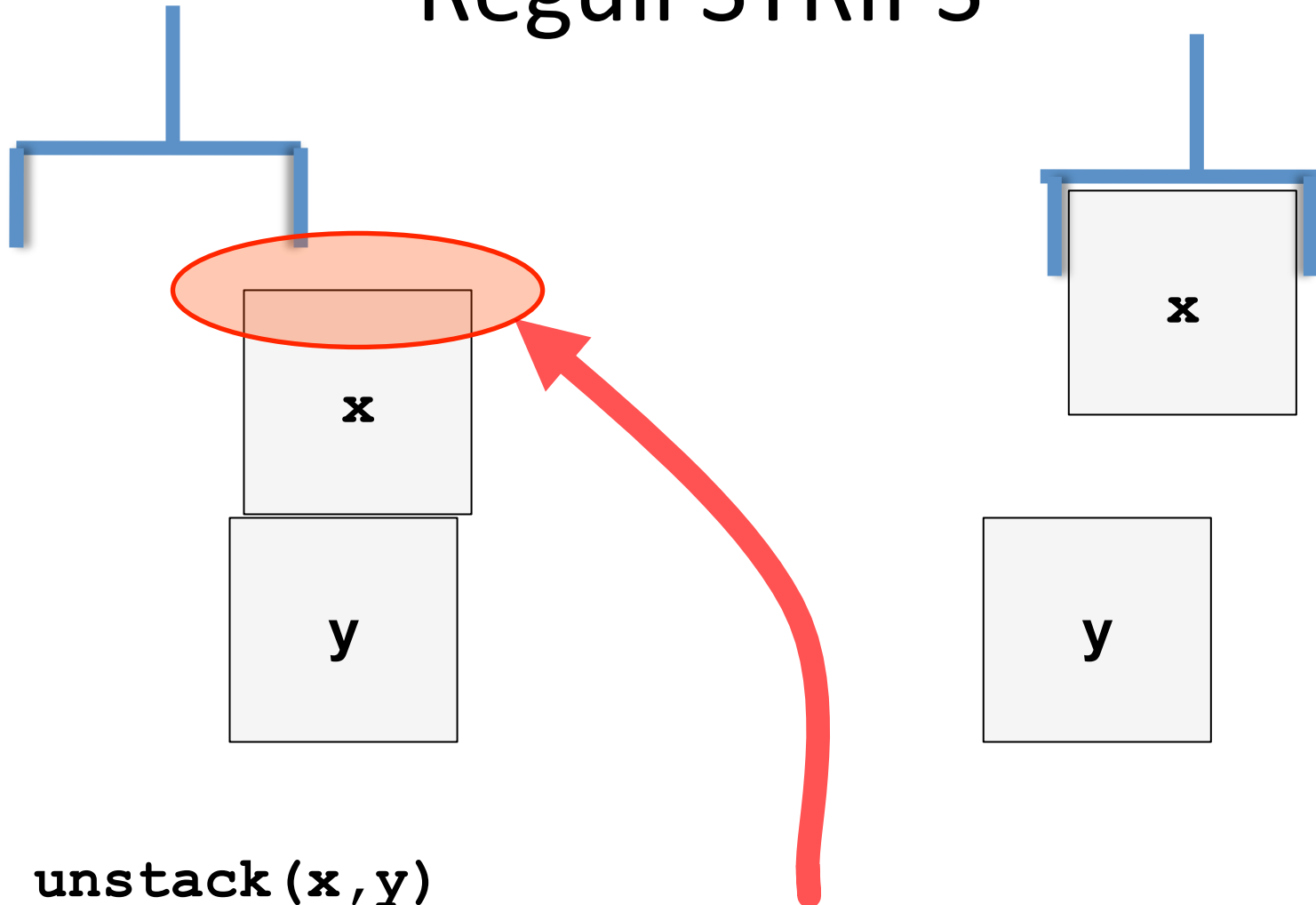


`unstack(x, y)`

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

A: `holding(x)`, `clear(y)`

# Reguli STRIPS

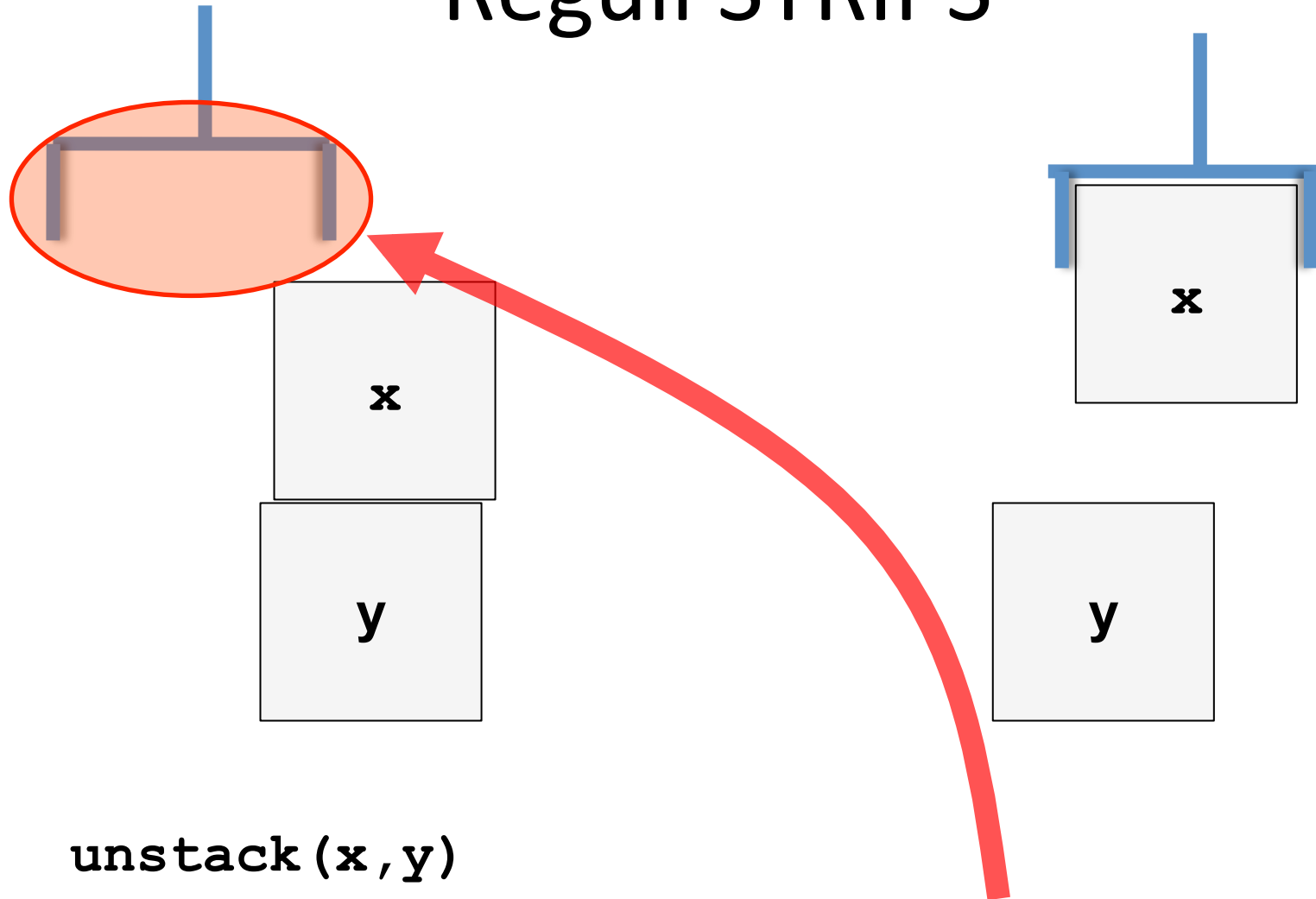


`unstack(x, y)`

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

A: `holding(x)` , `clear(y)`

# Reguli STRIPS

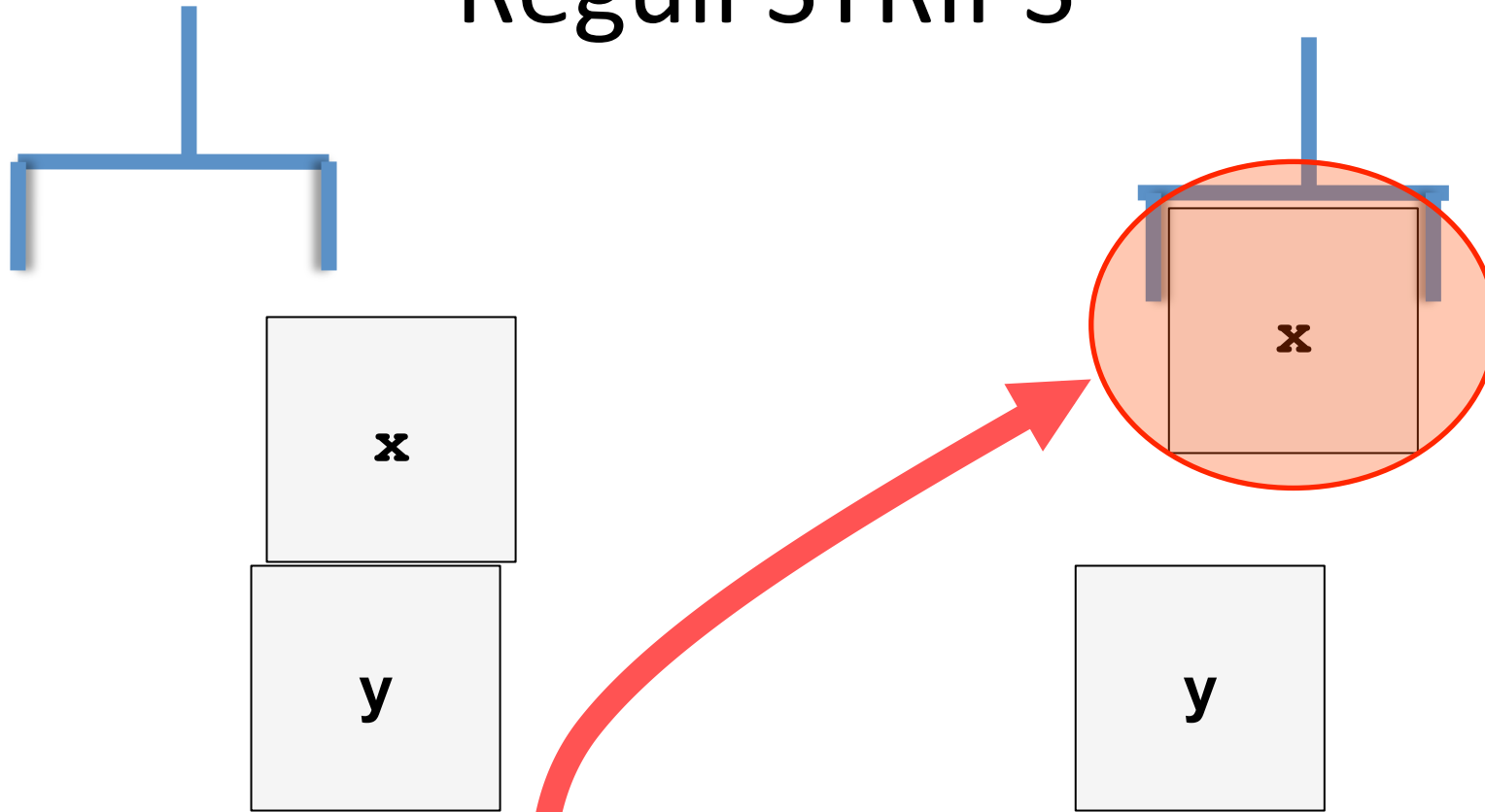


`unstack(x,y)`

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

A: `holding(x)` , `clear(y)`

# Reguli STRIPS



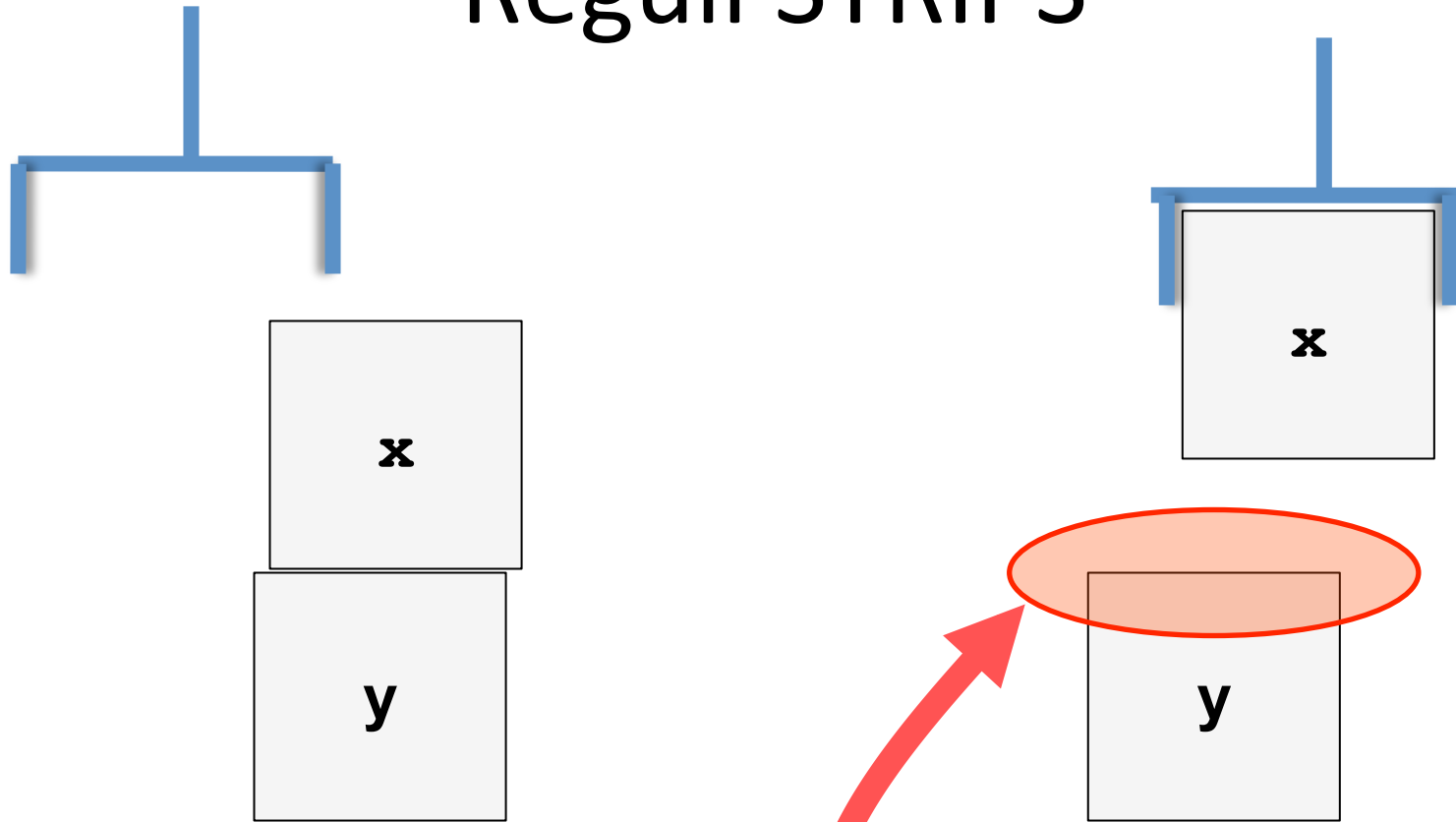
`unstack(x, y)`

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

A: **`holding(x)`** , `clear(y)`



# Reguli STRIPS

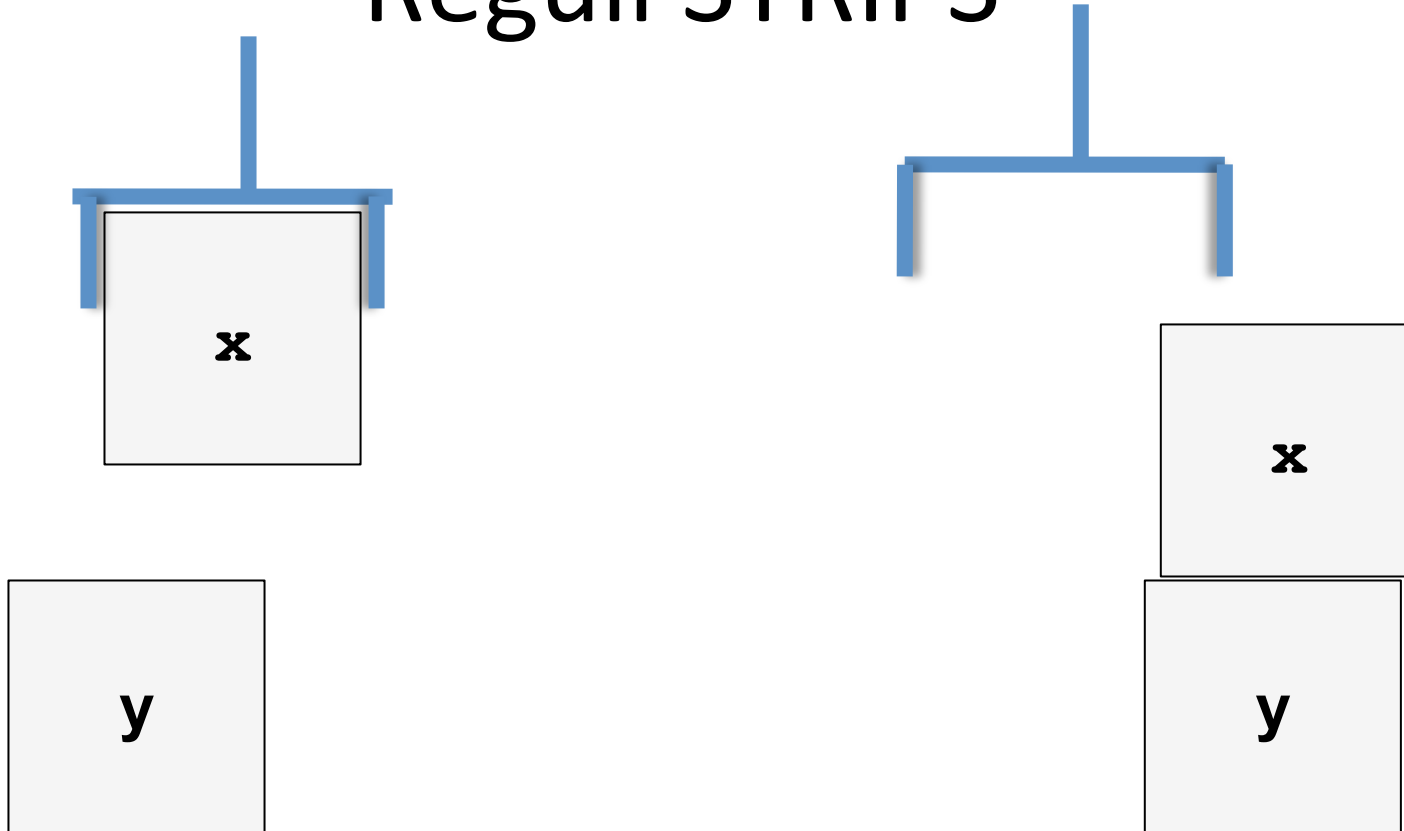


`unstack(x, y)`

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

A: `holding(x)` , `clear(y)`

# Reguli STRIPS

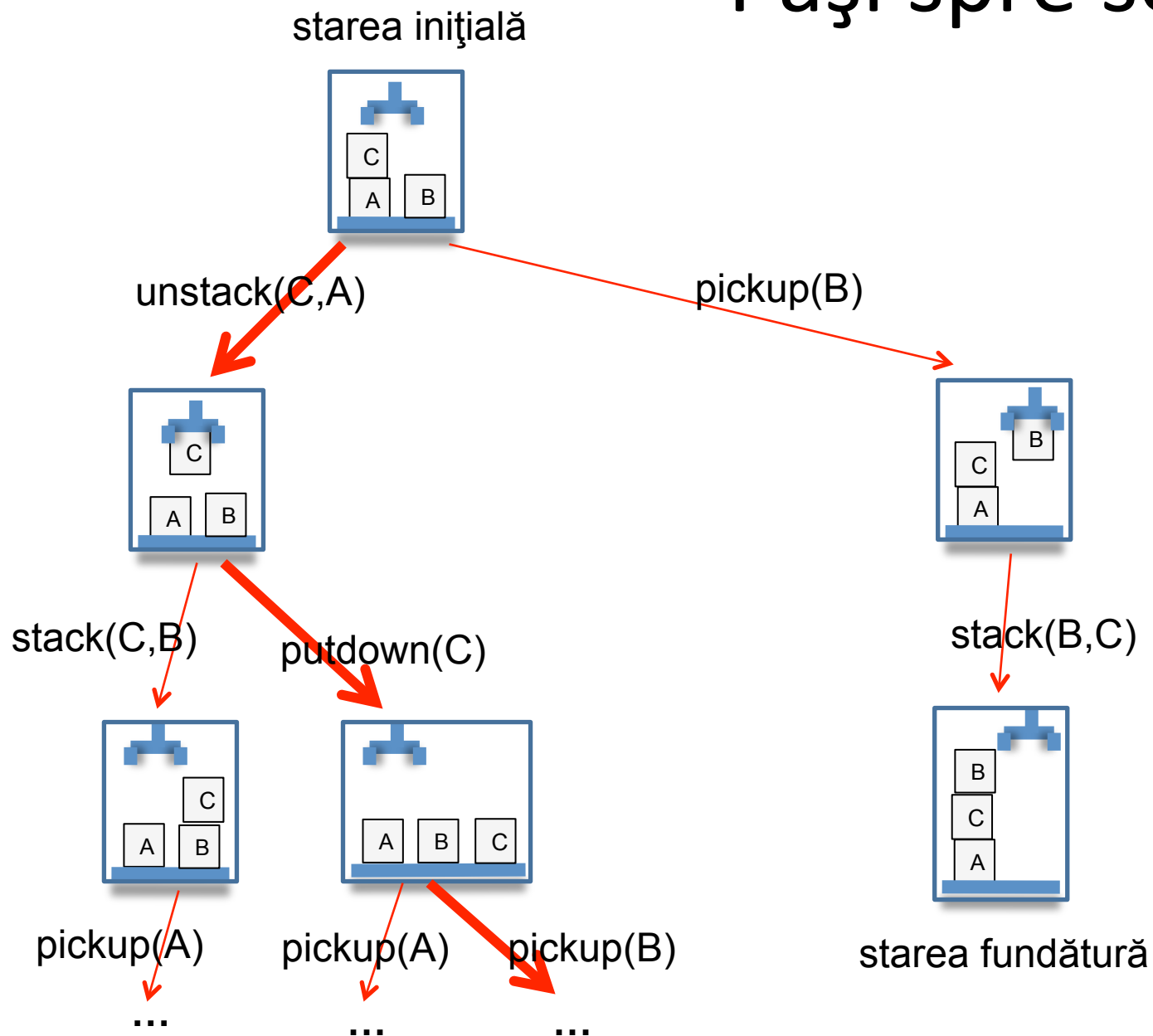


`stack(x,y)`

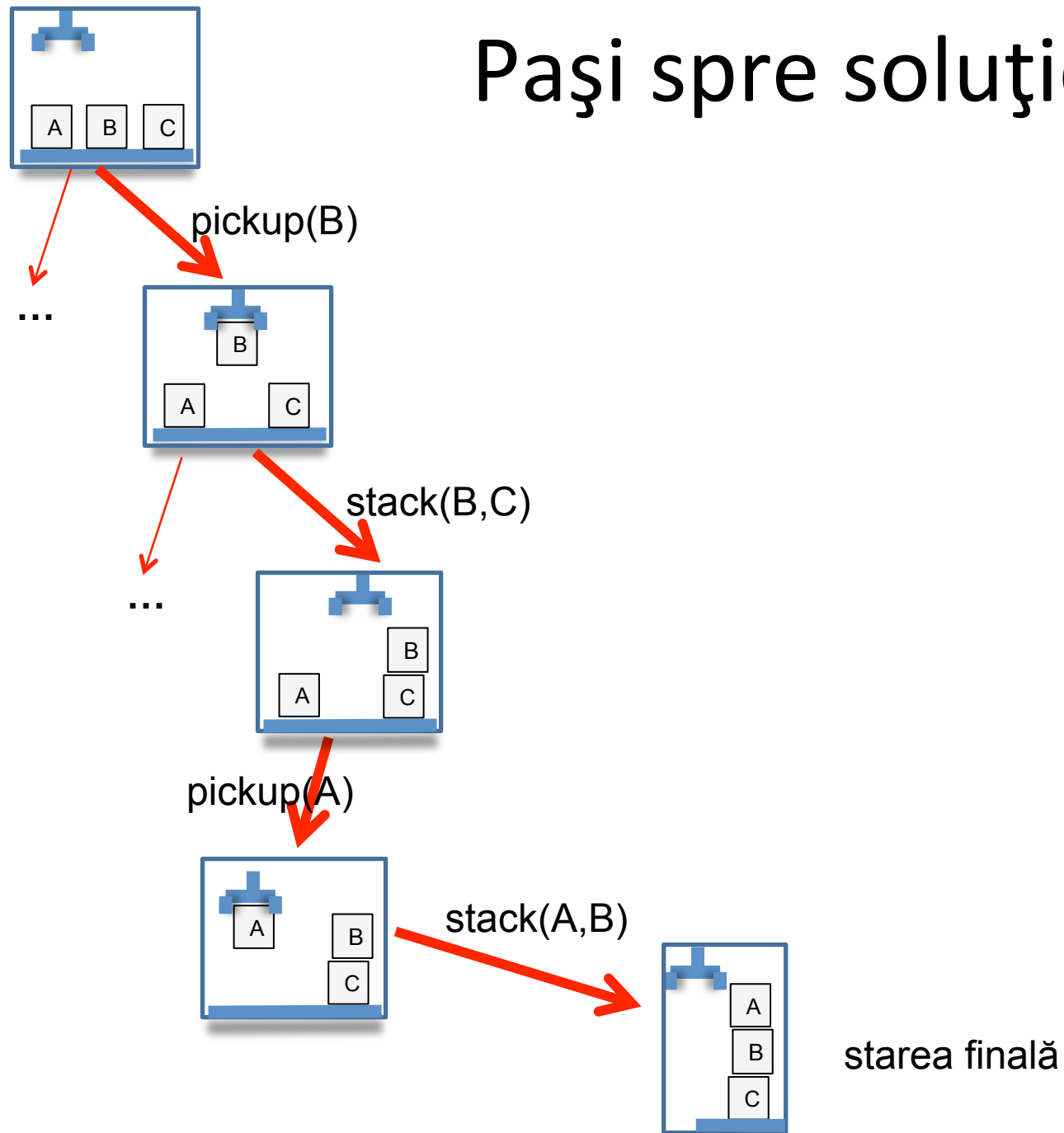
P&D: `holding(x)` , `clear(y)`

A: `on(x,y)` , `handempty` , `clear(x)`

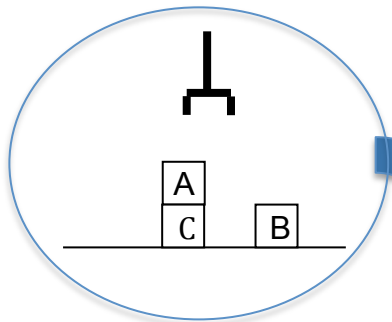
# Pași spre soluție



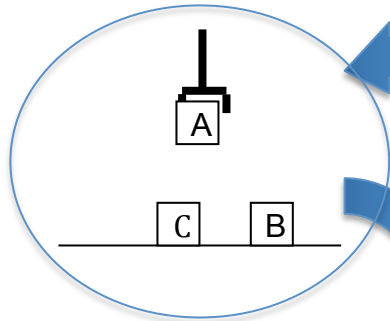
# Pași spre soluție



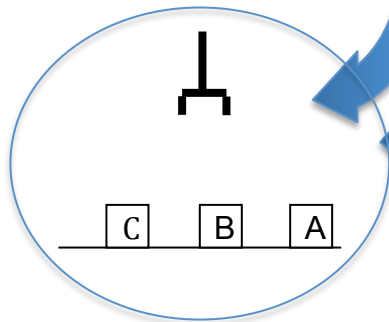
Starea inițială



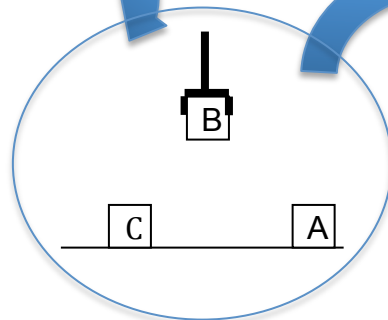
1: unstack(A,C)



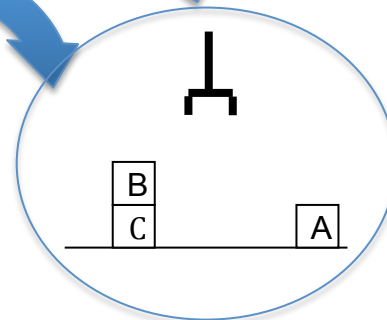
2: putdown(A)



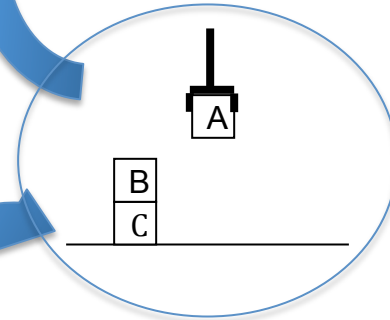
3: pickup(B)



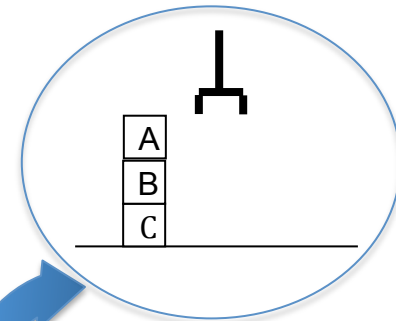
4: stack(B,C)



5: pickup(A)



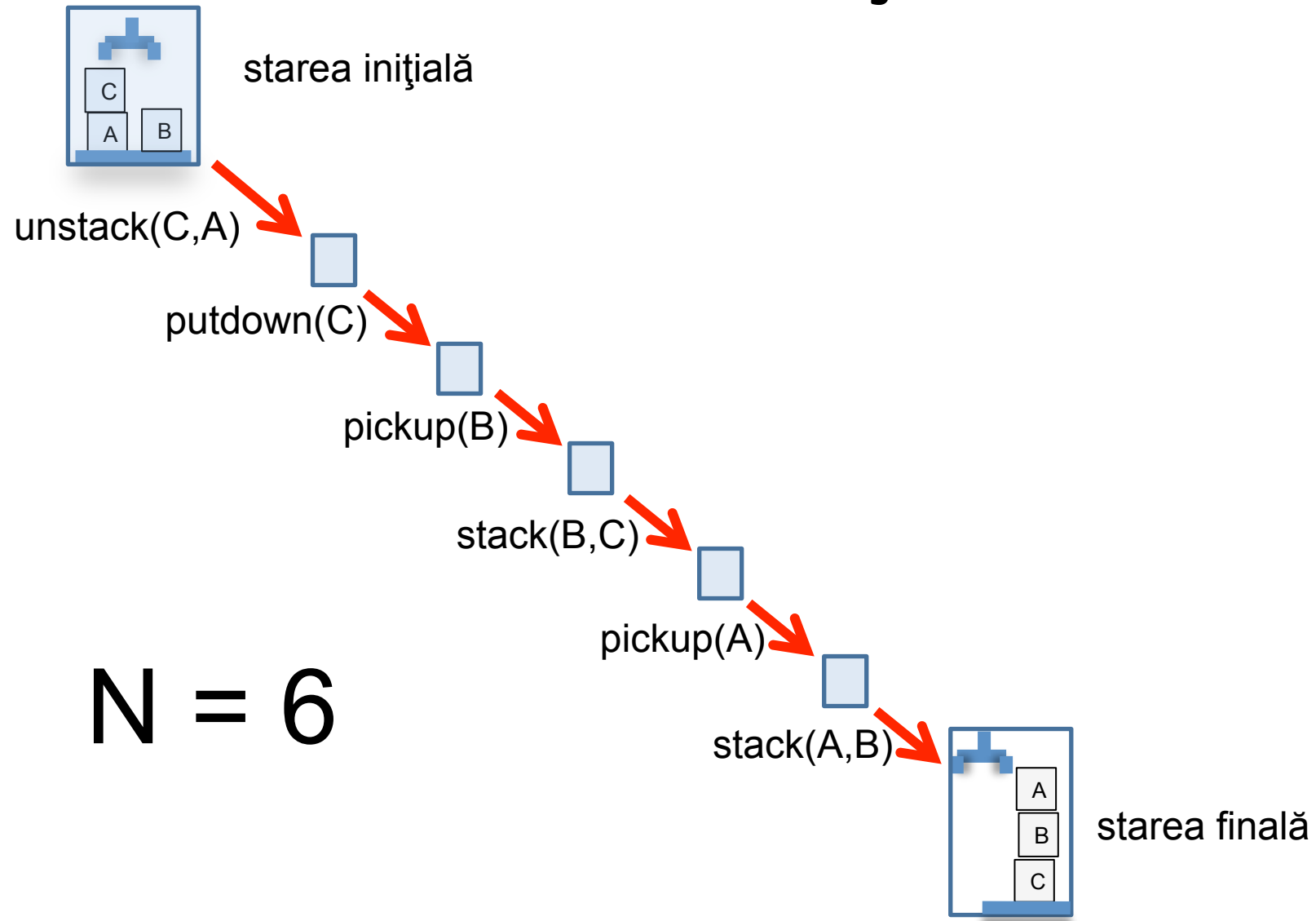
6: stack(A,B)



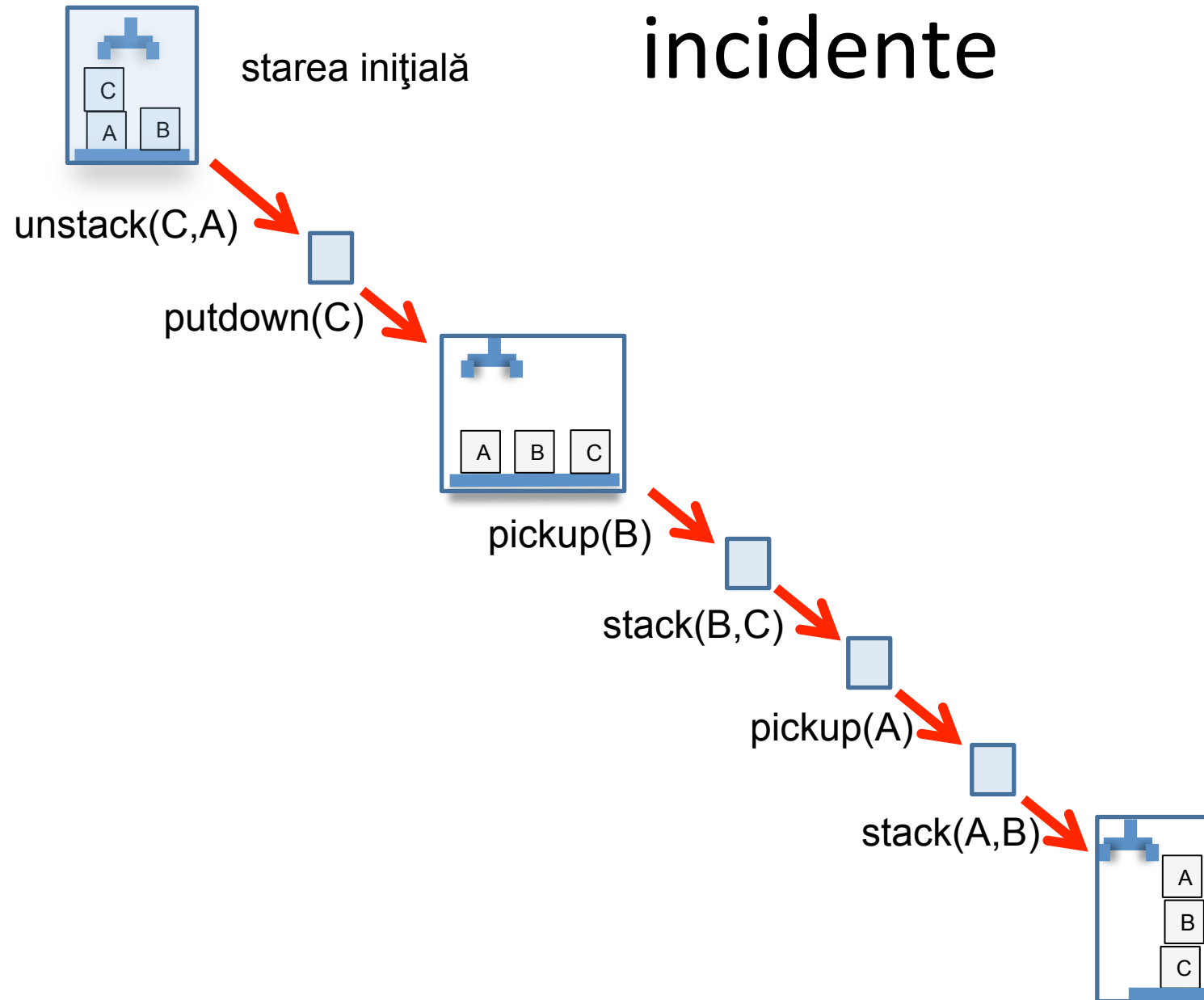
Starea finală

0 soluție

# Soluția



# Urmărirea execuției planului - incidente



# Urmărirea planului - incidente

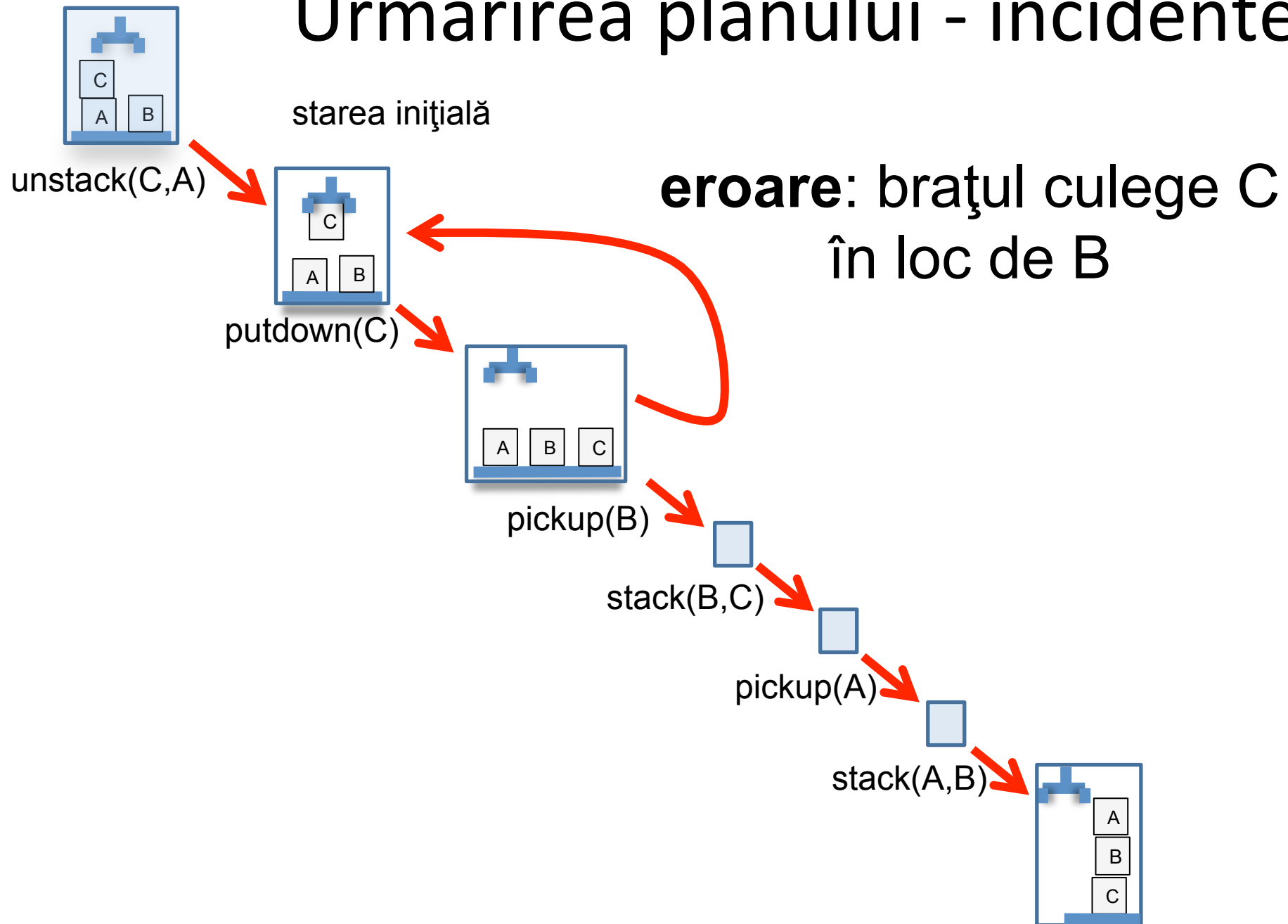
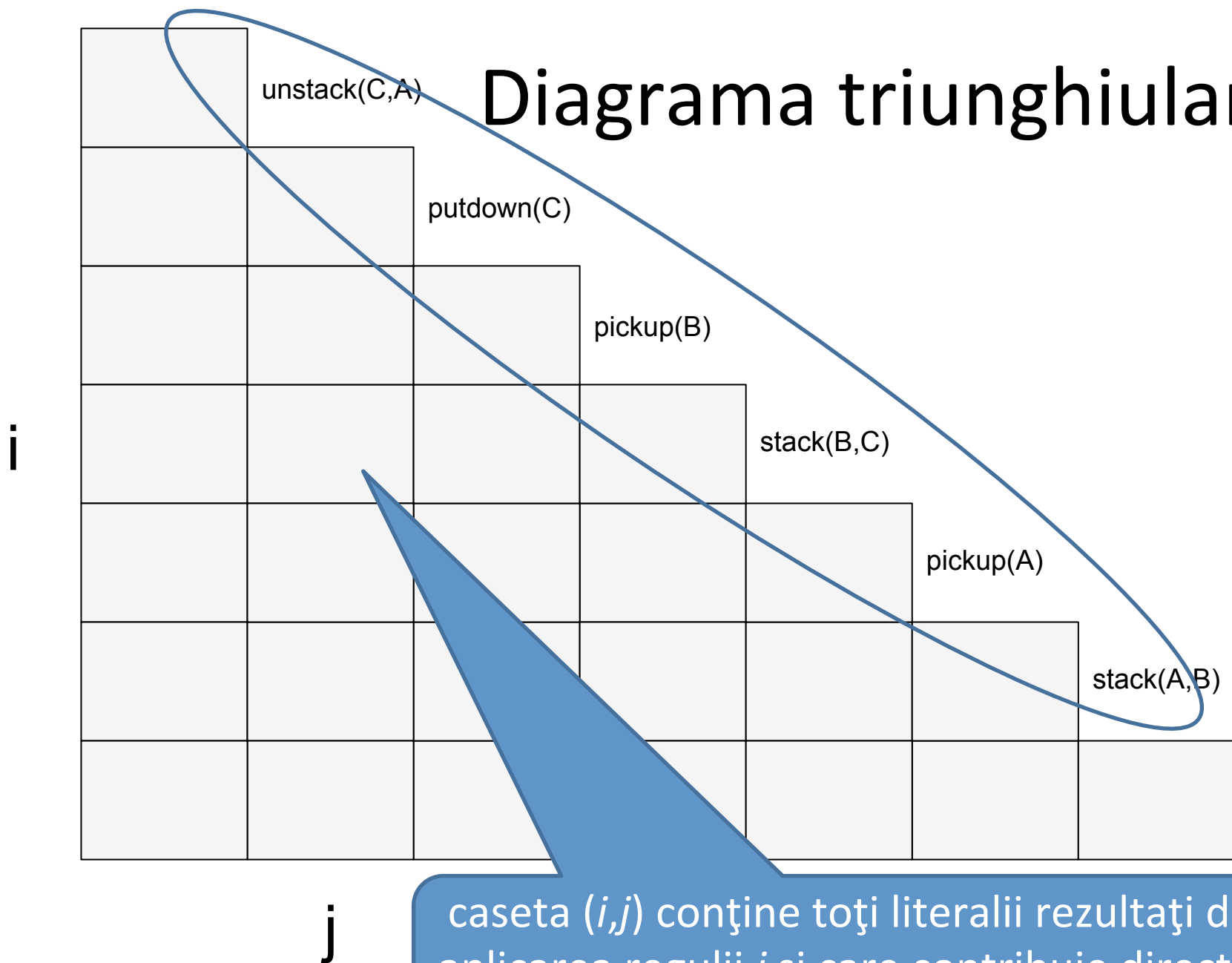




Diagram illustrating a triangular grid structure, likely representing a state space or a sequence of actions. The grid is composed of 7 rows and 7 columns of cells. A blue path is shown, starting from the top-left cell and ending at the bottom-right cell. The path is labeled with actions: unstack(C,A), putdown(C), pickup(B), stack(B,C), pickup(A), and stack(A,B). A blue arrow on the left indicates a height of 1, and a blue arrow at the bottom indicates a width of 1.

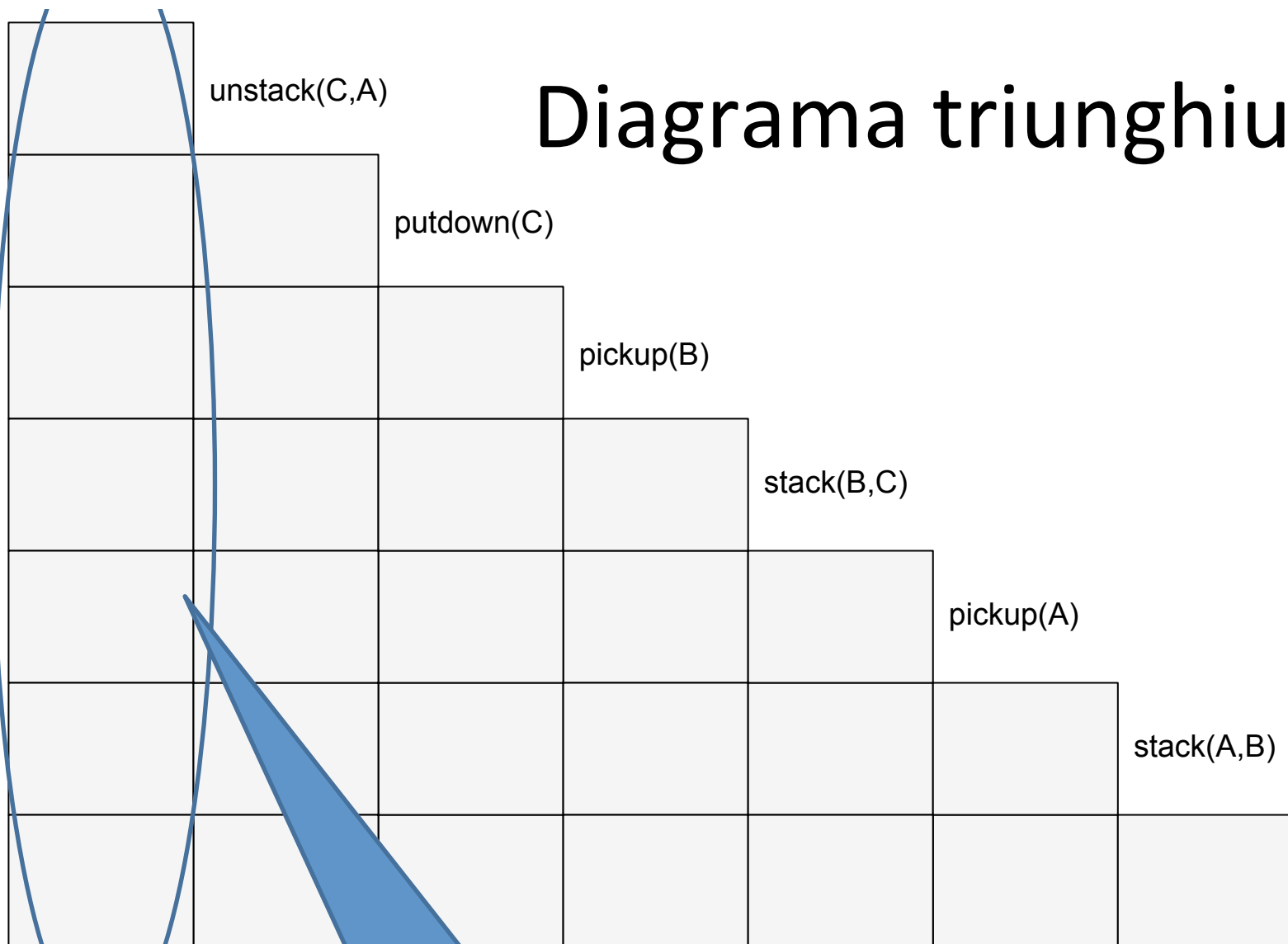
**soluția**

# Diagrama triunghiulară



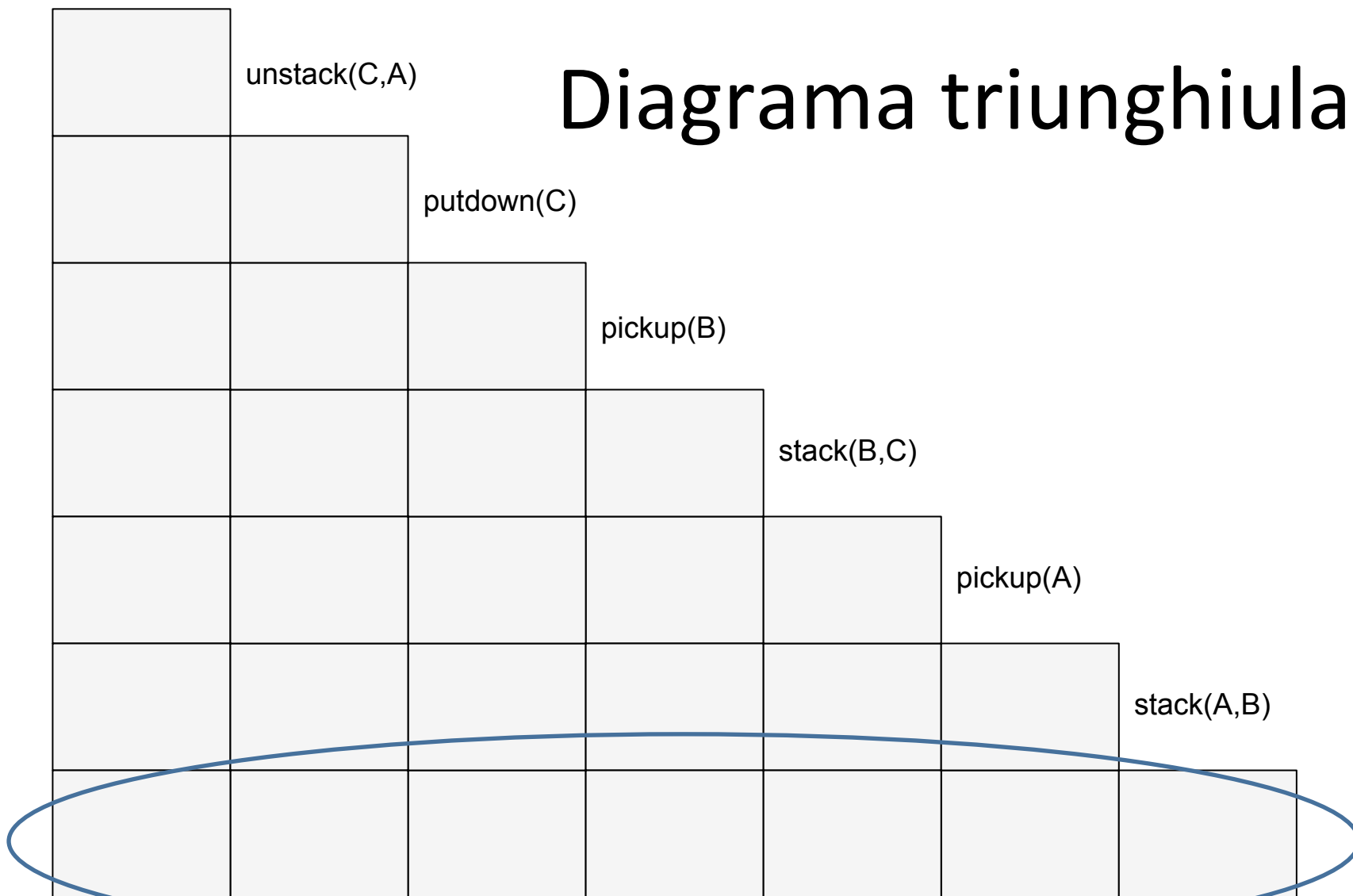
caseta  $(i,j)$  conține toți literalii rezultați din aplicarea regulii  $j$  și care contribuie direct la realizare unei condiții a pasului  $i$

# Diagrama triunghiulară



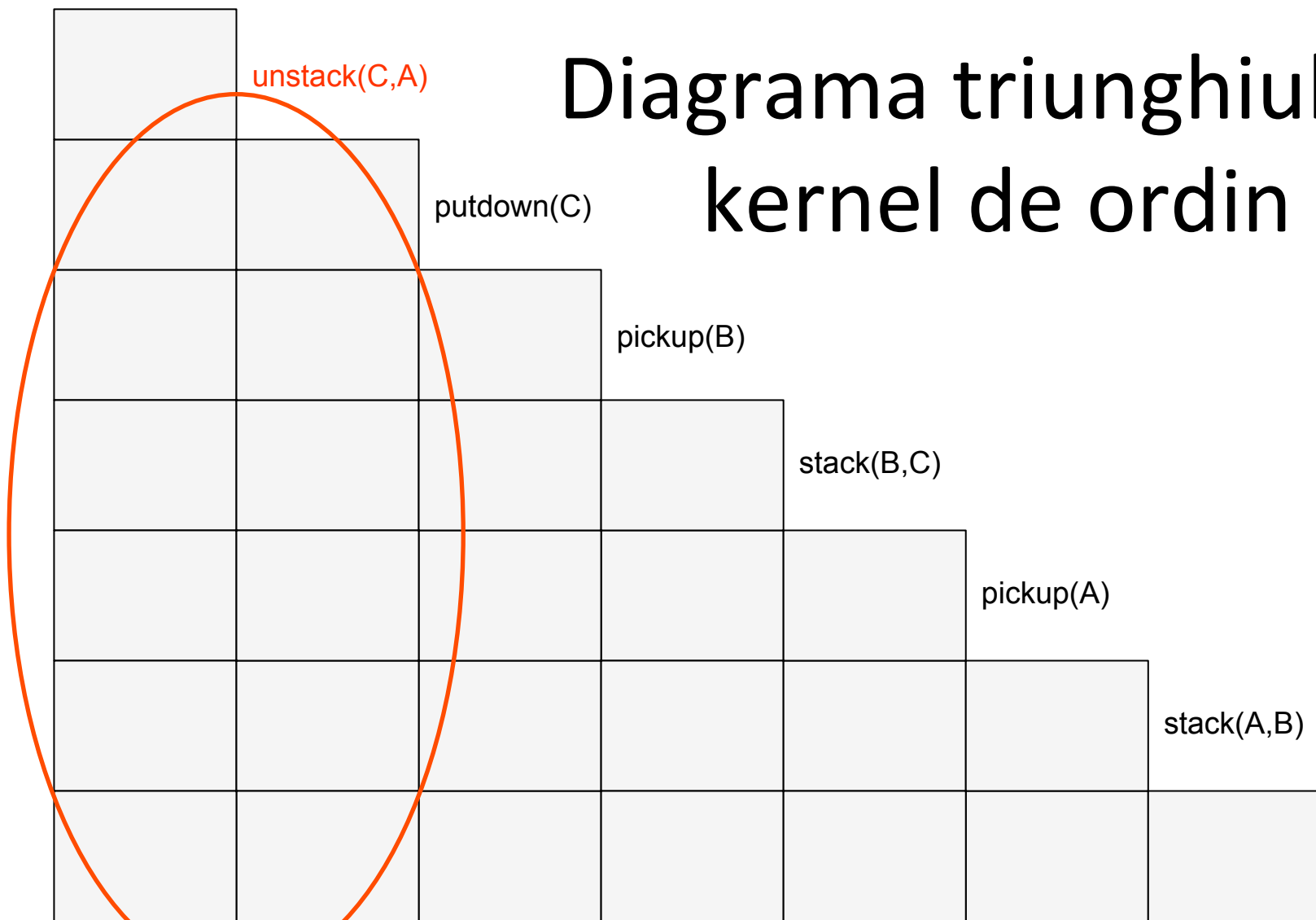
prima coloană va  
colecta predicatele  
stării inițiale

# Diagrama triunghiulară



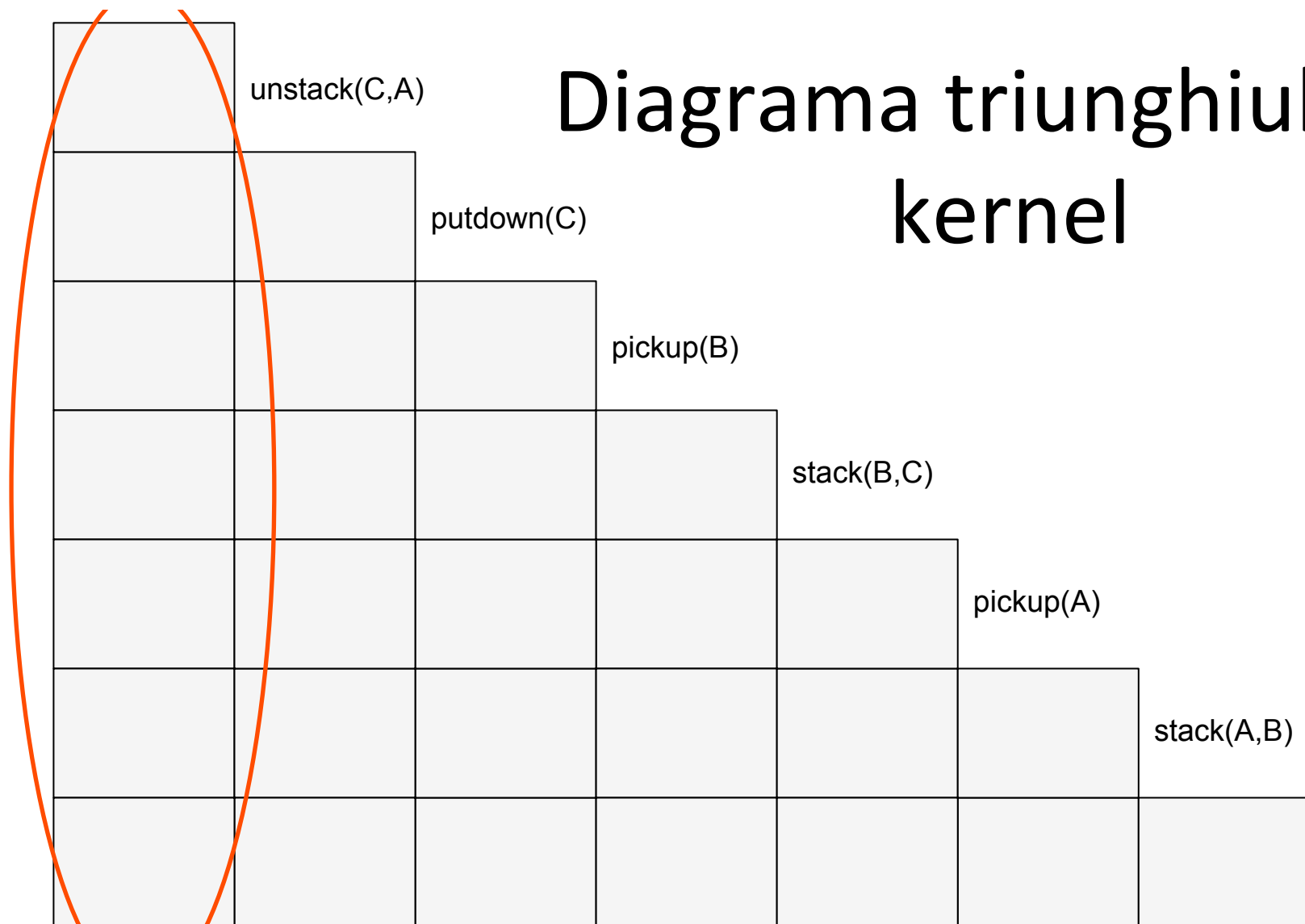
ultima linie va colecta predicatele  
stării finale

# Diagrama triunghiulară: kernel de ordin $i$



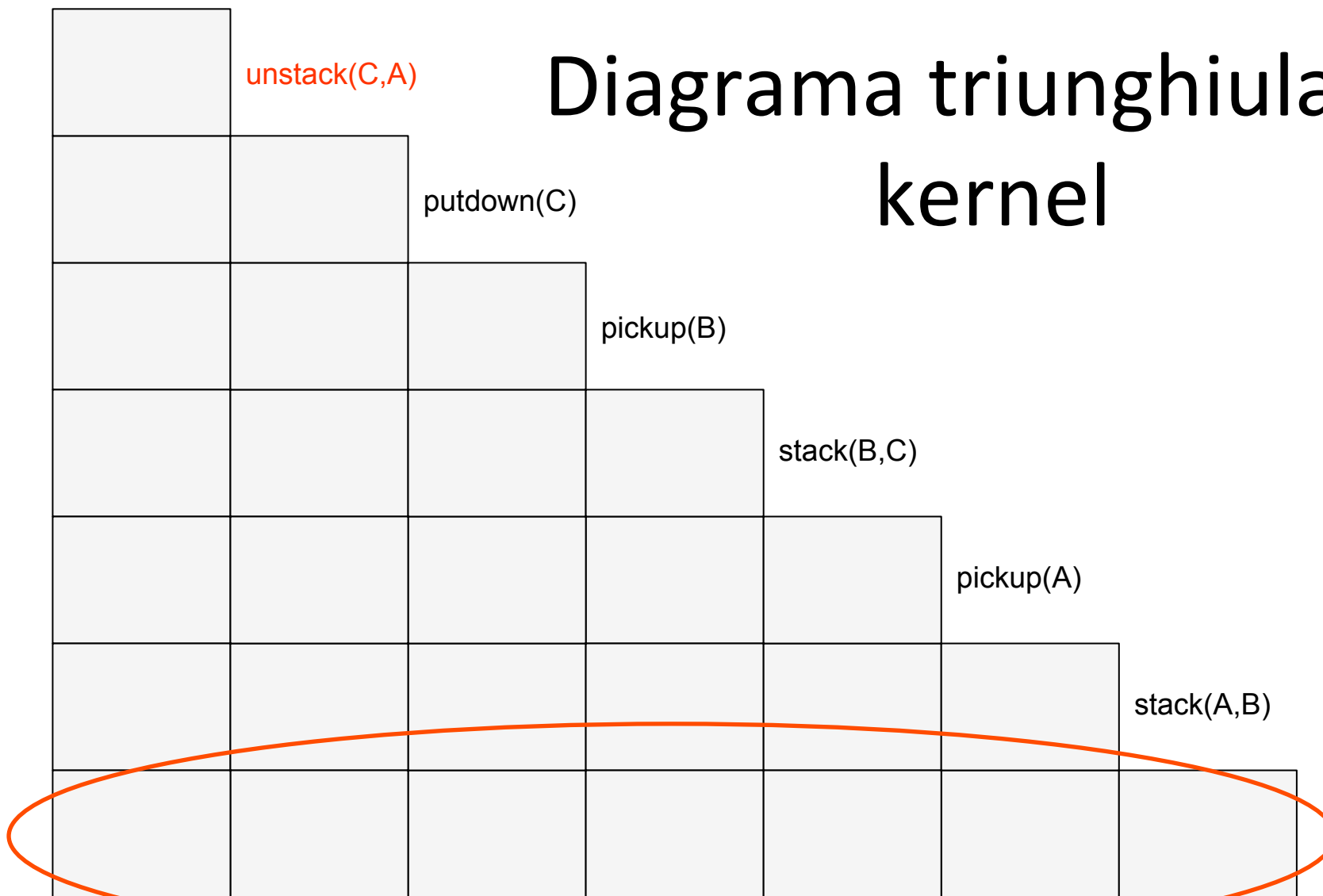
Mulțimea predicatelor din oricare dreptunghi  $[i,N] \times [1,j]$ : starea de după execuția regulii  $i$

# Diagrama triunghiulară: kernel



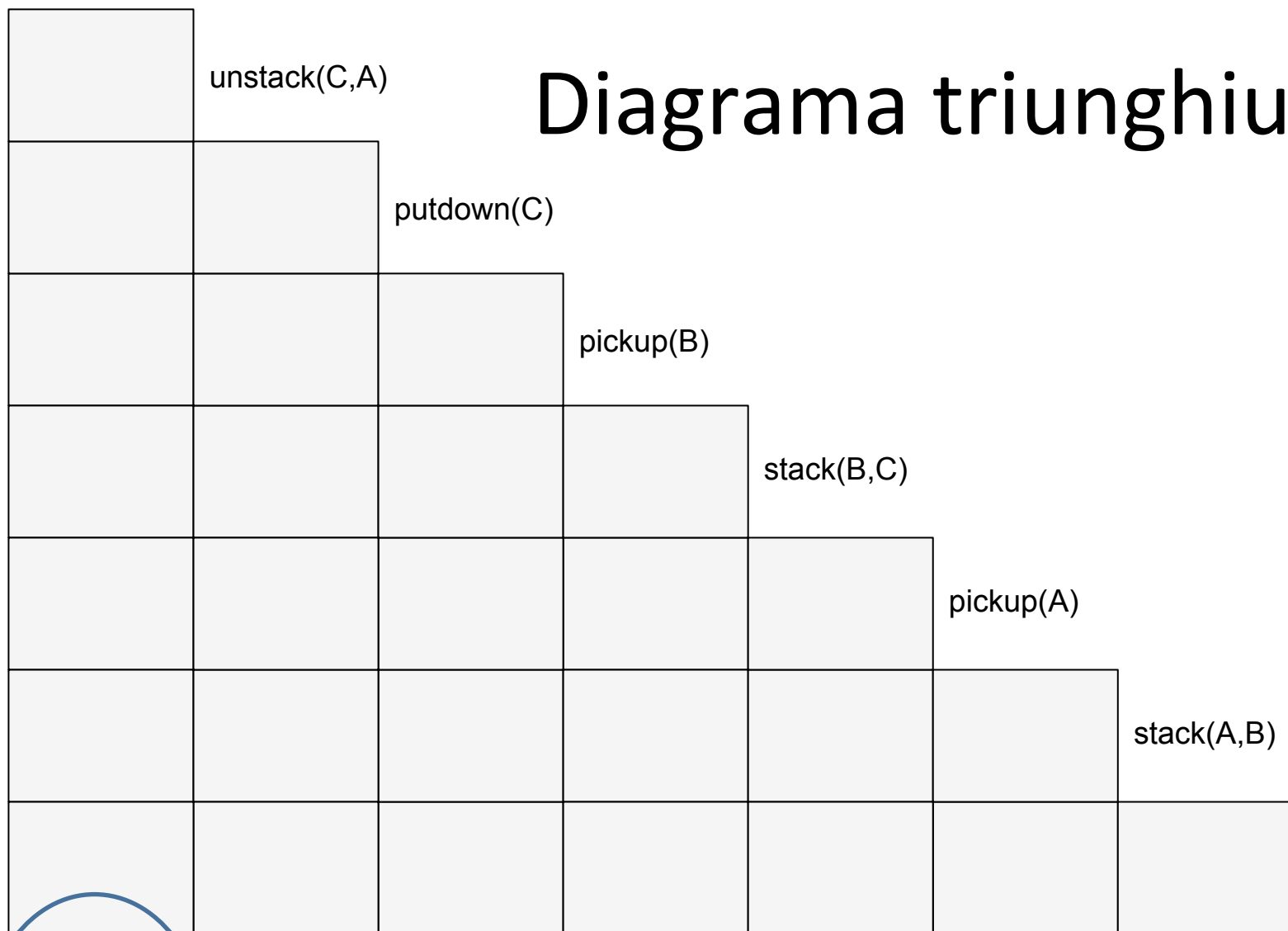
Starea inițială: kernul de ordin 0

# Diagrama triunghiulară: kernel



Starea finală: kernel de ordin N

# Diagrama triunghiulară

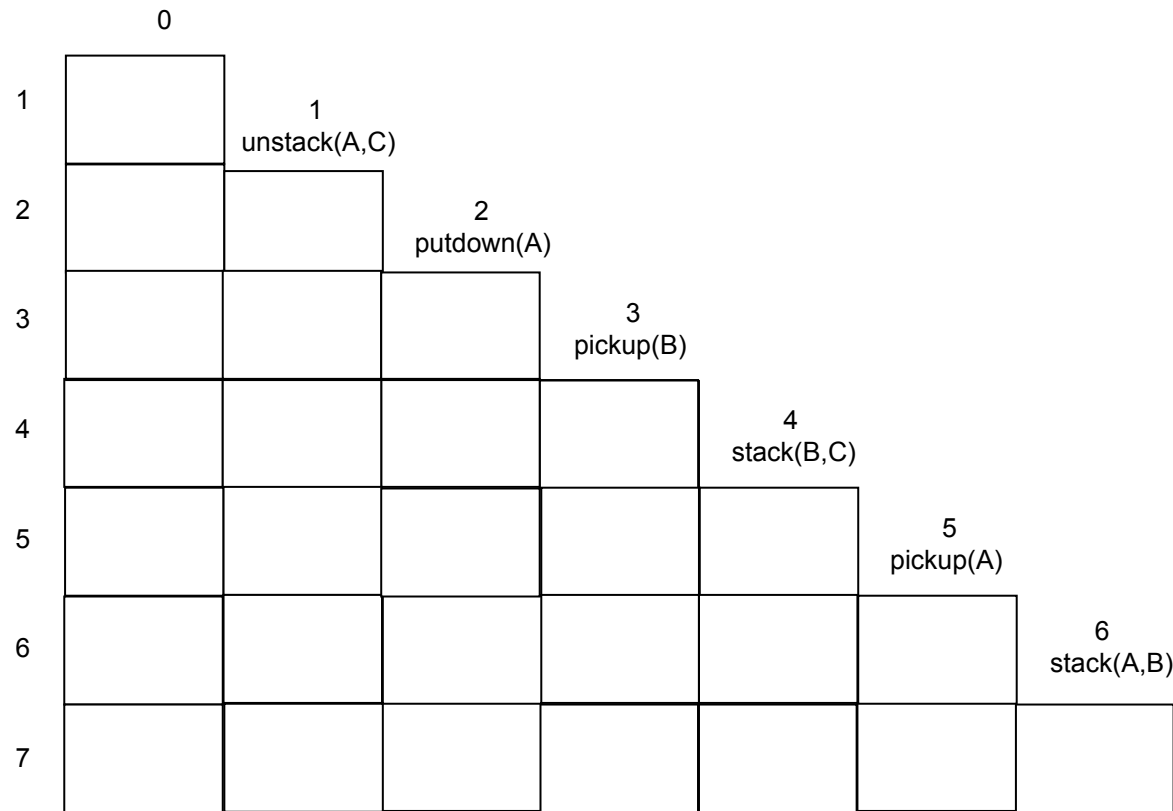


handempty  
ontable(A)  
ontable(B)  
on(C,A)  
clear (C)  
clear(B)

starea inițială e  
adunată în josul  
primei coloane



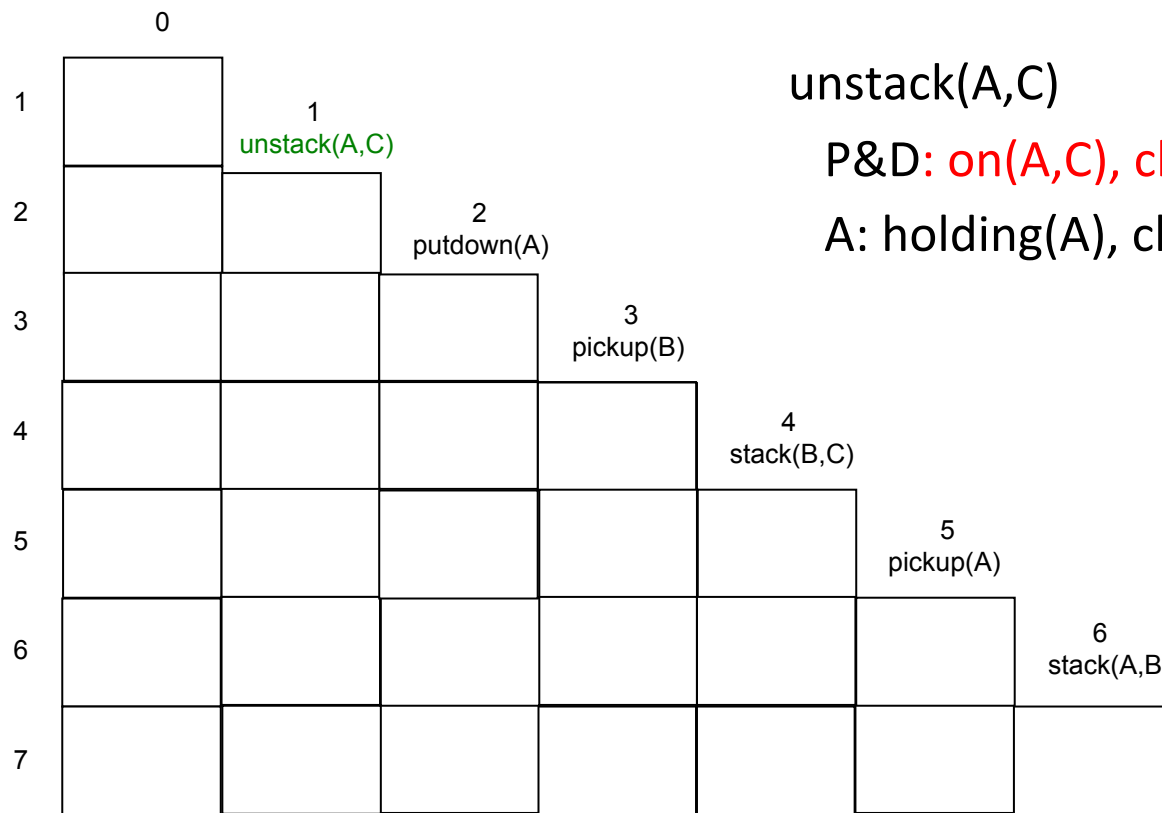
# Diagrama triunghiulară: inițializarea



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

Starea  
inițială

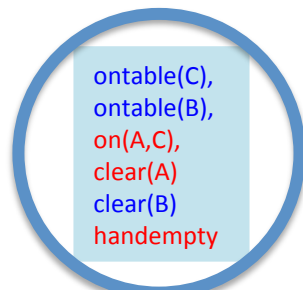
# Execuția planului: pasul 1



unstack(A,C)

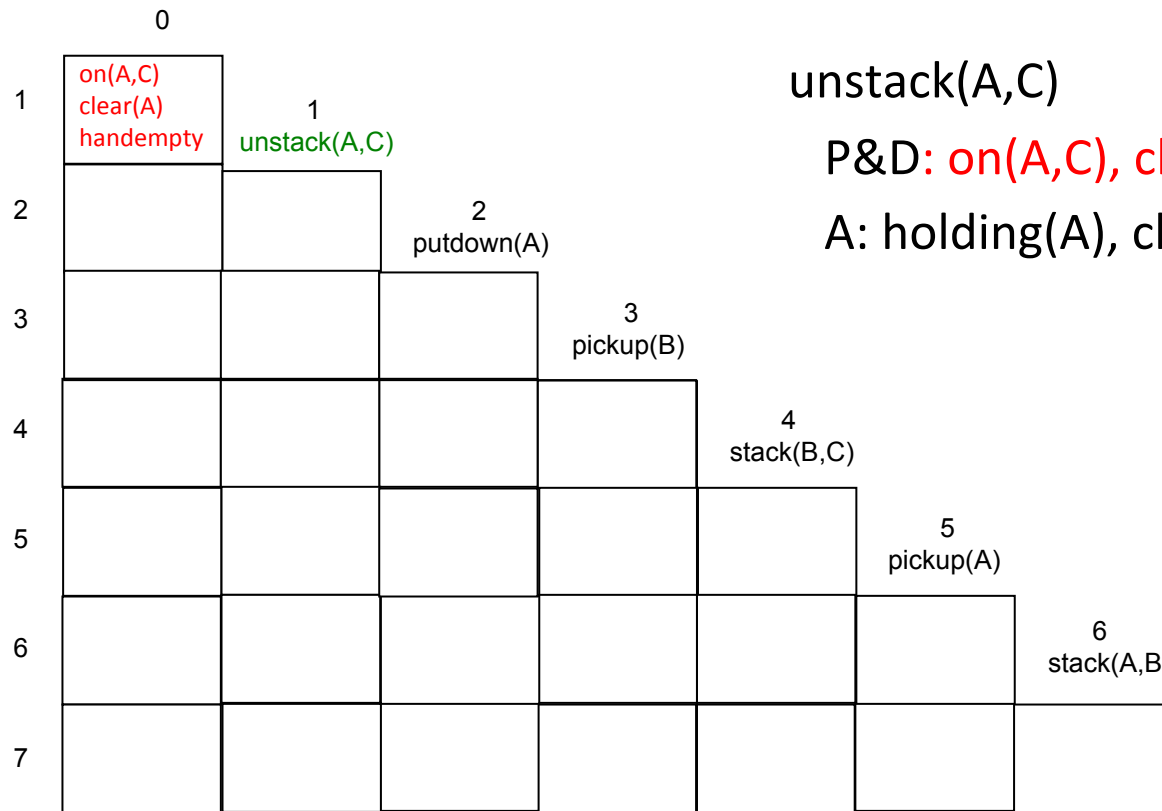
P&D: on(A,C), clear(A), handempty

A: holding(A), clear(C)



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

# Execuția planului: pasul 1



unstack(A,C)

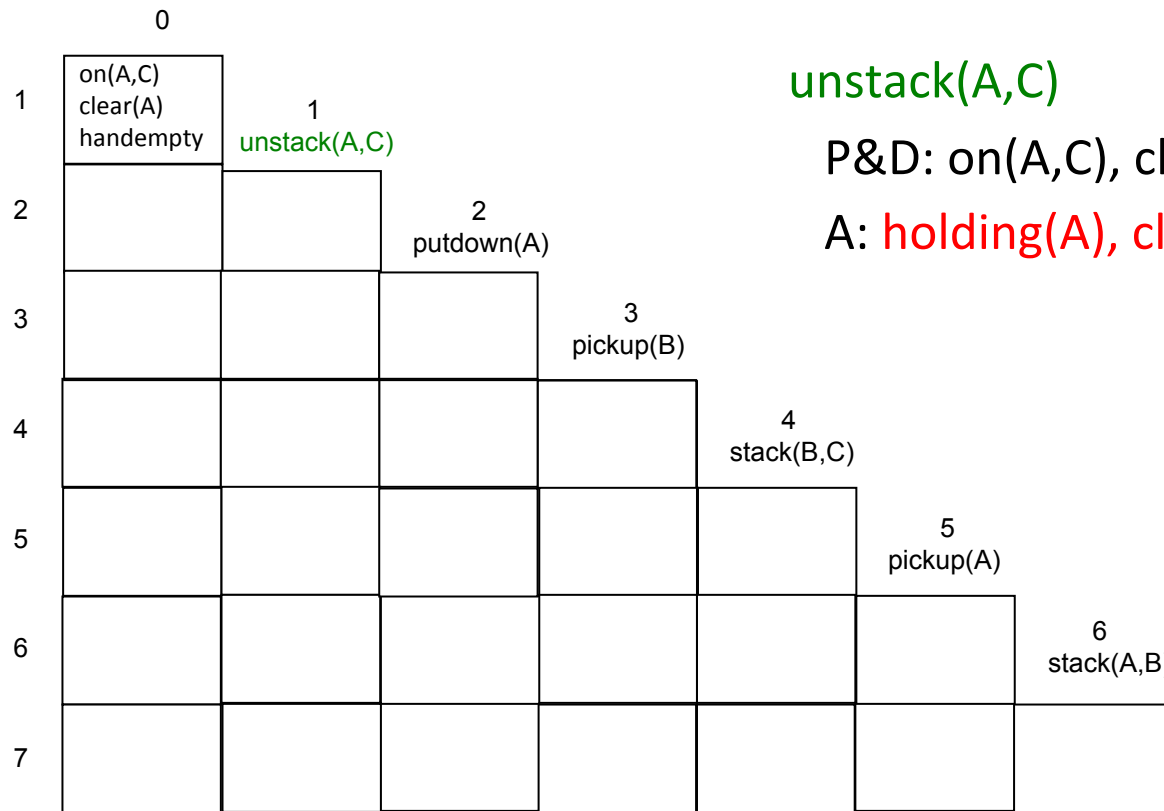
P&D: on(A,C), clear(A), handempty

A: holding(A), clear(C)

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

... și urcă-le în linia regulei 1

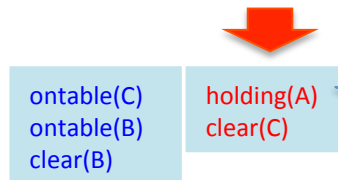
# Execuția planului: pasul 1



unstack(A,C)

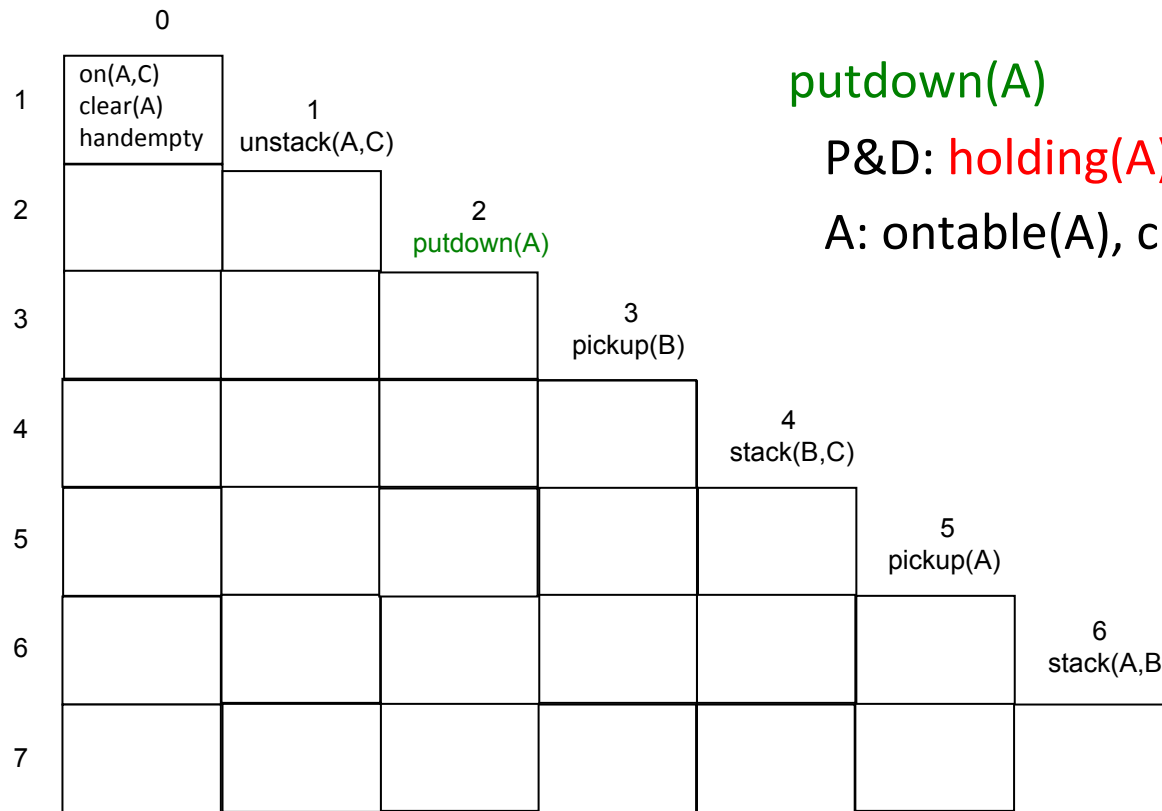
P&D: on(A,C), clear(A), handempty

A: holding(A), clear(C)



colectează predicatele listei A  
a regulii 1

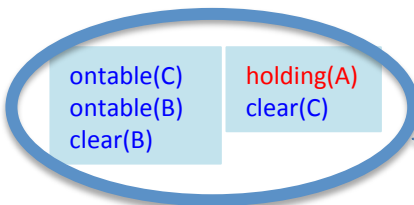
# Execuția planului: pasul 2



putdown(A)

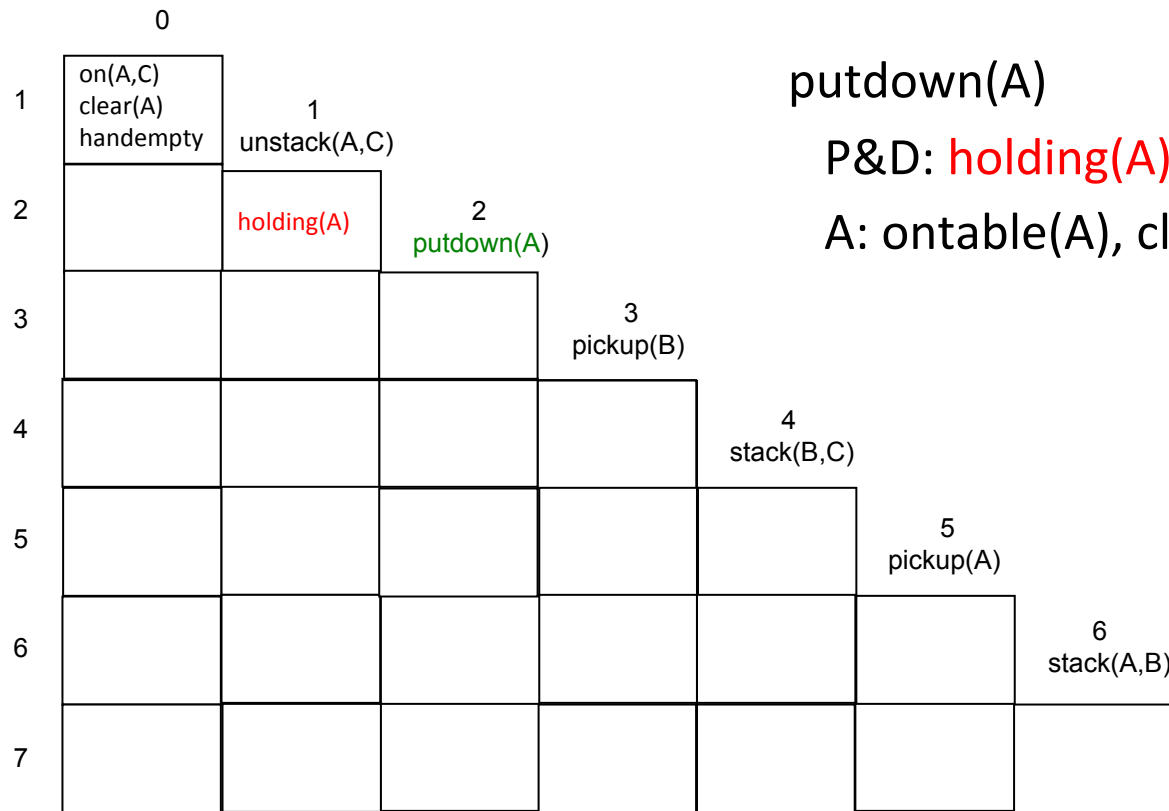
P&D: **holding(A)**

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



caută predicatele listei P&D  
a regulii 2

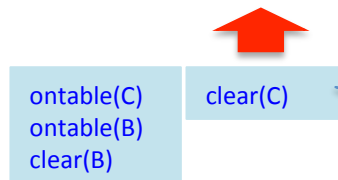
# Execuția planului: pasul 2



putdown(A)

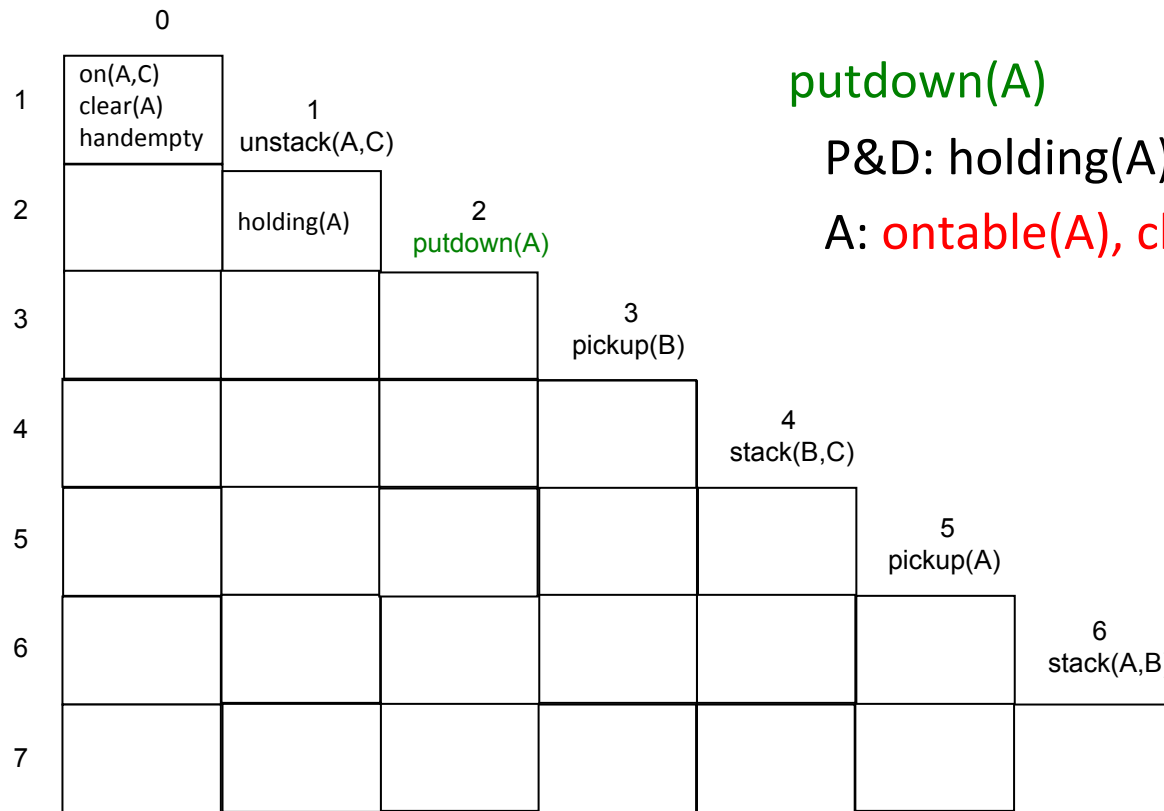
P&D: **holding(A)**

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



... și urcă-le în linia regulei 2

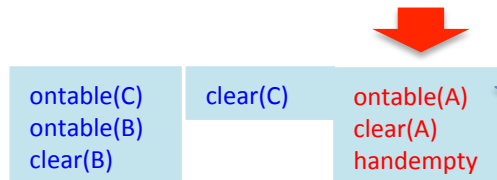
# Execuția planului: pasul 2



putdown(A)

P&D: holding(A)

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



colectează predicatele listei A  
a regulii 2

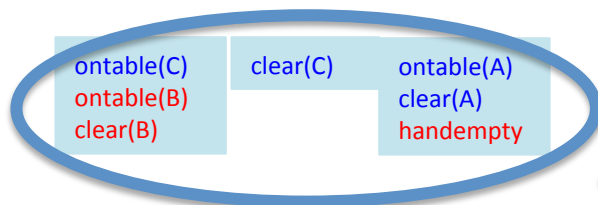
# Execuția planului: pasul 3

	0						
1	on(A,C) clear(A) handempty	1 unstack(A,C)					
2		holding(A)	2 putdown(A)				
3				3 pickup(B)			
4					4 stack(B,C)		
5						5 pickup(A)	
6							6 stack(A,B)
7							

pickup(B)

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

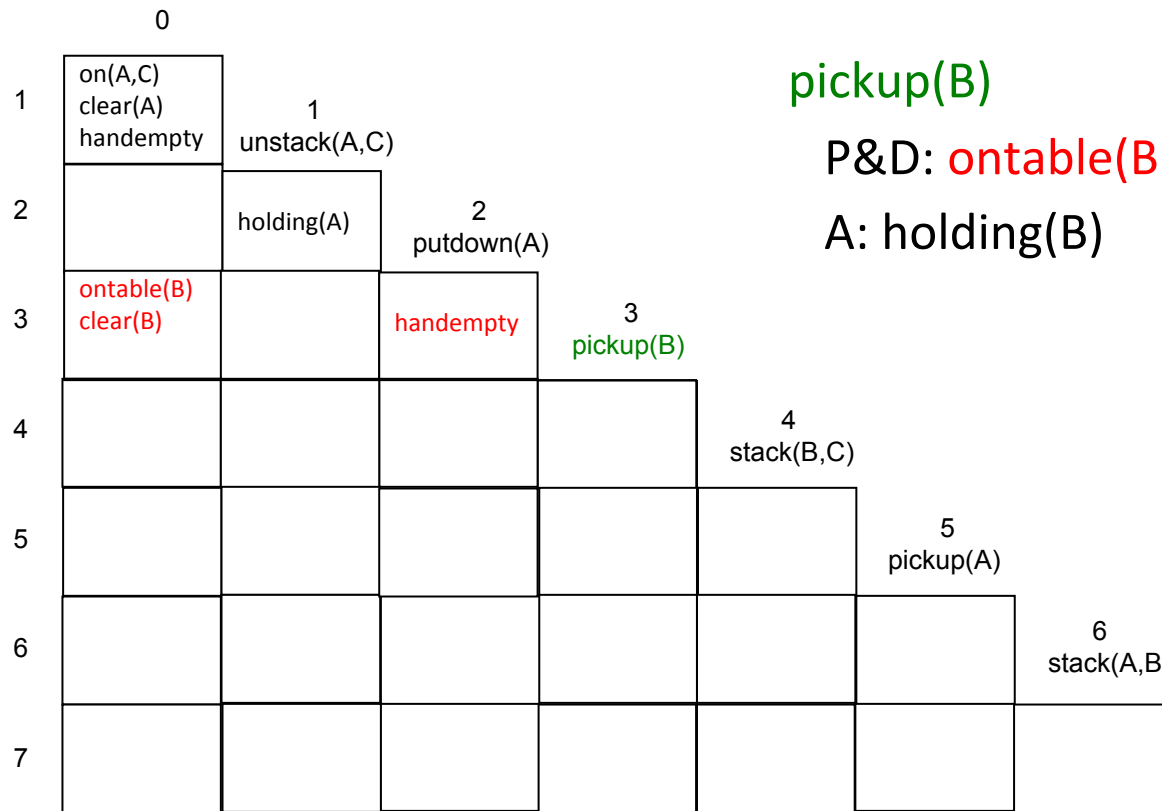
A: holding(B)



caută predicatele listei P&D  
a regulii 3



# Execuția planului: pasul 3



pickup(B)

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

A: holding(B)



ontable(C)

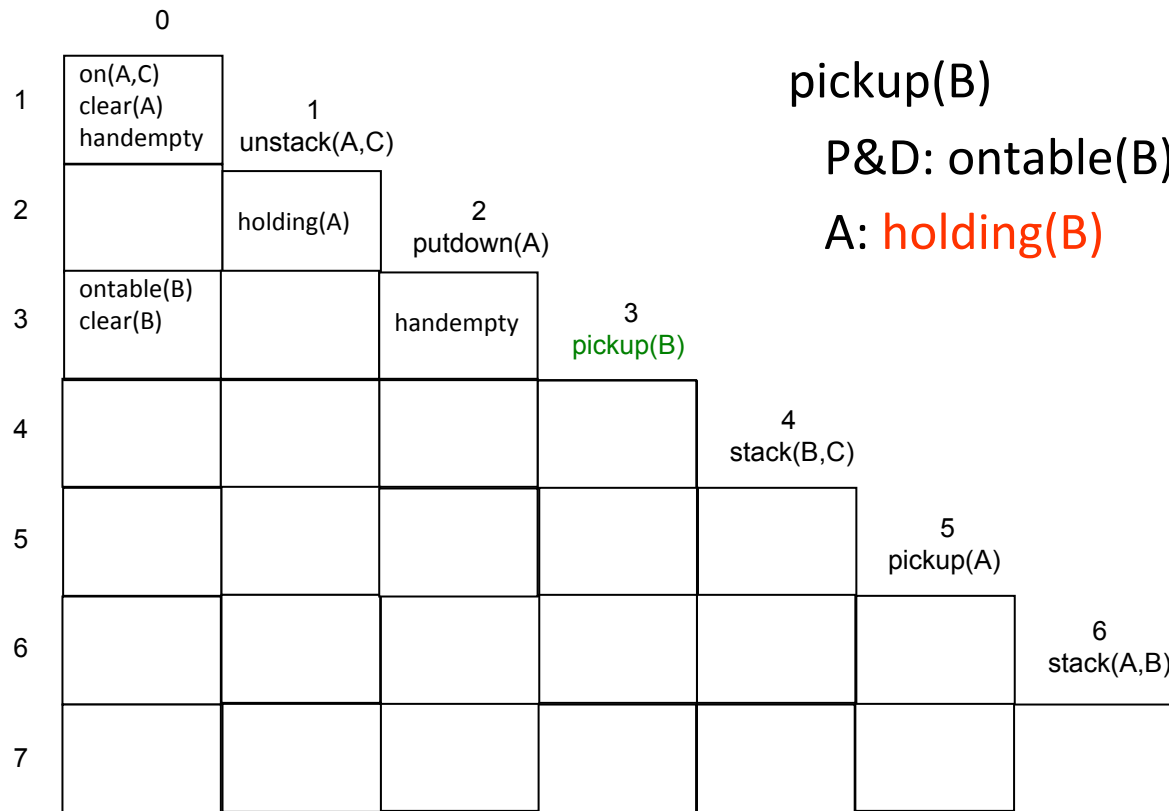


clear(C)

ontable(A)  
clear(A)

... și urcă-le în linia regulii 3

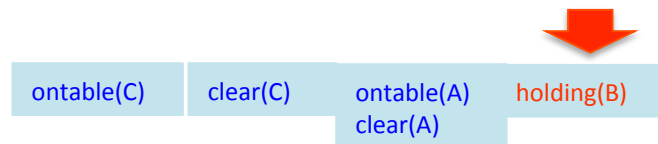
# Execuția planului: pasul 3



pickup(B)

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

A: **holding(B)**



colectează predicatul listei A  
a regulii 3

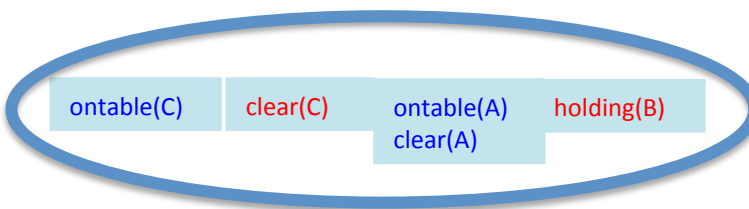
# Execuția planului: pasul 4

0							
1	on(A,C) clear(A) handempty	1 unstack(A,C)					
2		holding(A)	2 putdown(A)				
3	ontable(B) clear(B)		handempty	3 pickup(B)			
4					4 stack(B,C)		
5						5 pickup(A)	
6							6 stack(A,B)
7							

stack(B,C)

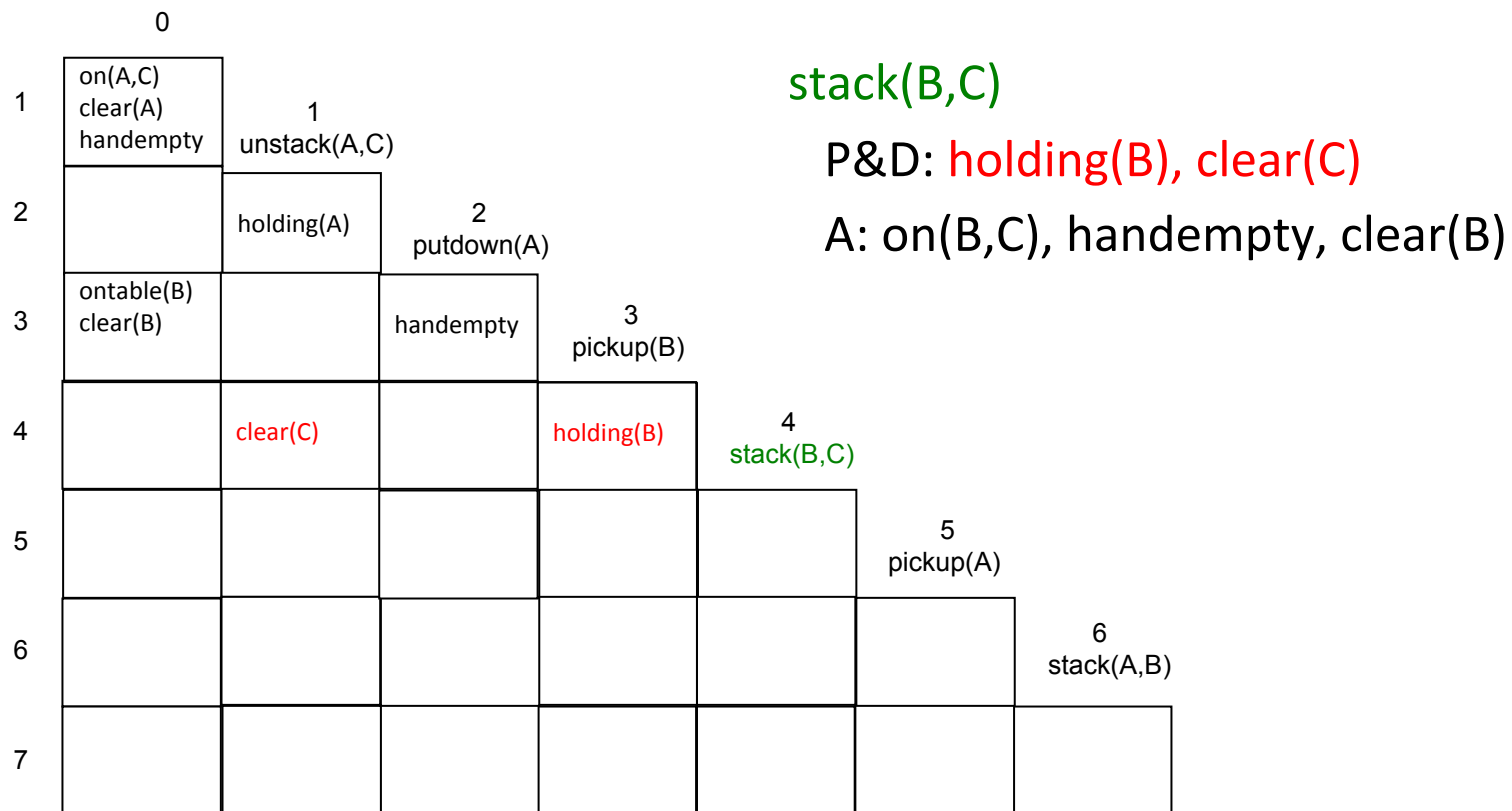
P&D: holding(B), clear(C)

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

ontable(C)

ontable(A)  
clear(A)

... și urcă-le în linia regulii 4

# Execuția planului: pasul 4

	0						
1	on(A,C) clear(A) handempty	1 unstack(A,C)					
2		holding(A)	2 putdown(A)				
3	ontable(B) clear(B)		handempty	3 pickup(B)			
4		clear(C)		holding(B)	4 stack(B,C)		
5						5 pickup(A)	
6							6 stack(A,E)
7							

stack(B,C)

P&D: holding(B)

A: on(B,C), handempty

stack(B,C)

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

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



ontable(C)

ontable(A)  
clear(A)

on(B,C)  
handempty  
clear(B)

colectează jos predicatele  
listei A a regulii 4

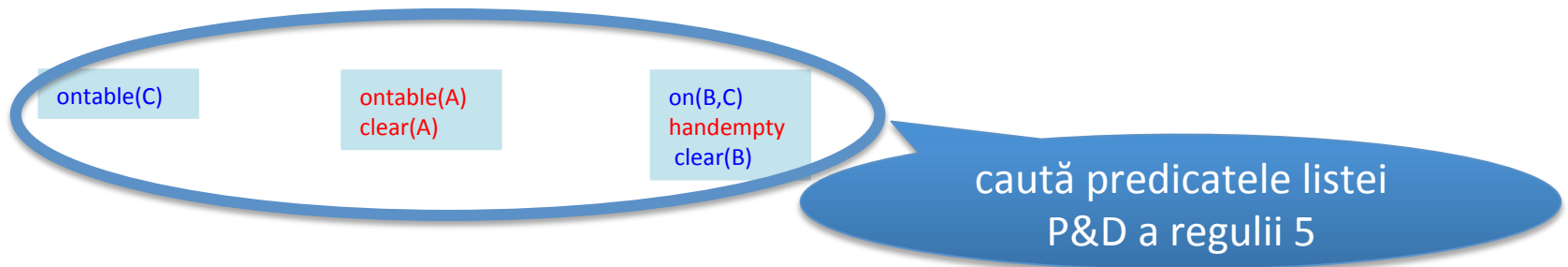
# Execuția planului: pasul 5

	0						
1	on(A,C) clear(A) handempty	1 unstack(A,C)					
2		holding(A)	2 putdown(A)				
3	ontable(B) clear(B)		handempty	3 pickup(B)			
4		clear(C)		holding(B)	4 stack(B,C)		
5						5 pickup(A)	
6							6 stack(A,B)
7							

pickup(A)

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

A: holding(A)



# Execuția planului: pasul 5

	0						
1	on(A,C) clear(A) handempty	1 unstack(A,C)					
2		holding(A)	2 putdown(A)				
3	ontable(B) clear(B)		handempty	3 pickup(B)			
4		clear(C)		holding(B)	4 stack(B,C)		
5			ontable(A) clear(A)		handempty	5 pickup(A)	
6							6 stack(A,B)
7							

pickup(A)

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

A: holding(A)

ontable(C)

on(B,C)  
clear(B)

... și urcă-le în linia  
regulii 5

# Execuția planului: pasul 5

	0						
1	on(A,C) clear(A) handempty	1 unstack(A,C)					
2		holding(A)	2 putdown(A)				
3	ontable(B) clear(B)		handempty	3 pickup(B)			
4		clear(C)		holding(B)	4 stack(B,C)		
5			ontable(A) clear(A)		handempty	5 pickup(A)	
6							6 stack(A,B)
7							

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

0						
1	on(A,C) clear(A) handempty	1 unstack(A,C)				
2		holding(A)	2 putdown(A)			
3	ontable(B) clear(B)		handempty	3 pickup(B)		
4		clear(C)		holding(B)	4 stack(B,C)	
5			ontable(A) clear(A)		handempty	5 pickup(A)
6						6 stack(A,B)
7						

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

	0						
1	on(A,C) clear(A) handempty	1 unstack(A,C)					
2		holding(A)	2 putdown(A)				
3	ontable(B) clear(B)		handempty	3 pickup(B)			
4		clear(C)		holding(B)	4 stack(B,C)		
5			ontable(A) clear(A)		handempty	5 pickup(A)	
6					clear(B)	holding(A)	6 stack(A,B)
7							

stack(A,B)

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

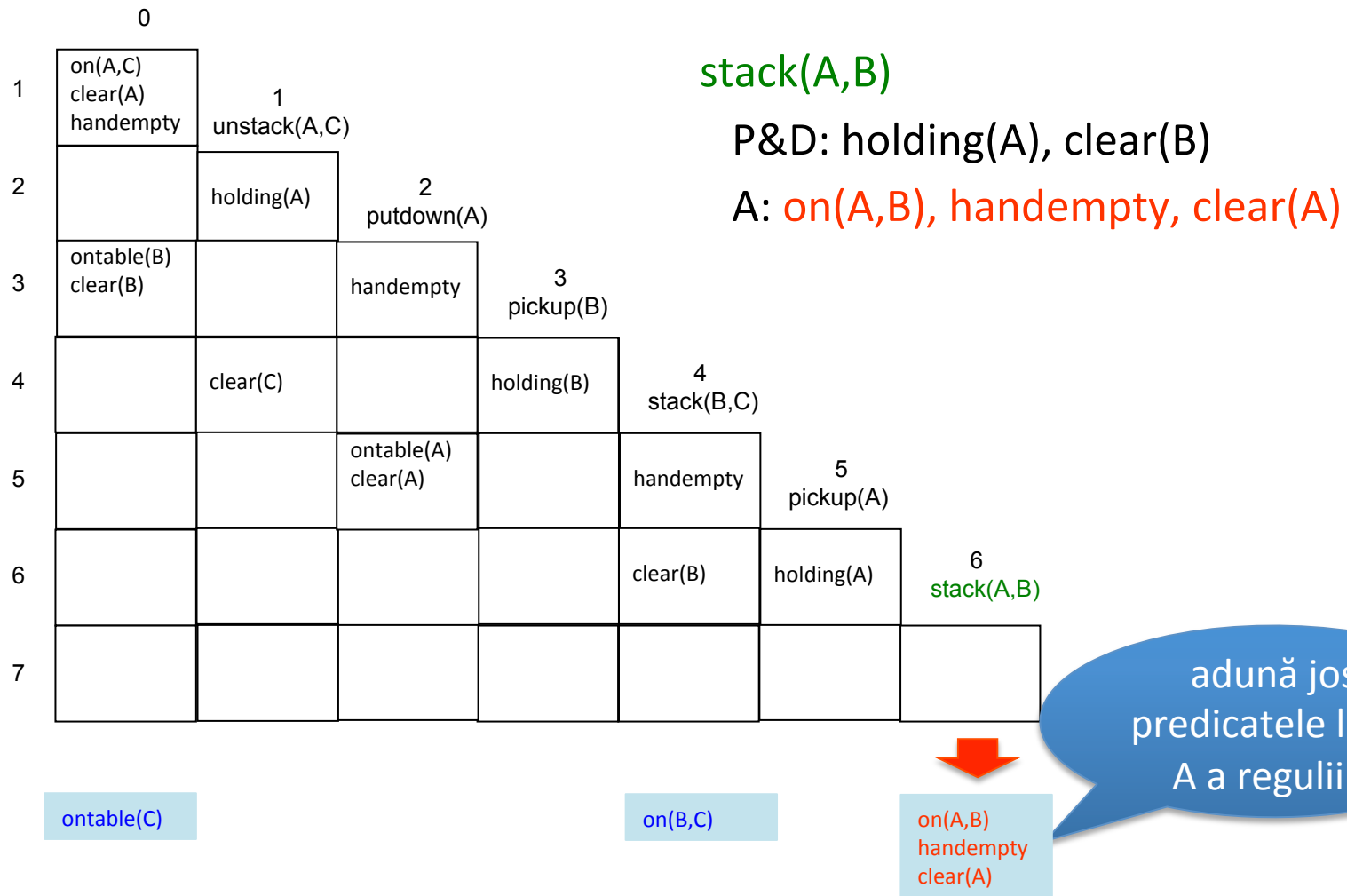
A: on(A,B), handempty, clear(A)

ontable(C)

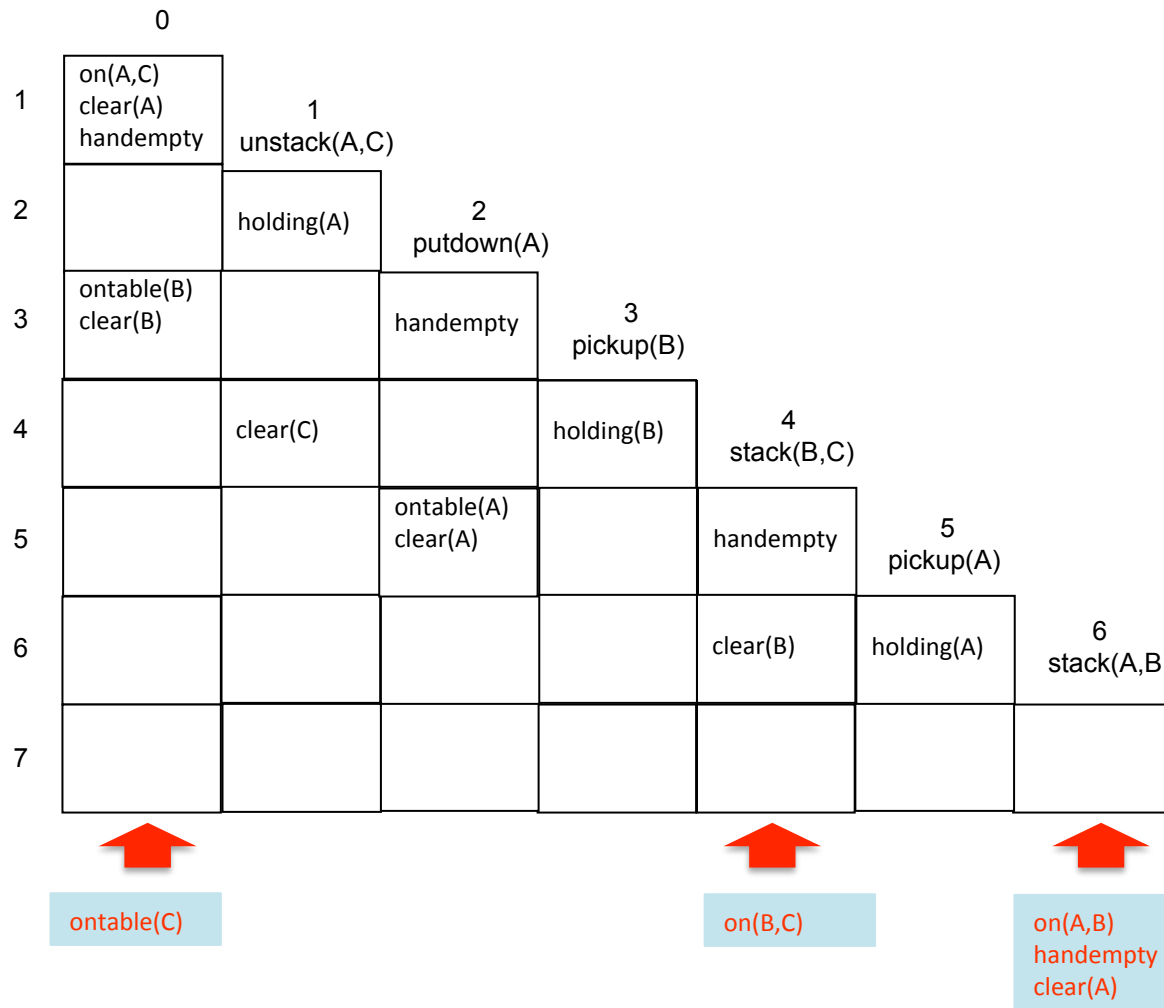
on(B,C)

... și urcă-le în  
linia regulii 6

# Execuția planului: pasul 6

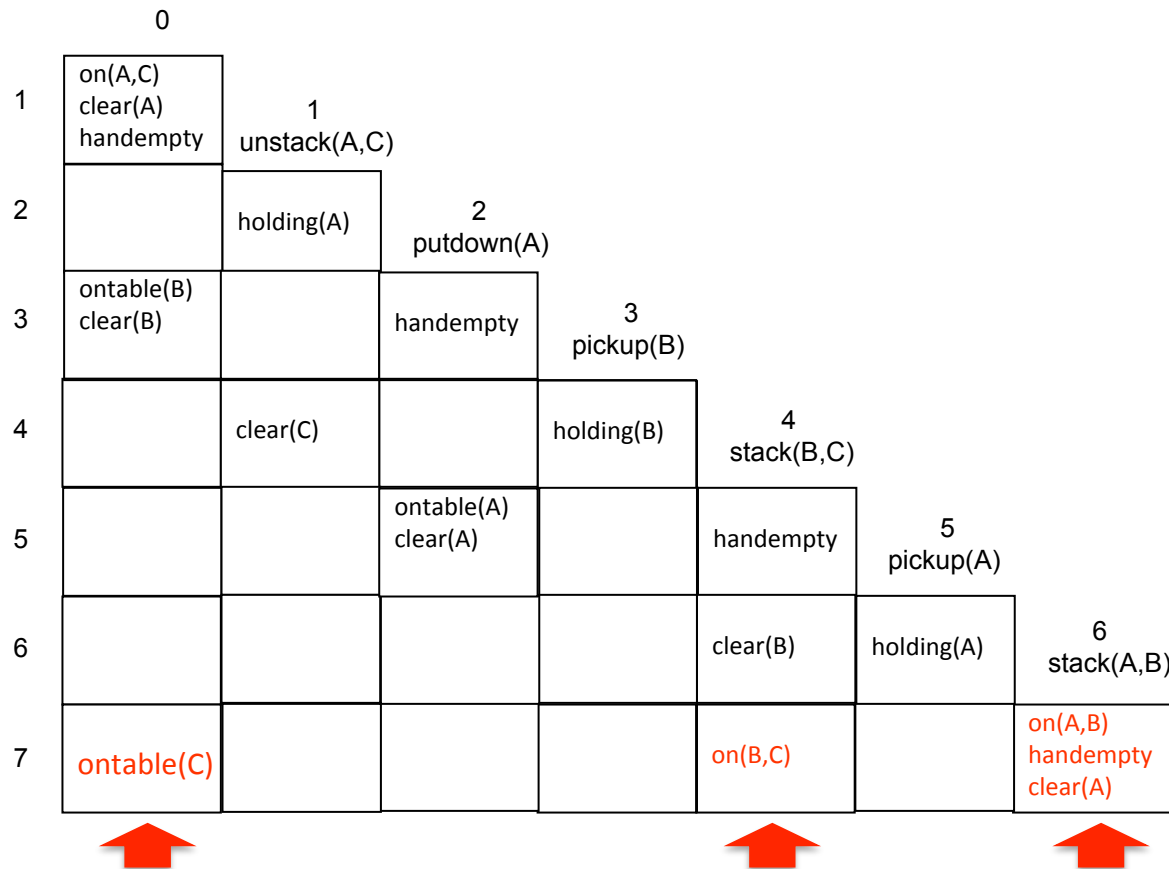


# Execuția planului: finalizare



culege pe  
ultimul rând  
predicatul  
rămase

# Execuția planului: finalizare



culege pe  
ultimul rând  
predicatele  
rămase

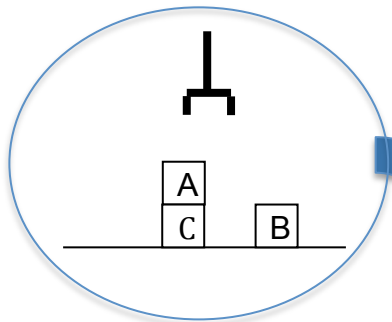
# Execuția planului: finalizare

	0						
1	on(A,C) clear(A) handempty	1 unstack(A,C)					
2		holding(A)	2 putdown(A)				
3	ontable(B) clear(B)		handempty	3 pickup(B)			
4		clear(C)		holding(B)	4 stack(B,C)		
5			ontable(A) clear(A)		handempty	5 pickup(A)	
6					clear(B)	holding(A)	6 stack(A,B)
7	ontable(C)				on(B,C)		on(A,B) handempty clear(A)

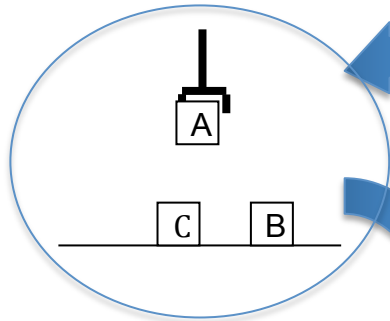
Starea  
finală

# Accident!

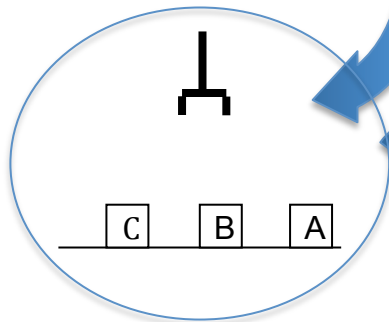
Starea inițială



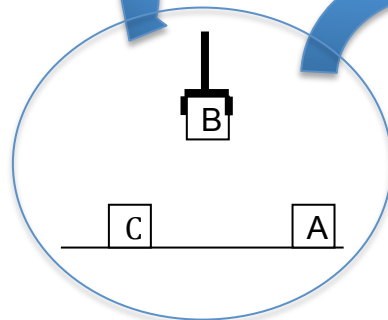
1: unstack(A,C)



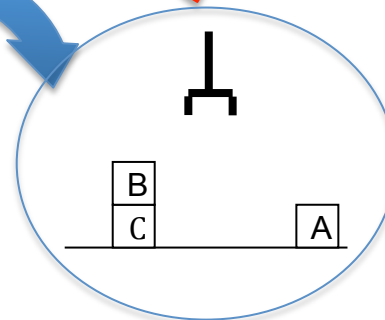
2: putdown(A)



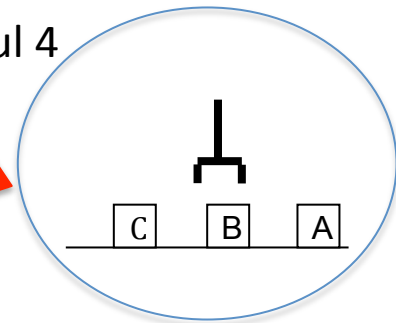
3: pickup(B)



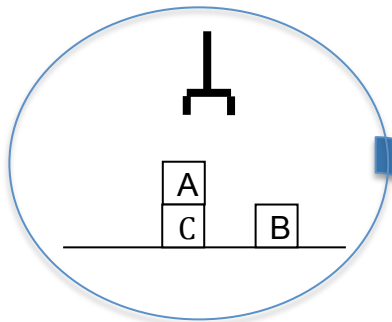
4: stack(B,C)



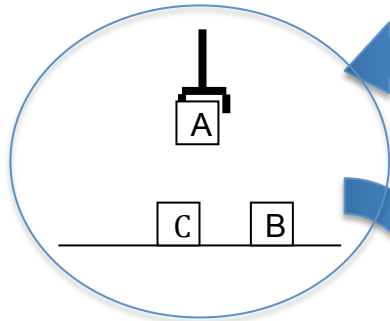
ACCIDENT: după pasul 4  
mâna dărmă stiva!



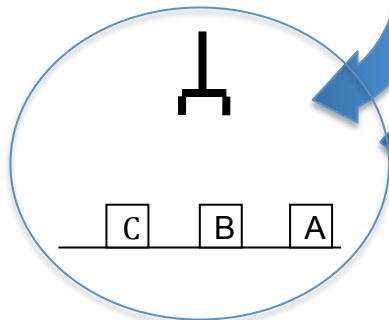
Starea inițială



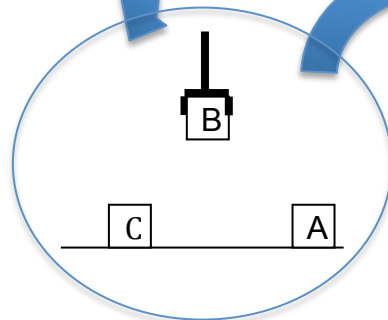
1: unstack(A,C)



2: putdown(A)



3: pickup(B)

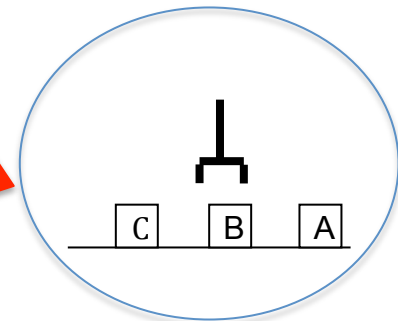


4: stack(B,C)

# Accident!

Descrierea stării de după accident:

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

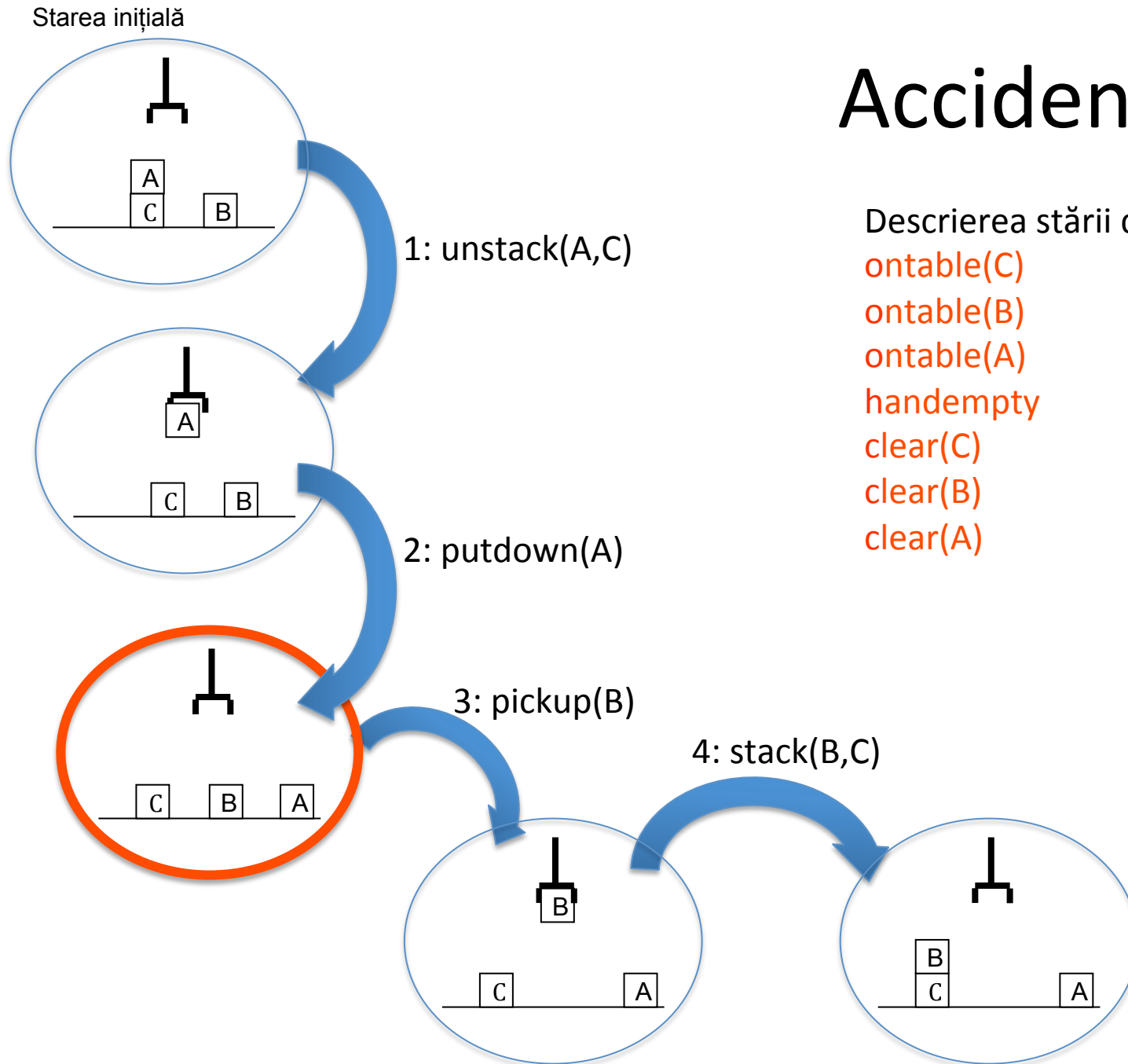




# Accident!

Descrierea stării de după accident:

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



# Kernel de rang 7

	0						
1	on(A,C) clear(A) handempty	1 unstack(A,C)					
2		holding(A)	2 putdown(A)				
3	ontable(B) clear(B)		handempty	3 pickup(B)			
4		clear(C)		holding(B)	4 stack(B,C)		
5			ontable(A) clear(A)		handempty	5 pickup(A)	
6					clear(B)	holding(A)	6 stack(A,B)
7	ontable(C)				on(B,C)		on(A,B) handempty clear(A)

Caută primul kernel  
care corespunde stării  
de după accident

NU

# Kernel de rang 6

	0						
1	on(A,C) clear(A) handempty	1 unstack(A,C)					
2		holding(A)	2 putdown(A)				
3	ontable(B) clear(B)		handempty	3 pickup(B)			
4		clear(C)		holding(B)	4 stack(B,C)		
5			ontable(A) clear(A)		handempty	5 pickup(A)	
6					clear(B)	holding(A)	6 stack(A,B)
7	ontable(C)				on(B,C)		on(A,B) handempty clear(A)

Caută primul kernel care corespunde stării de după accident

NU

# Kernel de rang 5

	0						
1	on(A,C) clear(A) handempty	1 unstack(A,C)					
2		holding(A)	2 putdown(A)				
3	ontable(B) clear(B)		handempty	3 pickup(B)			
4		clear(C)		holding(B)	4 stack(B,C)		
5			ontable(A) clear(A)		handempty	5 pickup(A)	
6					clear(B)	holding(A)	6 stack(A,B)
7	ontable(C)				on(B,C)		on(A,B) handempty clear(A)

Caută primul kernel care corespunde stării de după accident

NU

# Kernel de rang 4

	0						
1	on(A,C) clear(A) handempty	1 unstack(A,C)					
2		holding(A)	2 putdown(A)				
3	ontable(B) clear(B)		handempty	3 pickup(B)			
4		clear(C)		holding(B)	4 stack(B,C)		
5			ontable(A) clear(A)		handempty	5 pickup(A)	
6					clear(B)	holding(A)	6 stack(A,B)
7	ontable(C)				on(B,C)		on(A,B) handempty clear(A)

Caută primul kernel care corespunde stării de după accident

NU

# Kernel de rang 3

	0						
1	on(A,C) clear(A) handempty	1 unstack(A,C)					
2		holding(A)	2 putdown(A)				
3	ontable(B) clear(B)		handempty	3 pickup(B)			
4		clear(C)		holding(B)	4 stack(B,C)		
5			ontable(A) clear(A)		handempty	5 pickup(A)	
6					clear(B)	holding(A)	6 stack(A,B)
7	ontable(C)				on(B,C)		on(A,B) handempty clear(A)

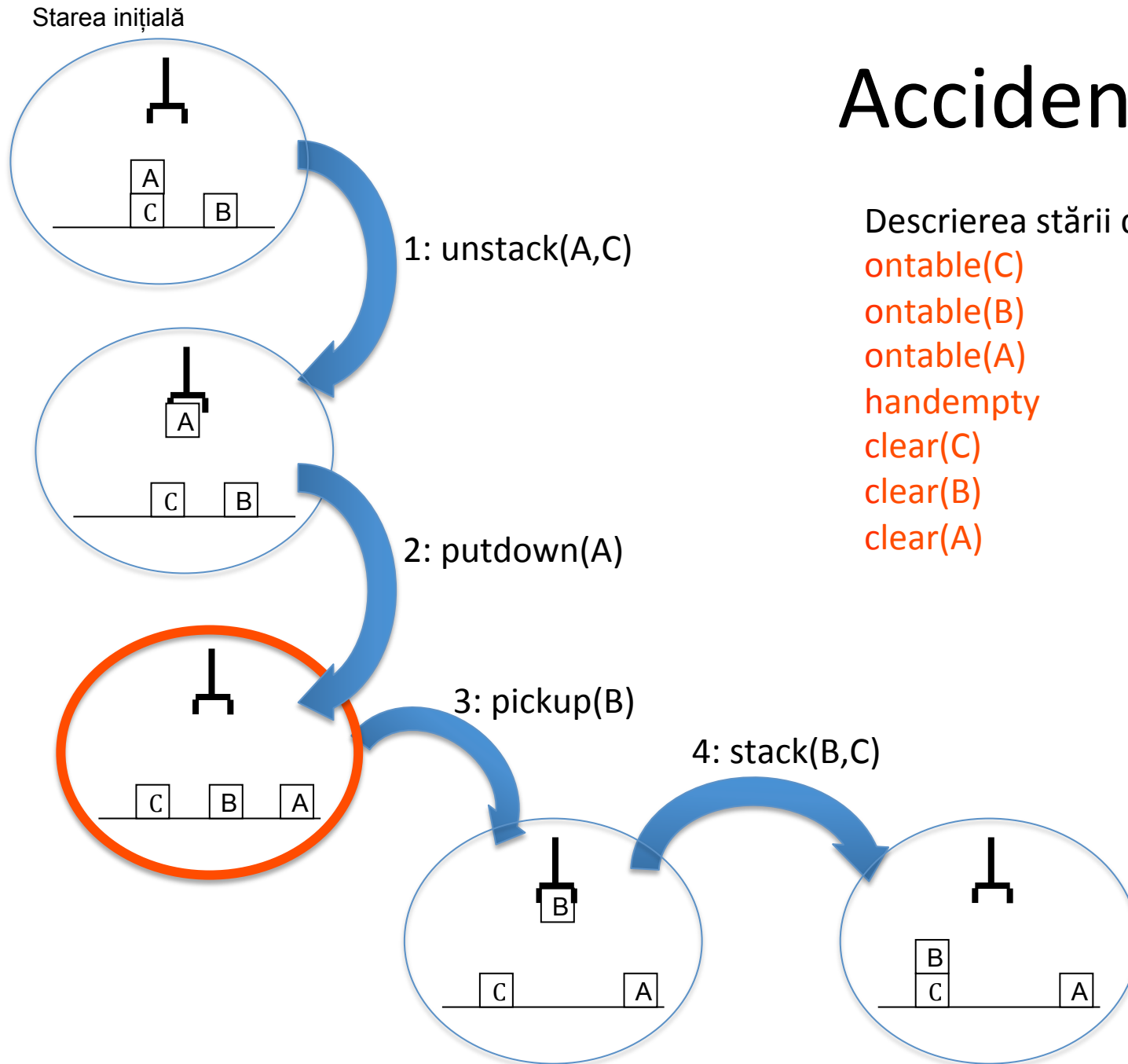
În caz de accident:  
dacă starea de după  
accident corespunde  
kernelului de rang  $k \rightarrow$   
continuă cu pasul  $k$

DA

# Accident!

Descrierea stării de după accident:

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



# leșirea din 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 => execuție