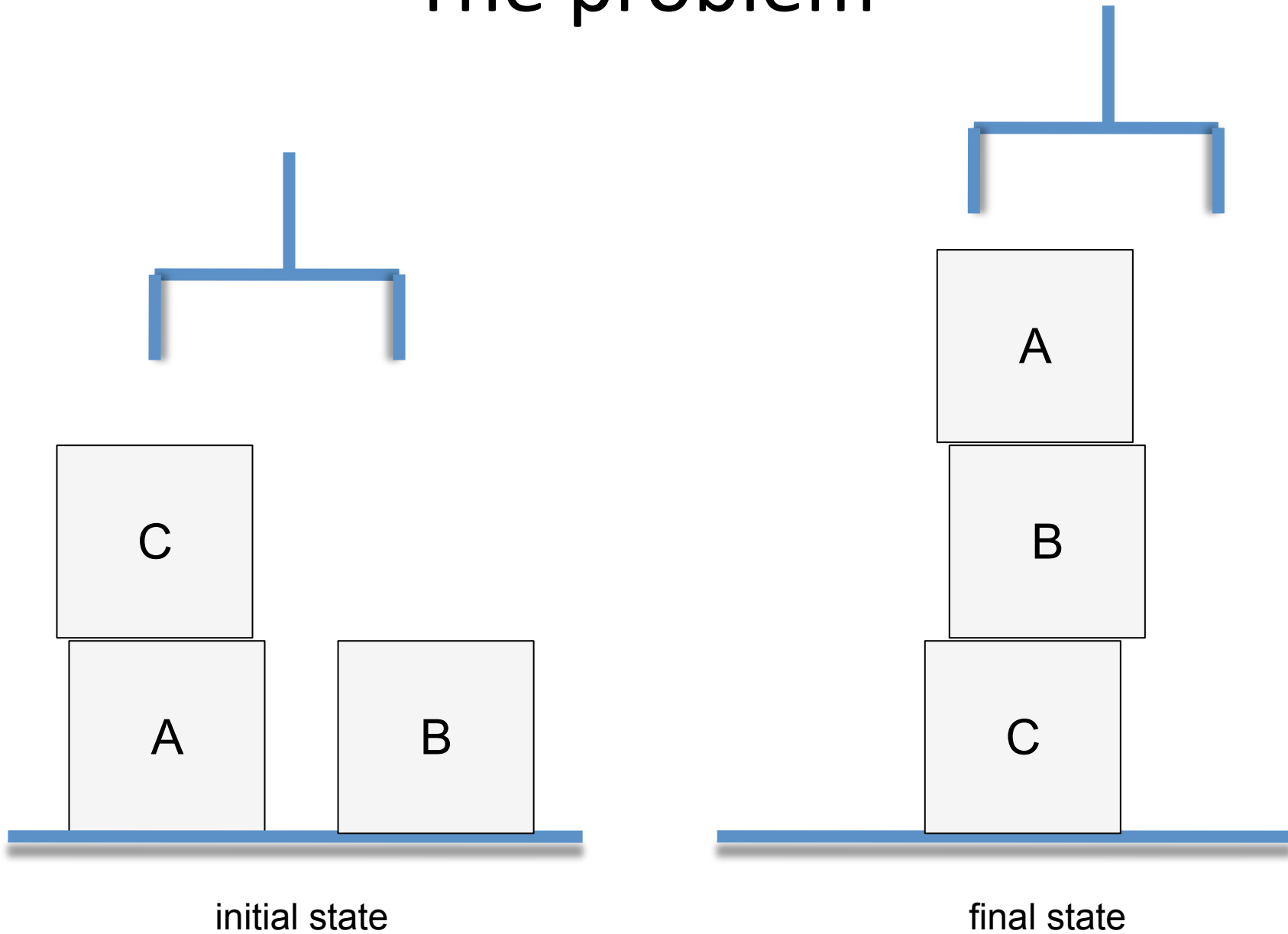


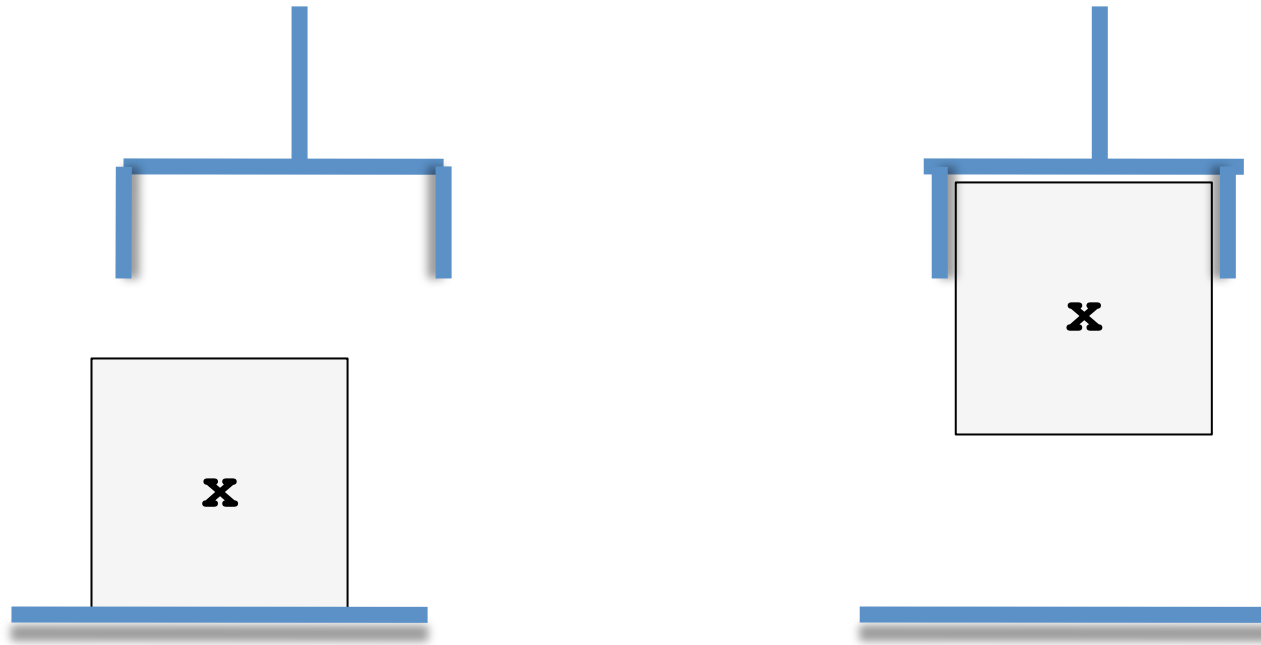
# Course 11

## Planning and plan execution

# The problem



# STRIPS rules

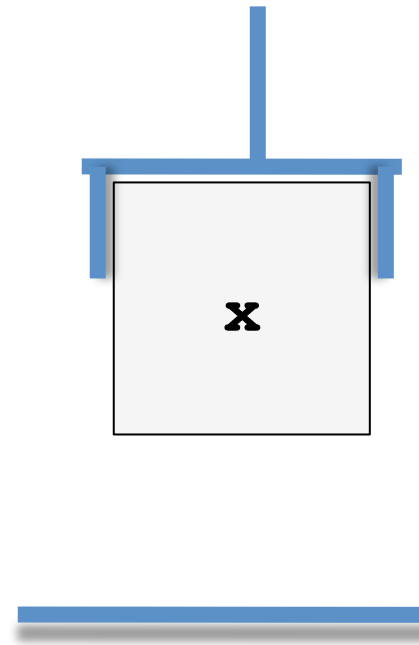
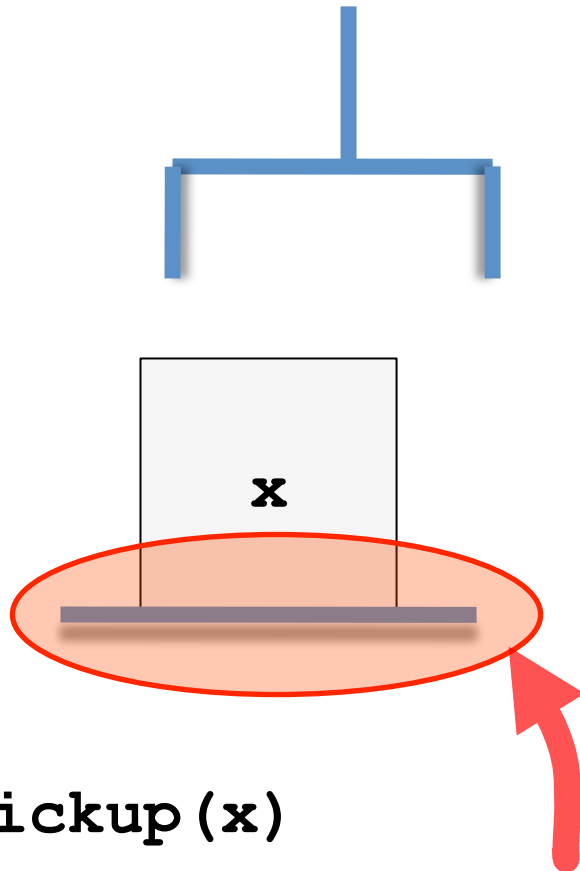


`pickup(x)`

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

A: `holding(x)`

# STRIPS rules

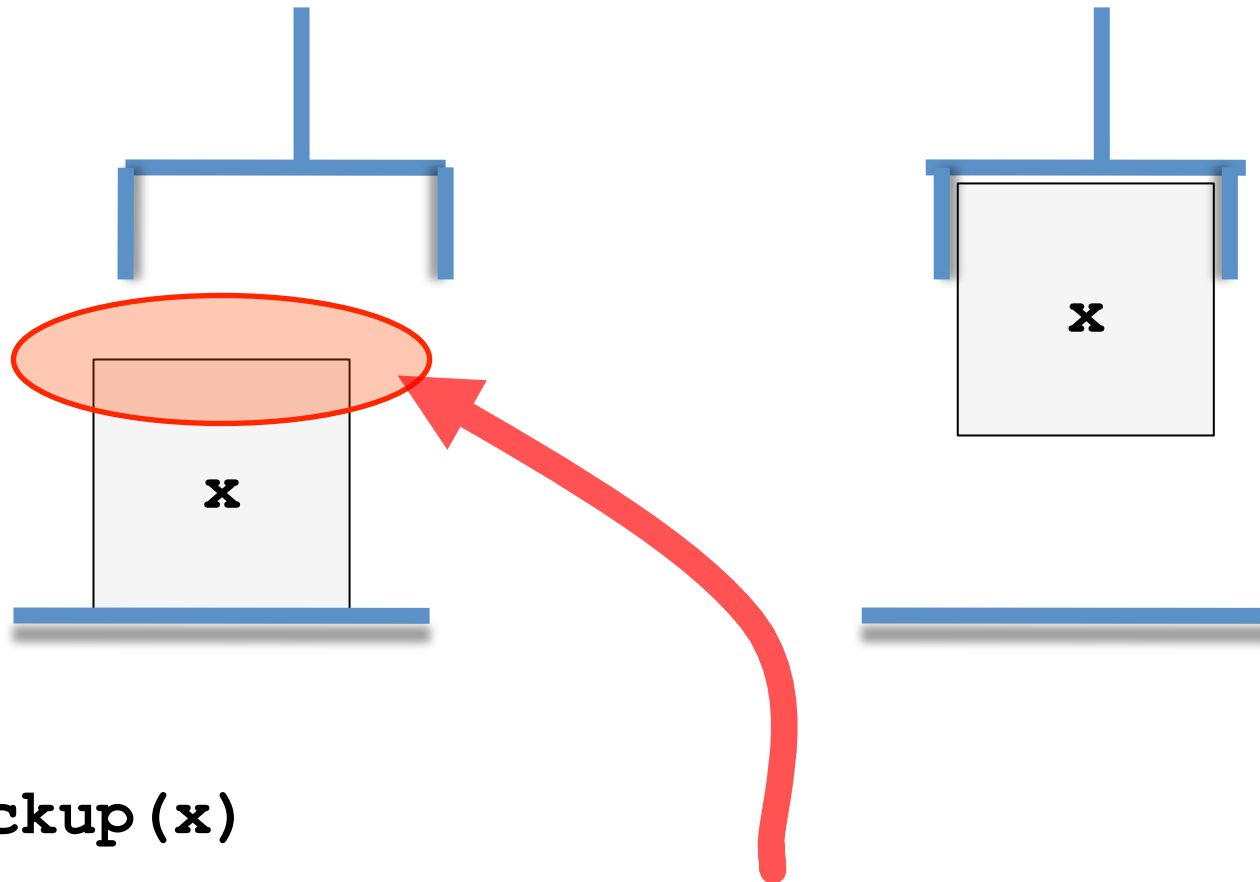


`pickup(x)`

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

A: `holding(x)`

# STRIPS rules

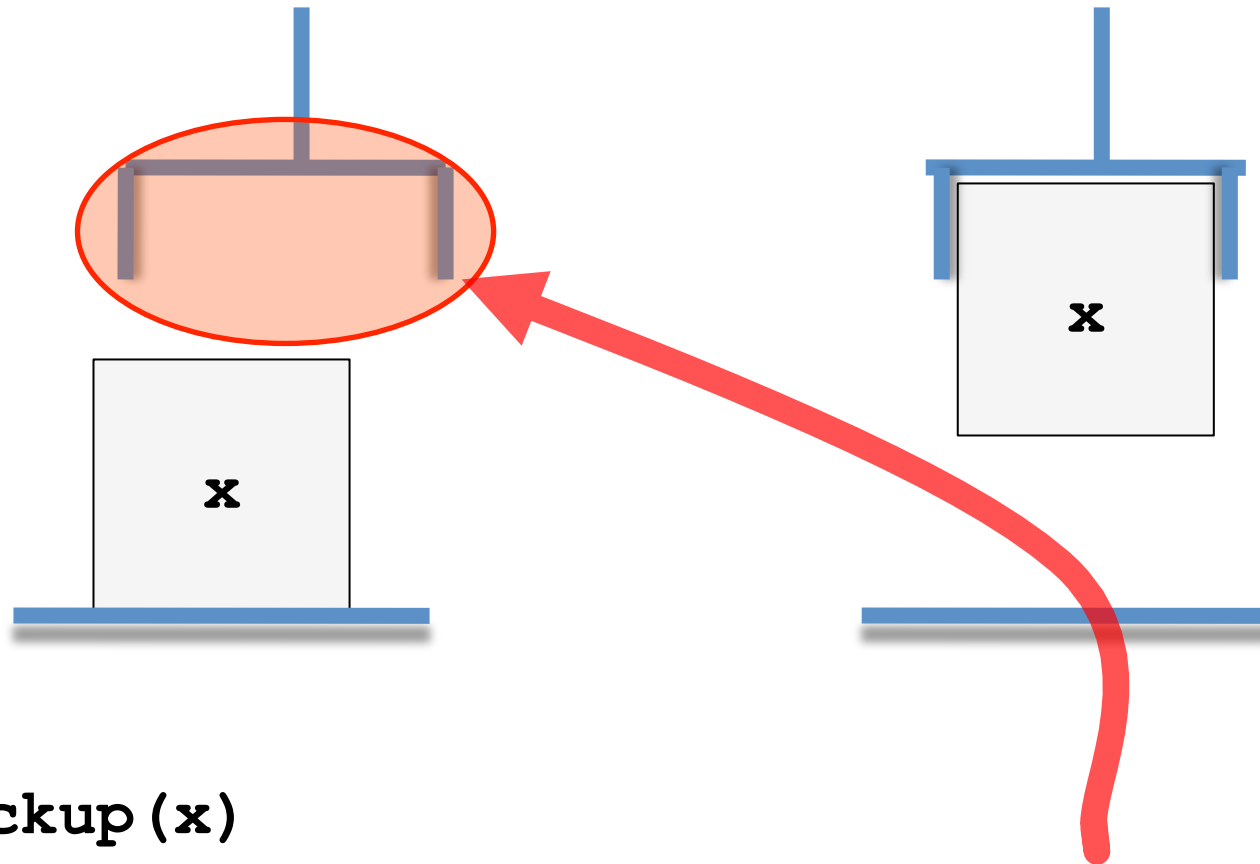


`pickup(x)`

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

A: `holding(x)`

# STRIPS rules

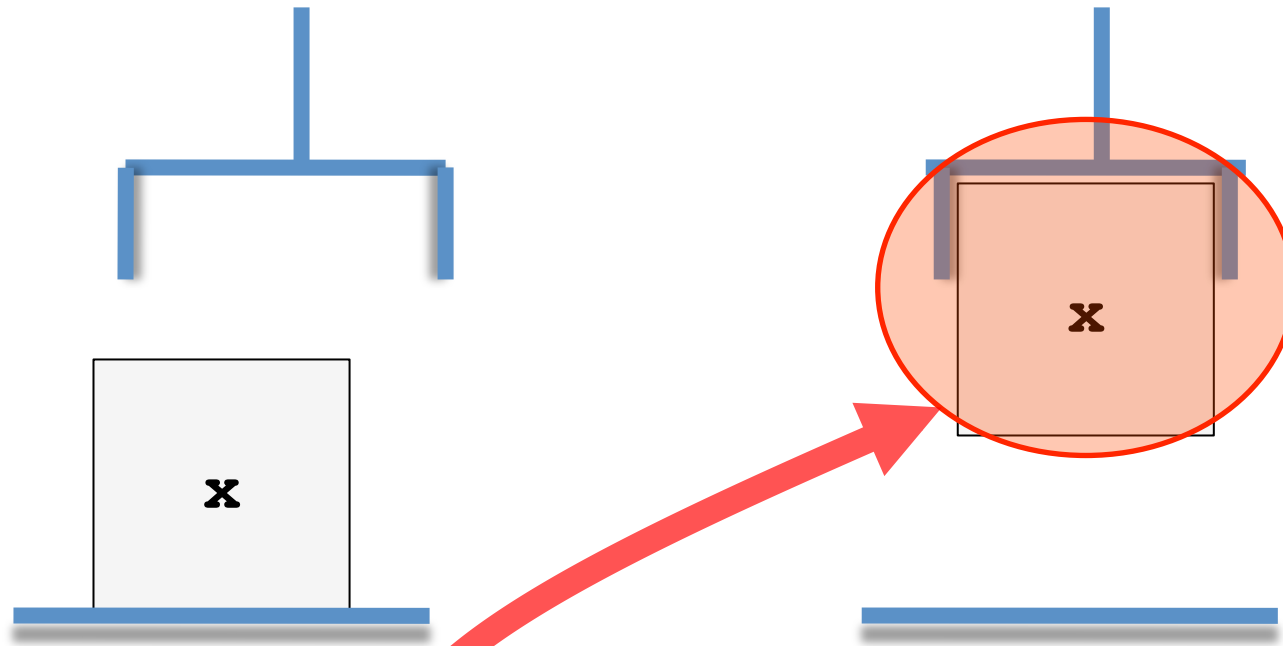


`pickup(x)`

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

A: `holding(x)`

# STRIPS rules

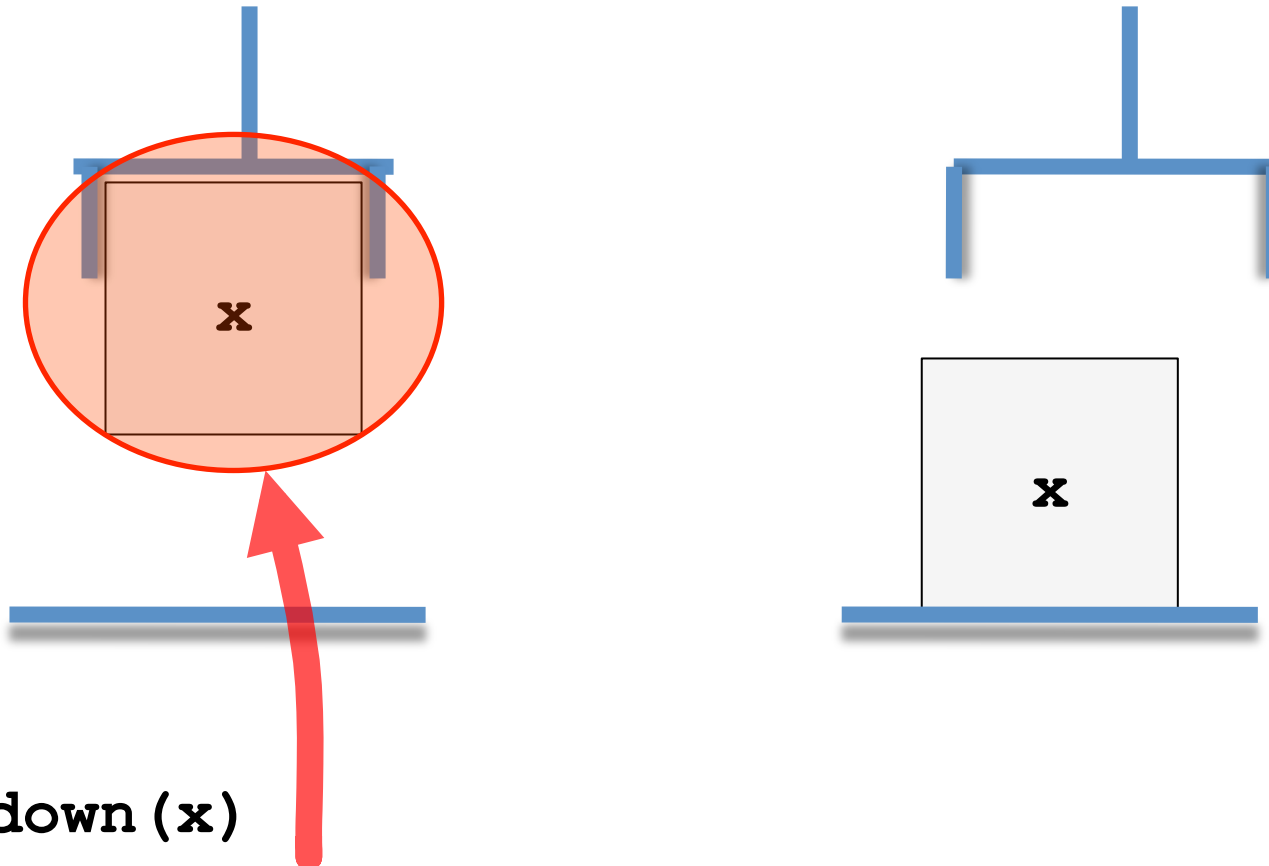


`pickup(x)`

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

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

# STRIPS rules



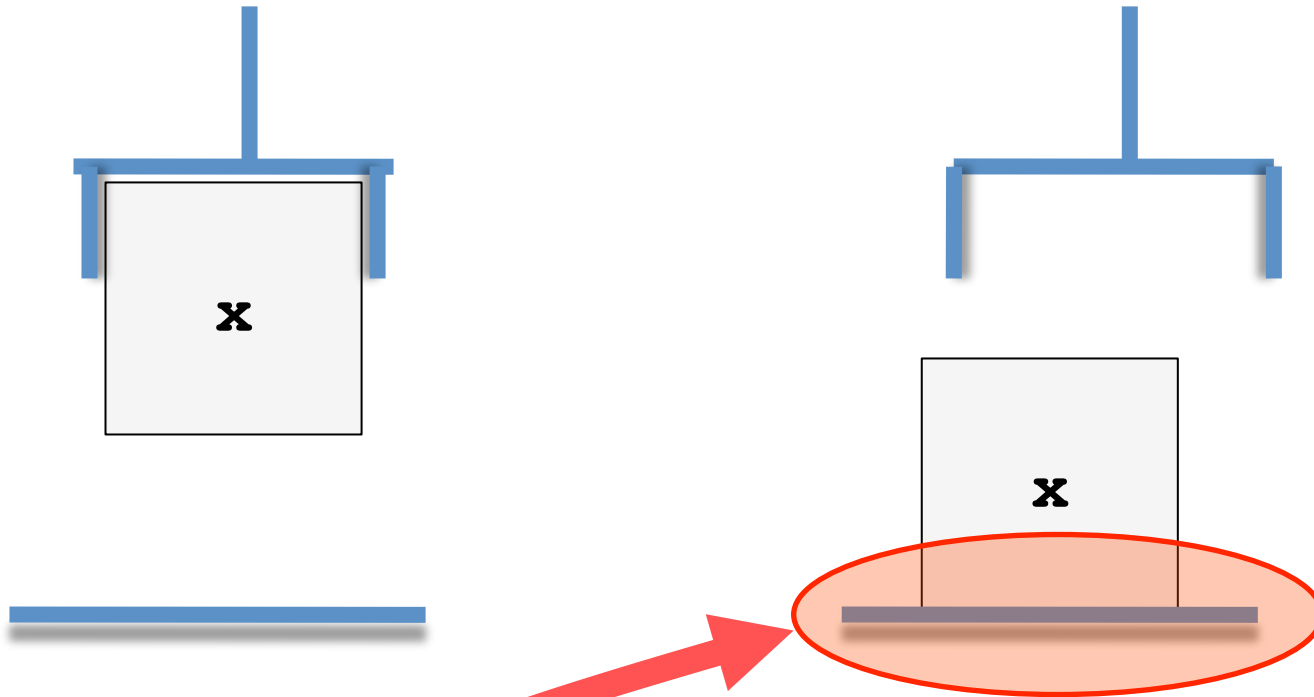
`putdown(x)`

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

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



# STRIPS rules

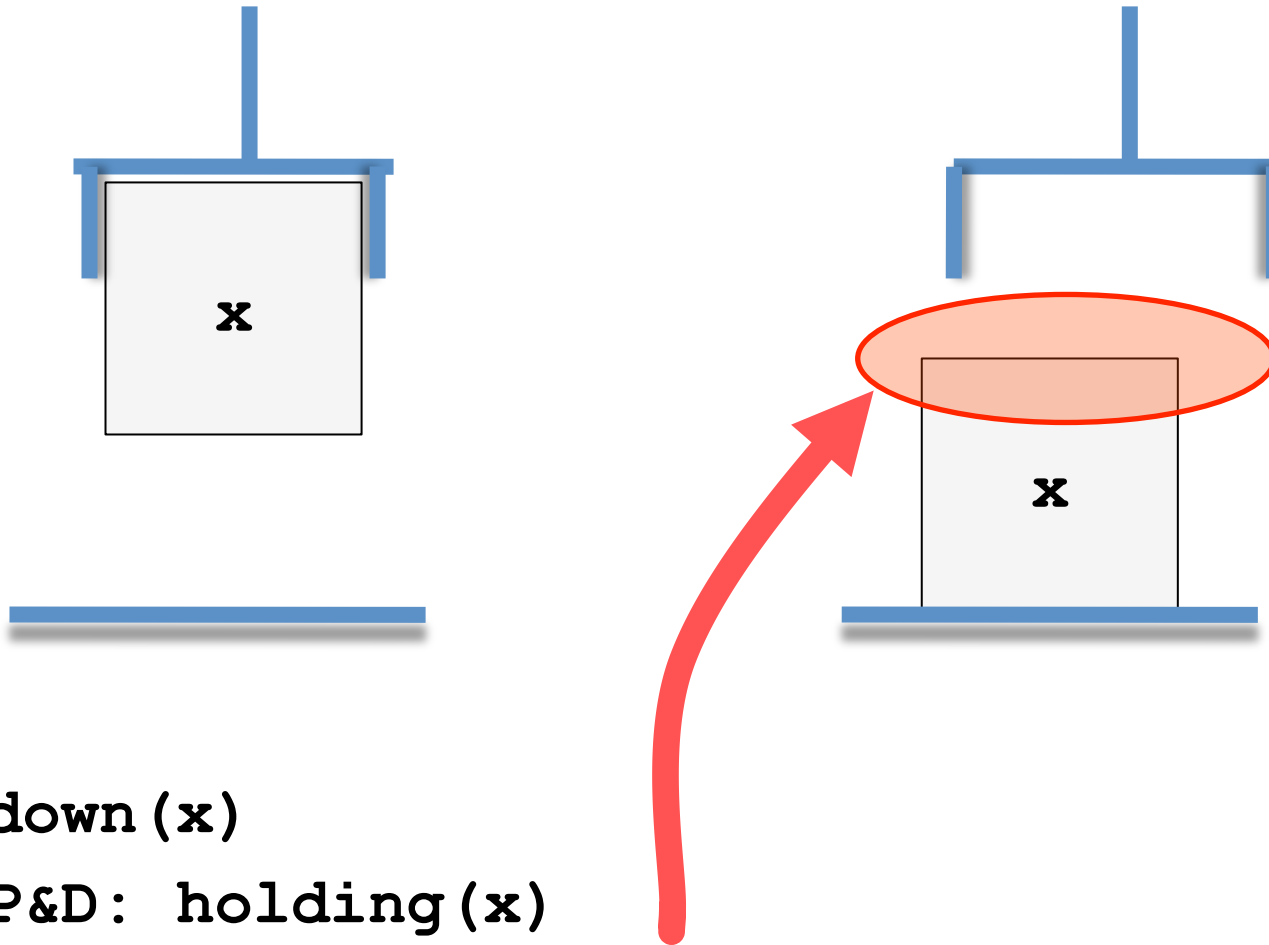


`putdown(x)`

P&D: `holding(x)`

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

# STRIPS rules

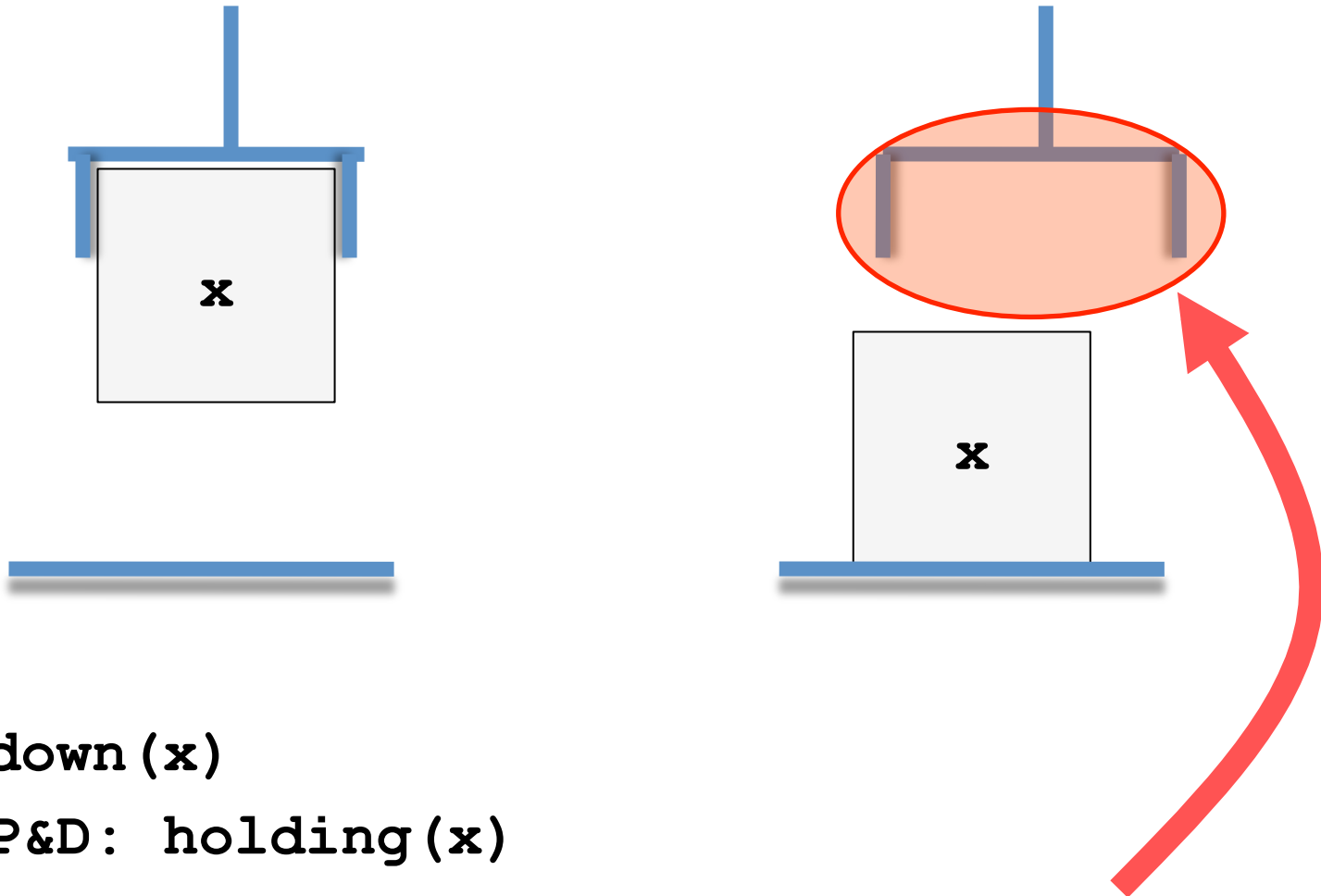


`putdown(x)`

P&D: `holding(x)`

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

# STRIPS rules

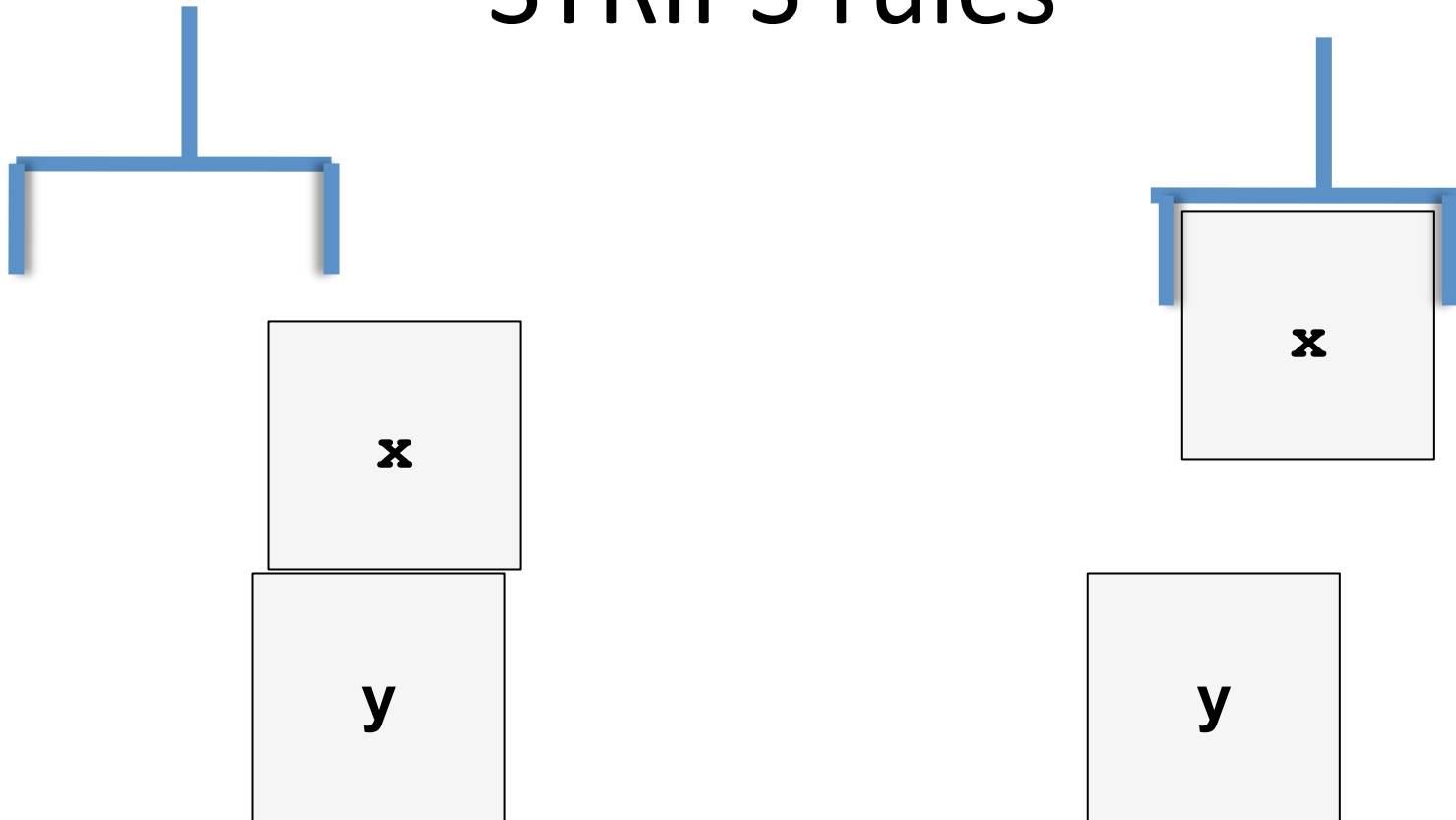


`putdown(x)`

P&D: `holding(x)`

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

# STRIPS rules

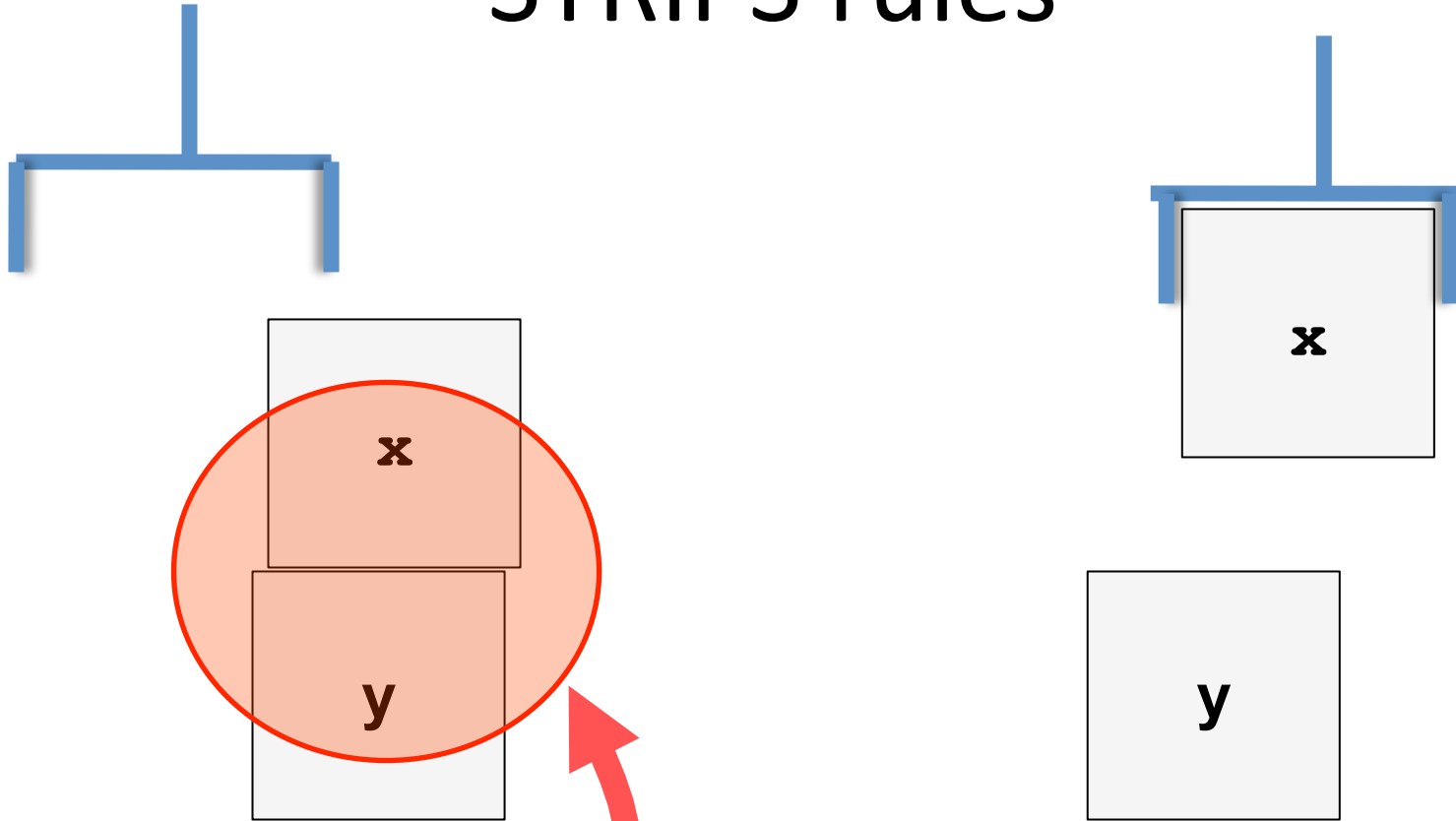


`unstack(x, y)`

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

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

# STRIPS rules

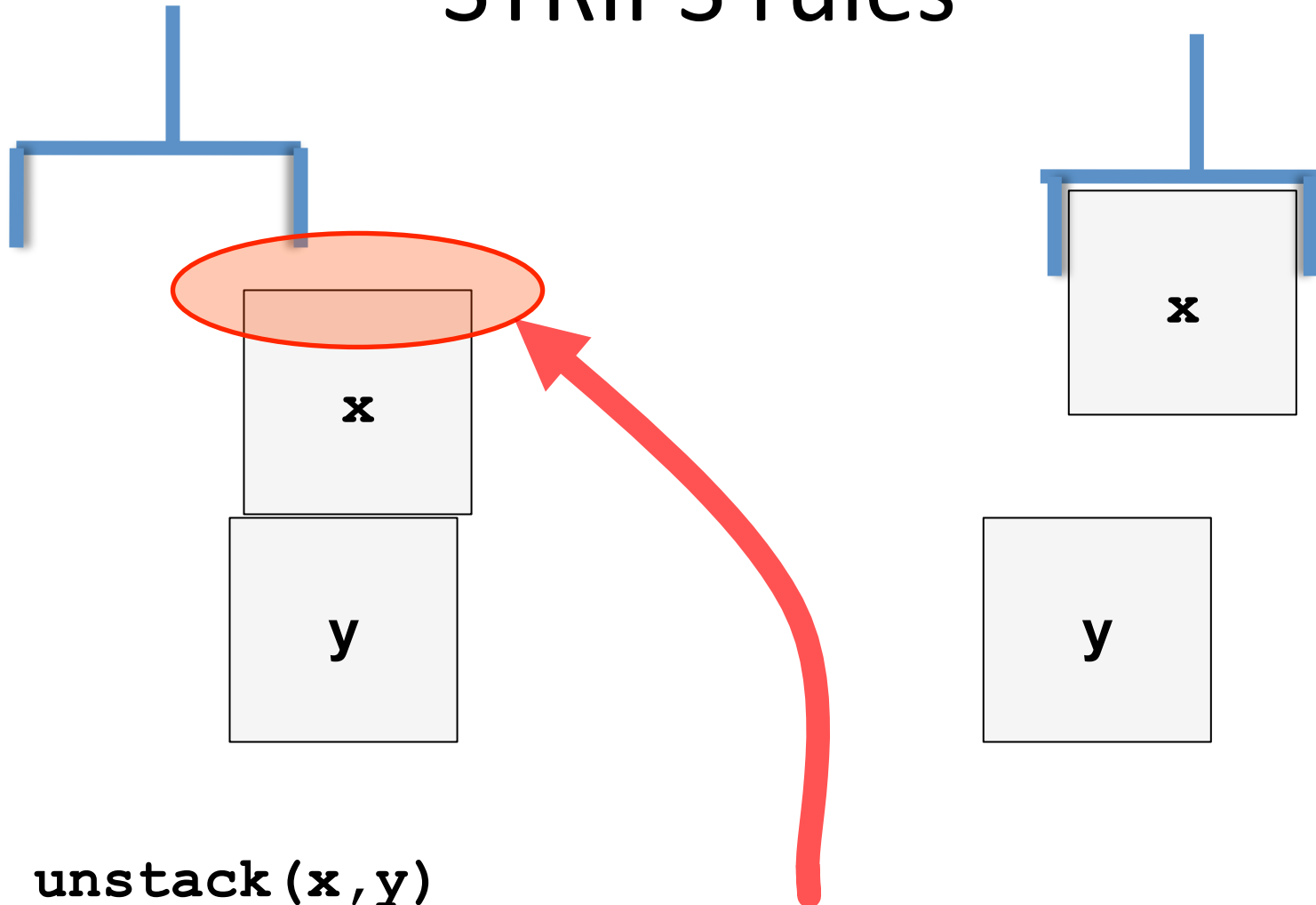


`unstack(x,y)`

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

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

# STRIPS rules

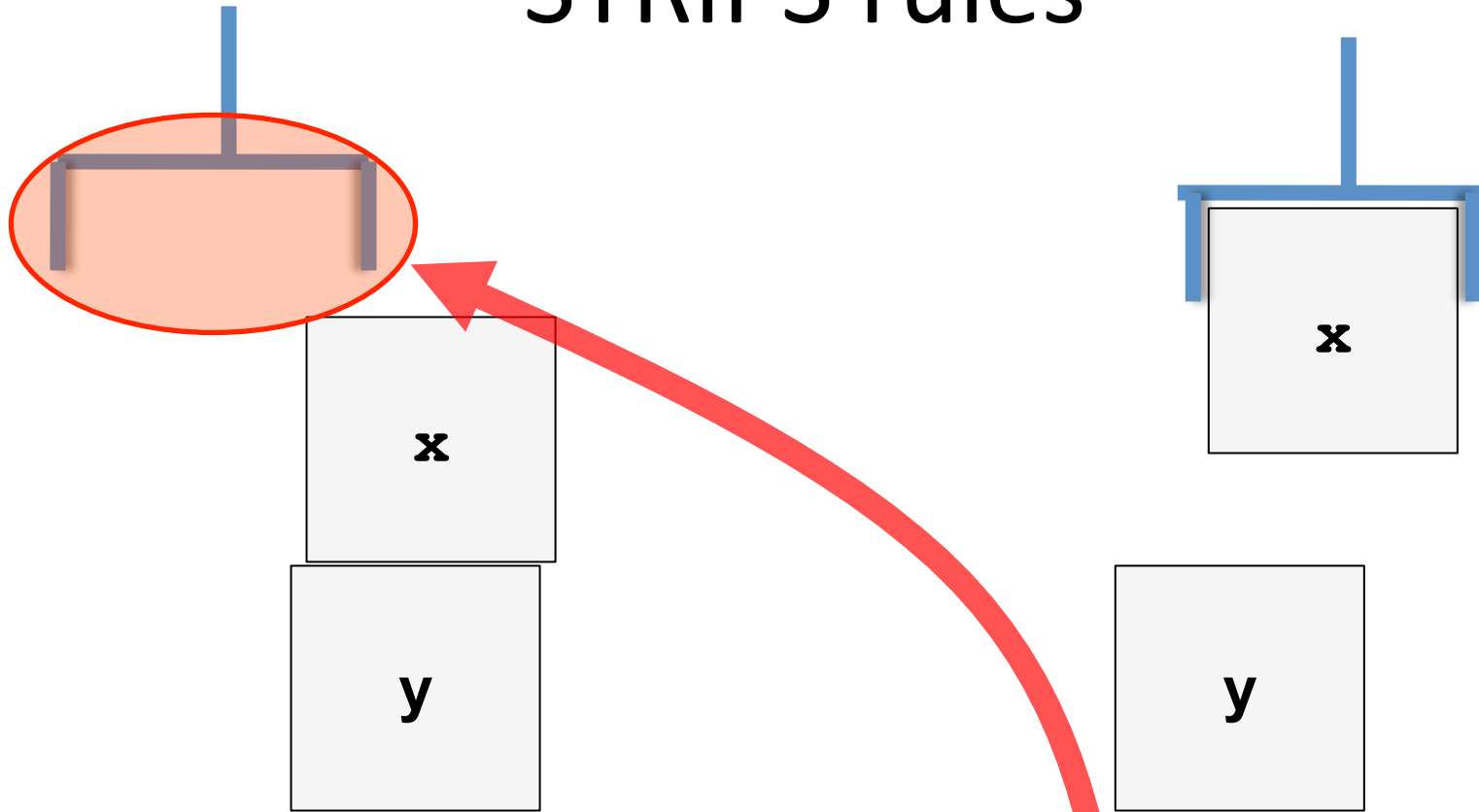


`unstack(x, y)`

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

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

# STRIPS rules

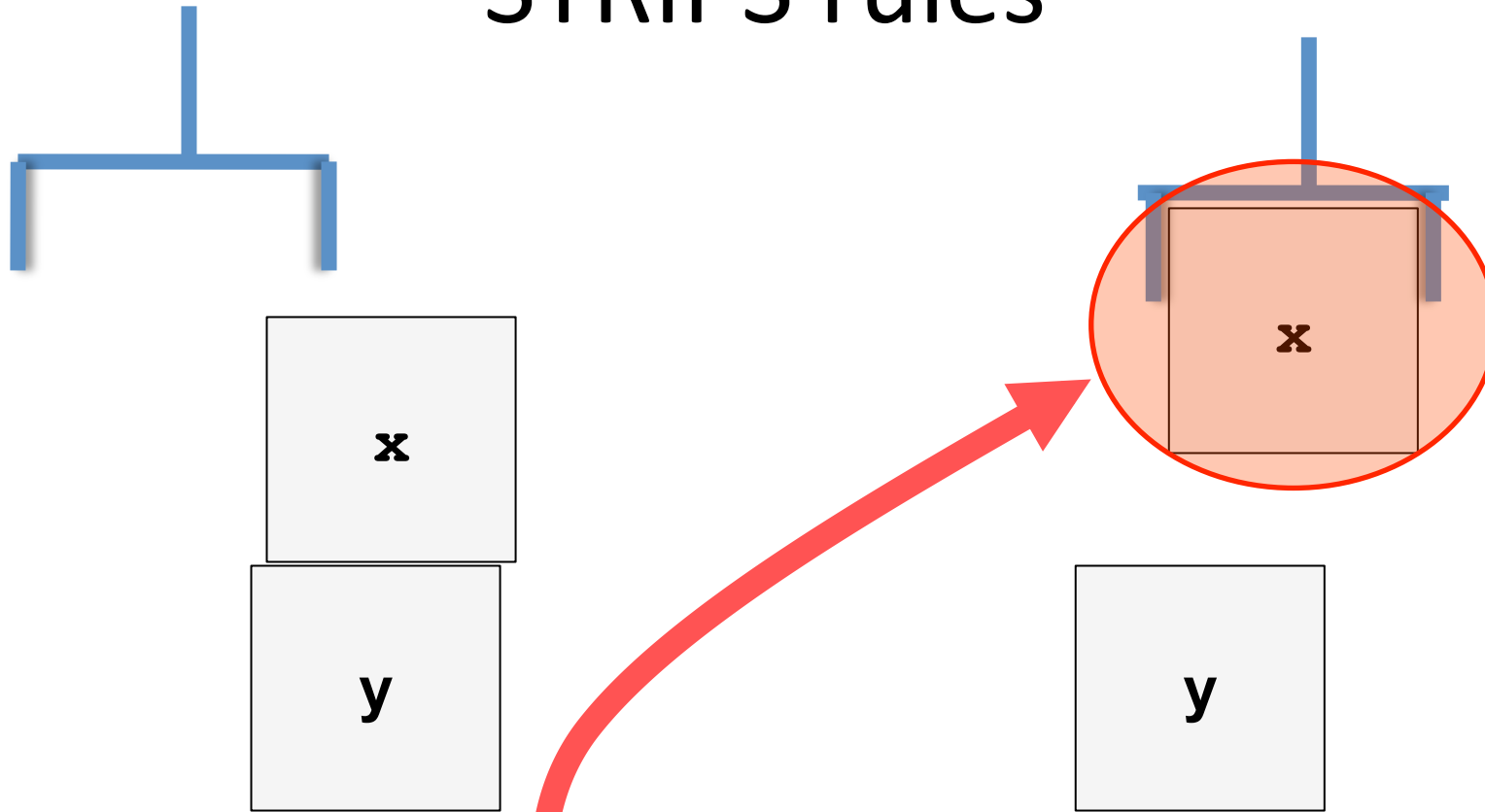


`unstack(x, y)`

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

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

# STRIPS rules



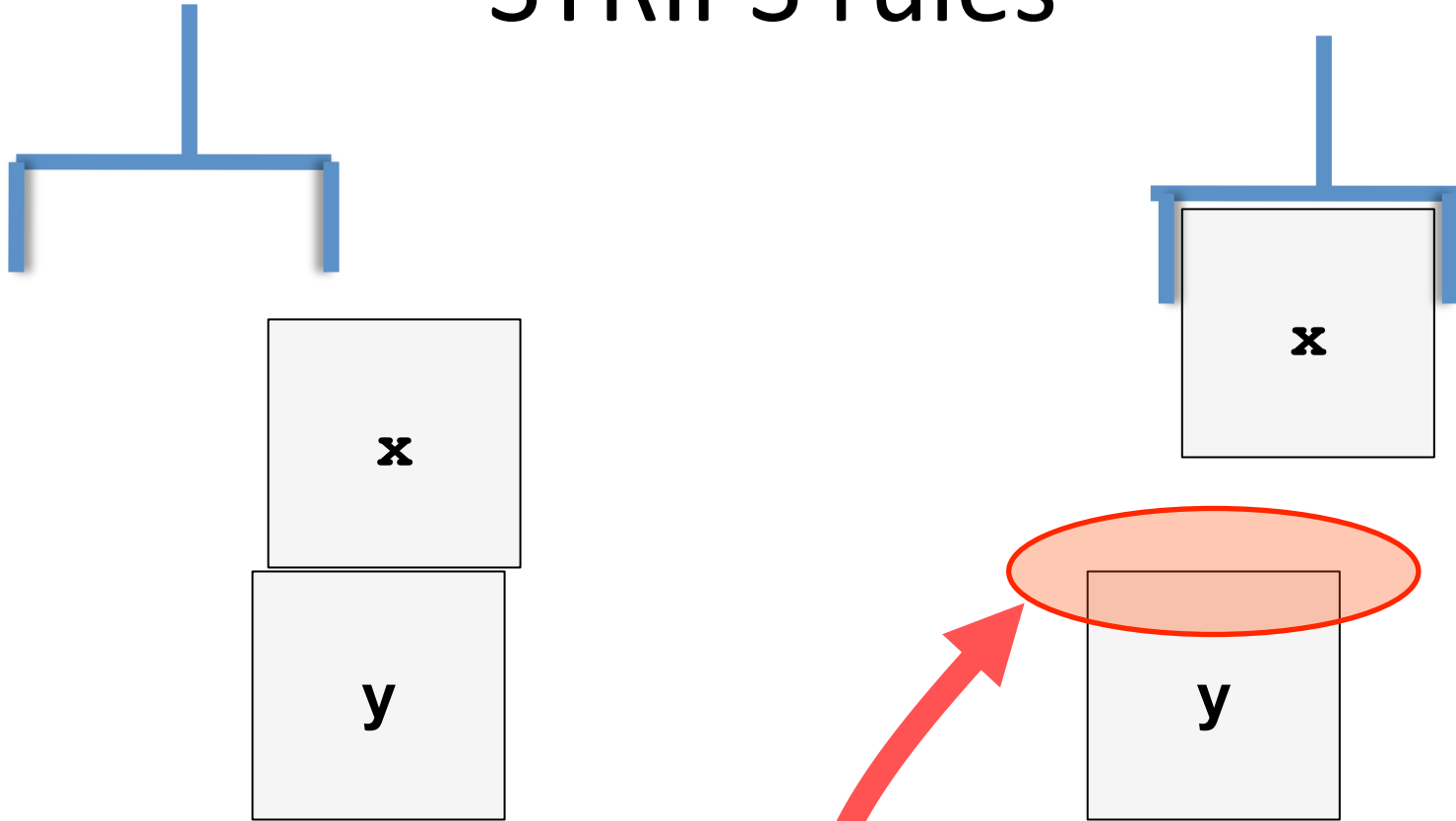
`unstack(x, y)`

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

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



# STRIPS rules

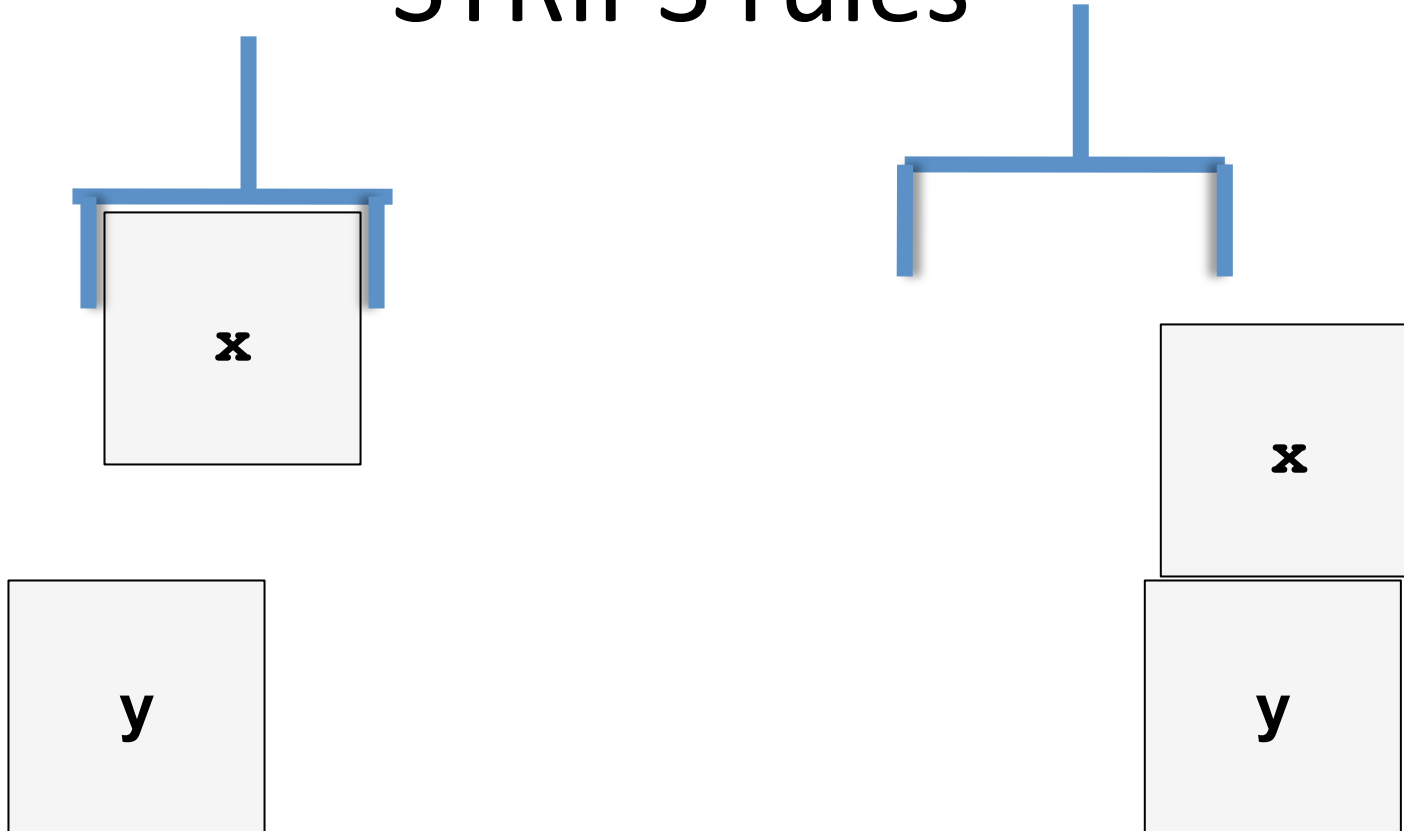


`unstack(x,y)`

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

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

# STRIPS rules

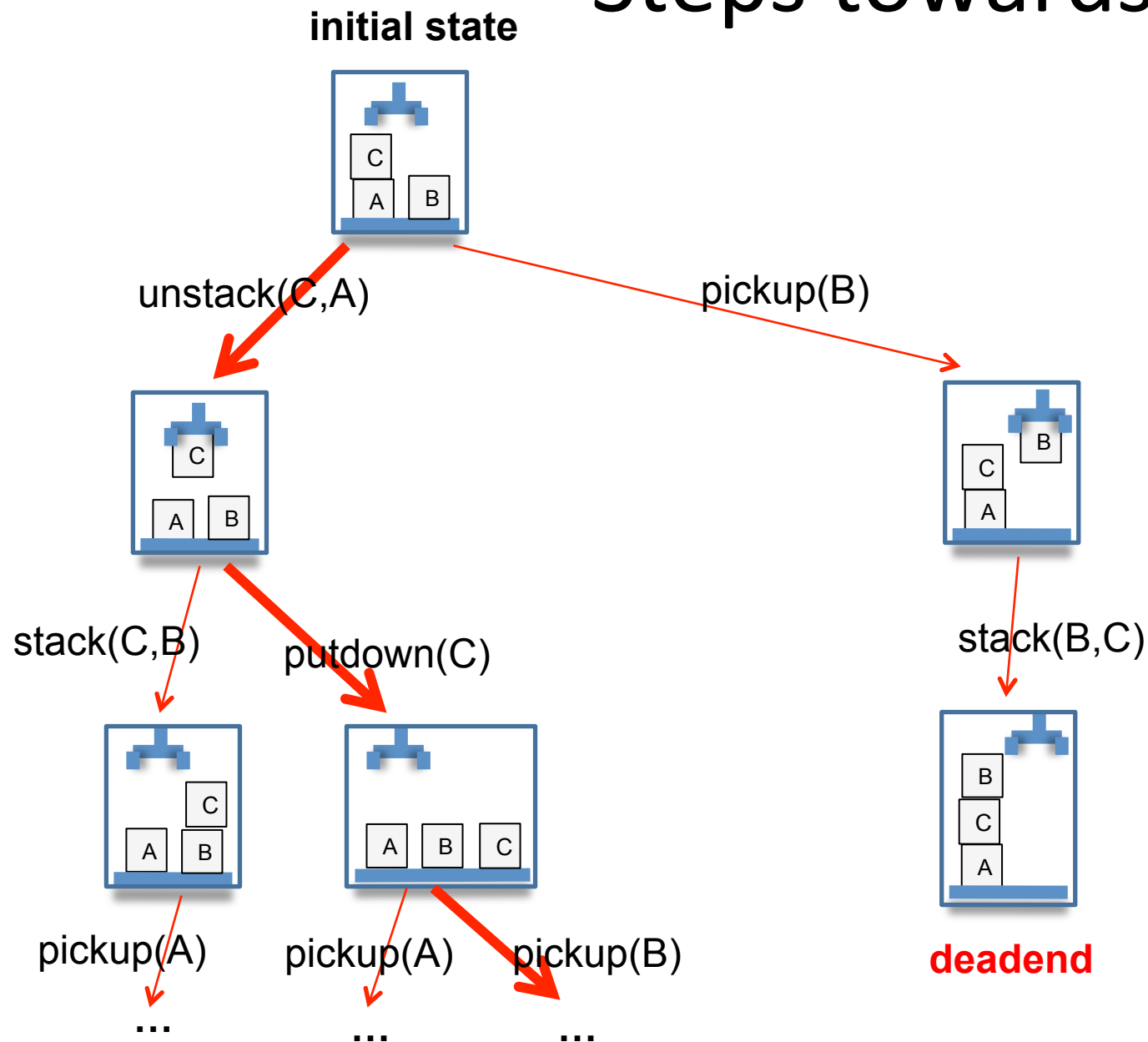


`stack(x,y)`

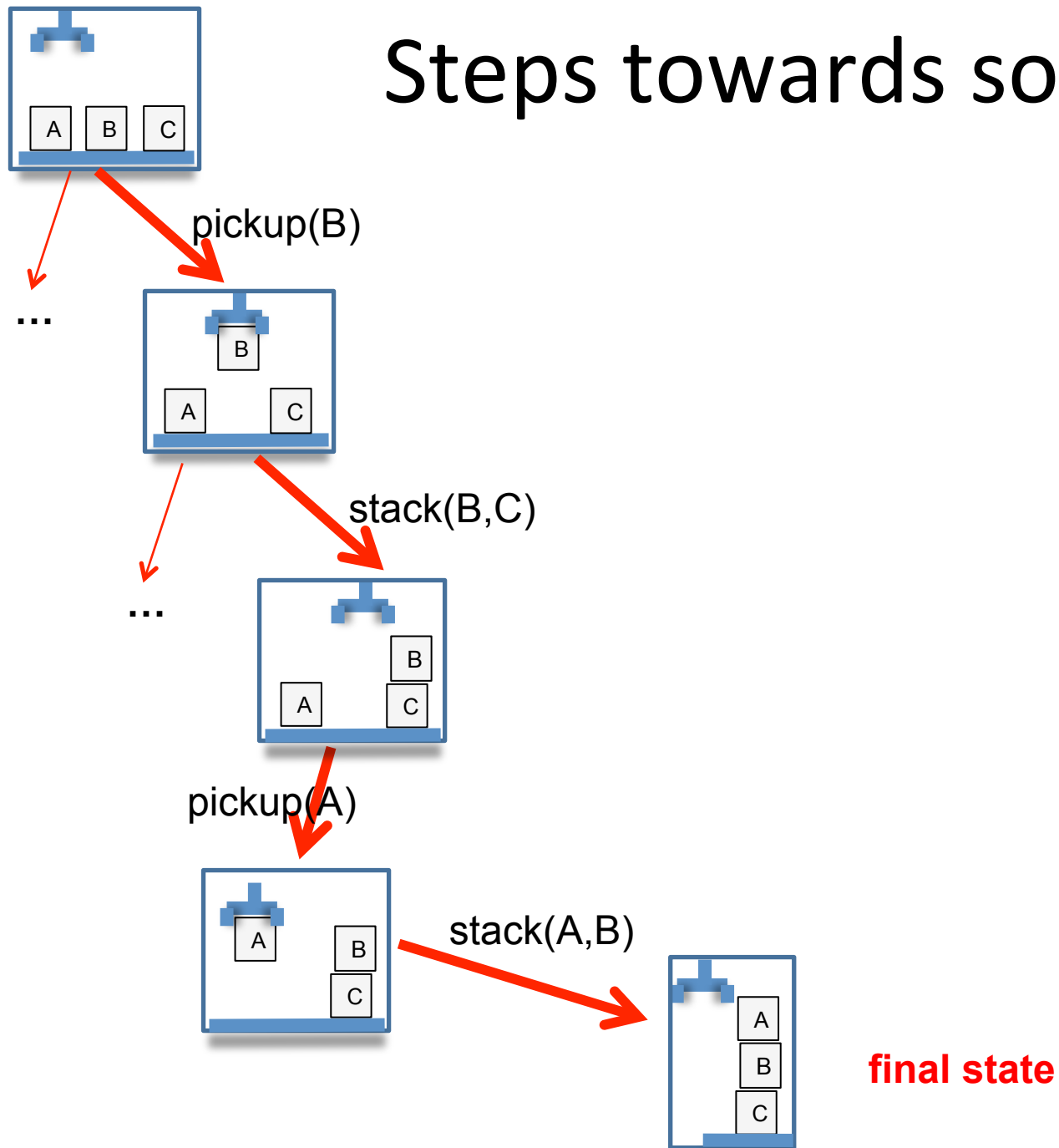
P&D: `holding(x) , clear(y)`

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

# Steps towards solution



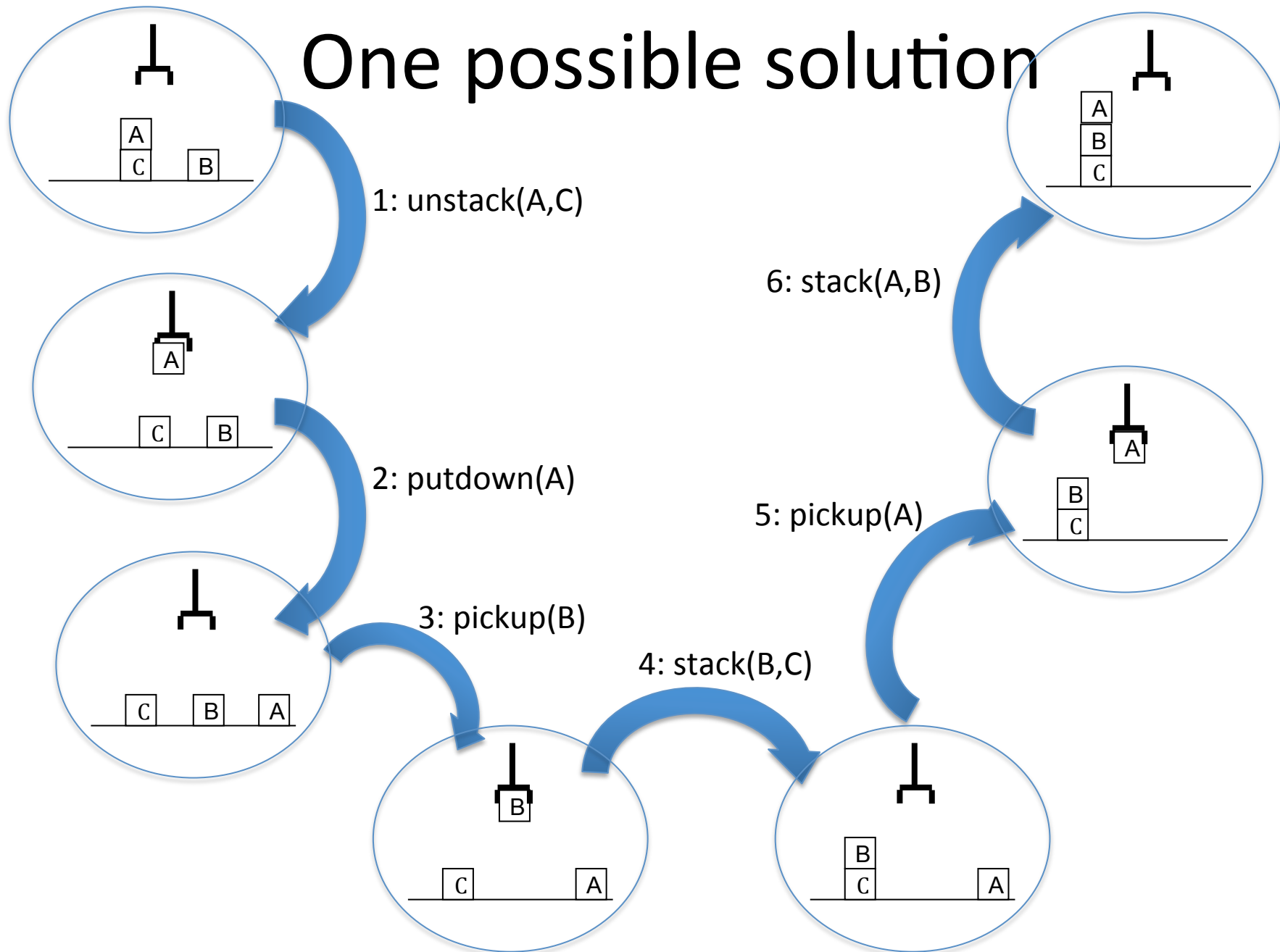
# Steps towards solution



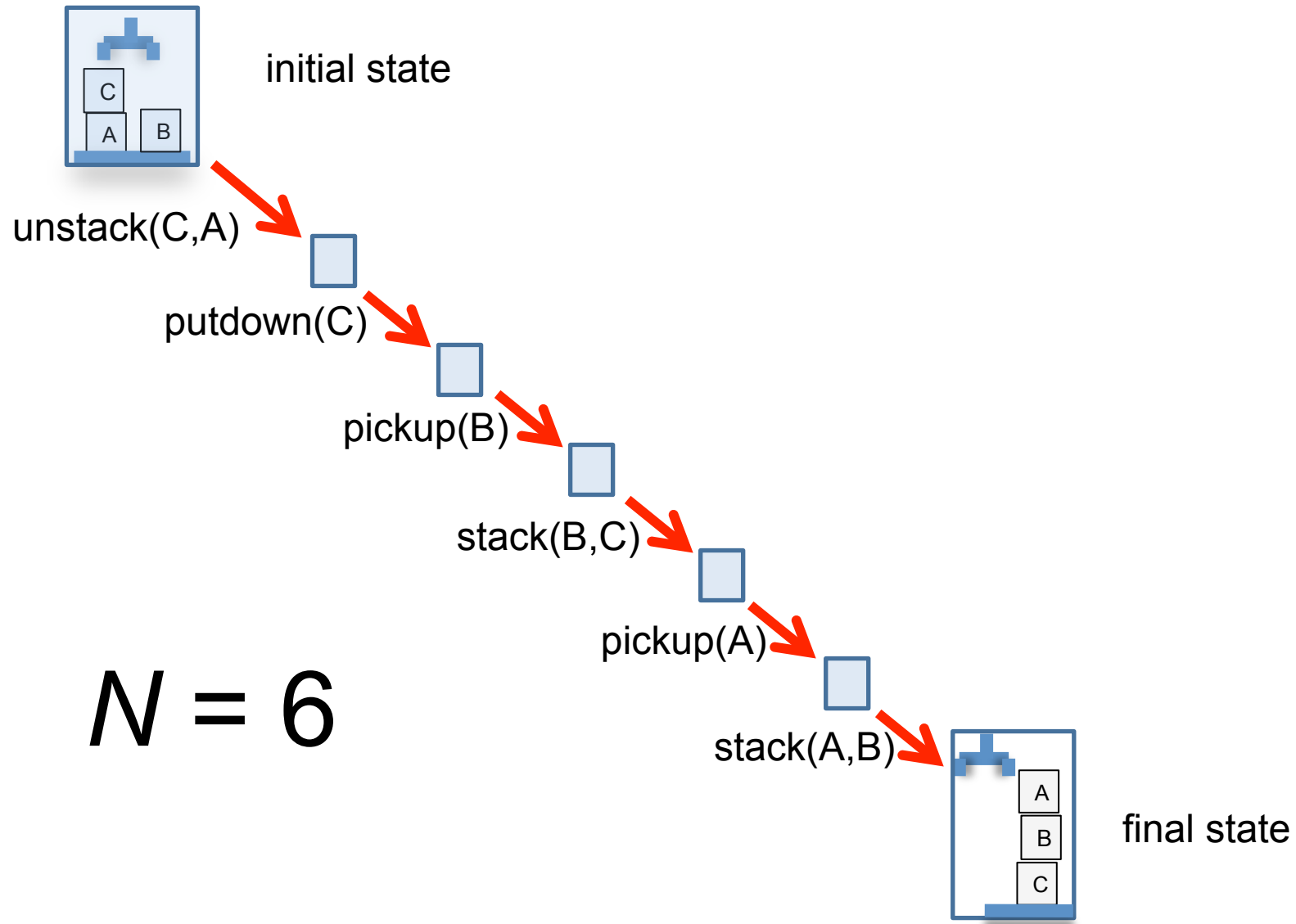
initial state

final state

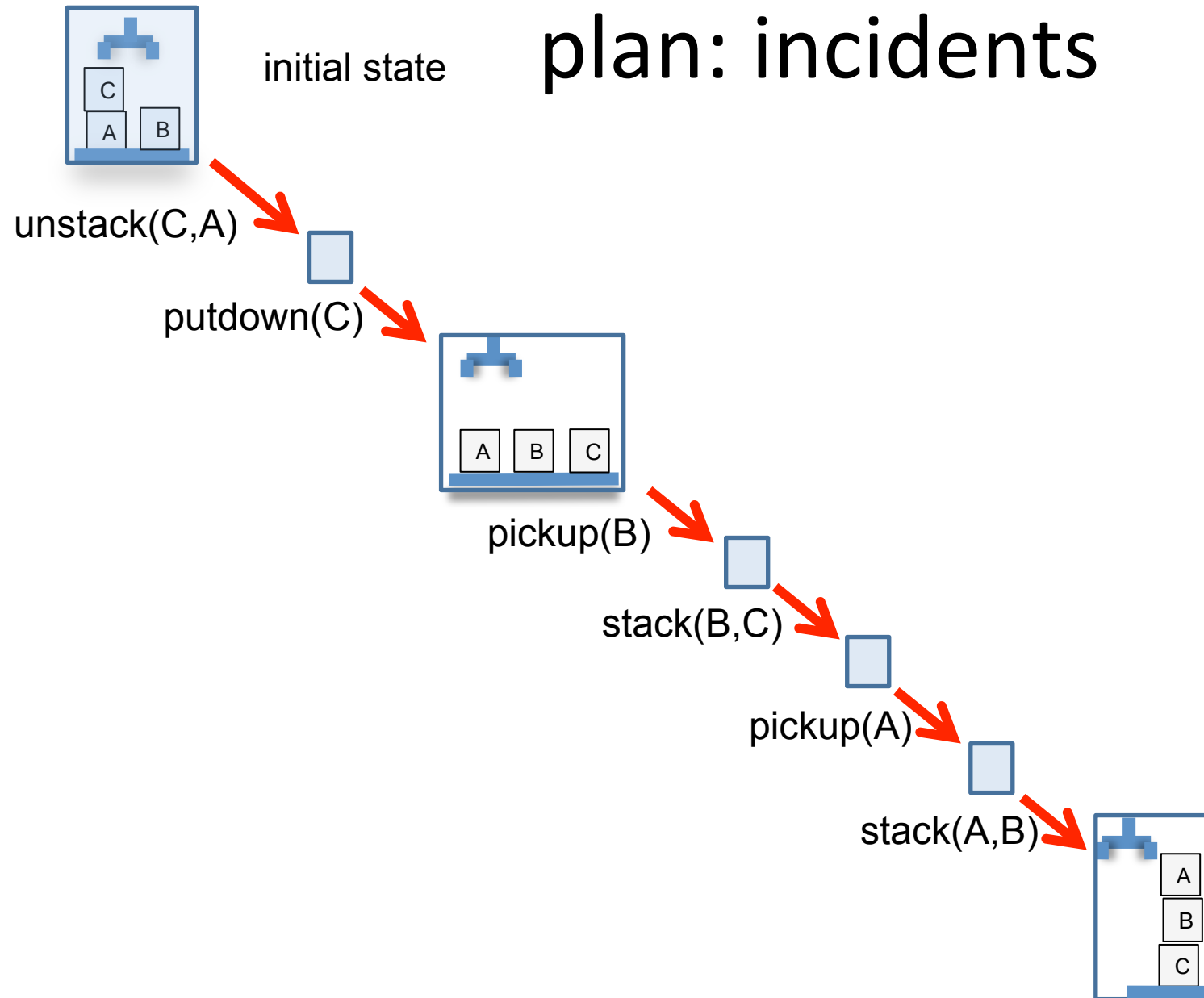
# One possible solution



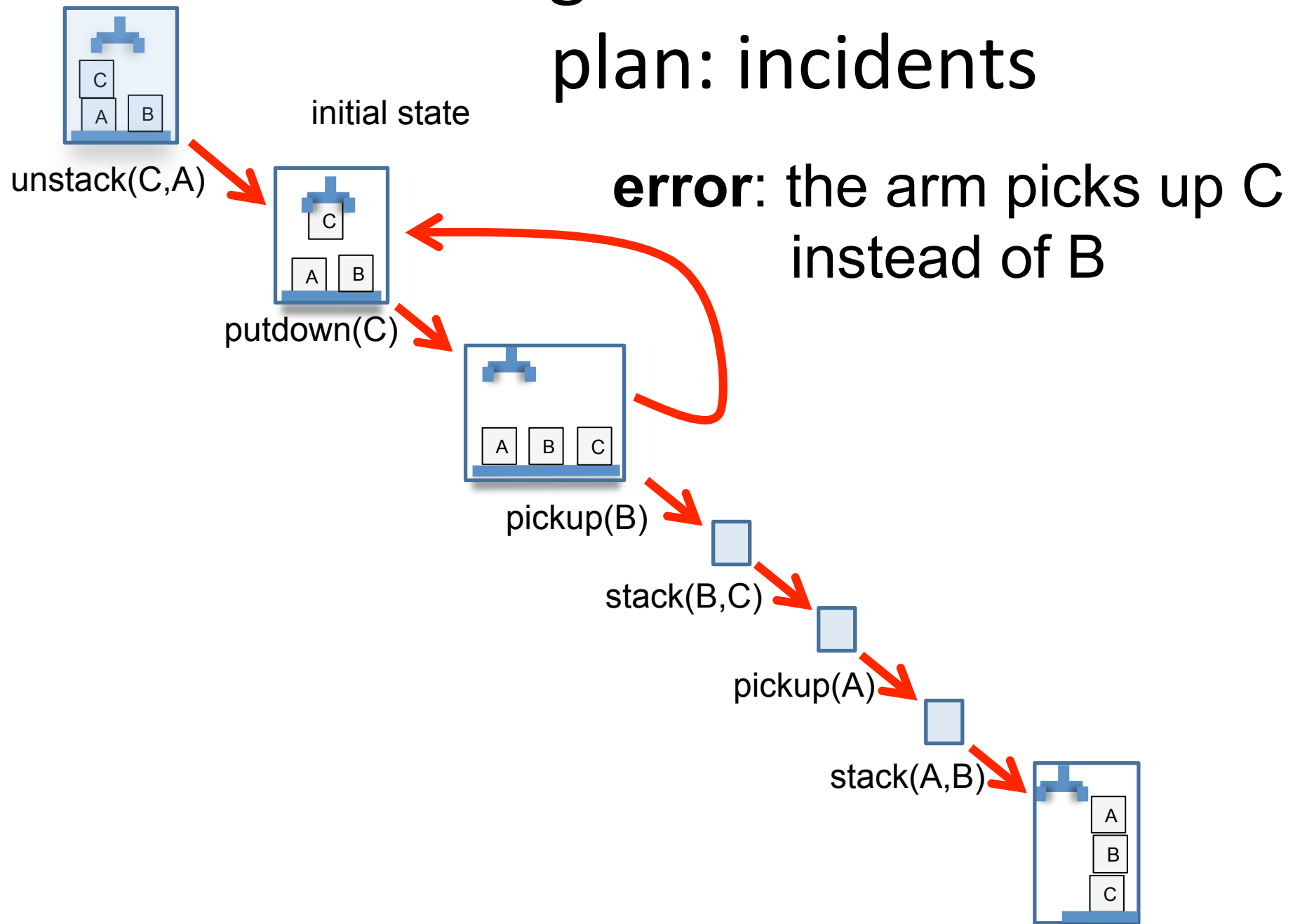
# The solution



# Tracking the execution of the plan: plan: incidents



# Tracking the execution of the plan: incidents

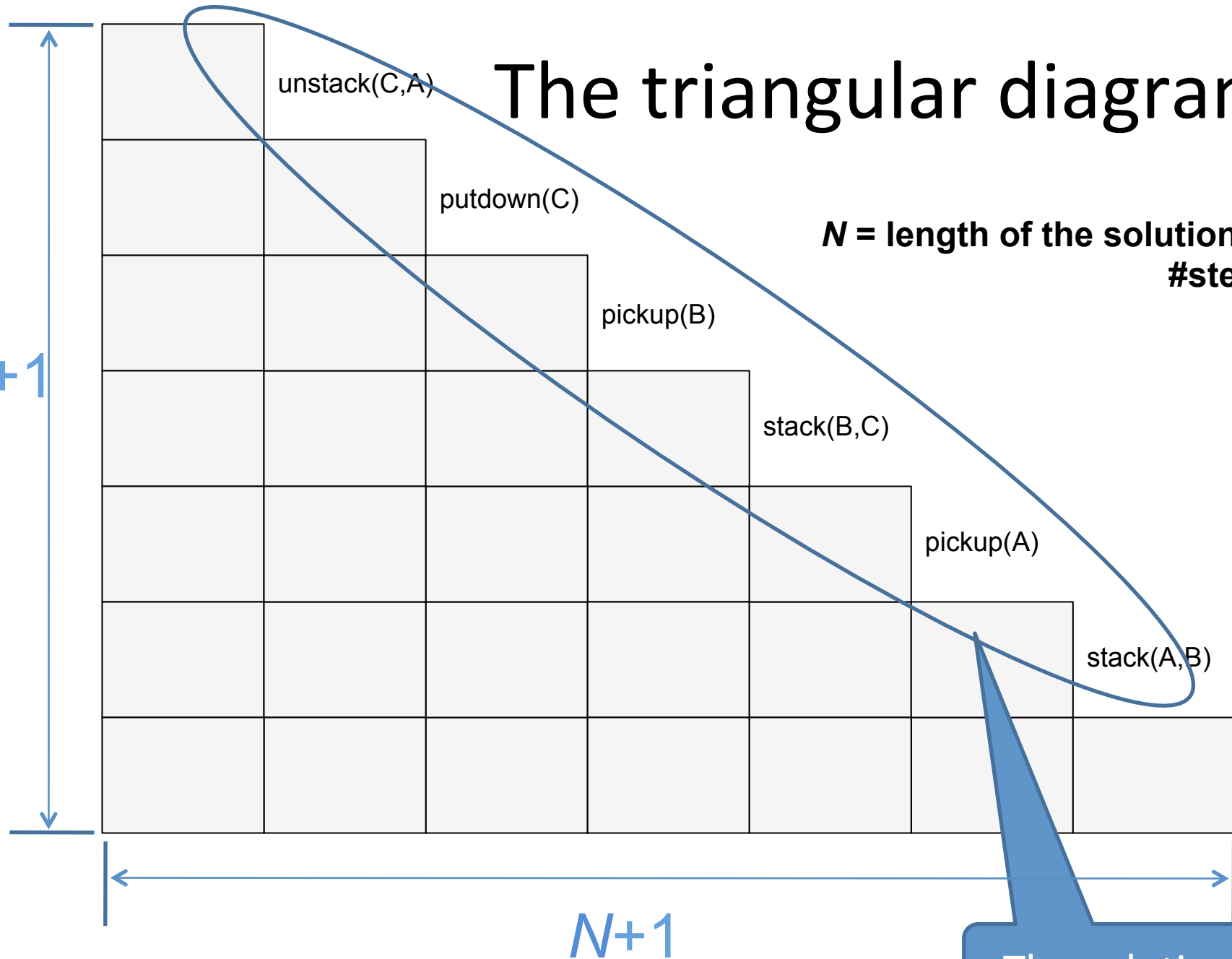




# The triangular diagram

**$N$  = length of the solution in  
#steps**

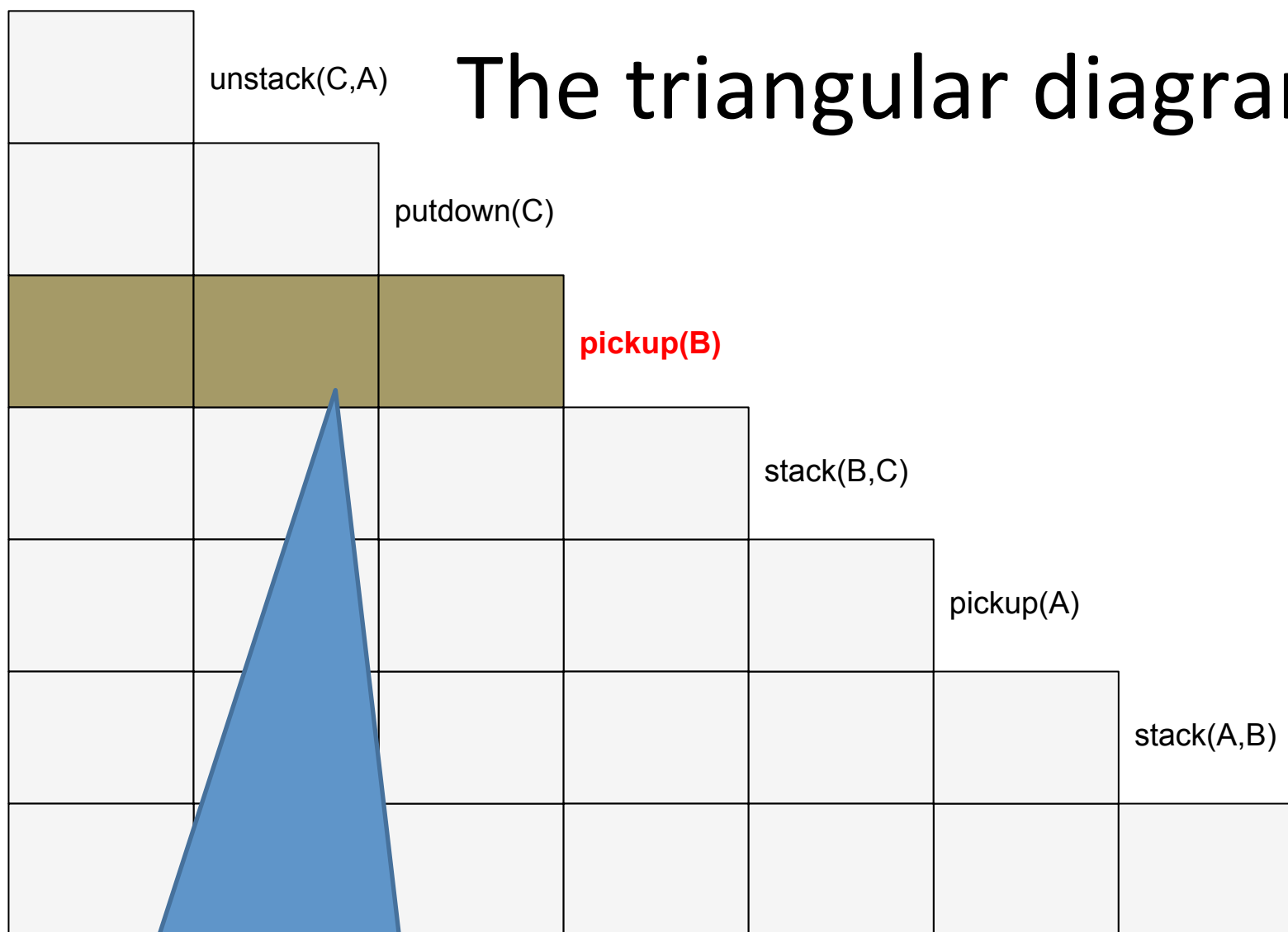
$N+1$



The solution

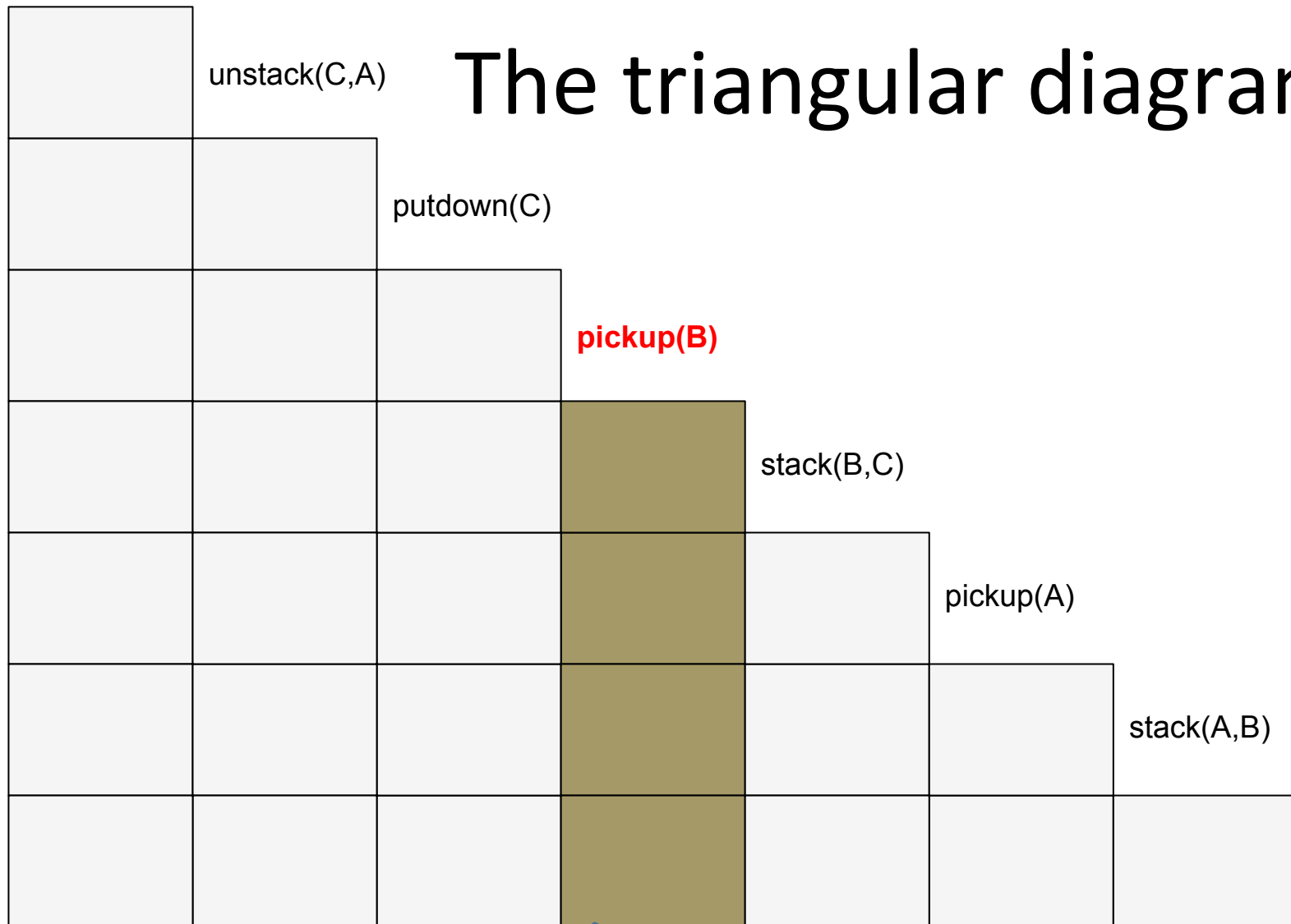
# The triangular diagram

**step  $i$**



These boxes concentrate all predicates  
in the lists P&D of the rule  $i$  step

# The triangular diagram

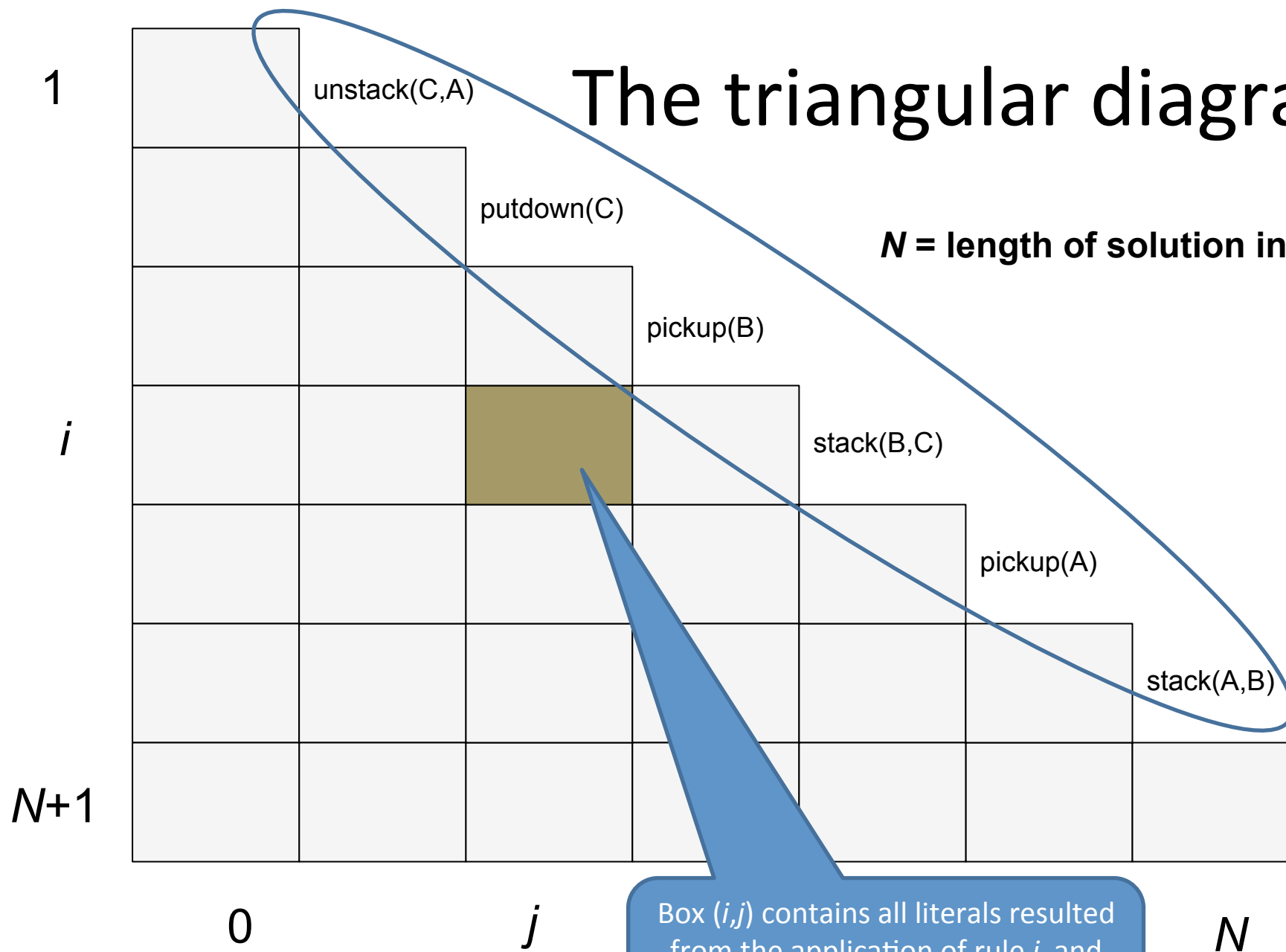


**step  $i$**

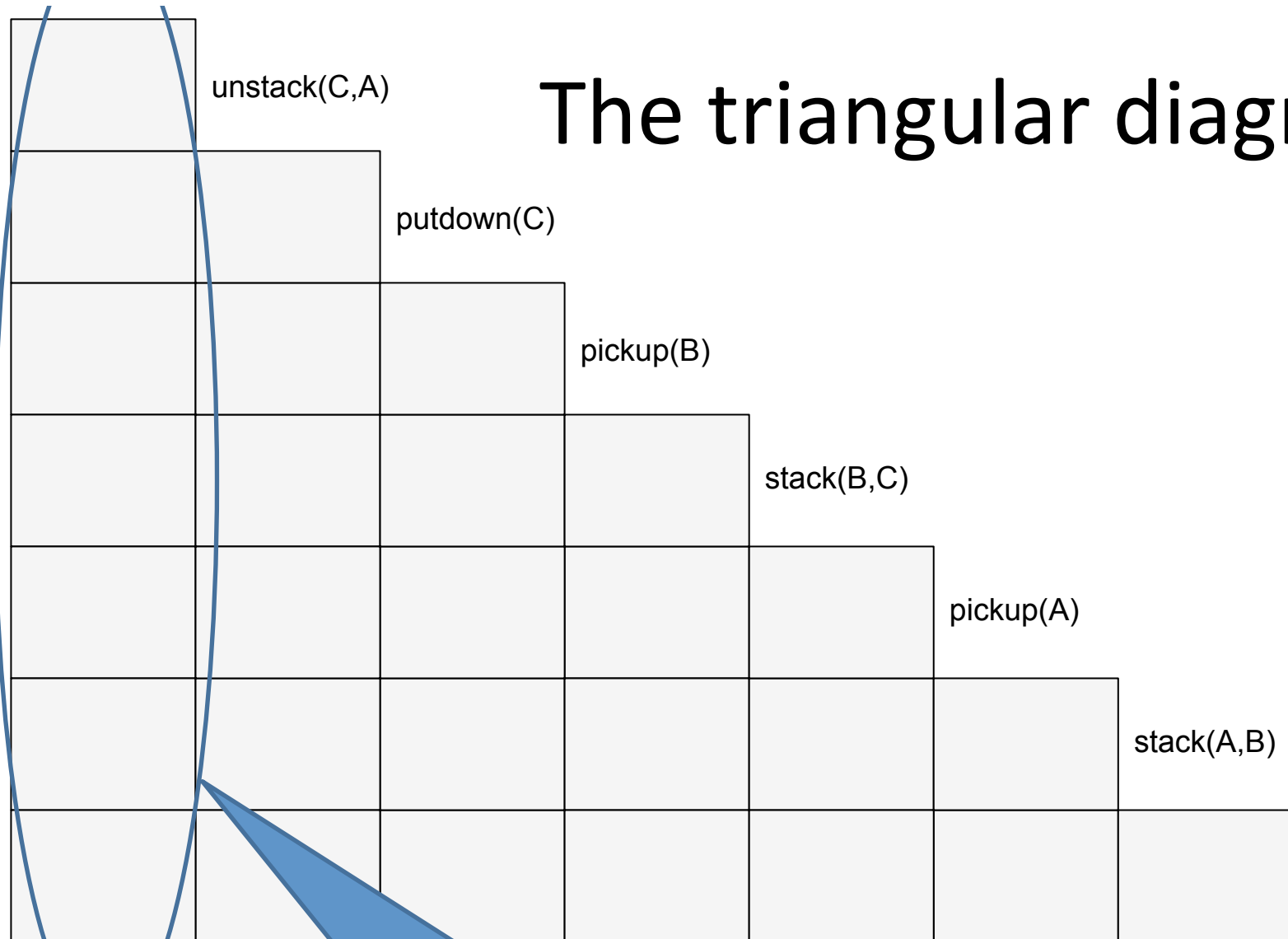
These boxes concentrate all predicates  
in the list  $A$  of the rule  $i$  step

# The triangular diagram

$N$  = length of solution in #steps

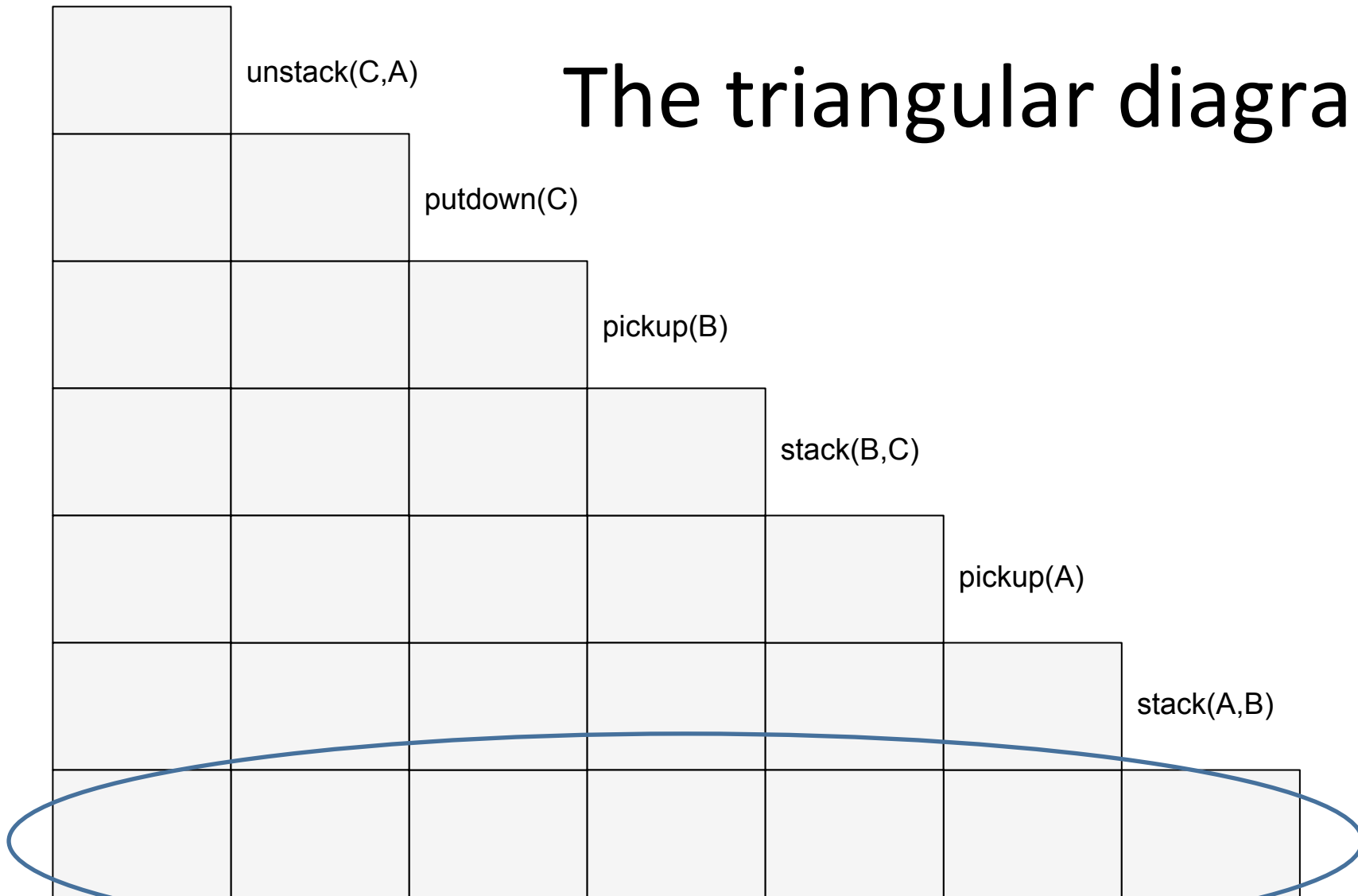


# The triangular diagram



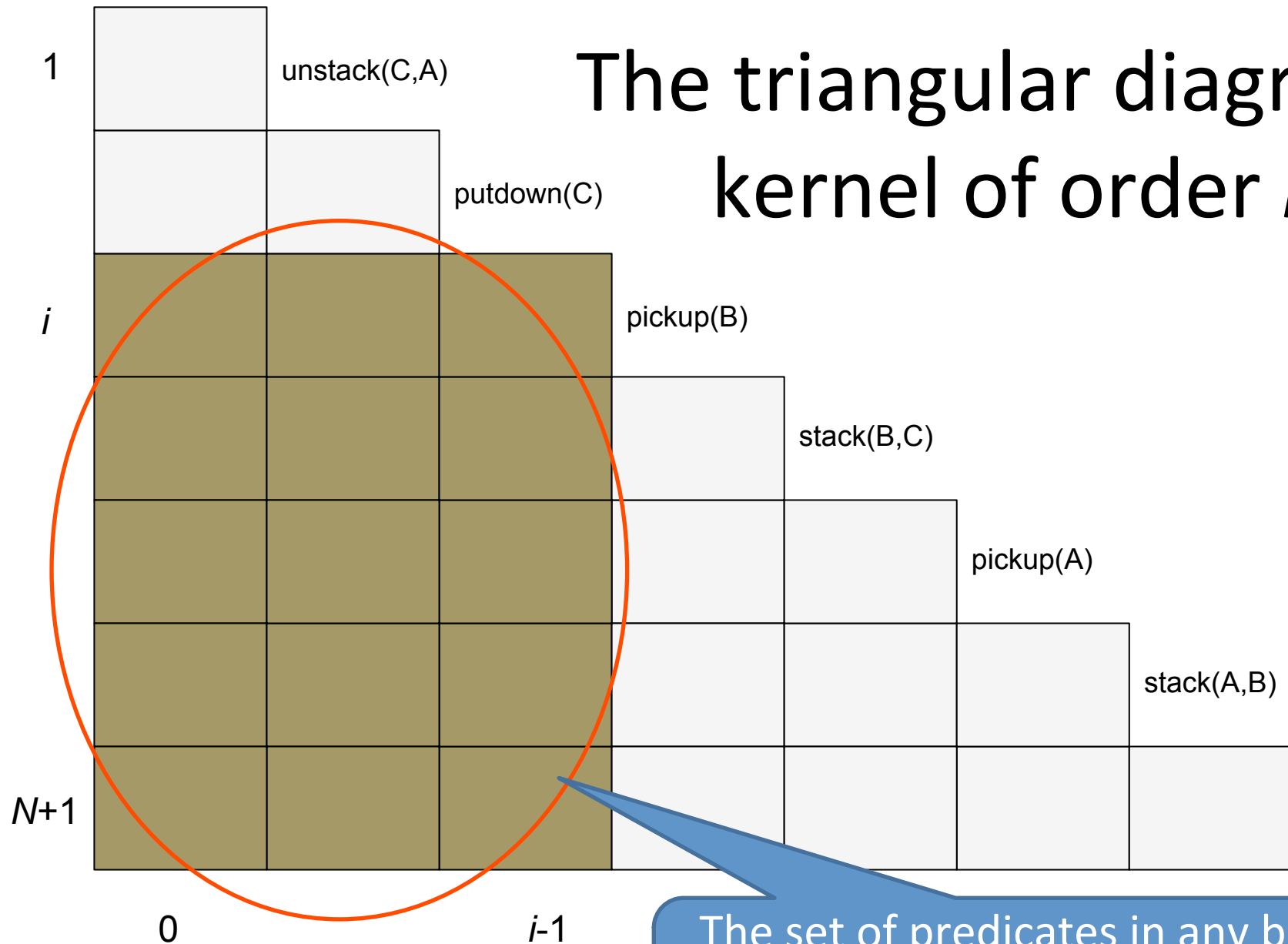
The first column concentrates all predicates in the initial state.

# The triangular diagram



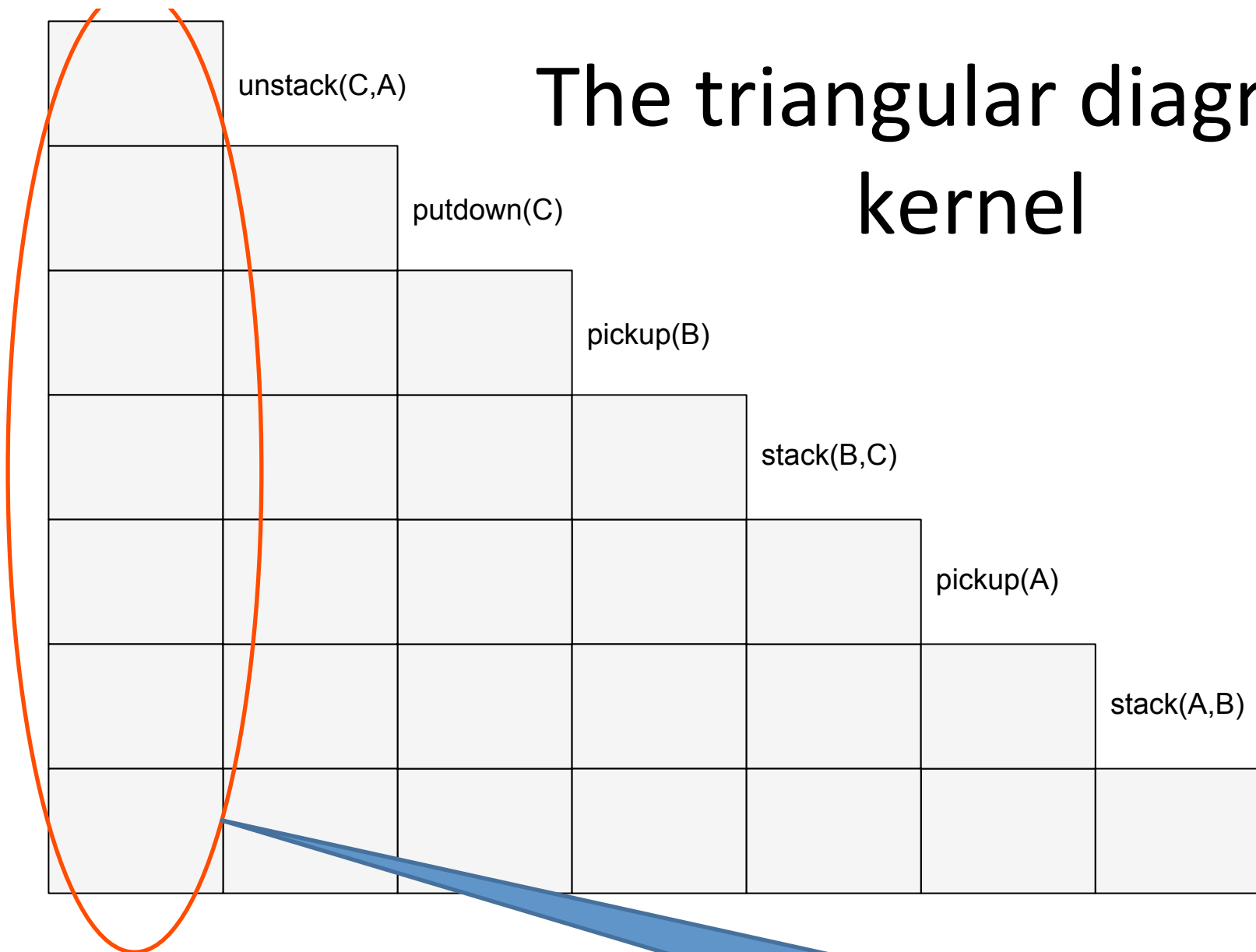
The last line concentrates all predicates in the final state.

# The triangular diagram: kernel of order $i$



The set of predicates in any box  $[i, N+1] \times [0, i-1]$ : the state after the execution of rule  $i-1$

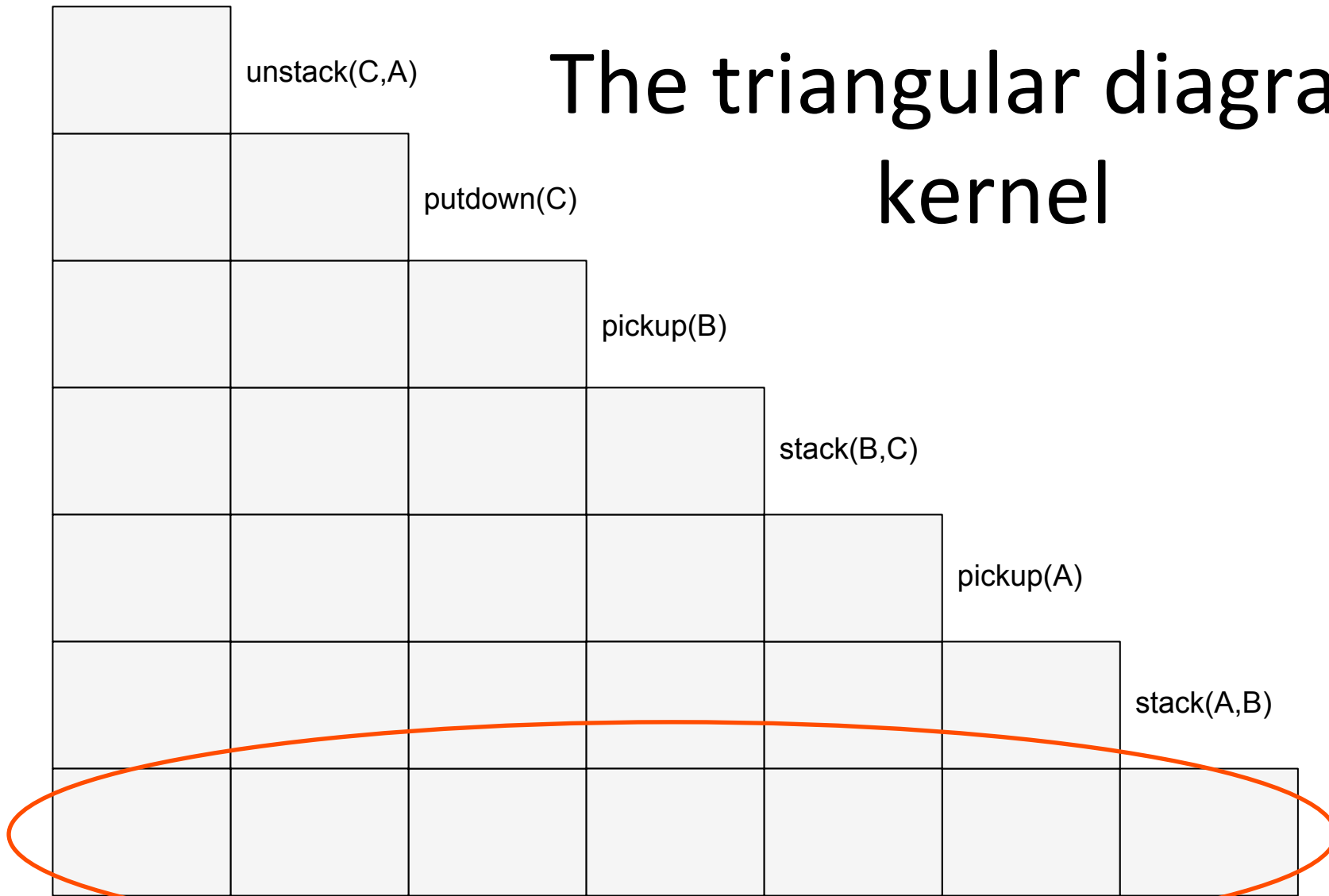
# The triangular diagram: kernel



Initial state: the kernel of order 1

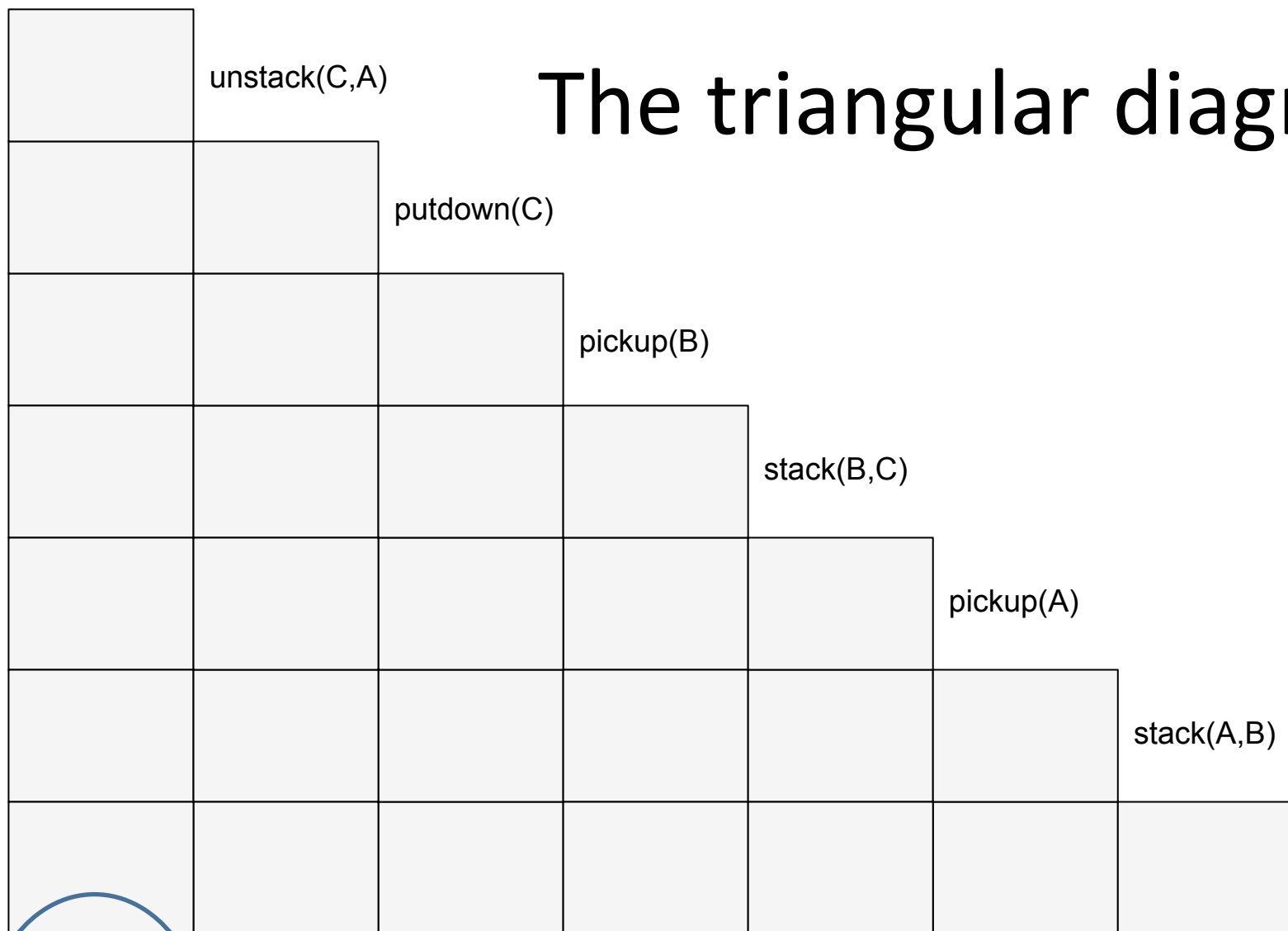


# The triangular diagram: kernel



Final state: the kernel of order  $N+1$

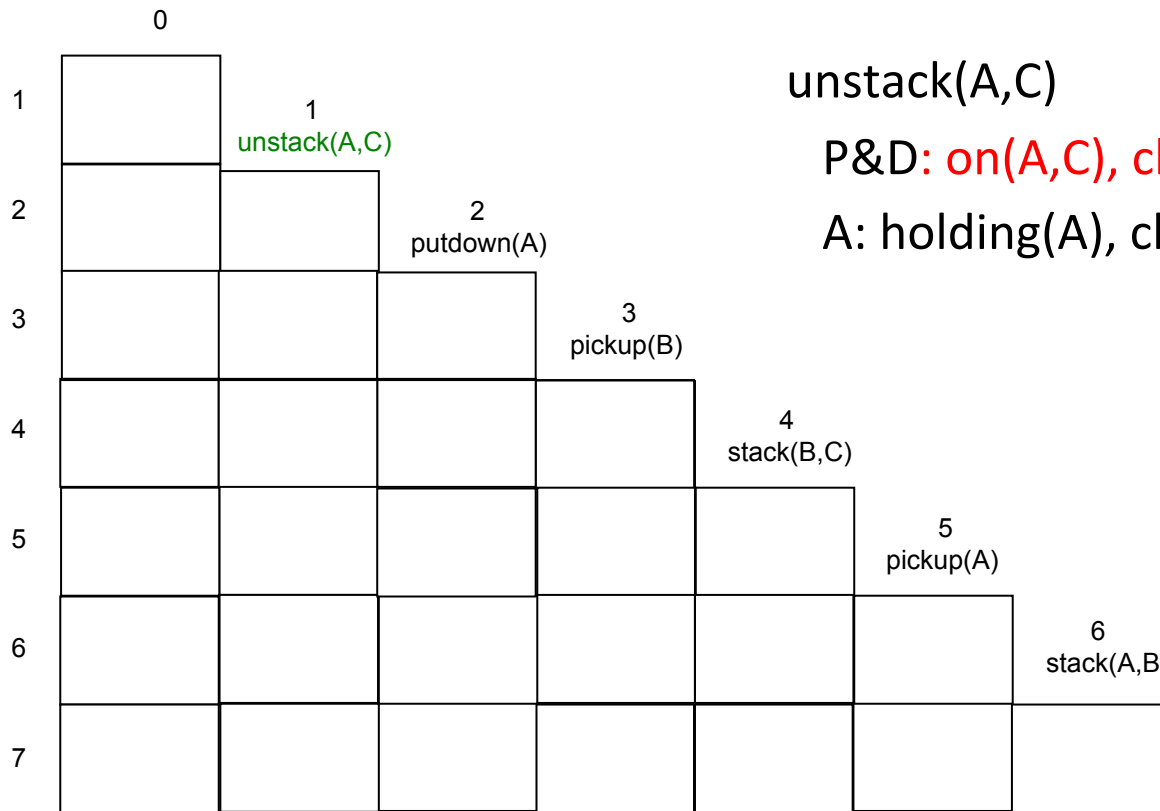
# The triangular diagram



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

The initial state should  
be placed here.

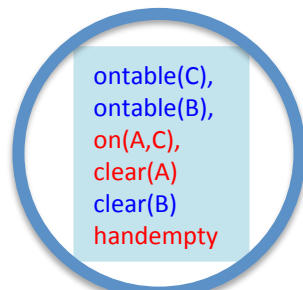
# Plan execution: step 1



unstack(A,C)

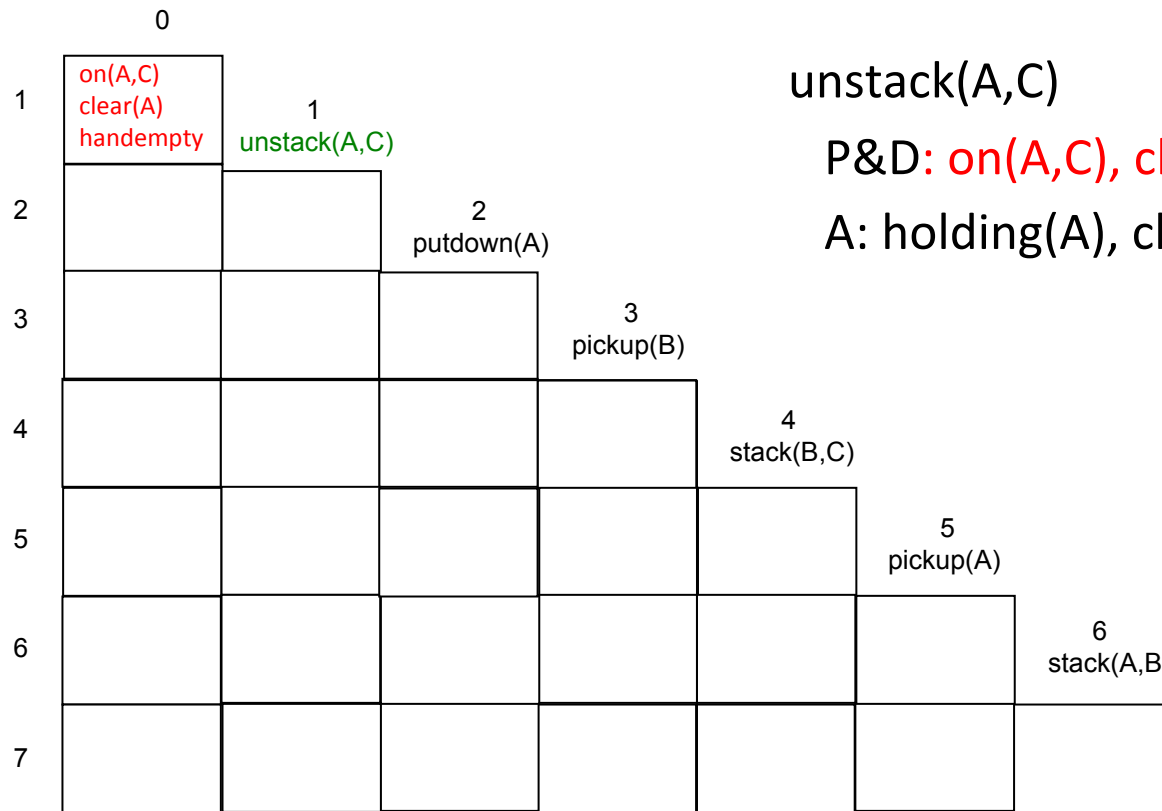
P&D: on(A,C), clear(A), handempty

A: holding(A), clear(C)



Out of the predicates of the initial state look for those that are part of the P&D list of rule 1

# Plan execution: step 1



unstack(A,C)

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

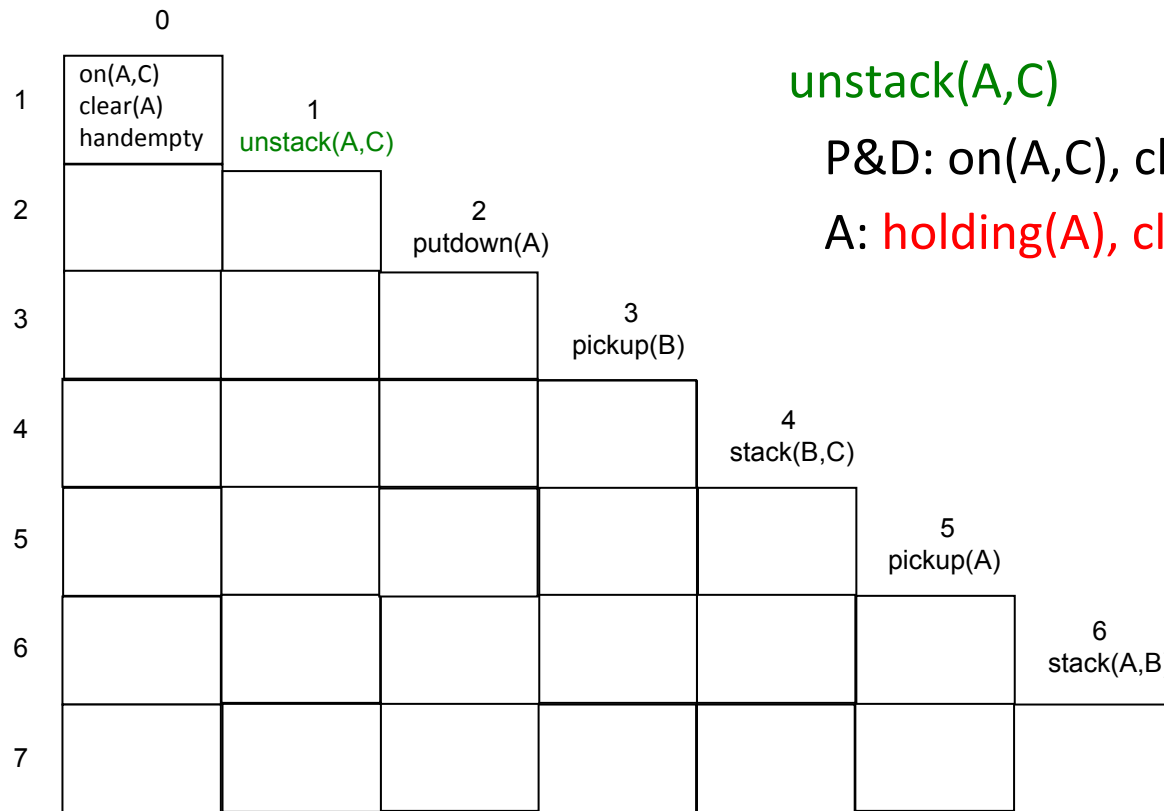
A: holding(A), clear(C)



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

... and bring them up in the line  
of rule 1.

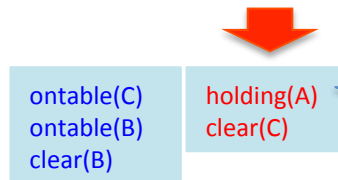
# Plan execution: step 1



unstack(A,C)

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

A: holding(A), clear(C)



Collect here all predicates of list  
A of rule 1.

# Plan execution: step 2

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

putdown(A)

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

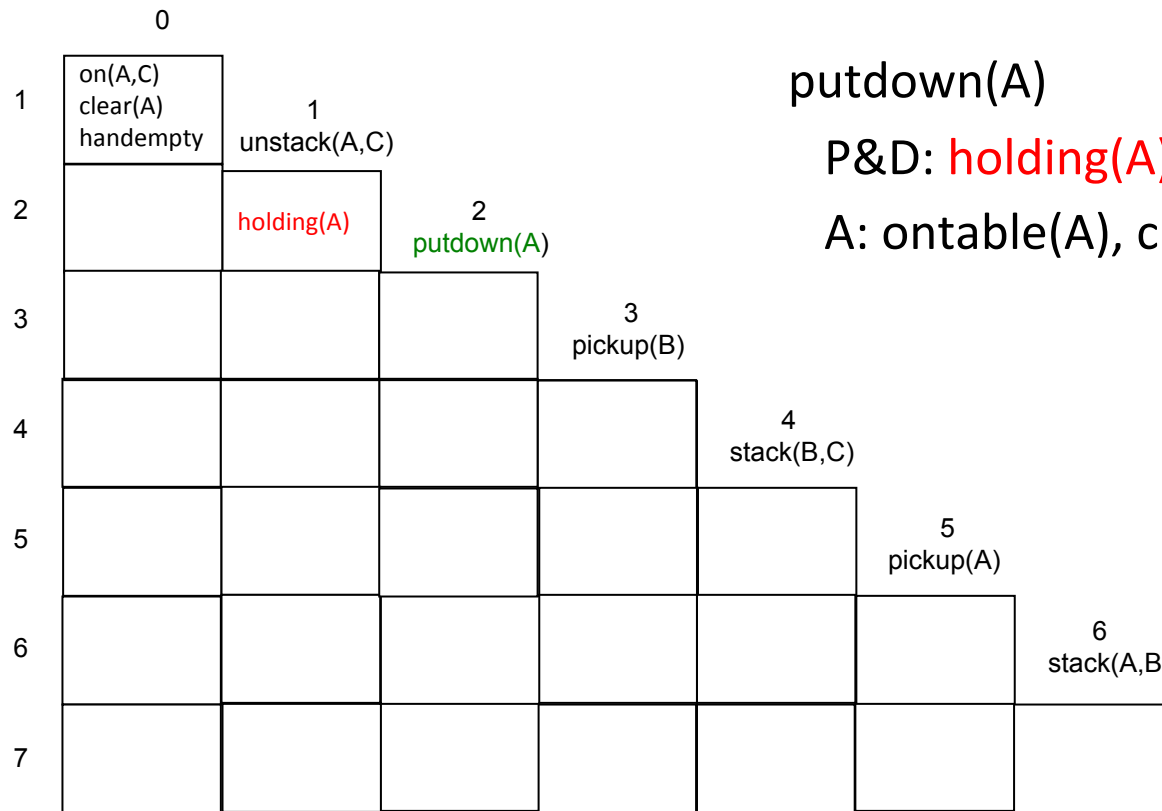
A: ontable(A), clear(A), handempty

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

**holding(A)**  
clear(C)

Out of all predicates remained down  
look for those in the P&D list of rule 2.

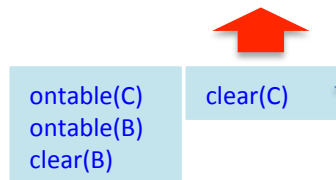
# Plan execution: step 2



putdown(A)

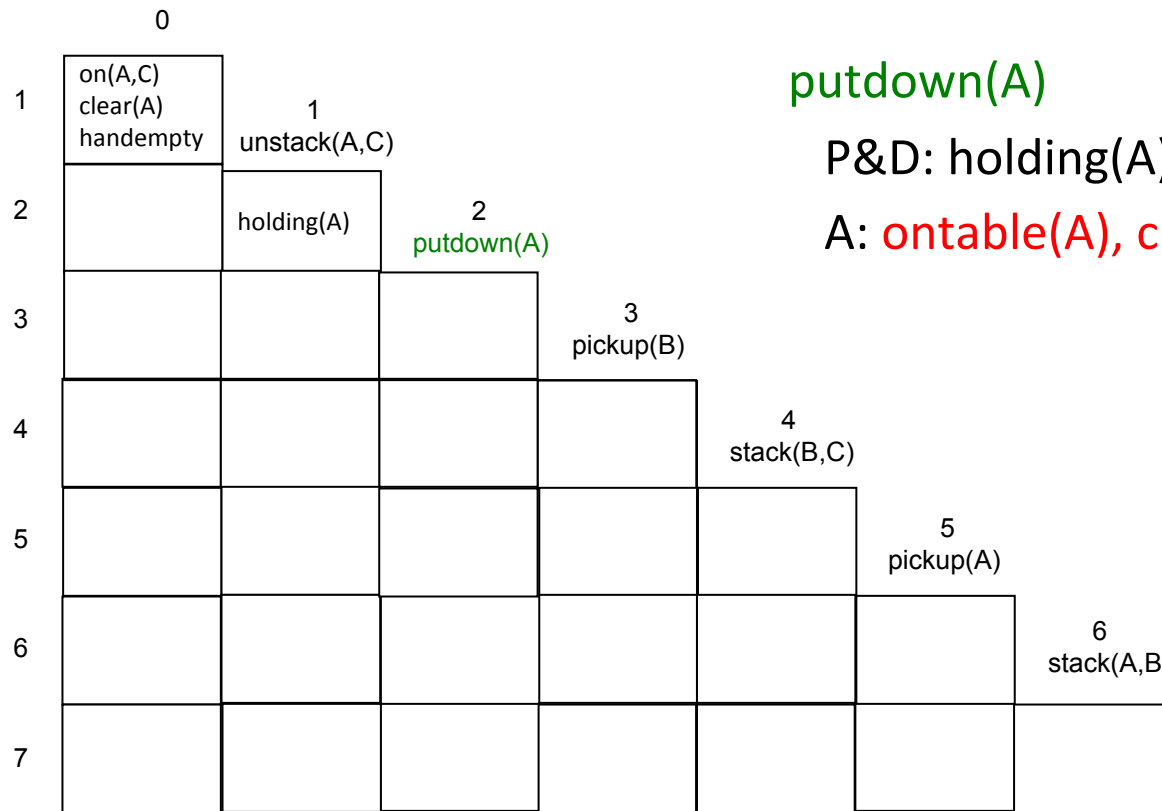
P&D: **holding(A)**

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



... and bring them up in the line of rule 2.

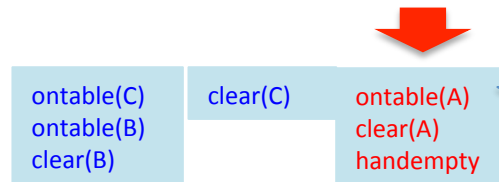
# Plan execution: step 2



putdown(A)

P&D: holding(A)

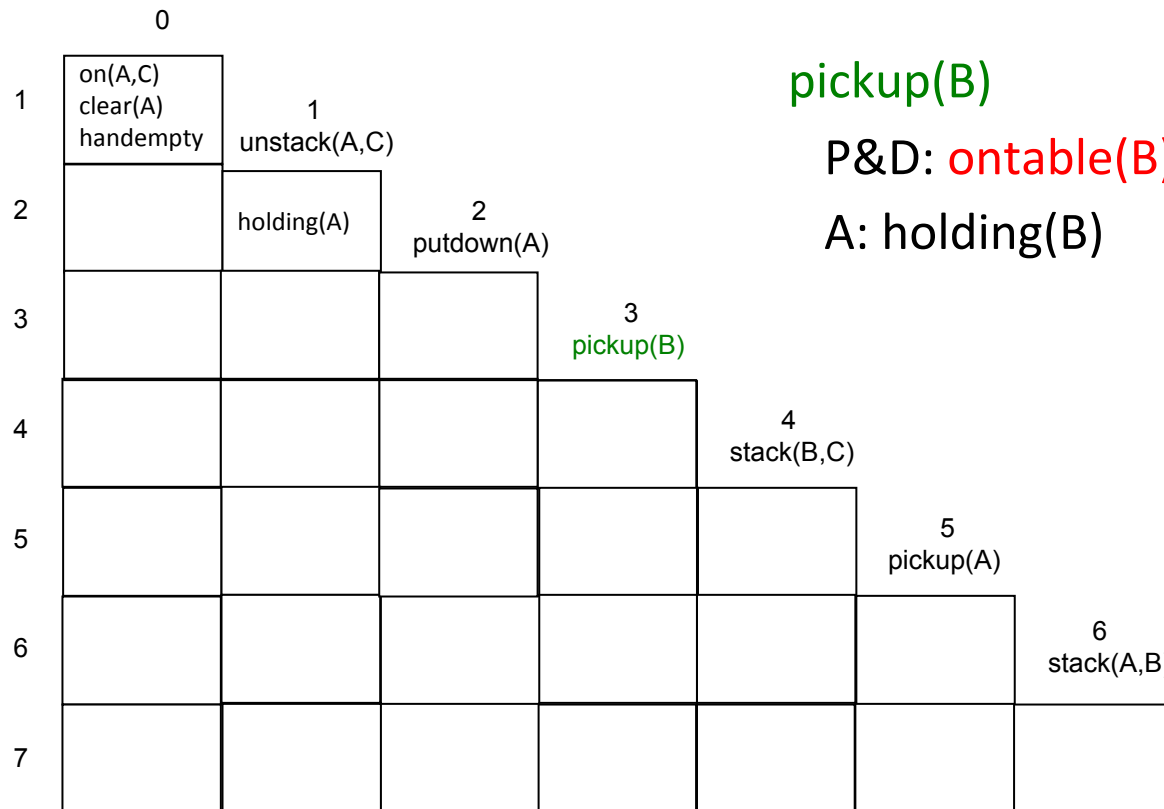
A: ontable(A), clear(A), handempty



Collect here all predicates of list A of rule 2.



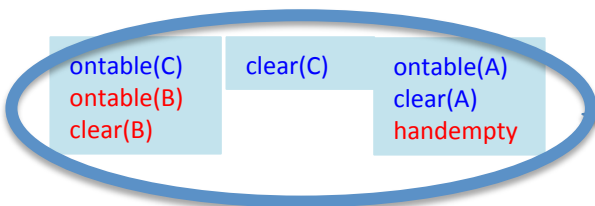
# Plan execution: step 3



pickup(B)

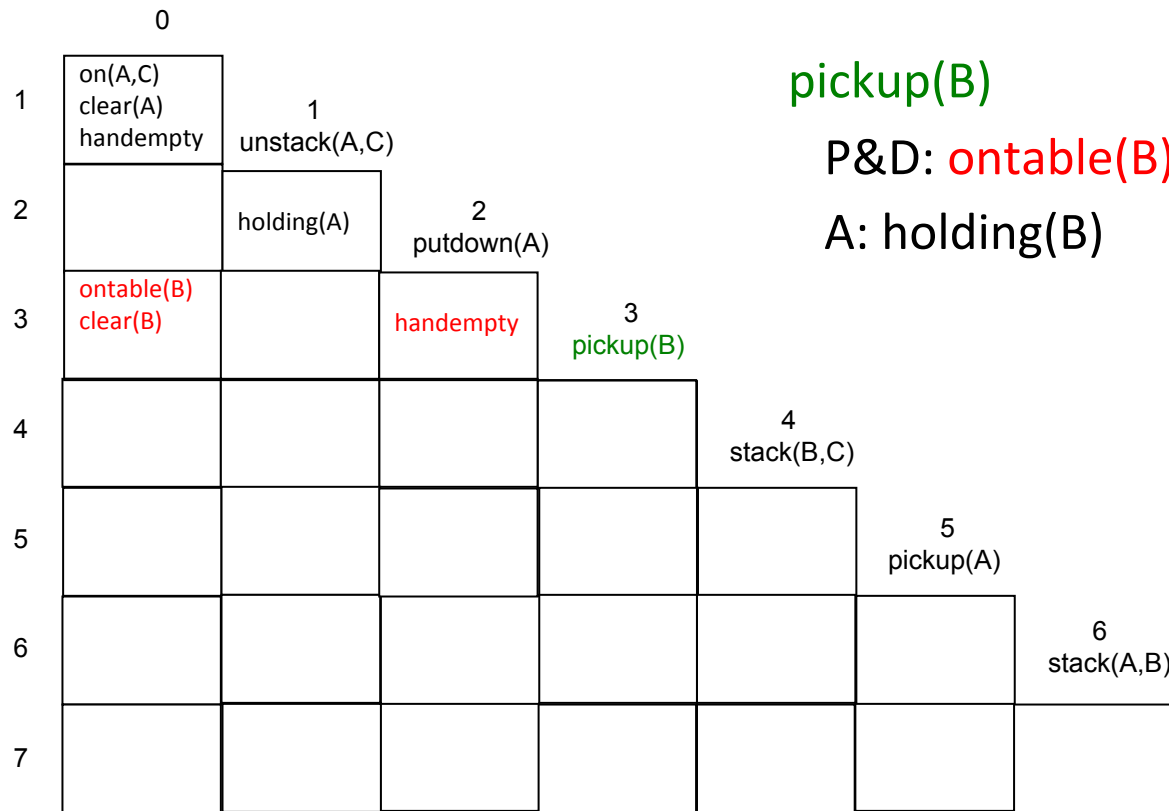
P&D: **ontable(B)**, **clear(B)**, **handempty**

A: holding(B)



Out of all predicates remained down look for those in the P&D list of rule 3.

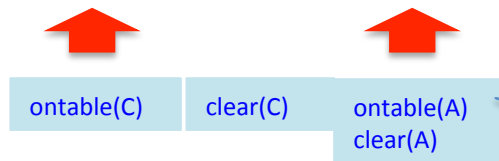
# Plan execution: step 3



pickup(B)

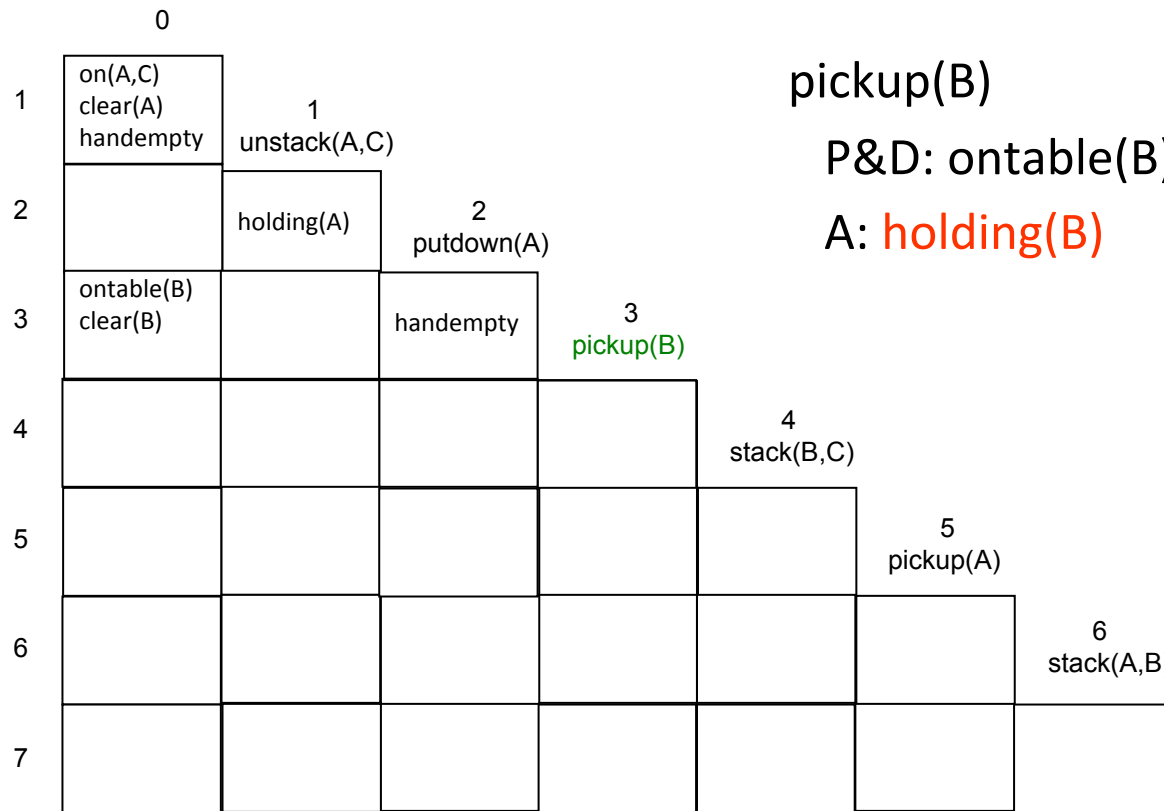
P&D: ontable(B), clear(B), handempty

A: holding(B)



... and bring them up in the line of rule 3

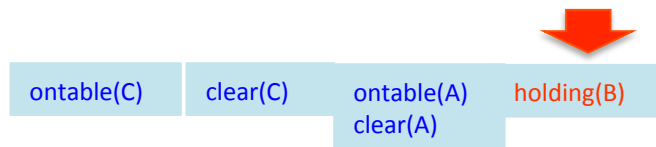
# Plan execution: step 3



pickup(B)

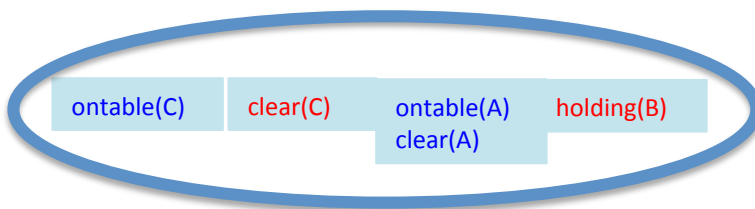
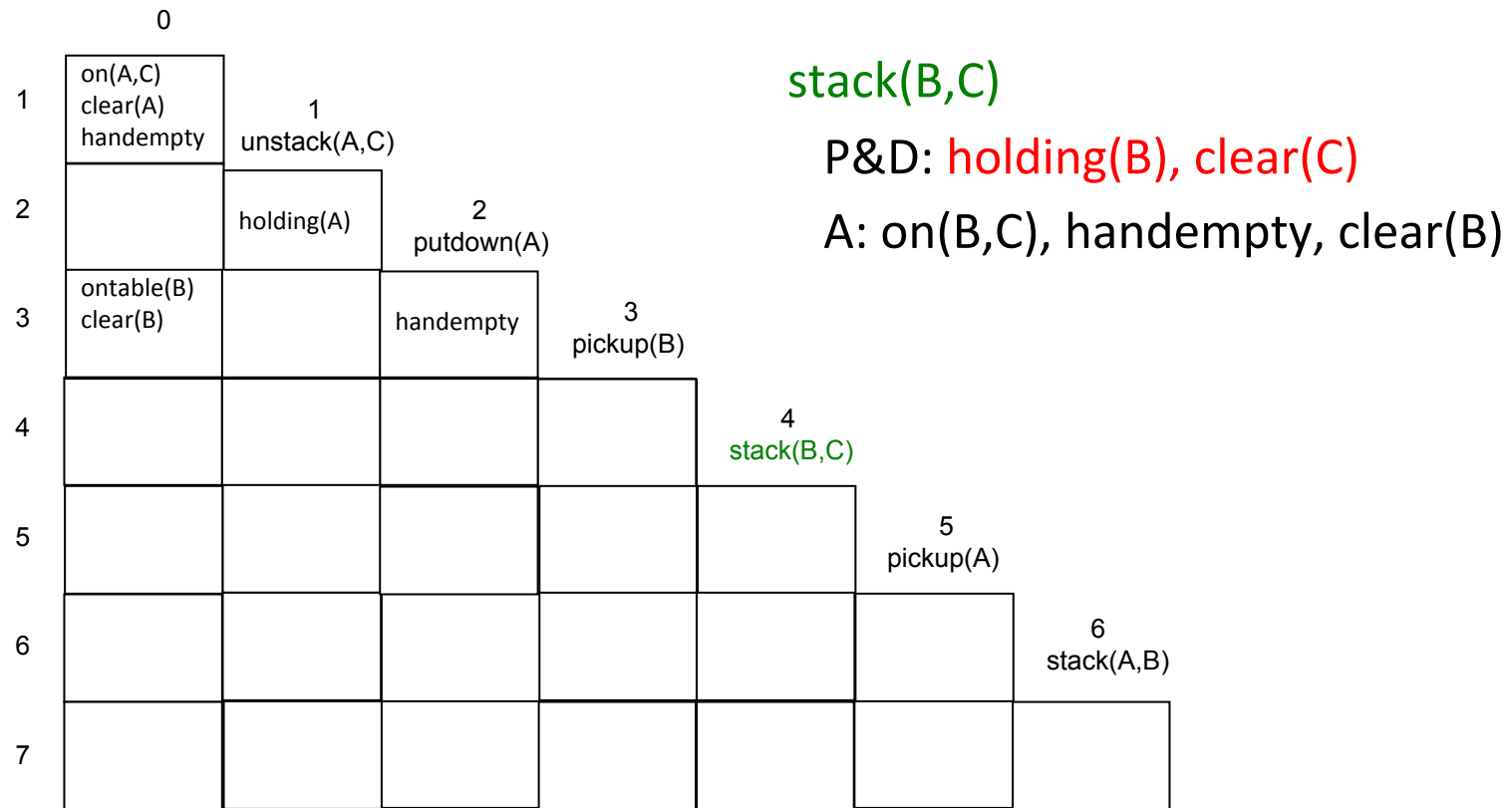
P&D: ontable(B), clear(B), handempty

A: holding(B)



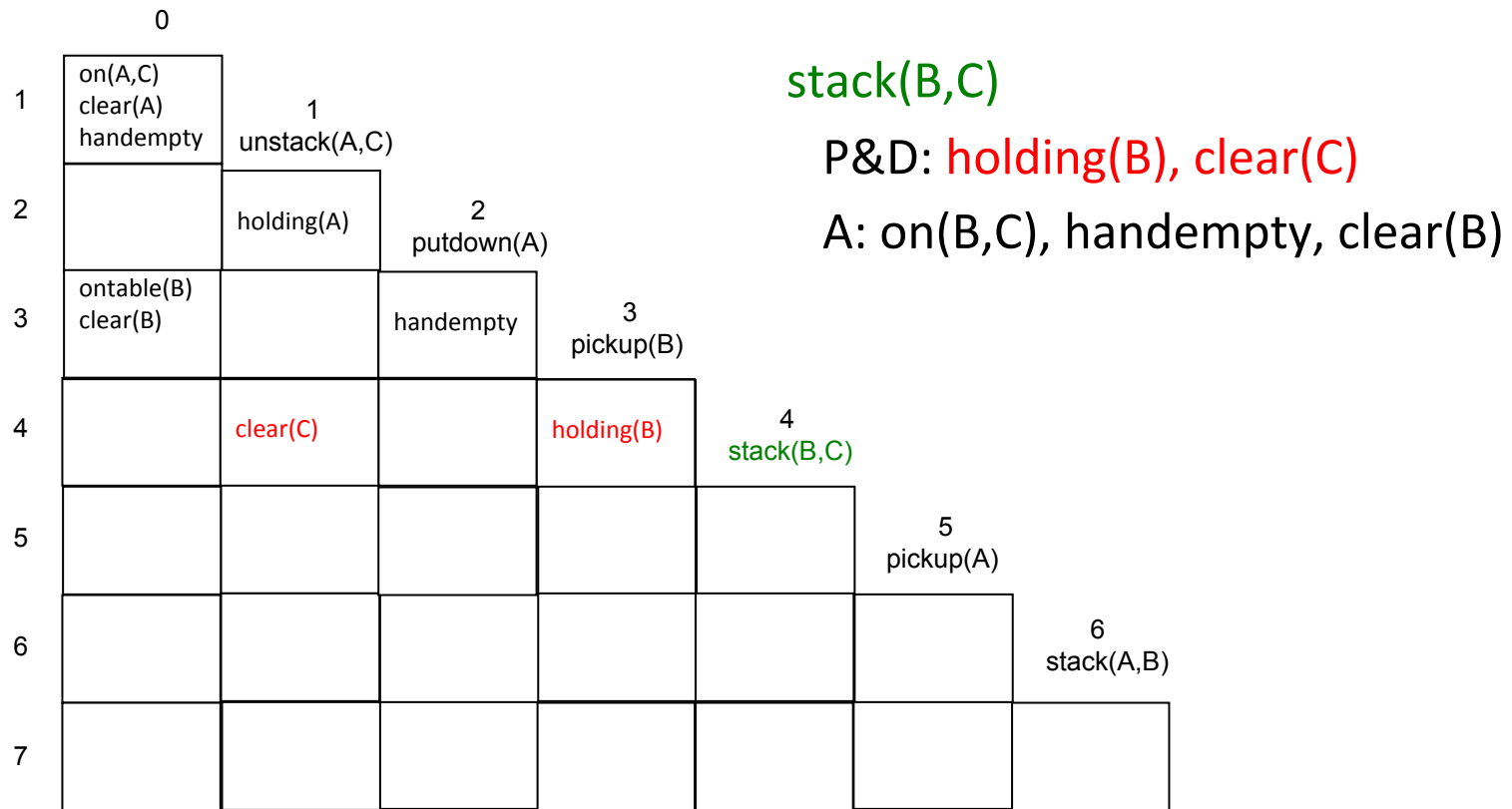
Collect list A predicates of rule 3.

# Plan execution: step 4



Look down for predicates of the  
P&D list of rule 4.

# Plan execution: step 4



stack(B,C)

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

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

ontable(C)

ontable(A)  
clear(A)

... and bring them up in the line of rule 4

# Plan execution: step 4

	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				</
--	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----

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)

Collect here down all list A  
predicates of rule 4.

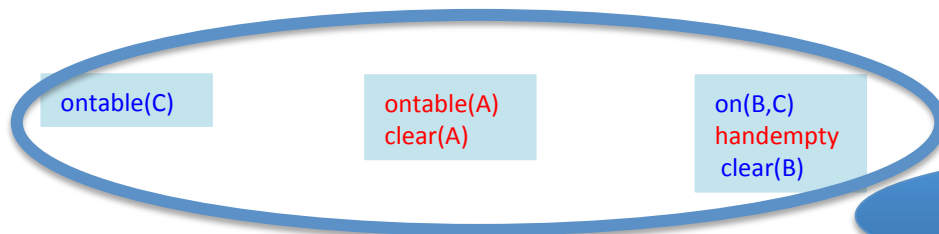
# Plan execution: step 5

	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)



Look down for predicates  
of the P&D list of rule 5.

# Plan execution: step 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)

... and bring them up  
in the line of rule 5.



# Plan execution: step 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)

Collect down all list A predicates of rule 5.

# Plan execution: step 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)

Look down for predicates  
of the P&D list of rule 6.

# Plan execution: step 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)
7						6 stack(A,B)

stack(A,B)

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

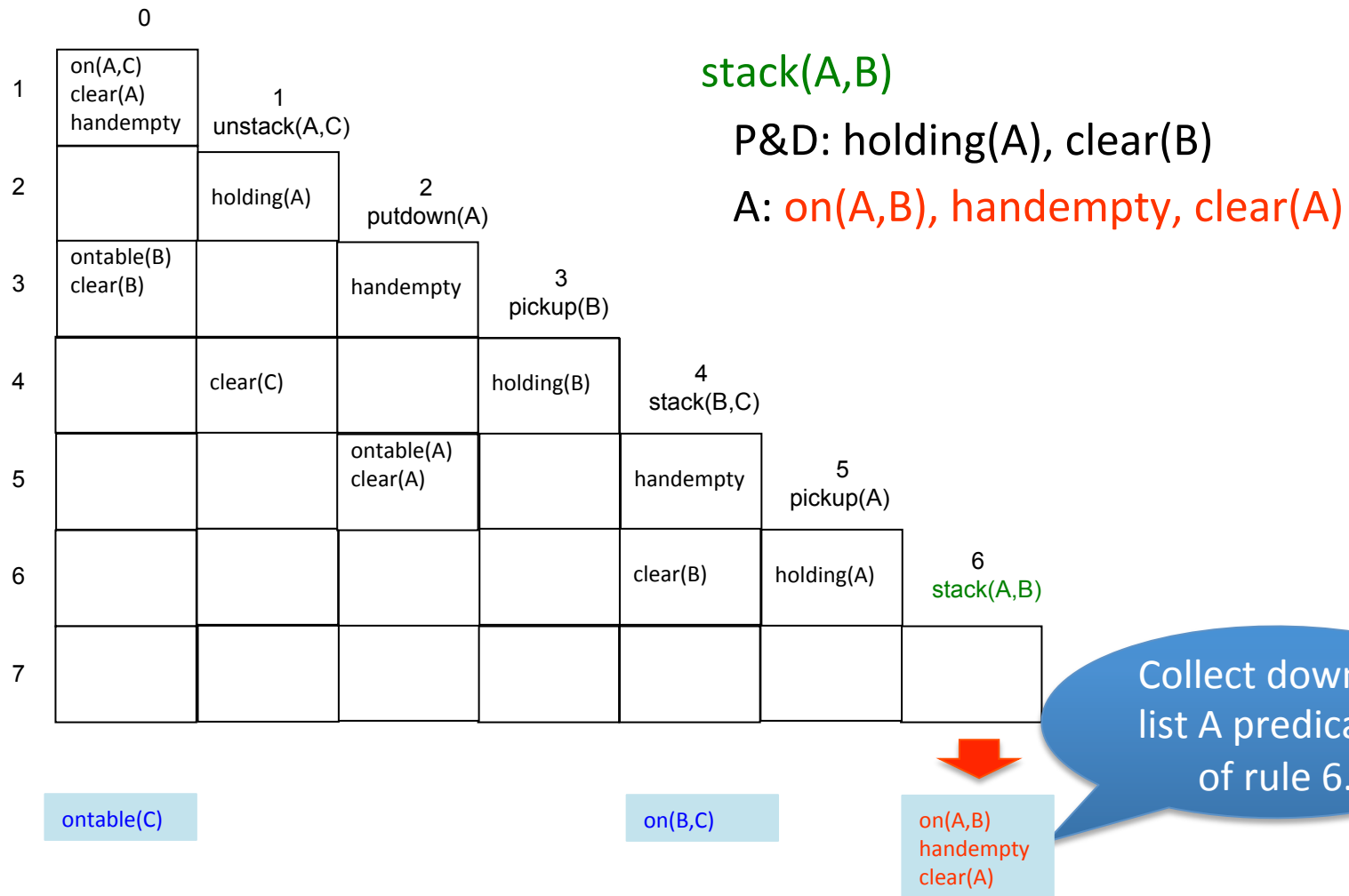
A: on(A,B), handempty, clear(A)

ontable(C)

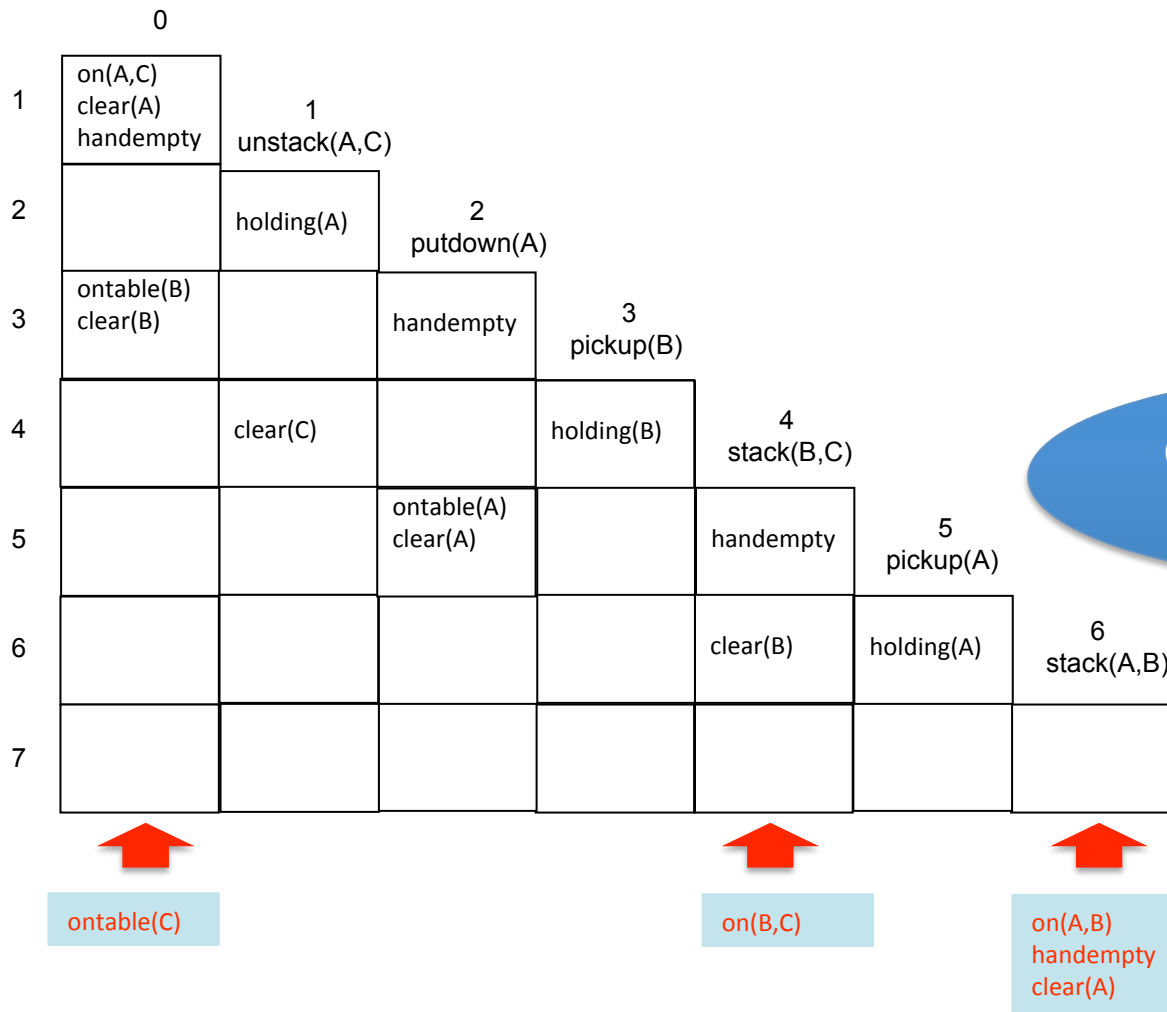
on(B,C)

... and bring them up in the line of rule 6.

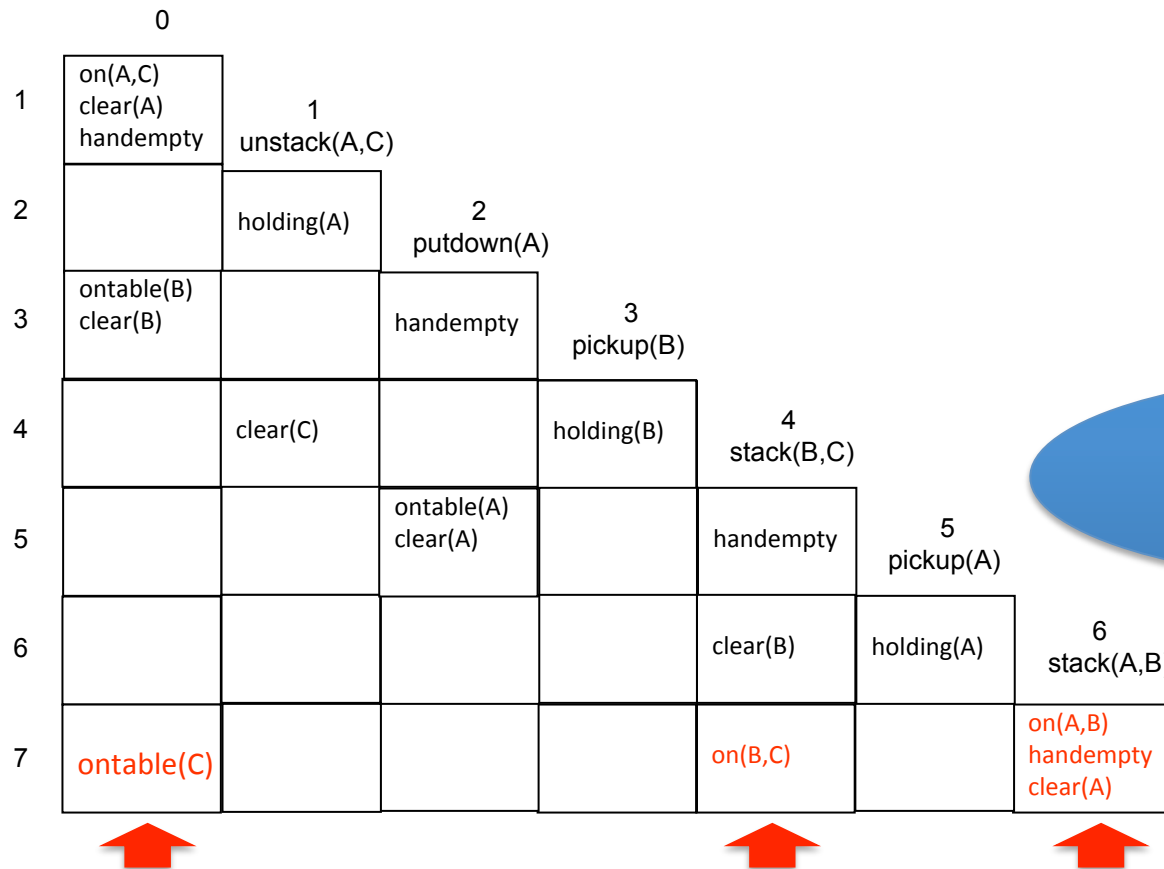
# Plan execution: step 6



# Plan execution: catching up



# Plan execution: catching up

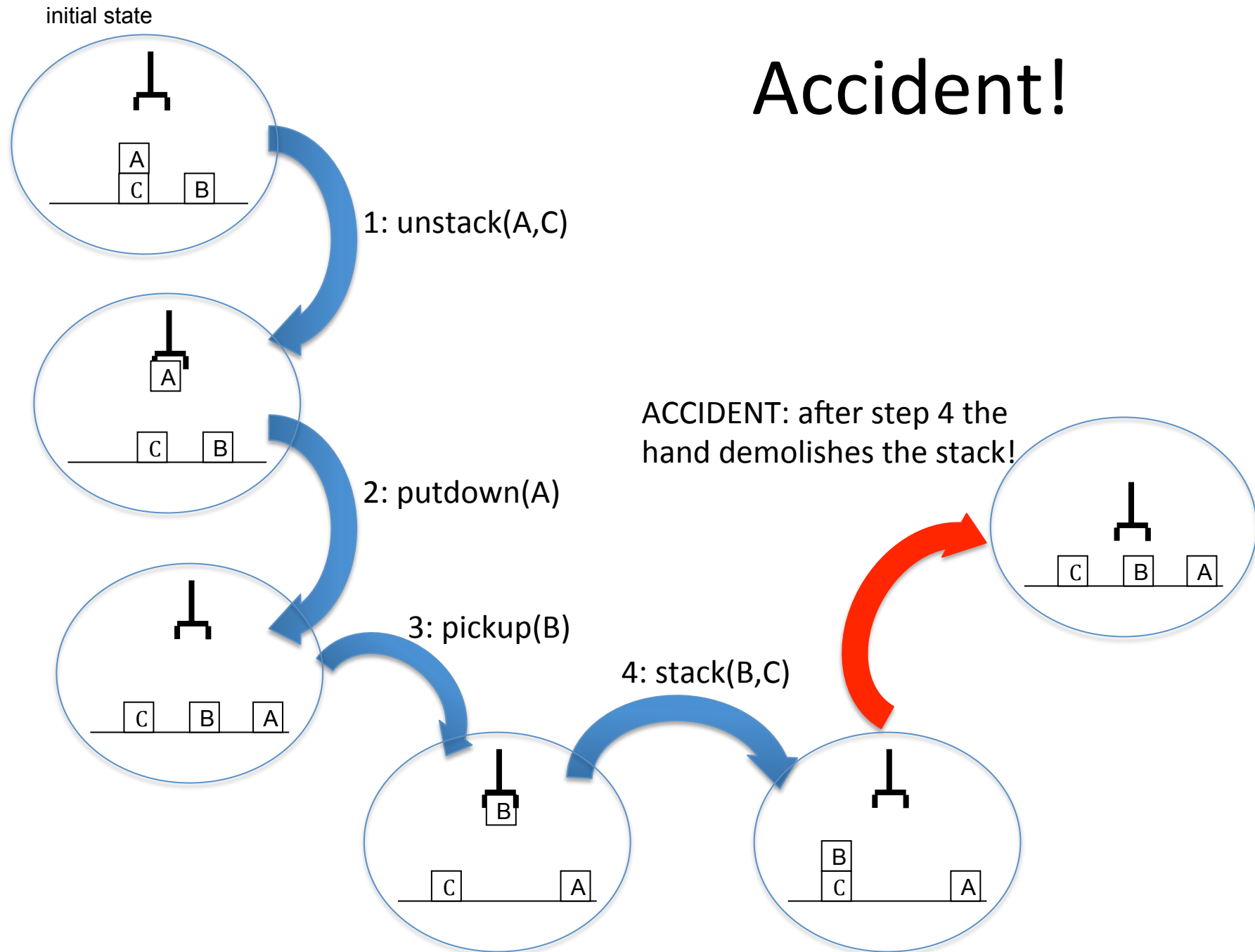


# Plan execution: catching up

	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)

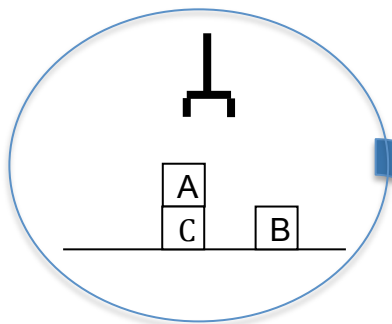
The final state.

# Accident!

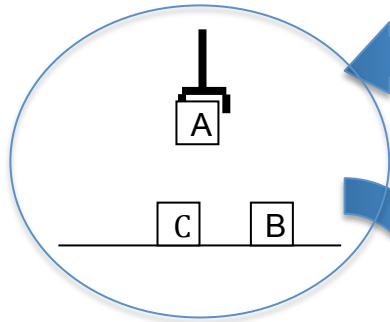




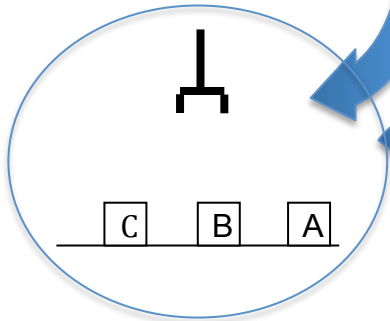
initial state



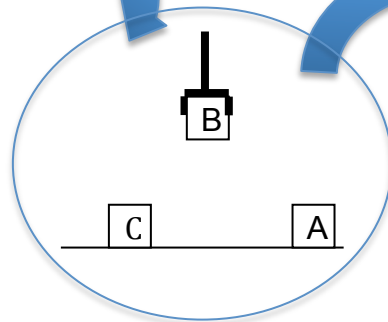
1: unstack(A,C)



2: putdown(A)



3: pickup(B)

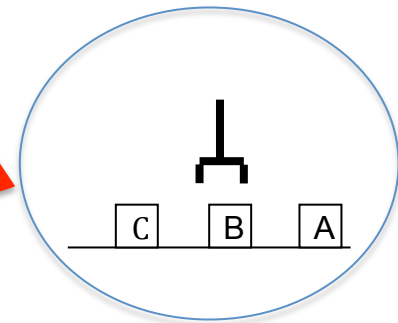


4: stack(B,C)

# Accident!

Description of the state following the accident

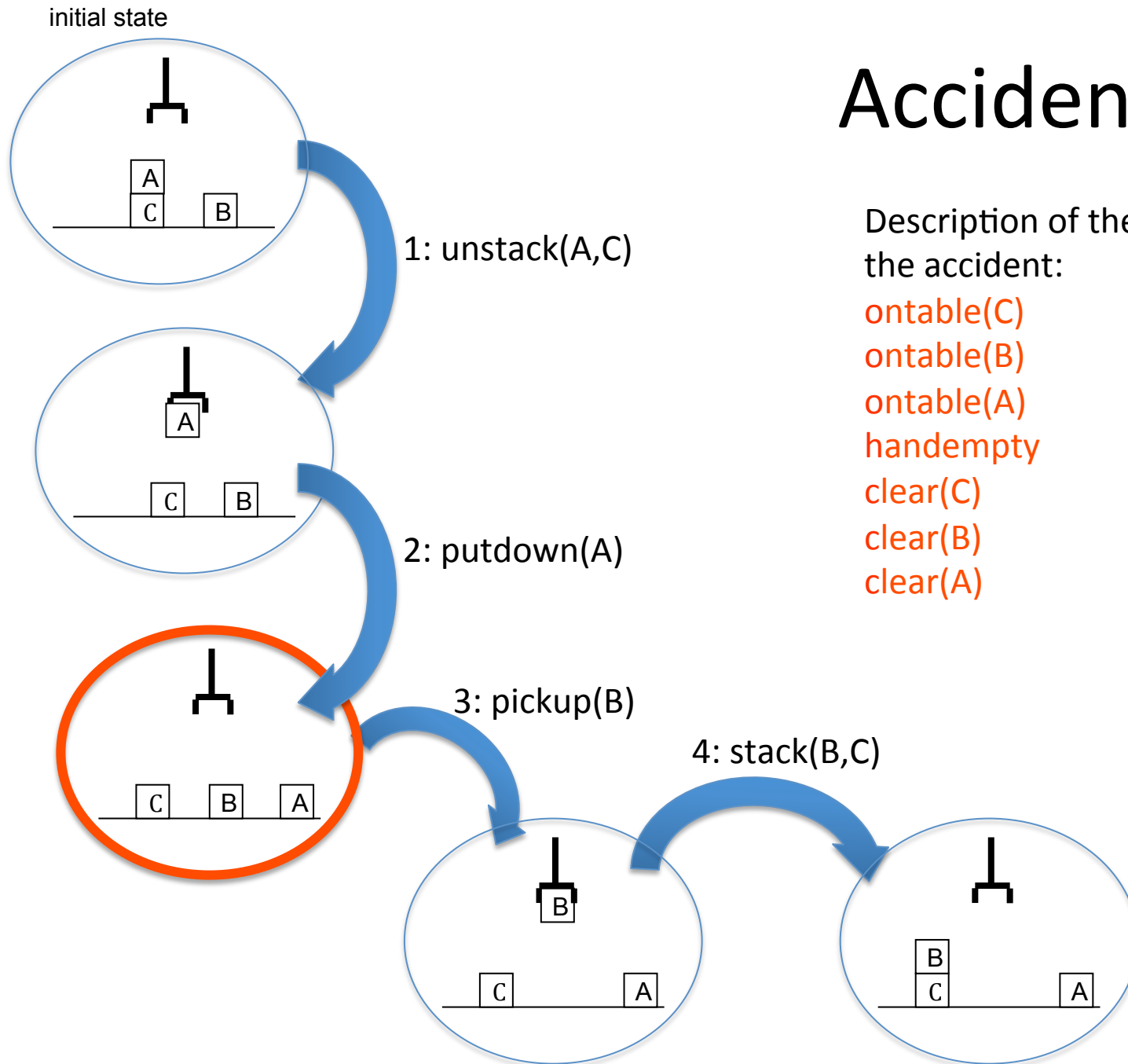
ontable(C)  
ontable(B)  
ontable(A)  
handempty  
clear(C)  
clear(B)  
clear(A)



# Accident!

Description of the state following the accident:

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



# Kernel of order 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)

Look for the first kernel that corresponds to the state following the accident.

NO

# Kernel of order 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)

Look for the first kernel that corresponds to the state following the accident.

NO

# Kernel or order 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)

Look for the first kernel that corresponds to the state following the accident.

NO

# Kernel of order 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)

Look for the first kernel that corresponds to the state following the accident.

NO

# Kernel of order 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)

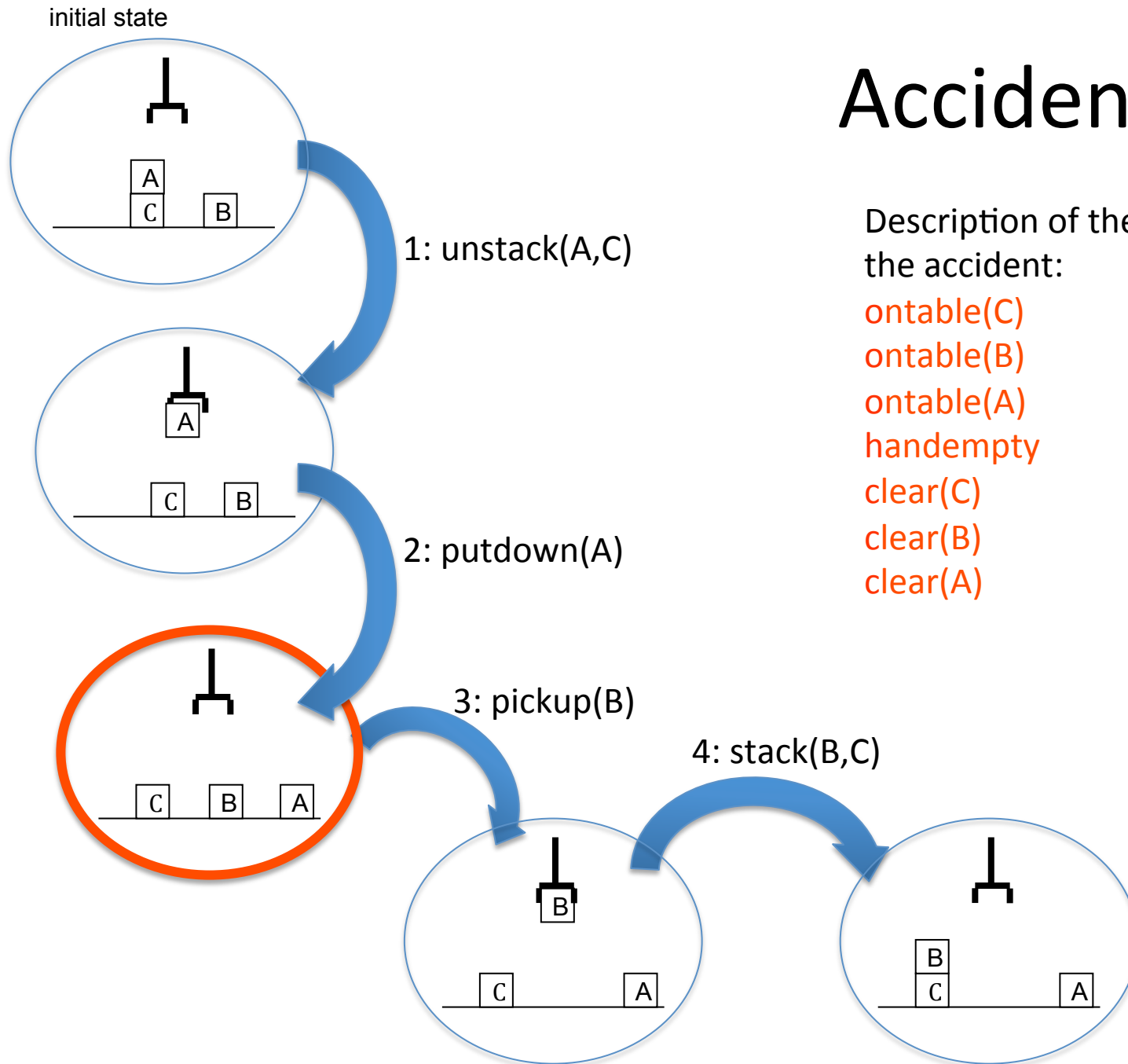
If the state following the accident corresponds with the order  $k$  kernel  $\rightarrow$  go on with step  $k$ .

YES

# Accident!

Description of the state following the accident:

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





# Conclusion:

## what to do after an accident?

- If the state following the accident:
  - is among the set of kernels => go on with the execution from there on!
  - is not among the set of kernels => build a new plan in which the initial state is this accident state!  
=> execute the plan!