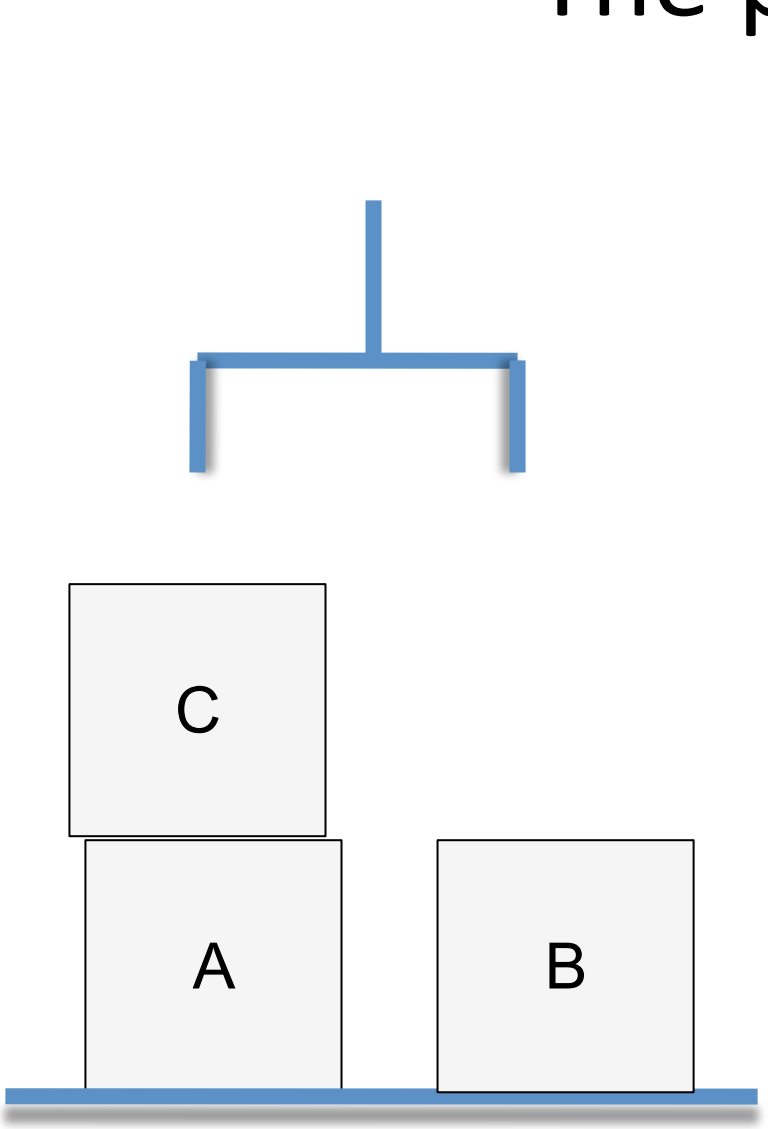
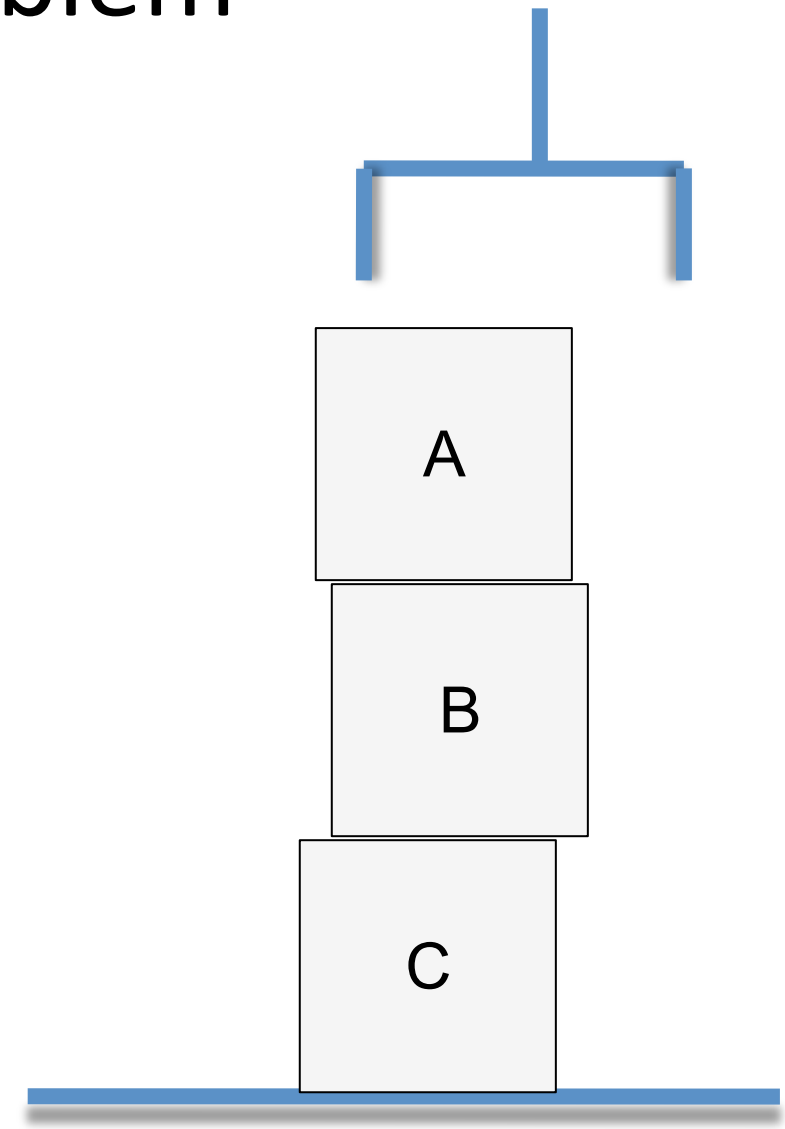
Course 11
Planning and plan execution

The problem

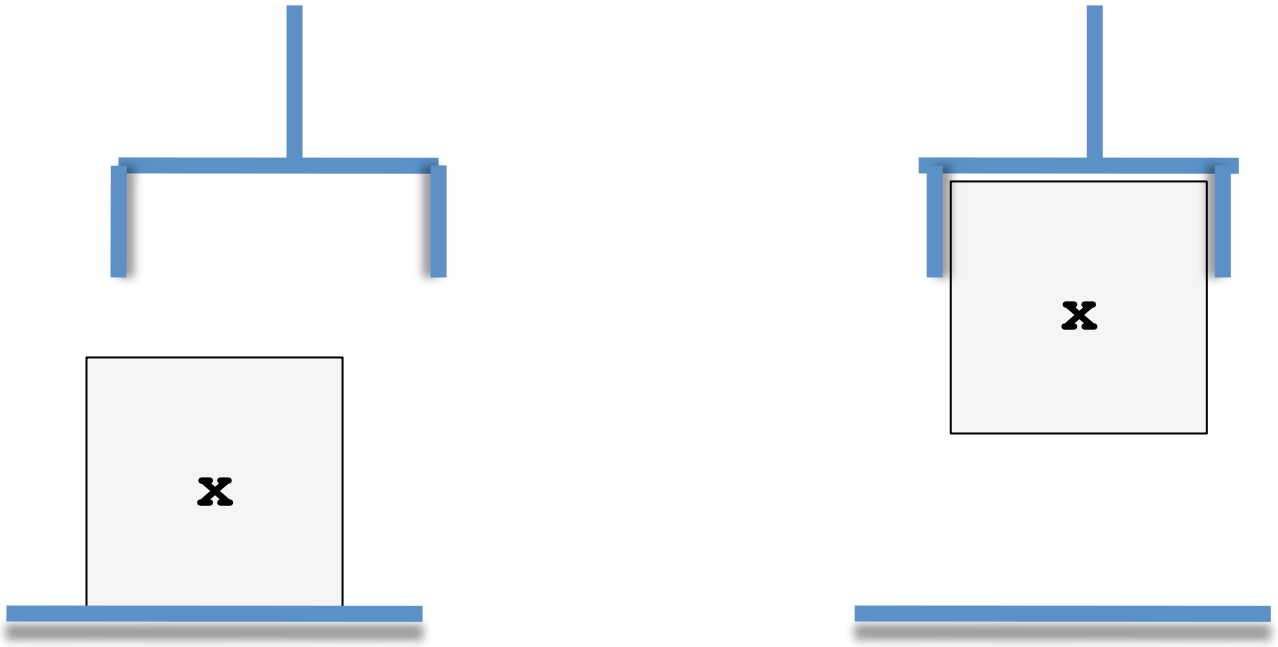


initial state



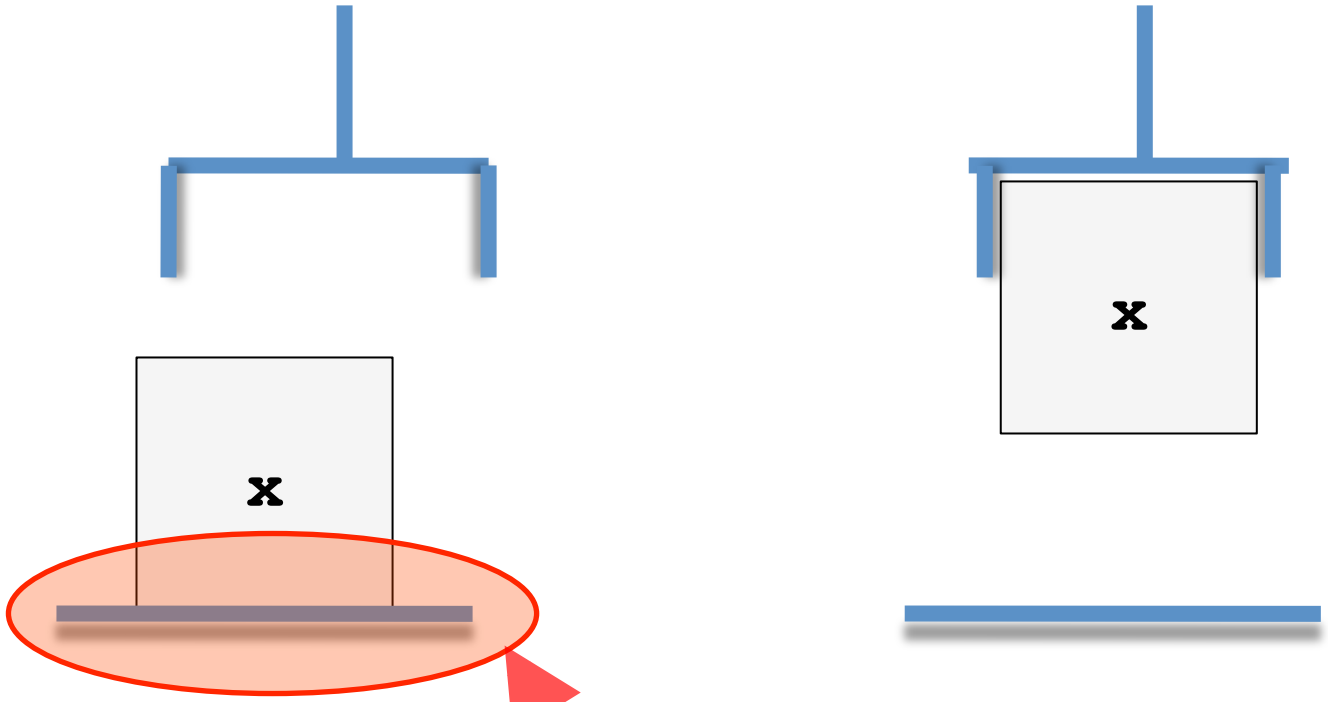
final state

STRIPS rules



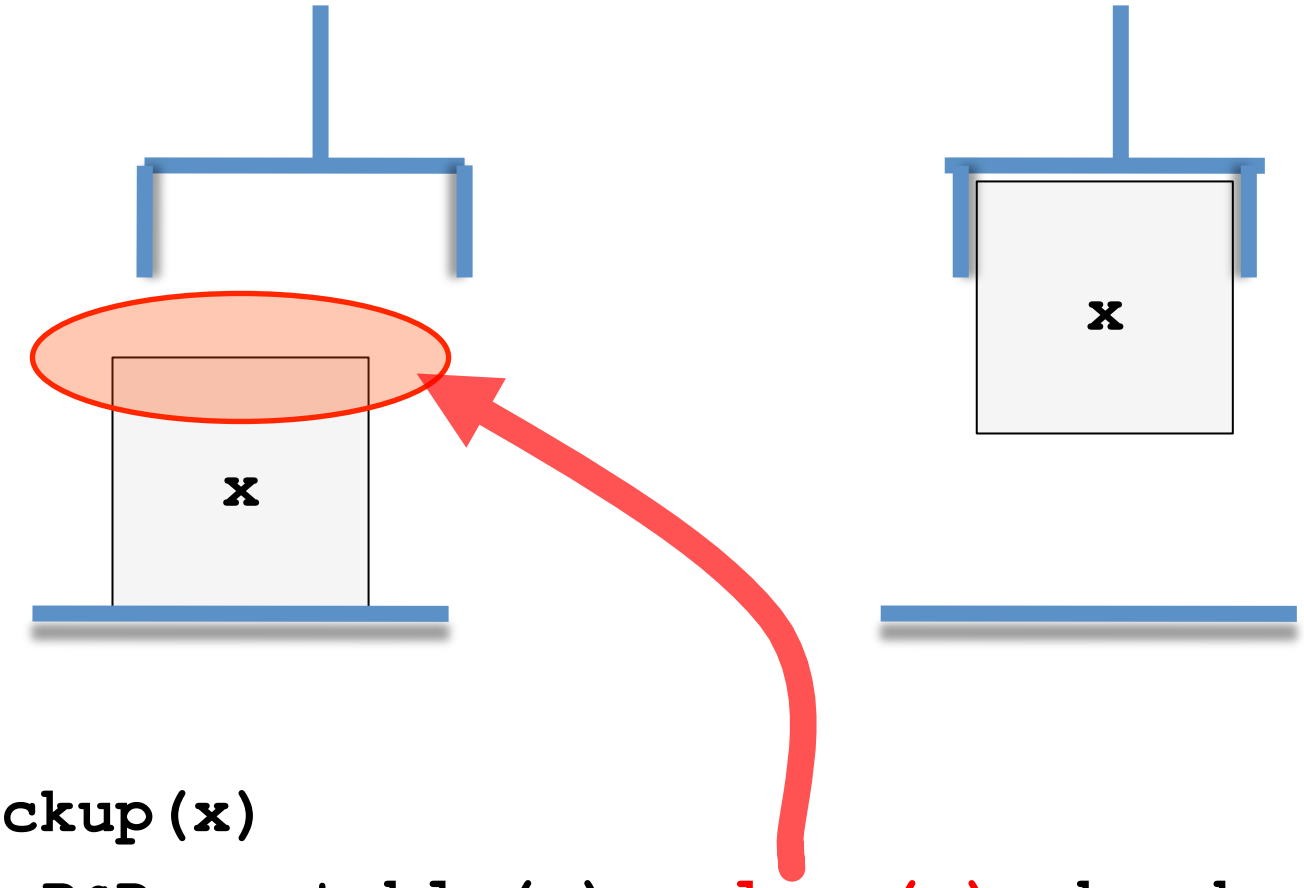

```
pickup(x)
    P&D: ontable(x), clear(x), handempty
    A: holding(x)
```

STRIPS rules



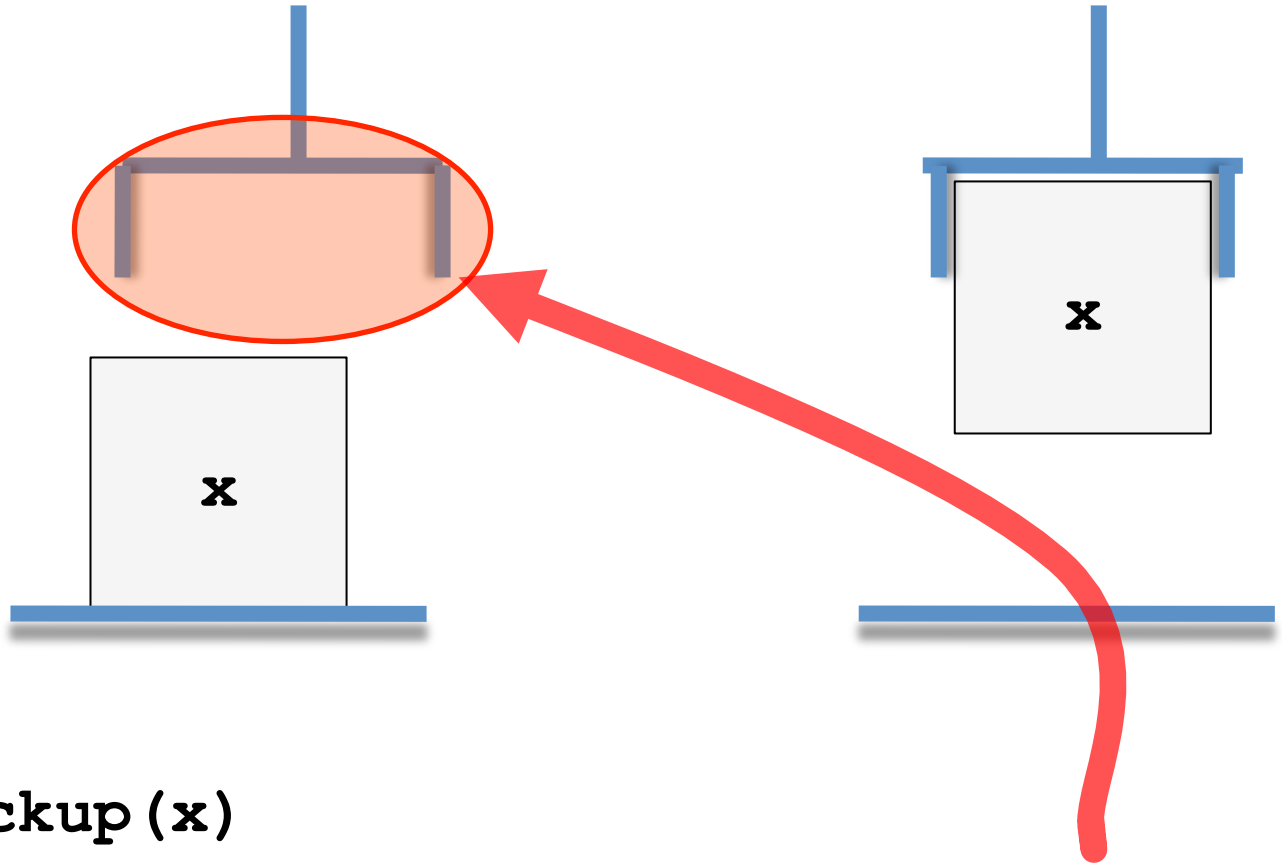

```
pickup(x)
    P&D: ontable(x), clear(x), handempty
    A: holding(x)
```

STRIPS rules



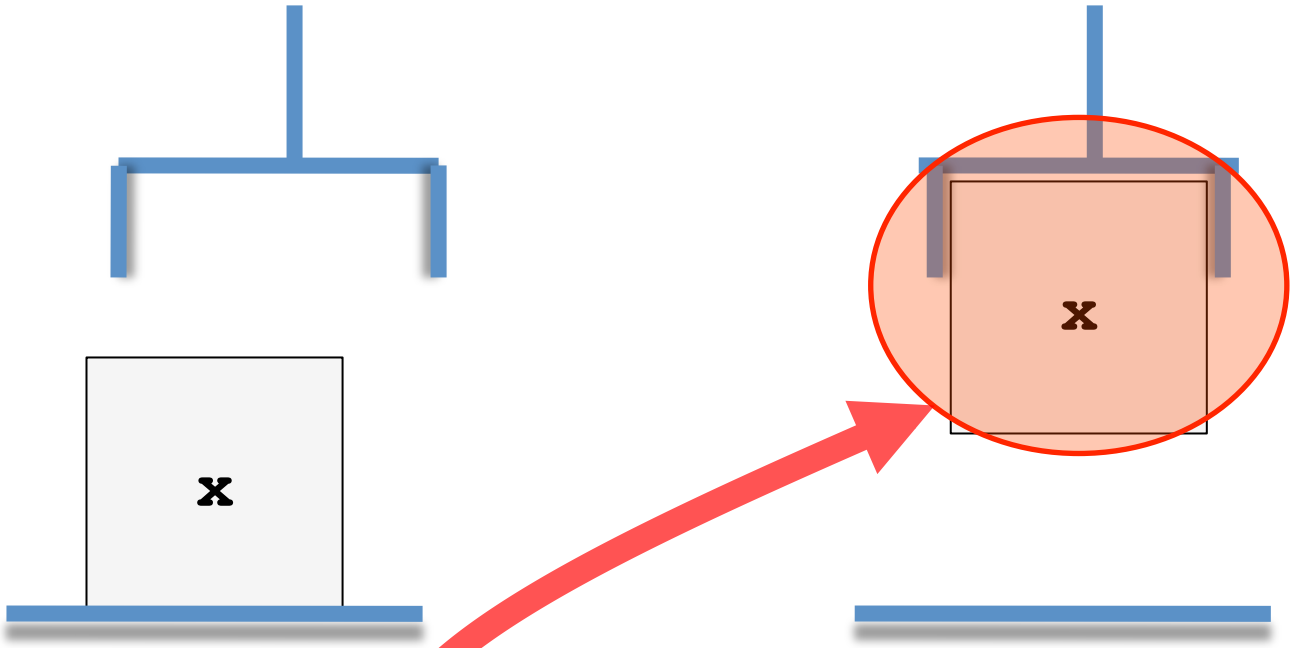

```
pickup(x)
    P&D: ontable(x), clear(x), handempty
    A: holding(x)
```

STRIPS rules



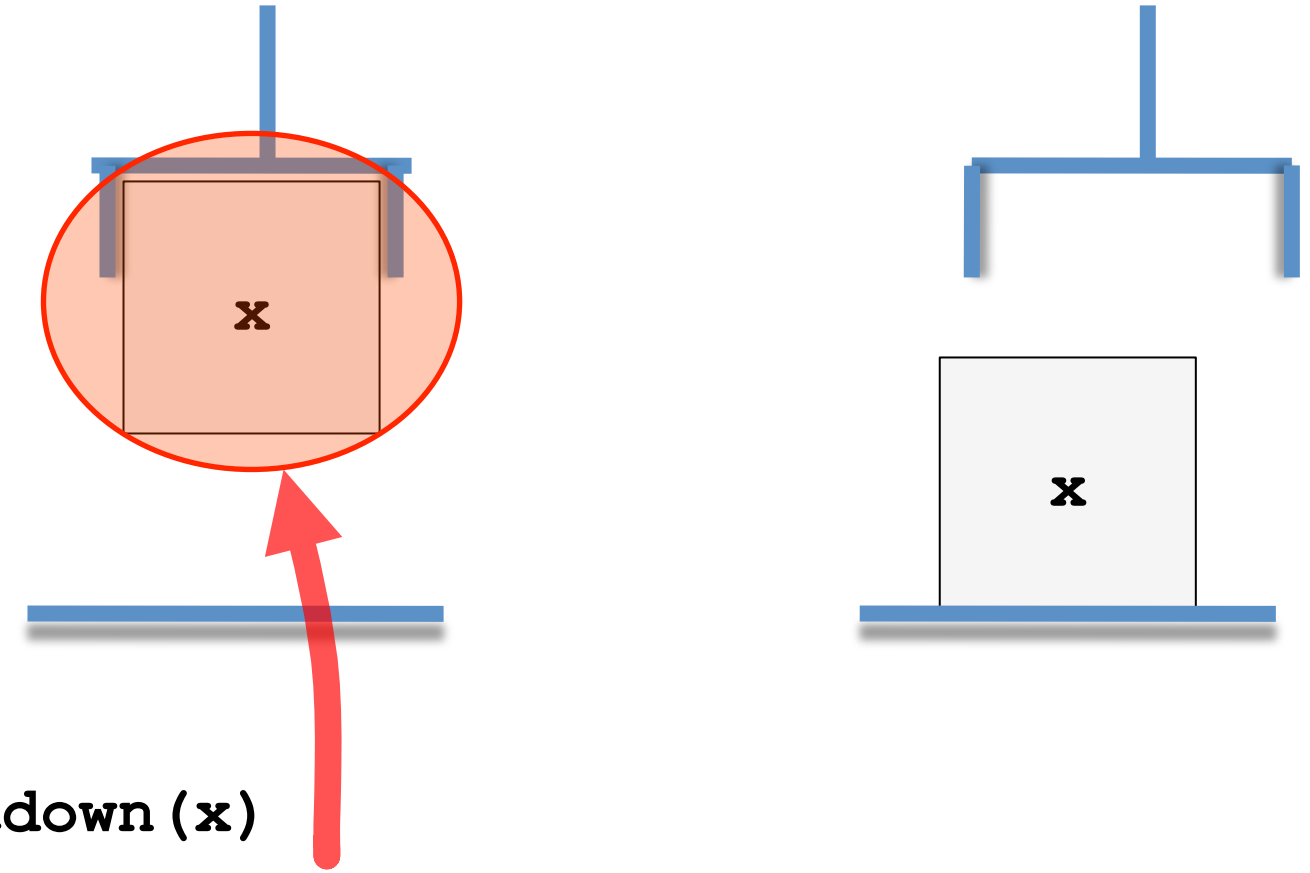

```
pickup(x)
    P&D: ontable(x), clear(x), handempty
    A: holding(x)
```

STRIPS rules




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

STRIPS rules

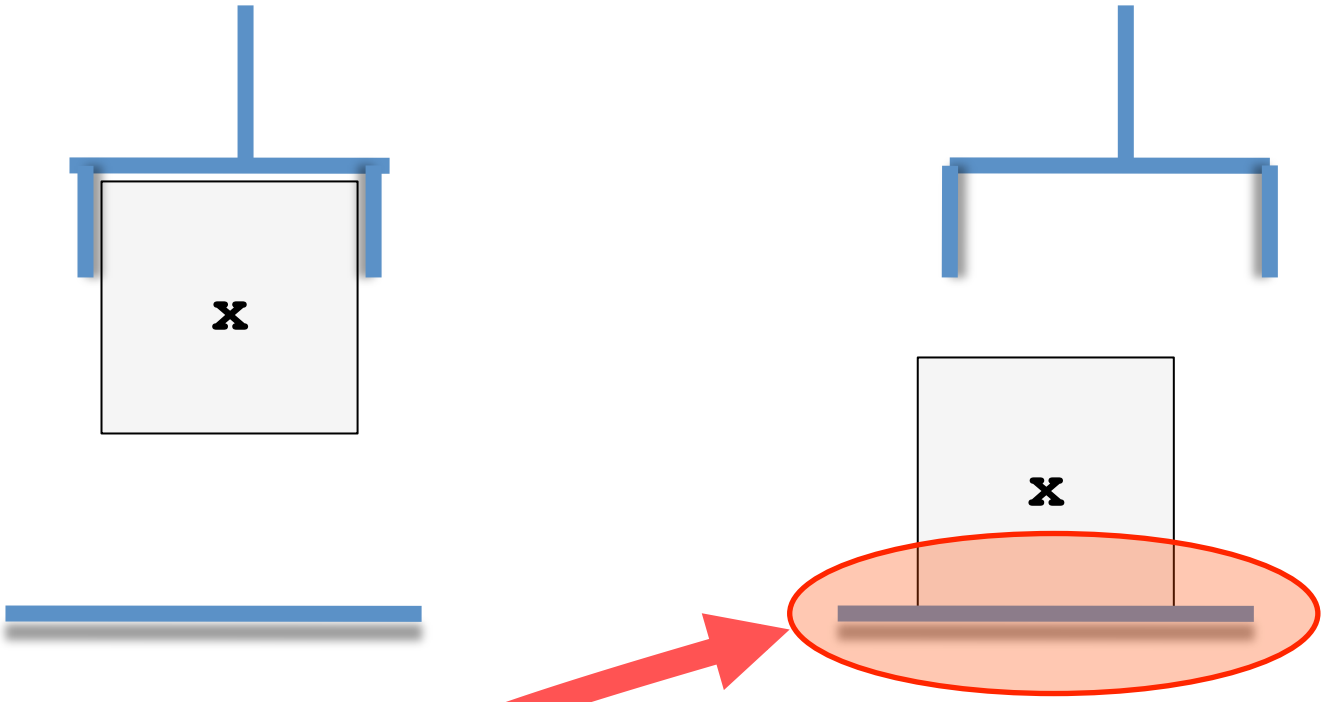


putdown(x)

P&D: holding(x)

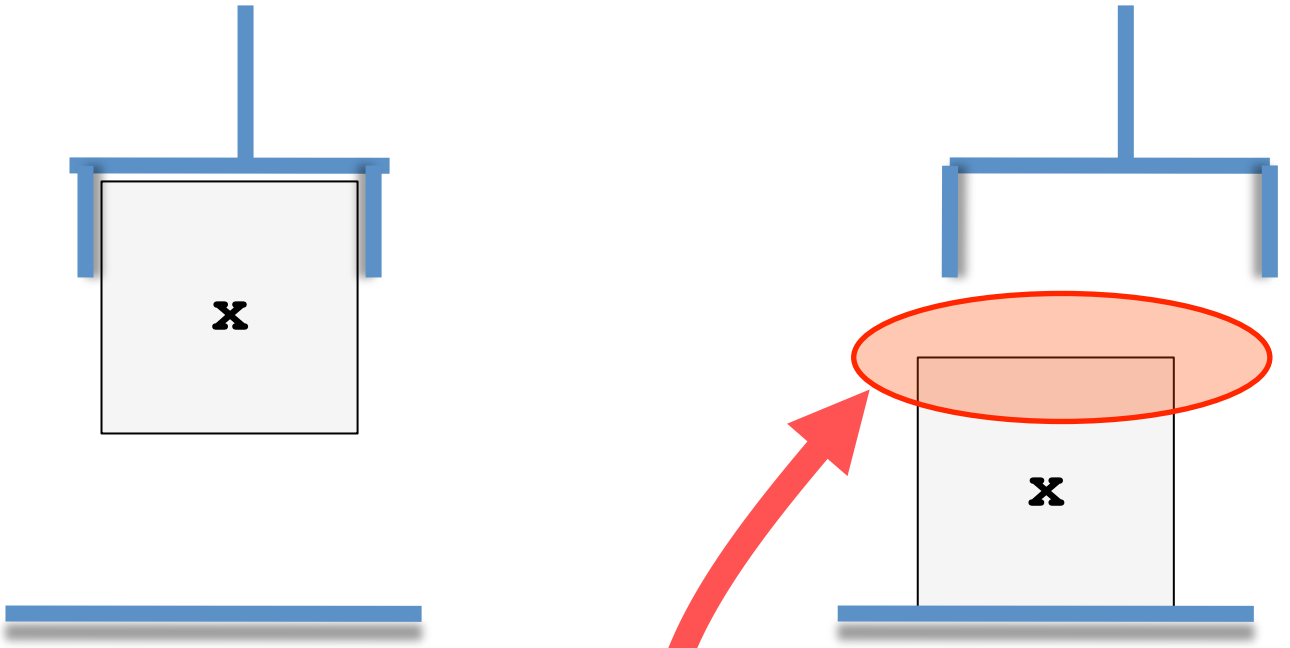
A: ontable(x), clear(x), handempty

STRIPS rules




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

STRIPS rules

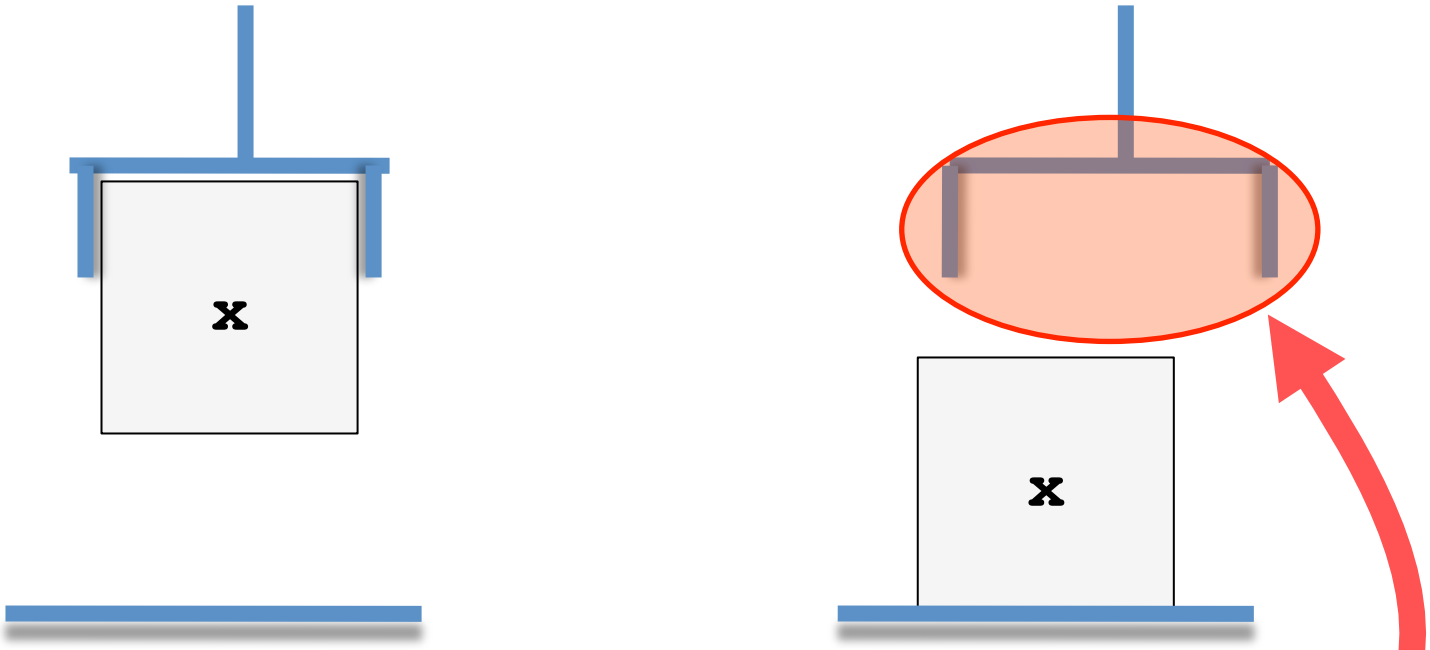


putdown(x)

P&D: holding(x)

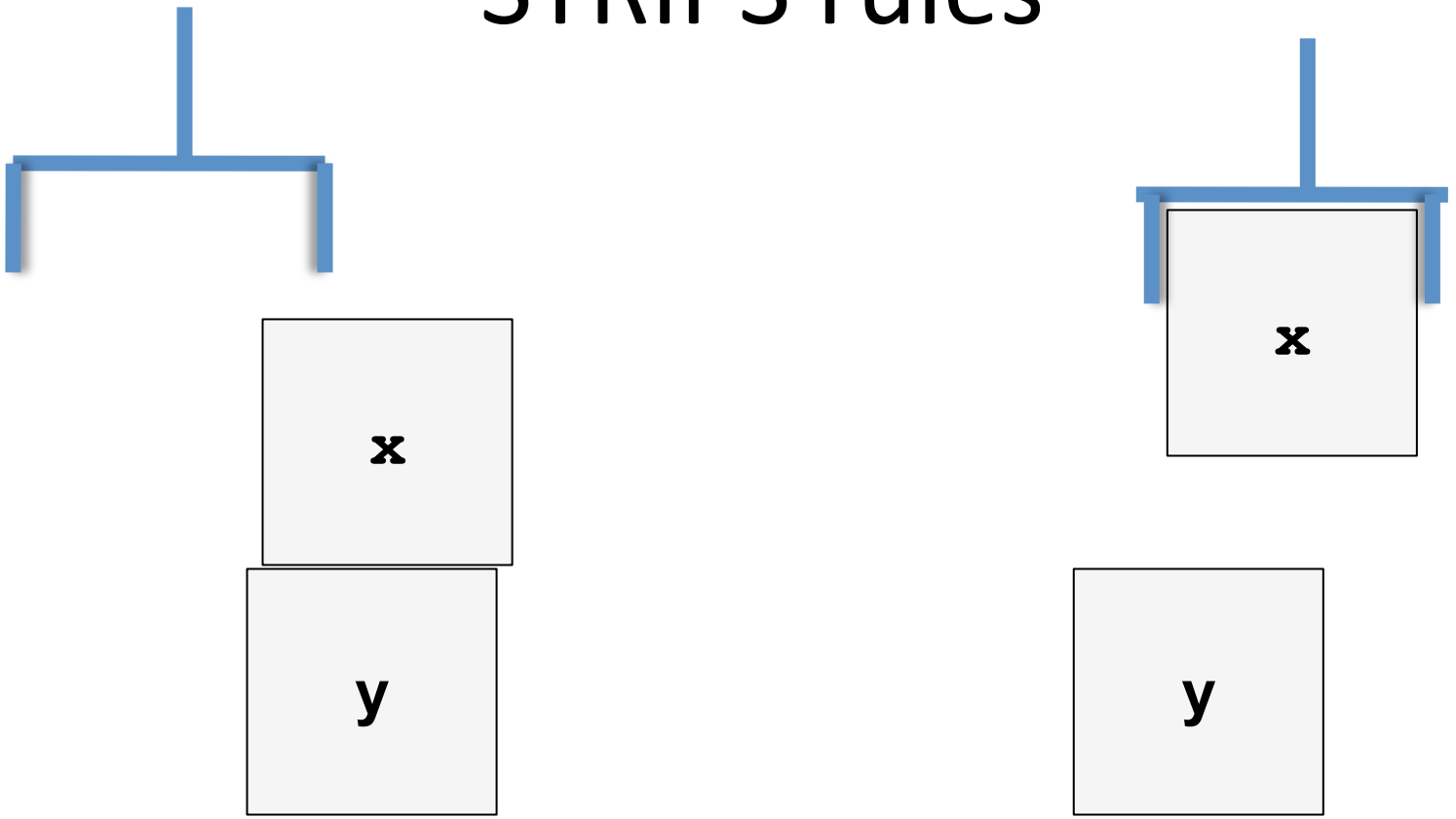
A: ontable(x), clear(x), handempty

STRIPS rules



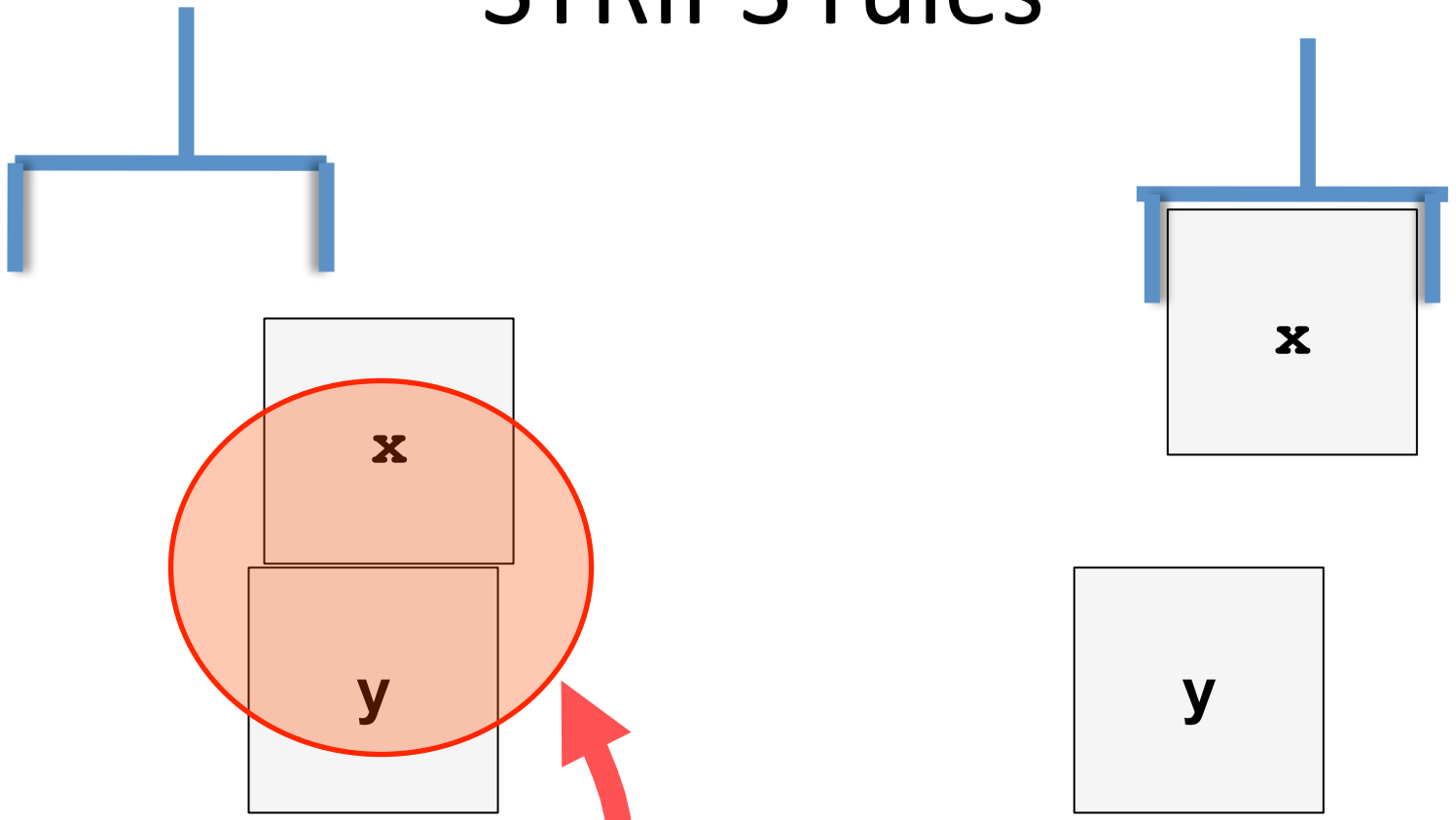

```
putdown(x)
    P&D: holding(x)
    A: ontable(x), clear(x), handempty
```

STRIPS rules



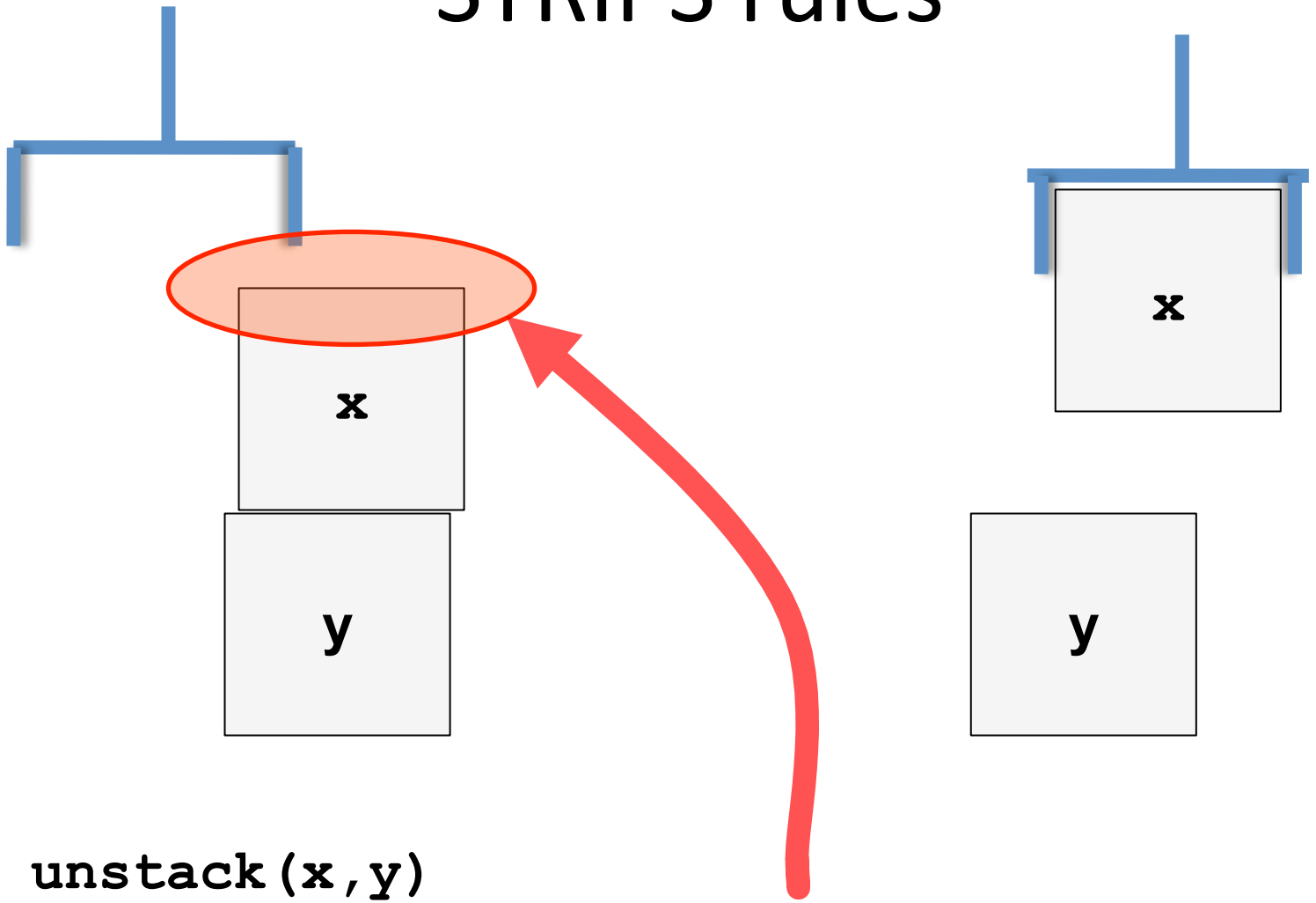

```
unstack(x,y)
    P&D: on(x,y), clear(x), handempty
    A: holding(x), clear(y)
```

STRIPS rules



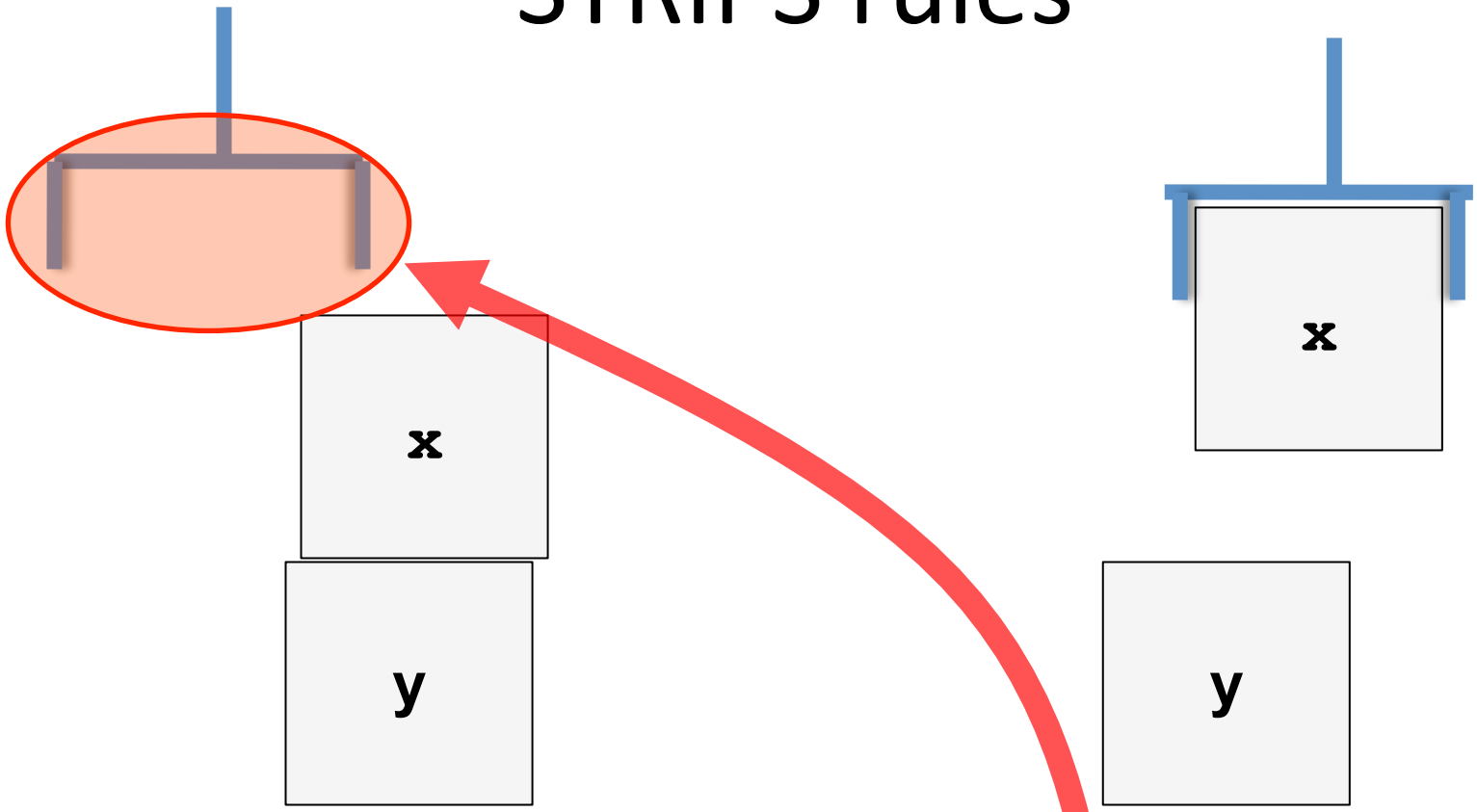

```
unstack(x,y)
    P&D: on(x,y), clear(x), handempty
    A: holding(x), clear(y)
```

STRIPS rules



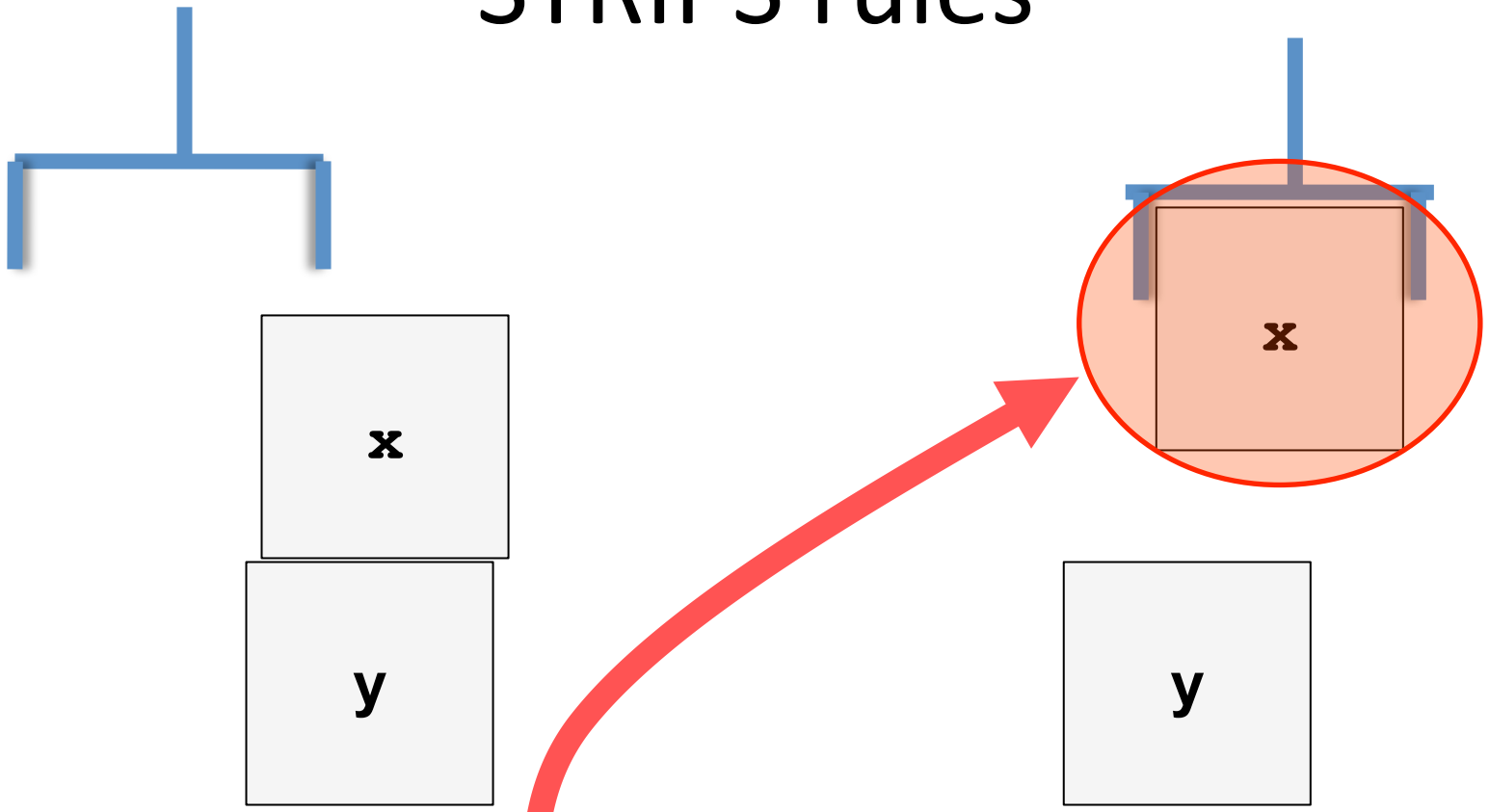

```
unstack(x,y)
    P&D: on(x,y), clear(x), handempty
    A: holding(x), clear(y)
```

STRIPS rules



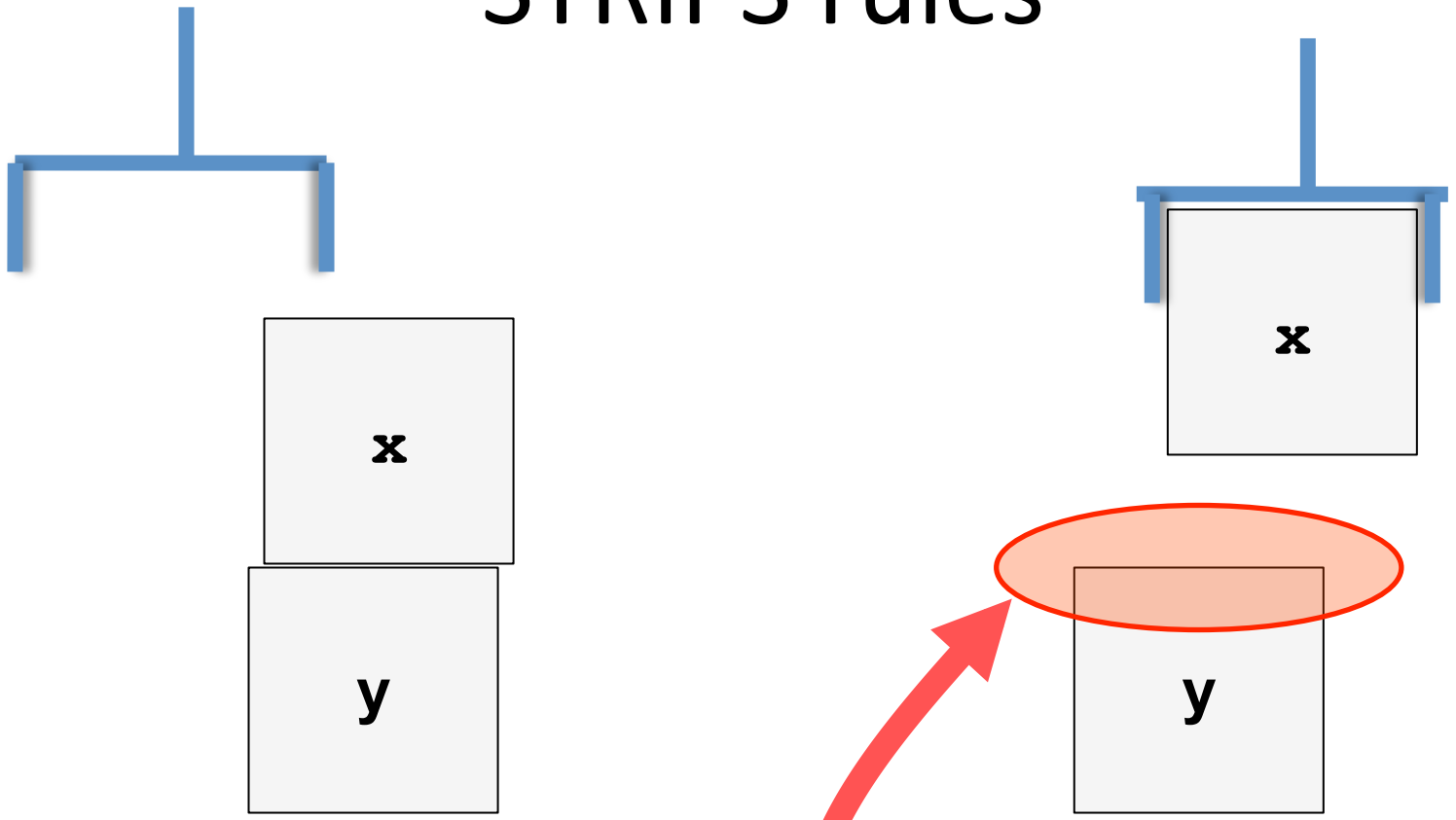

```
unstack(x,y)
    P&D: on(x,y), clear(x), handempty
    A: holding(x), clear(y)
```

STRIPS rules



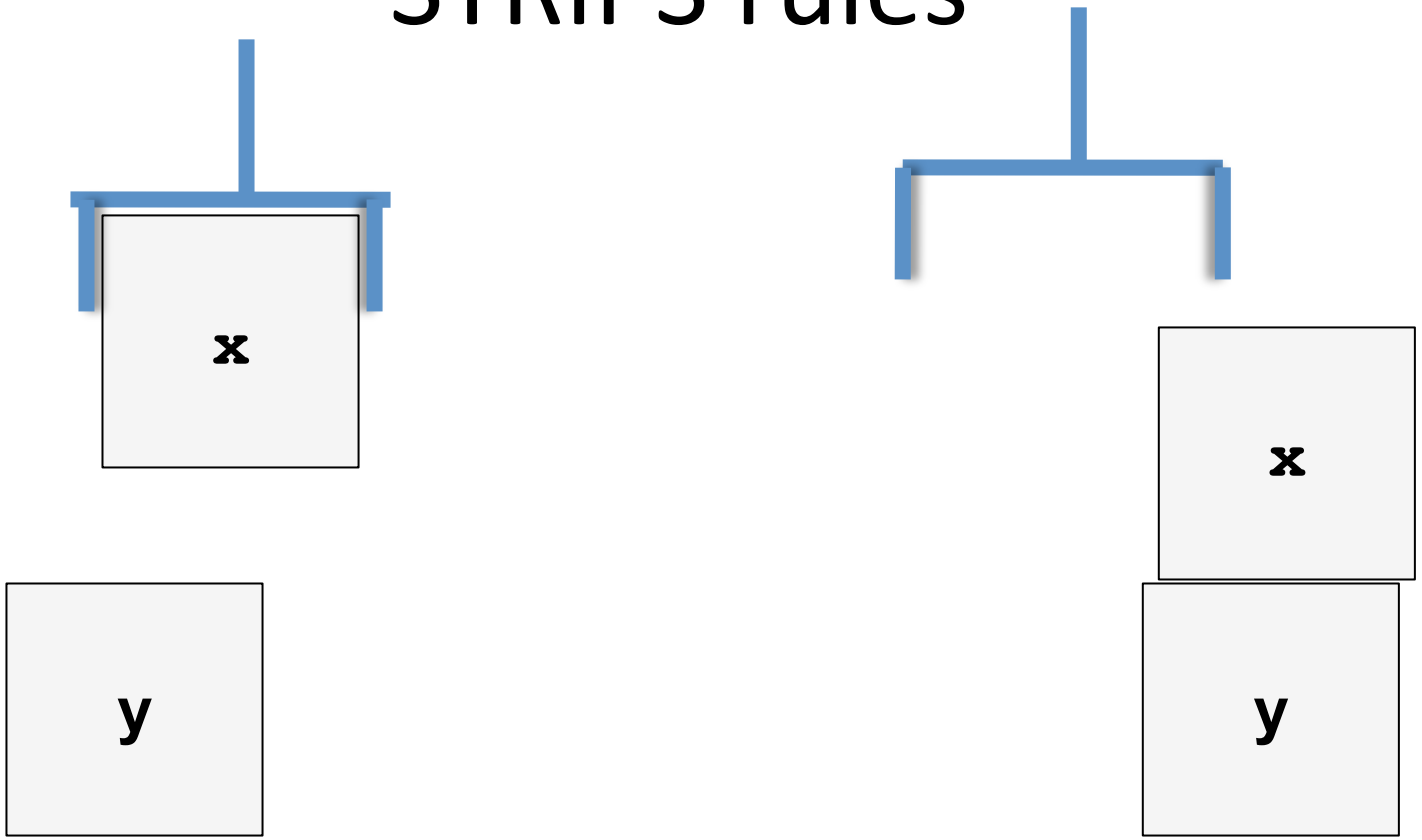

```
unstack(x,y)
    P&D: on(x,y), clear(x), handempty
    A: holding(x), clear(y)
```

STRIPS rules



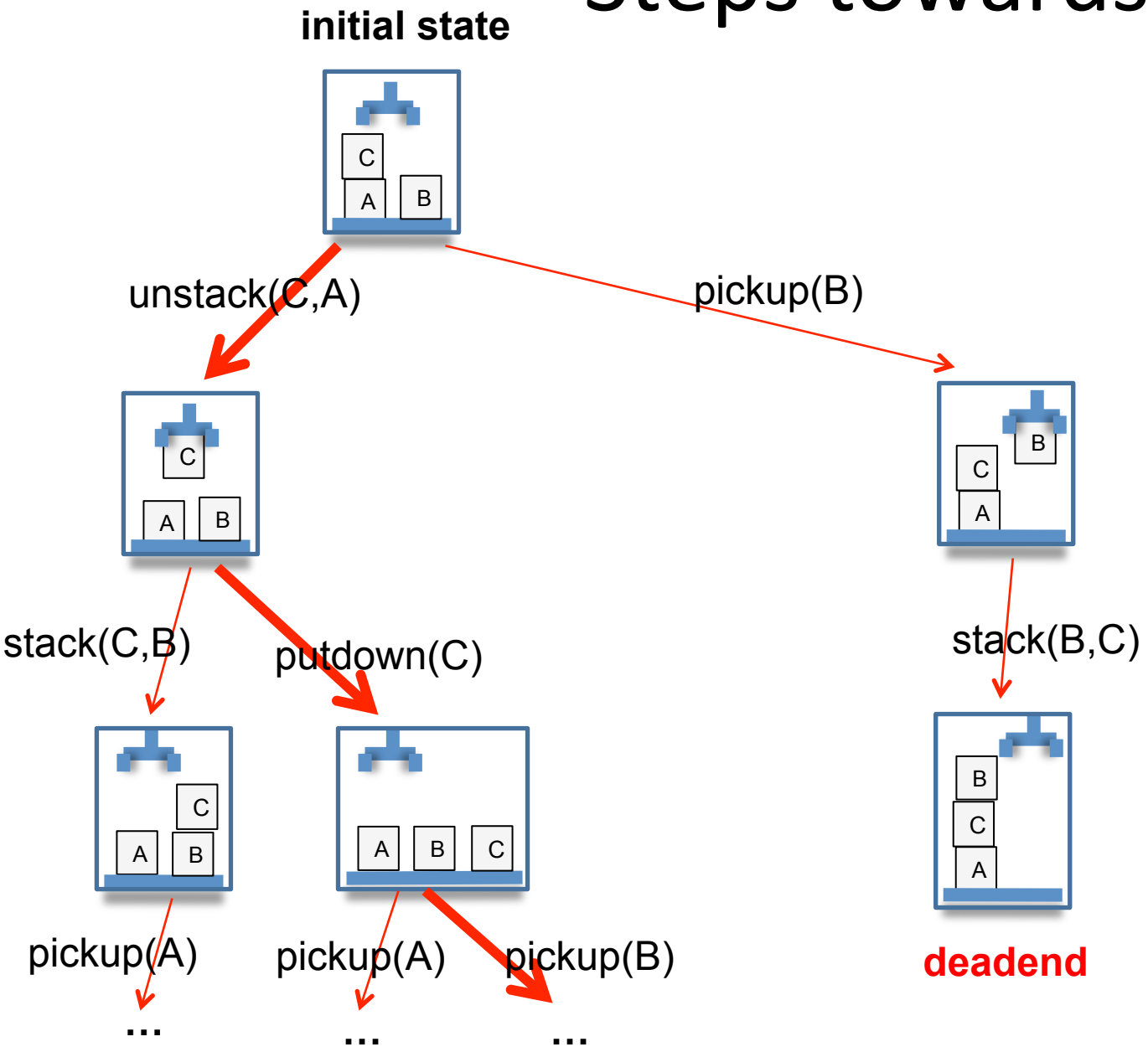

```
unstack(x,y)
    P&D: on(x,y), clear(x), handempty
    A: holding(x), clear(y)
```

STRIPS rules

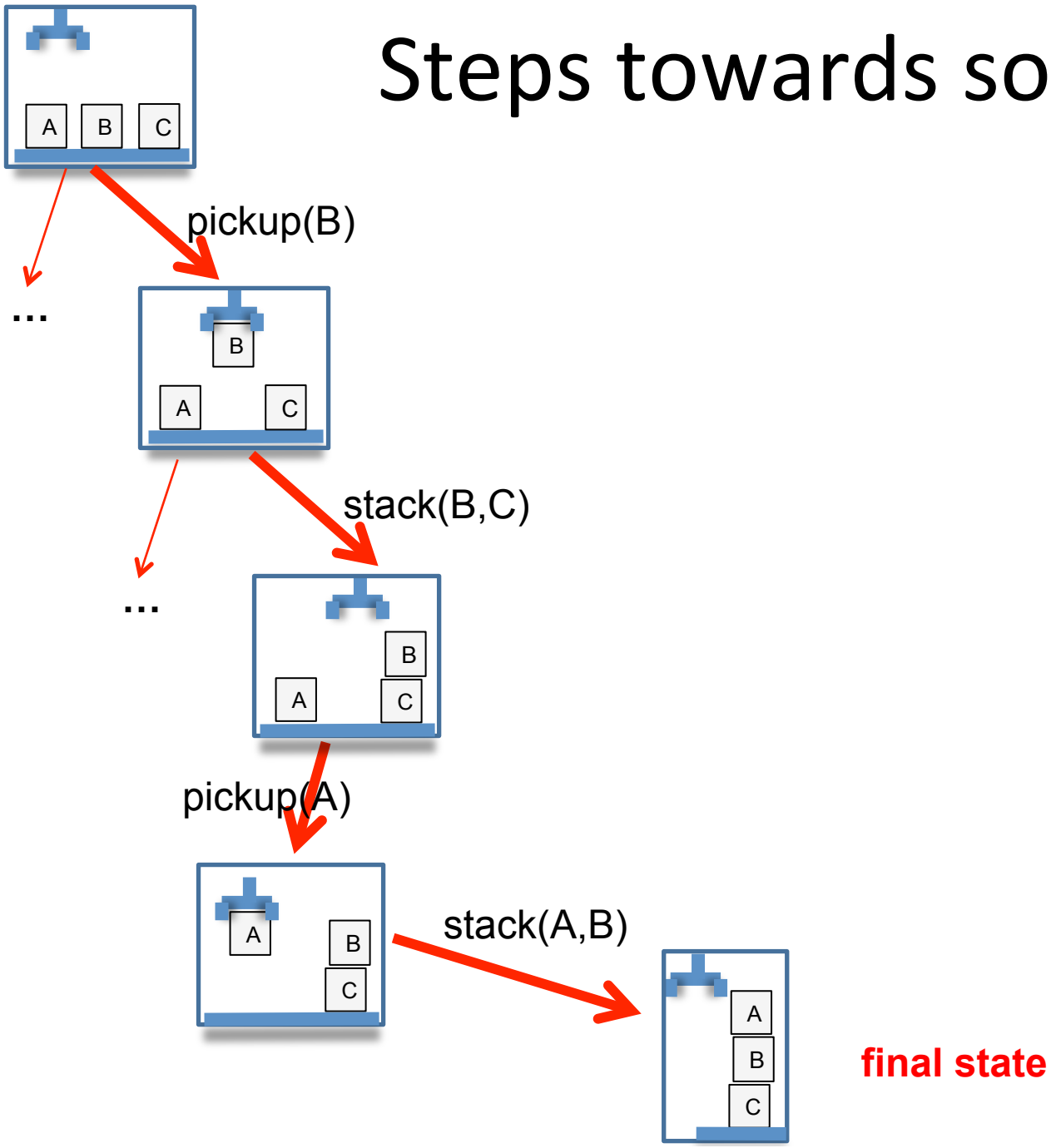



```
stack(x,y)
    P&D: holding(x), clear(y)
    A: on(x,y), handempty, clear(x)
```

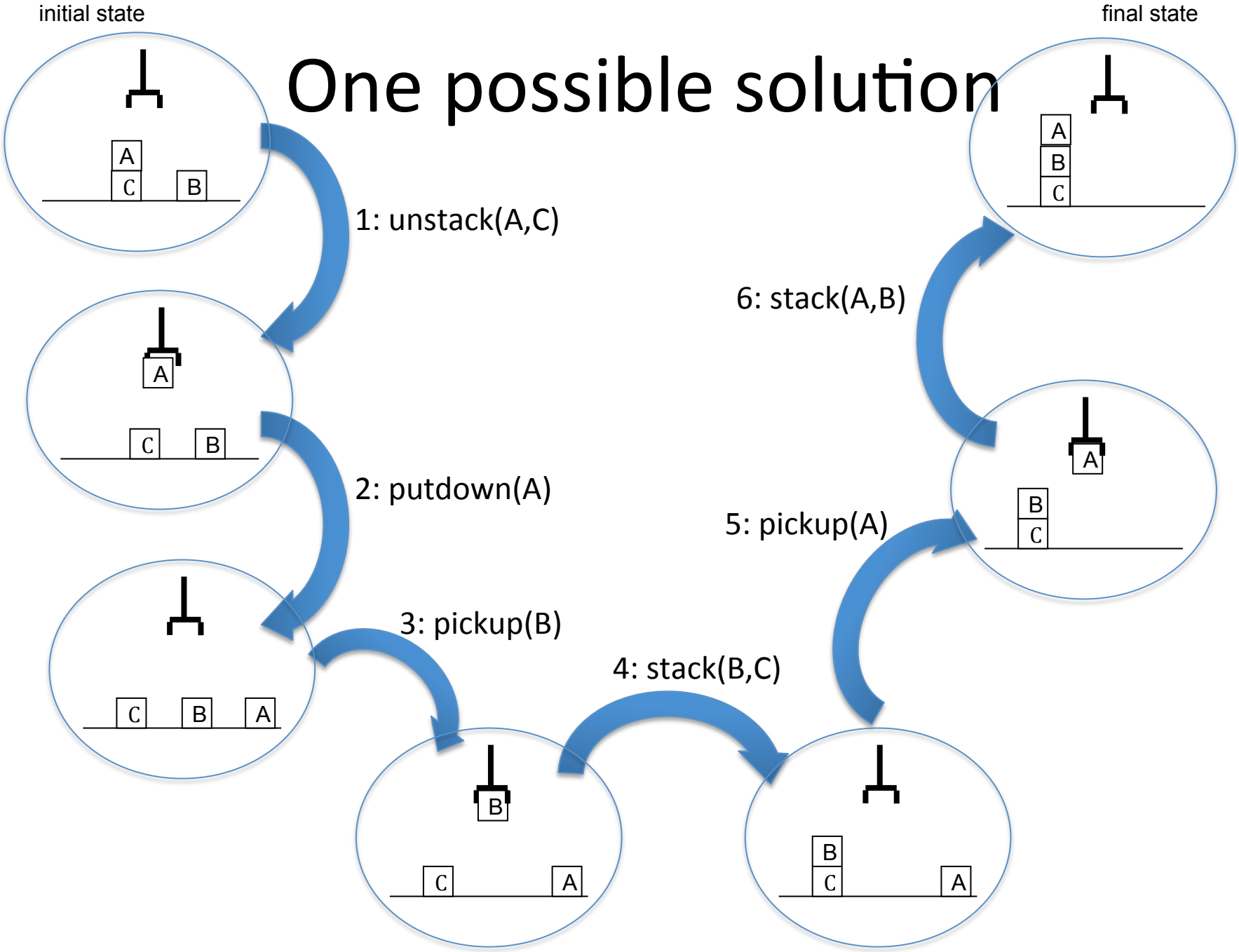
Steps towards solution



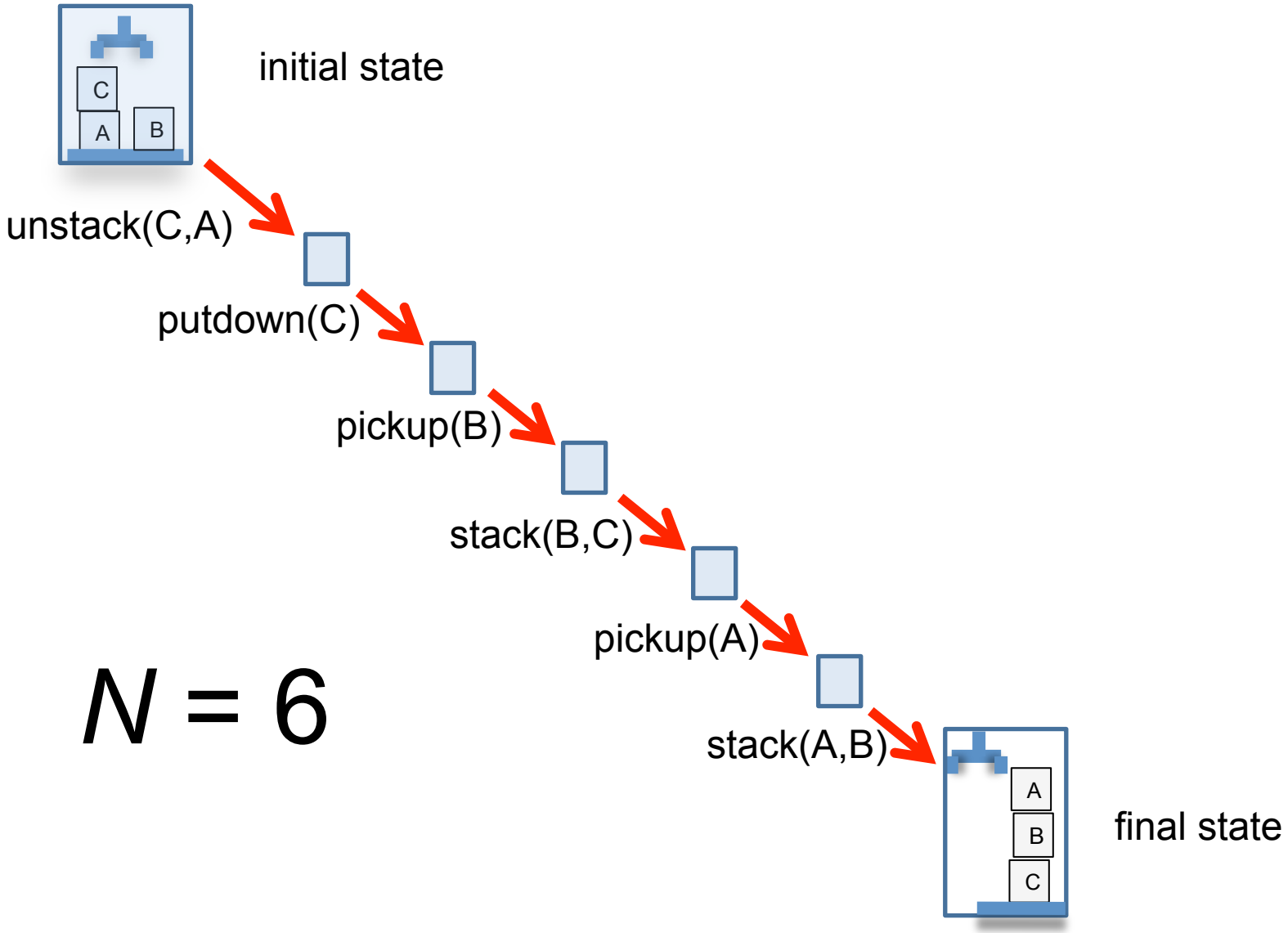
Steps towards solution



One possible solution

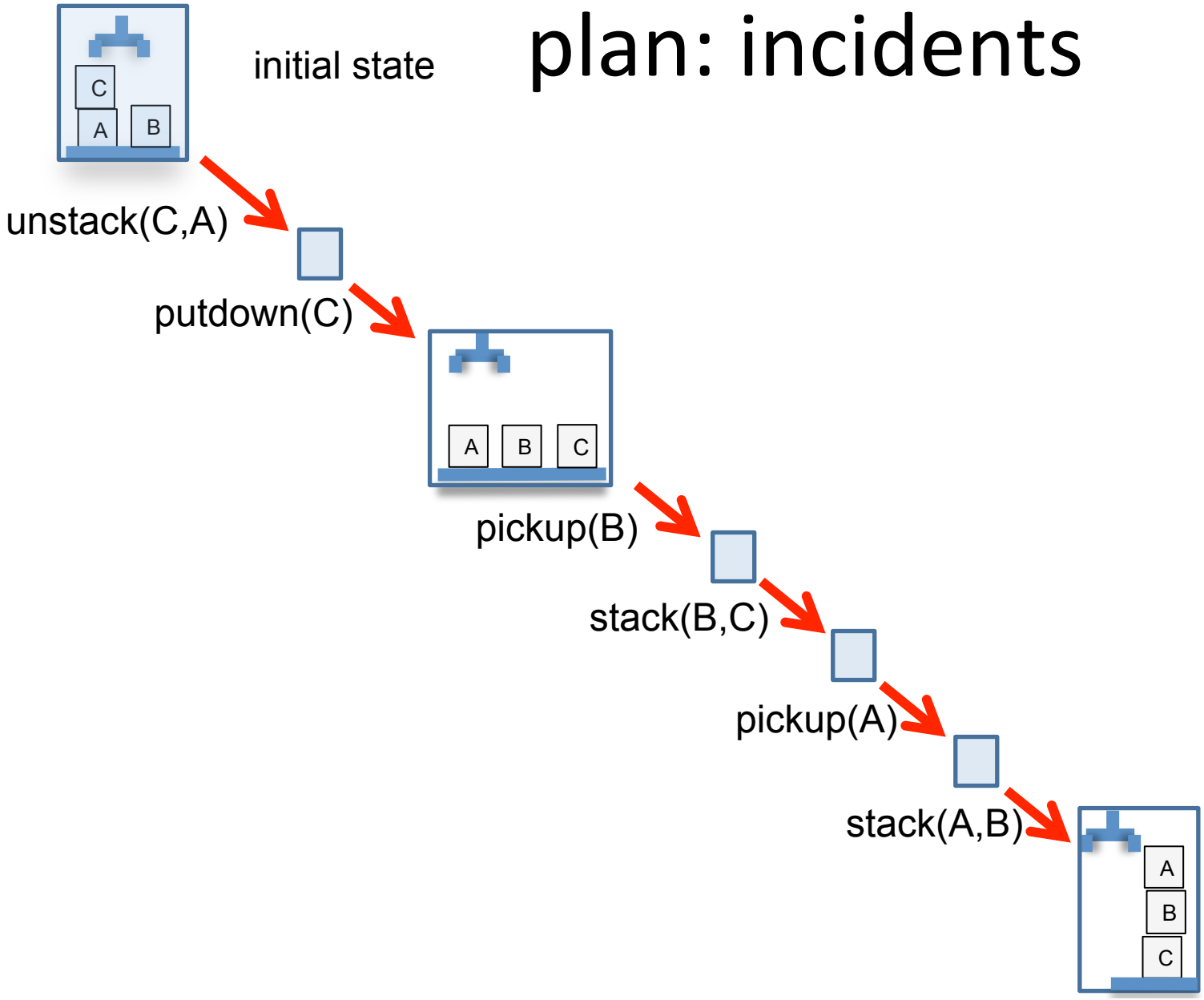


The solution

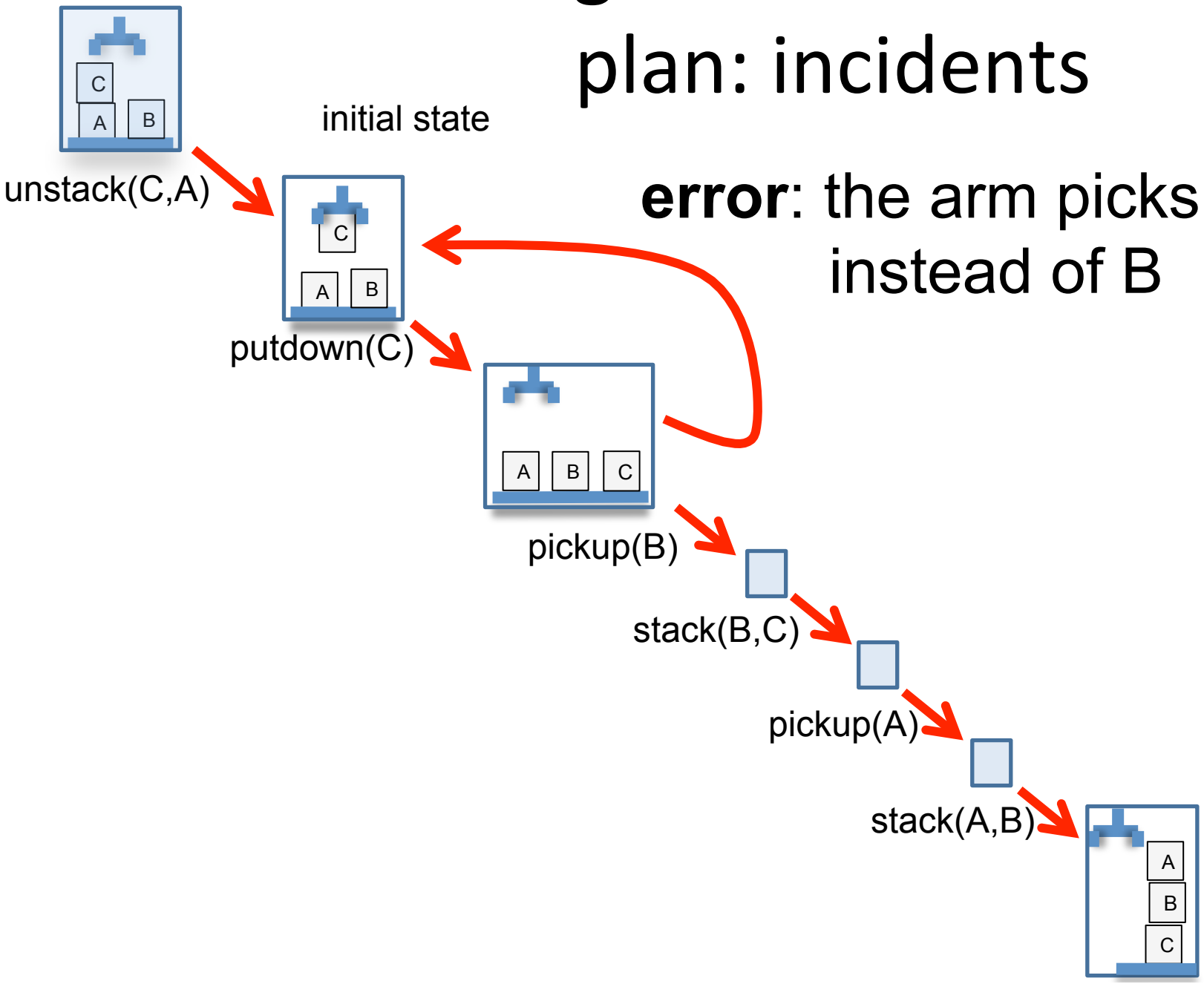


$N = 6$

Tracking the execution of the plan: incidents

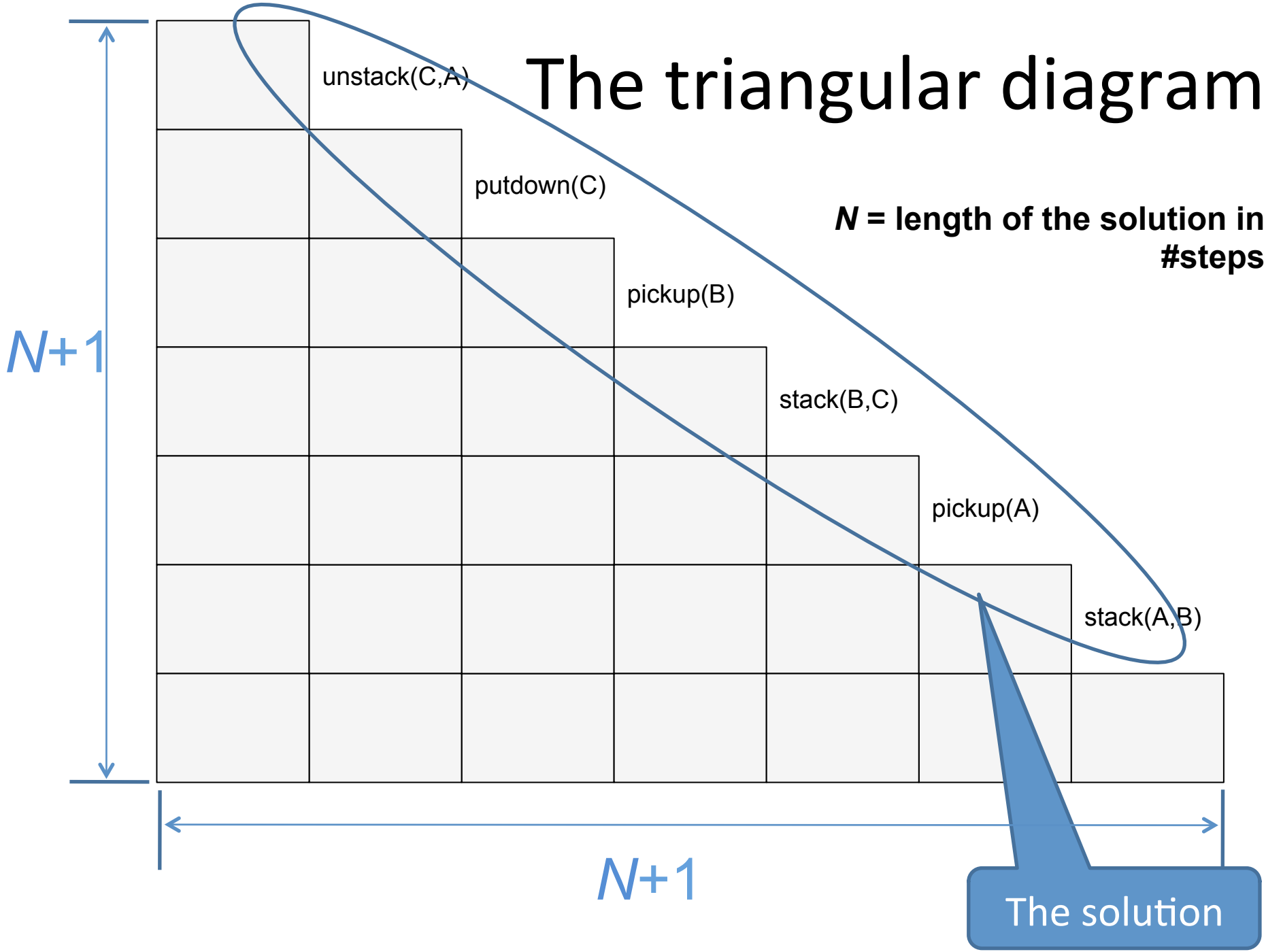


Tracking the execution of the plan: incidents



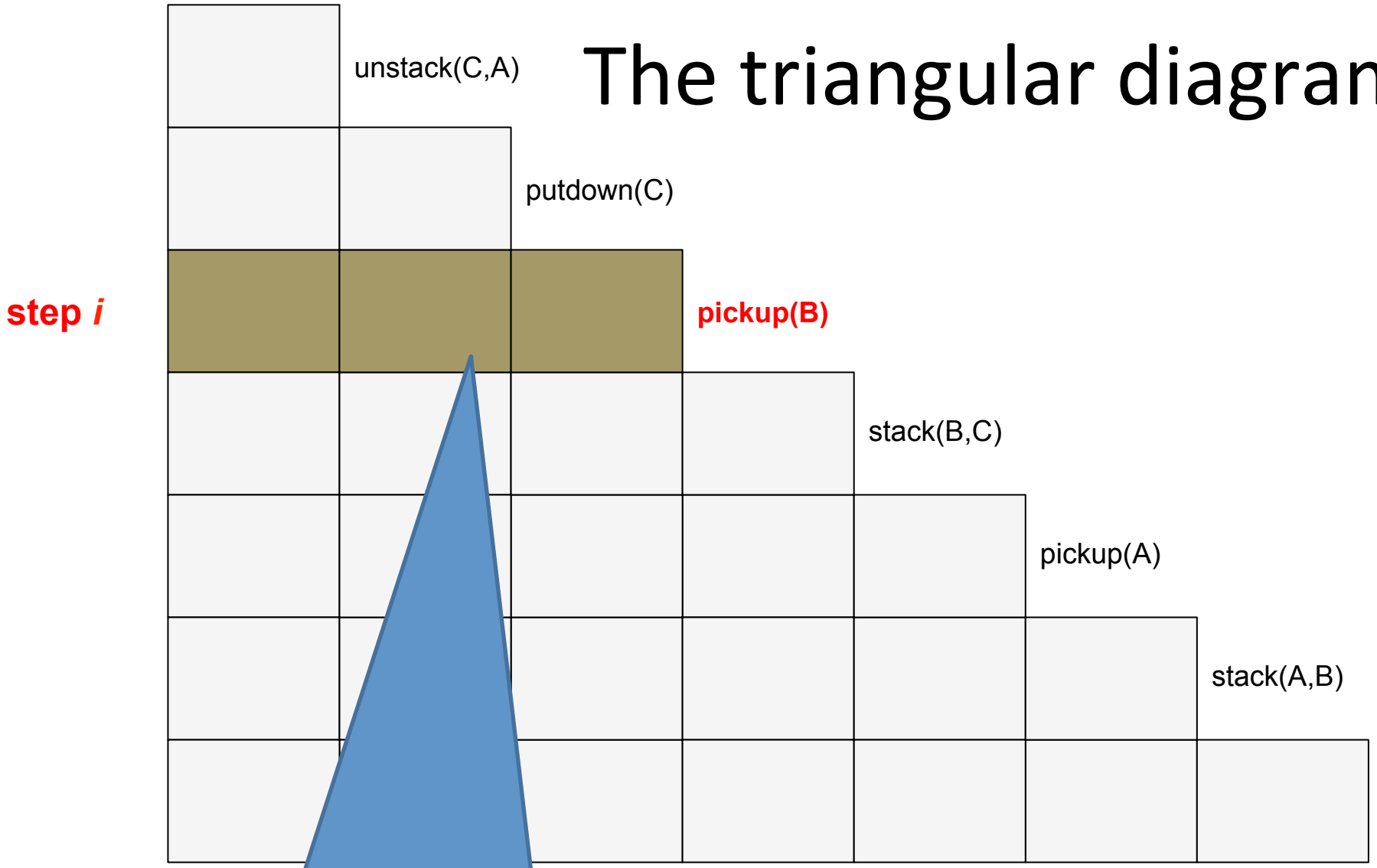
error: the arm picks up C instead of B

The triangular diagram

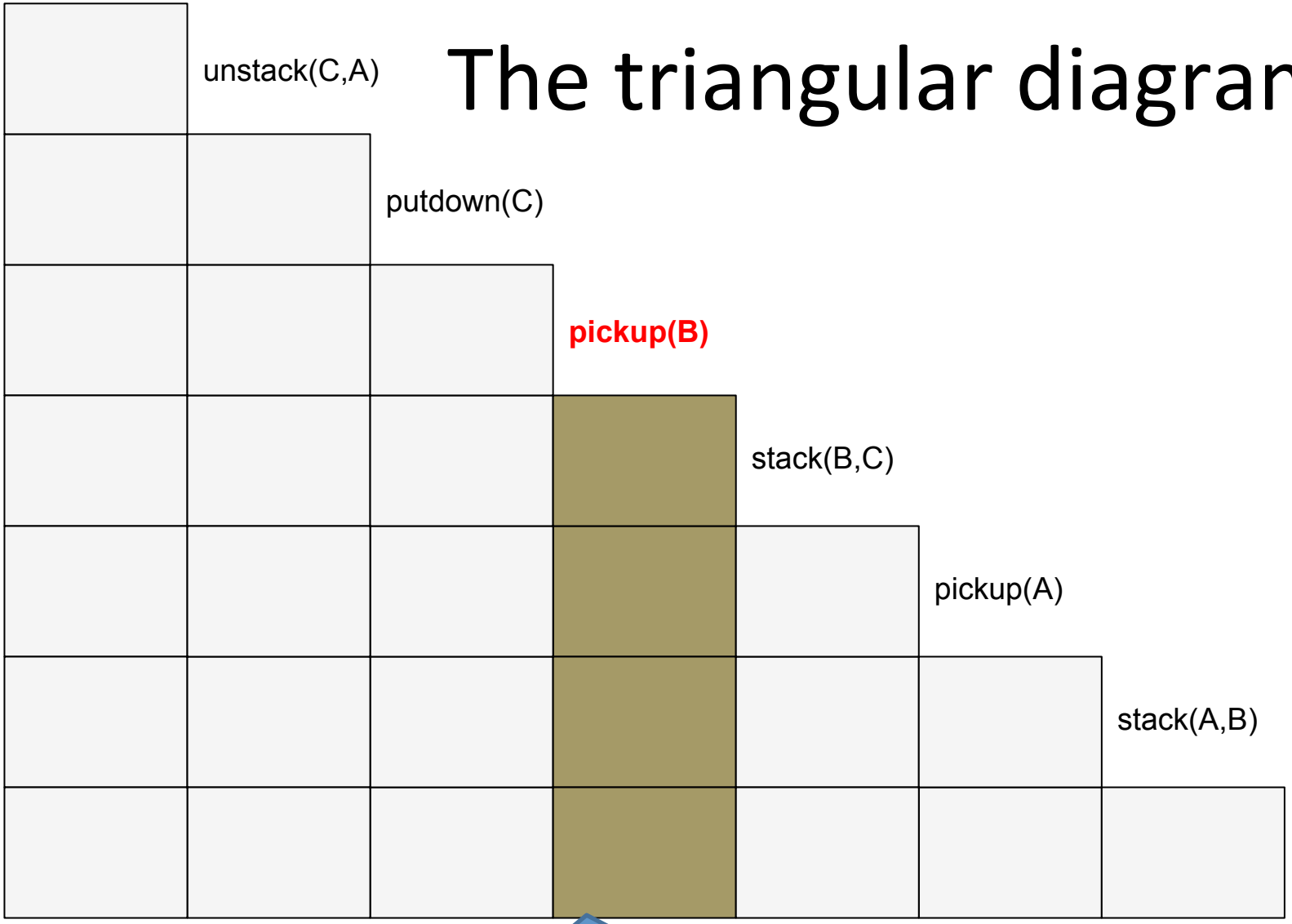


***N* = length of the solution in #steps**

The triangular diagram

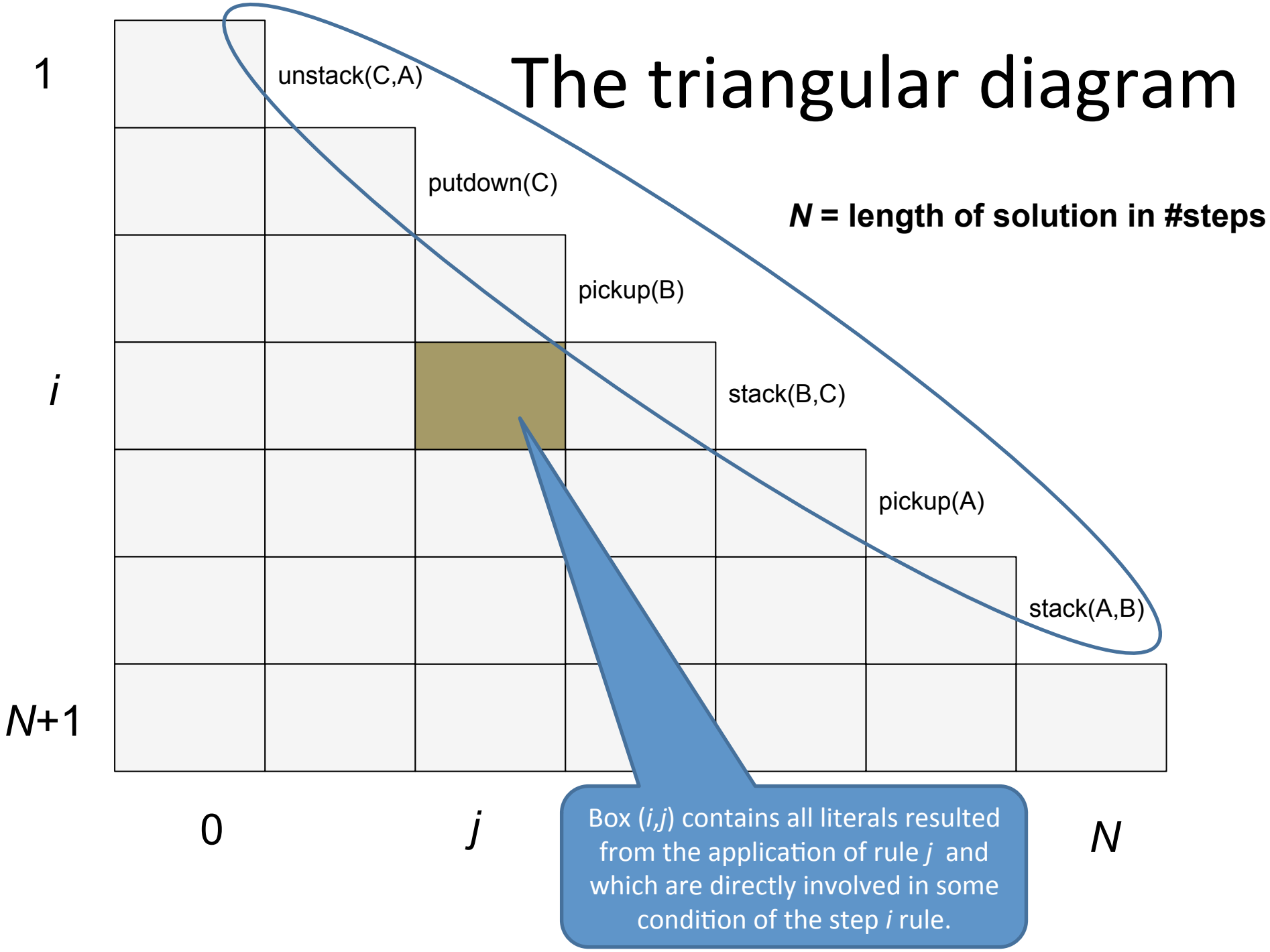


The triangular diagram

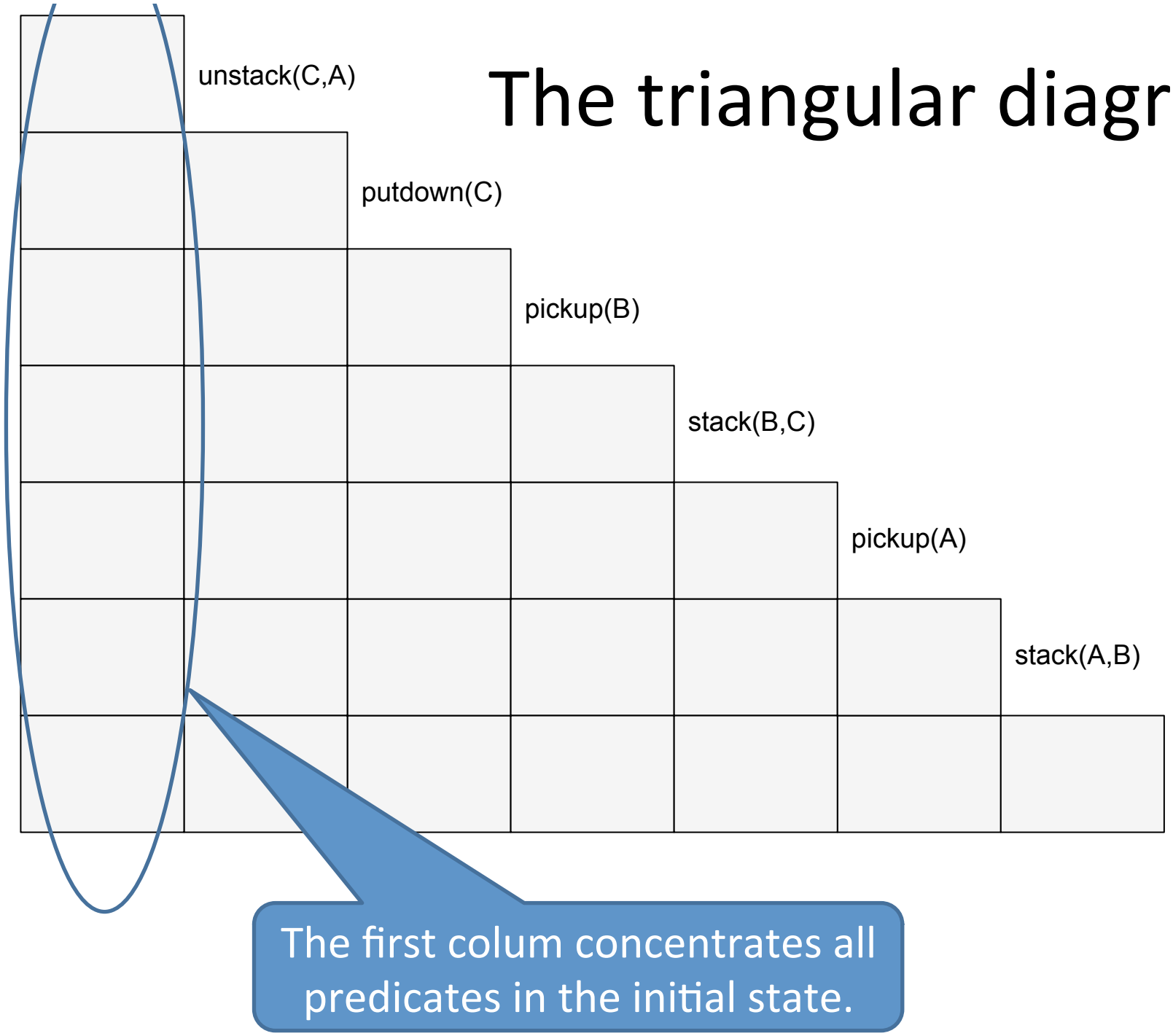


These boxes concentrate all predicates in the list A of the rule *i* step

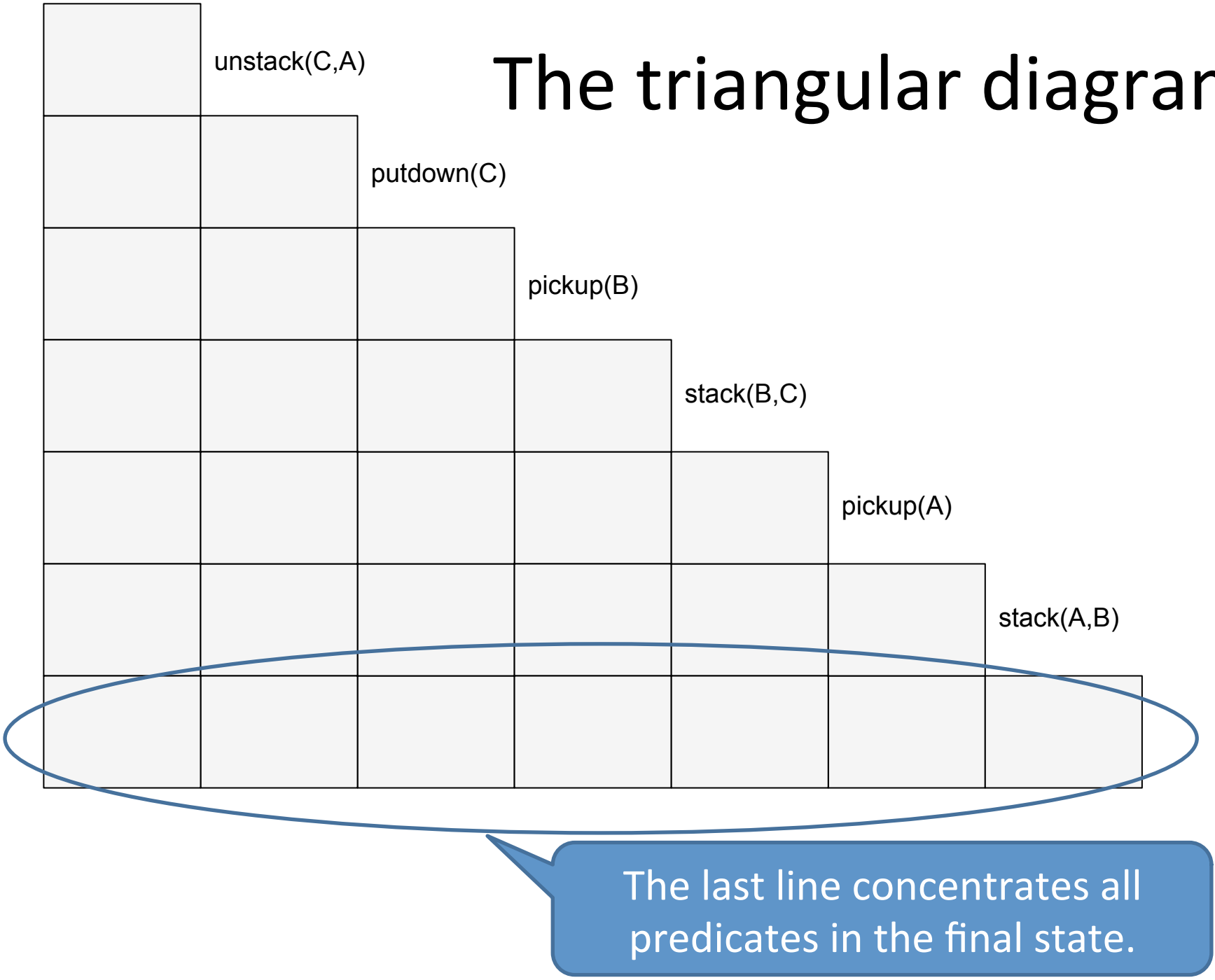
The triangular diagram



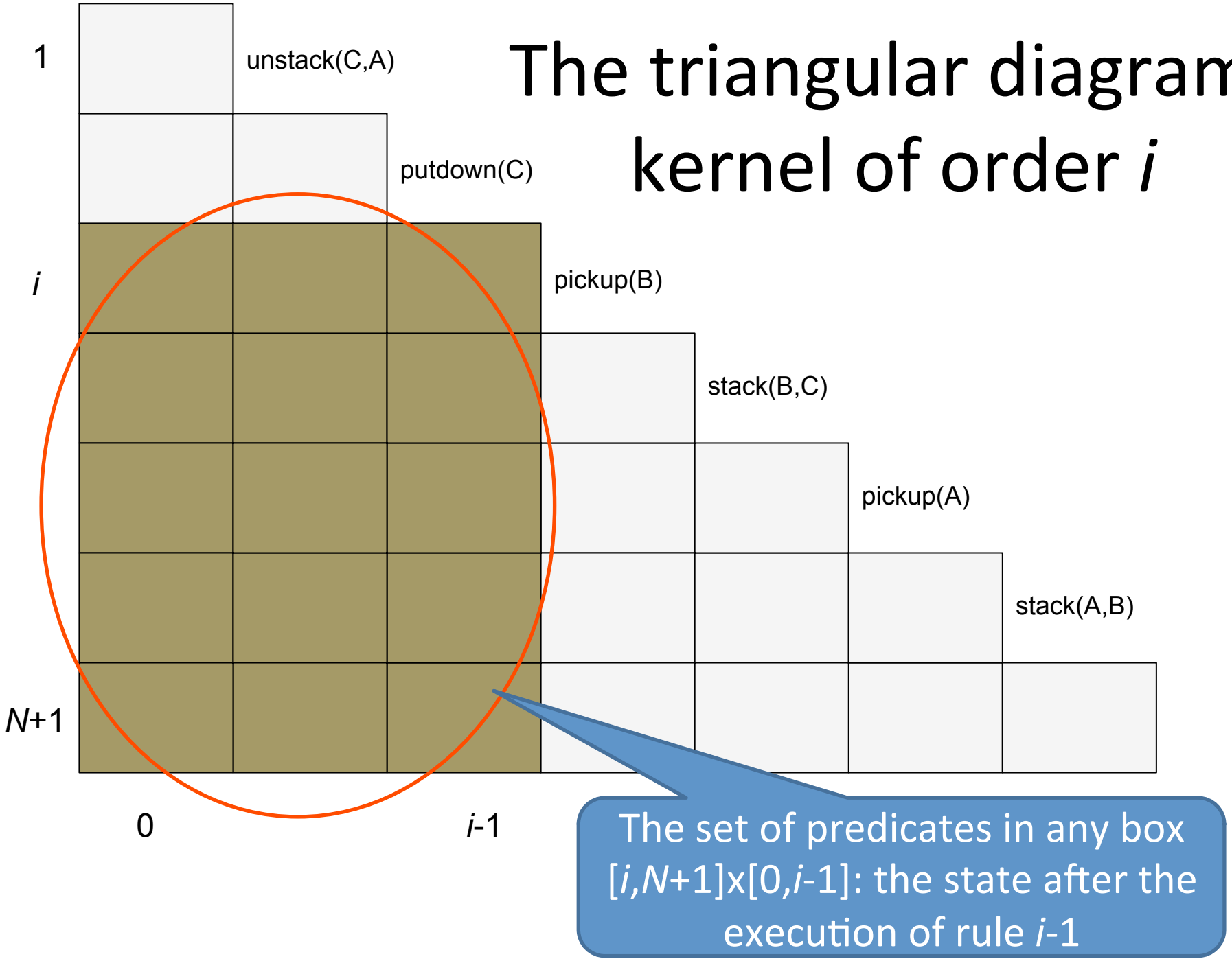
The triangular diagram



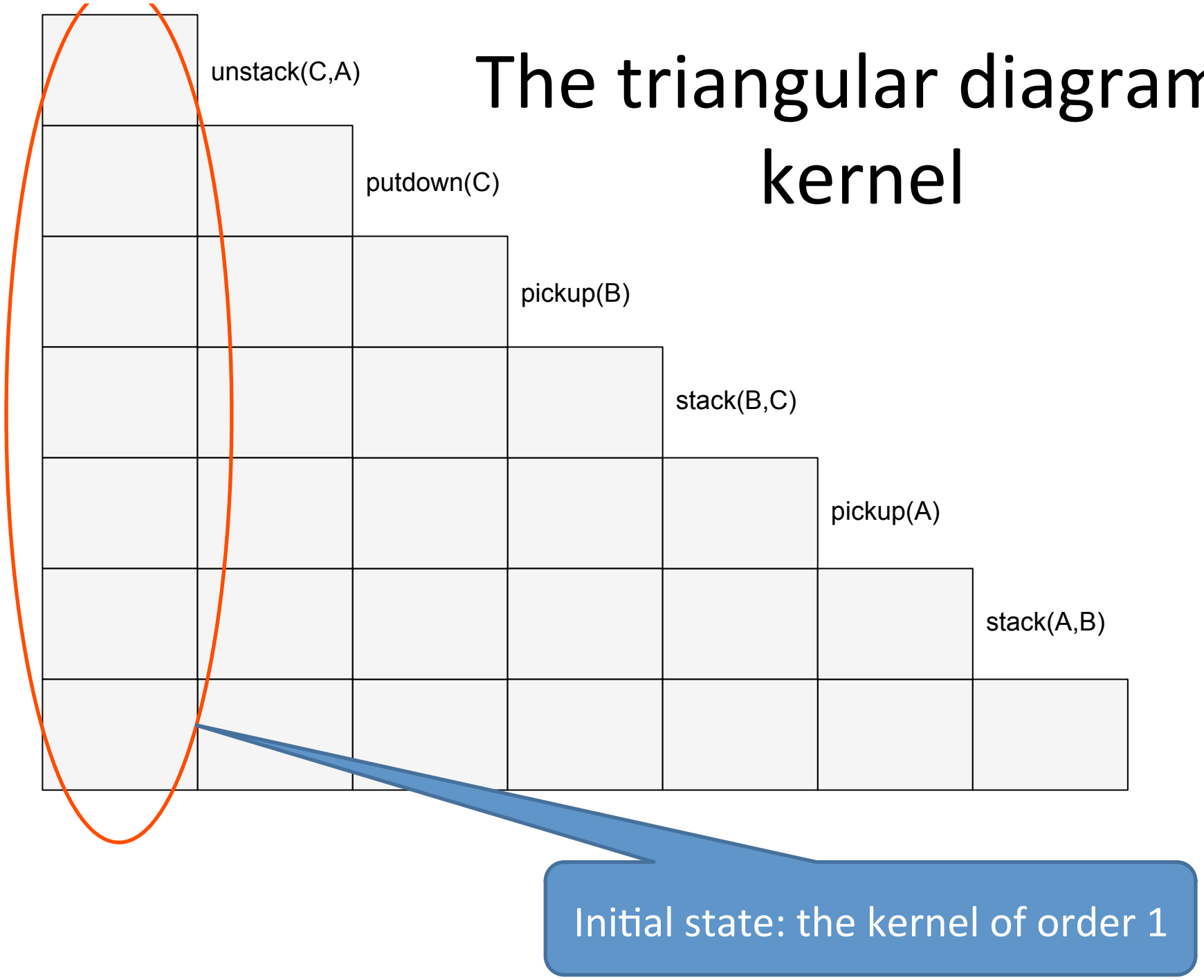
The triangular diagram



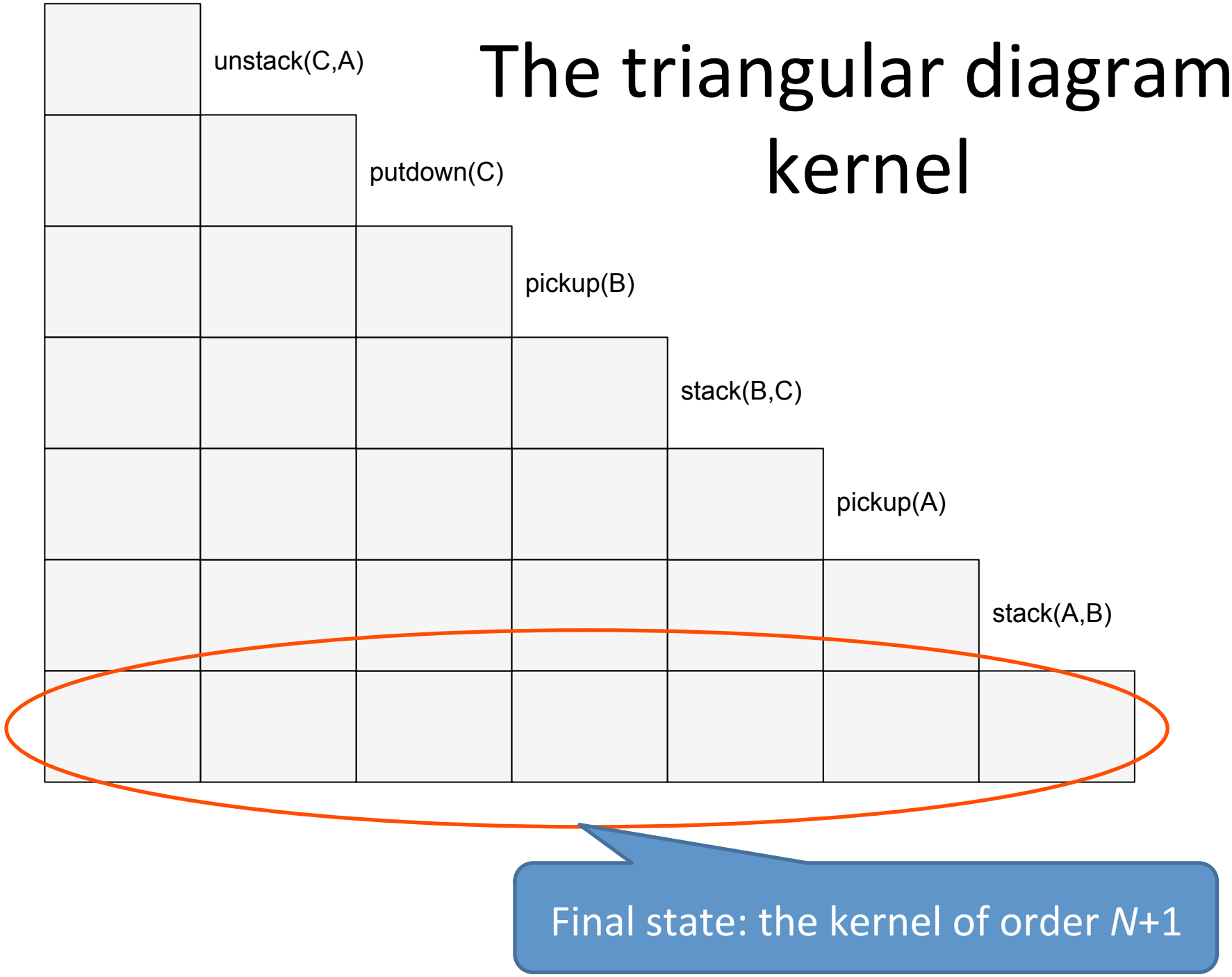
The triangular diagram: kernel of order *i*



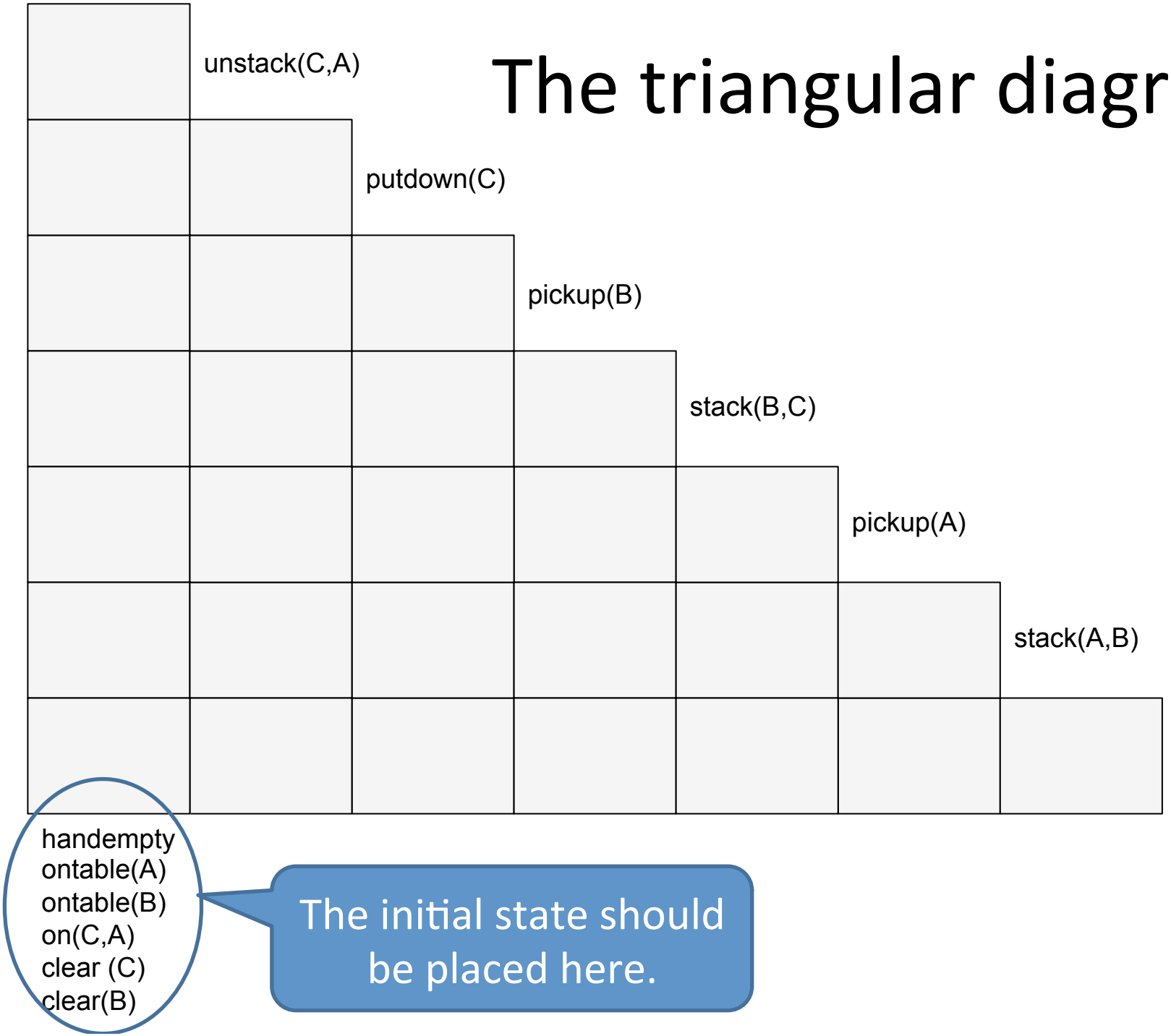
The triangular diagram: kernel



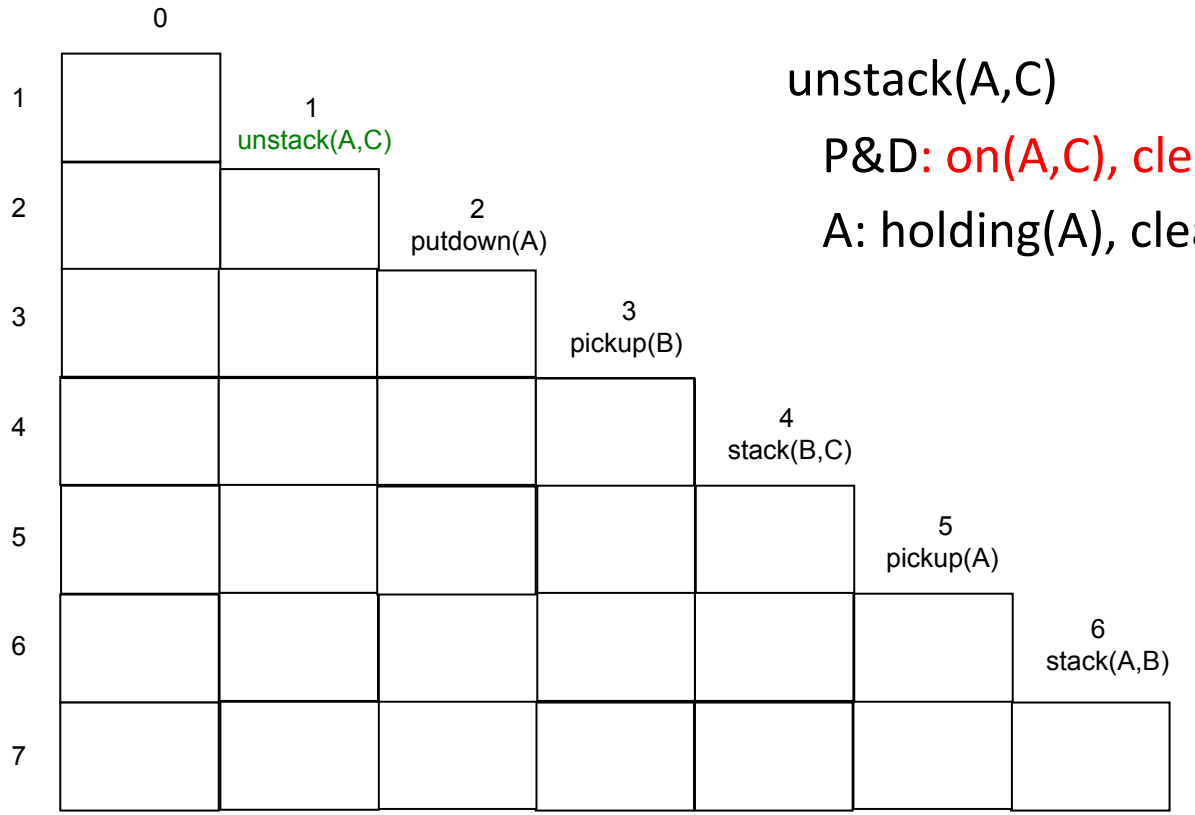
The triangular diagram: kernel



The triangular diagram



Plan execution: step 1



unstack(A,C)

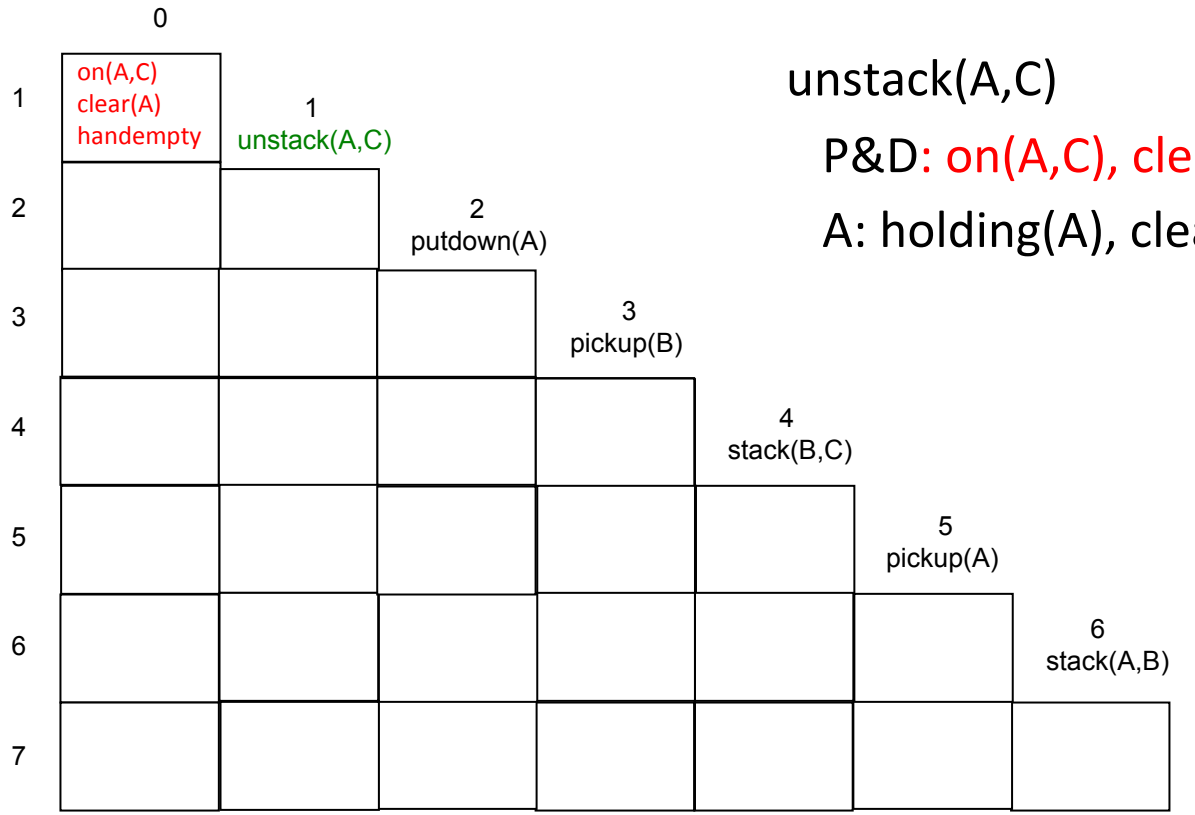
P&D: on(A,C), clear(A), handempty

A: holding(A), clear(C)

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

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



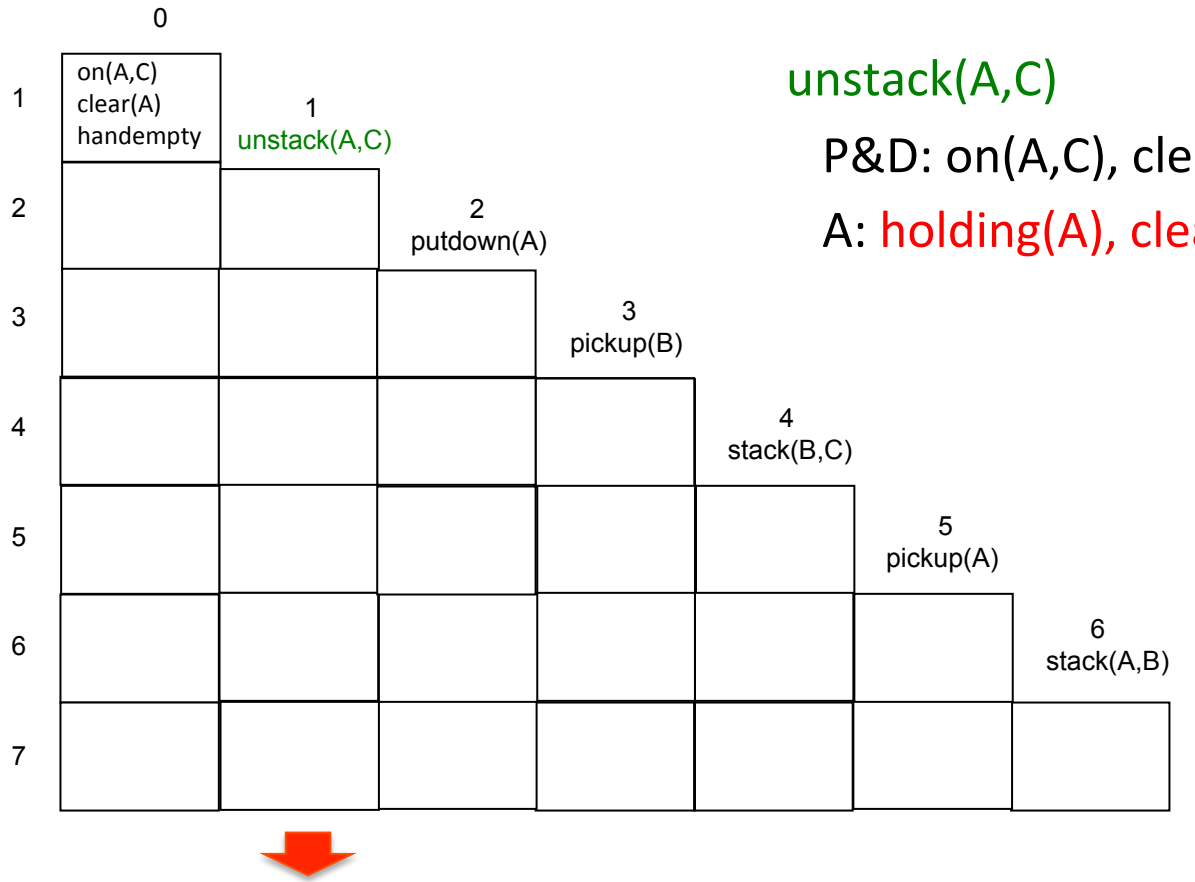
unstack(A,C)

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

A: holding(A), clear(C)

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

Plan execution: step 1



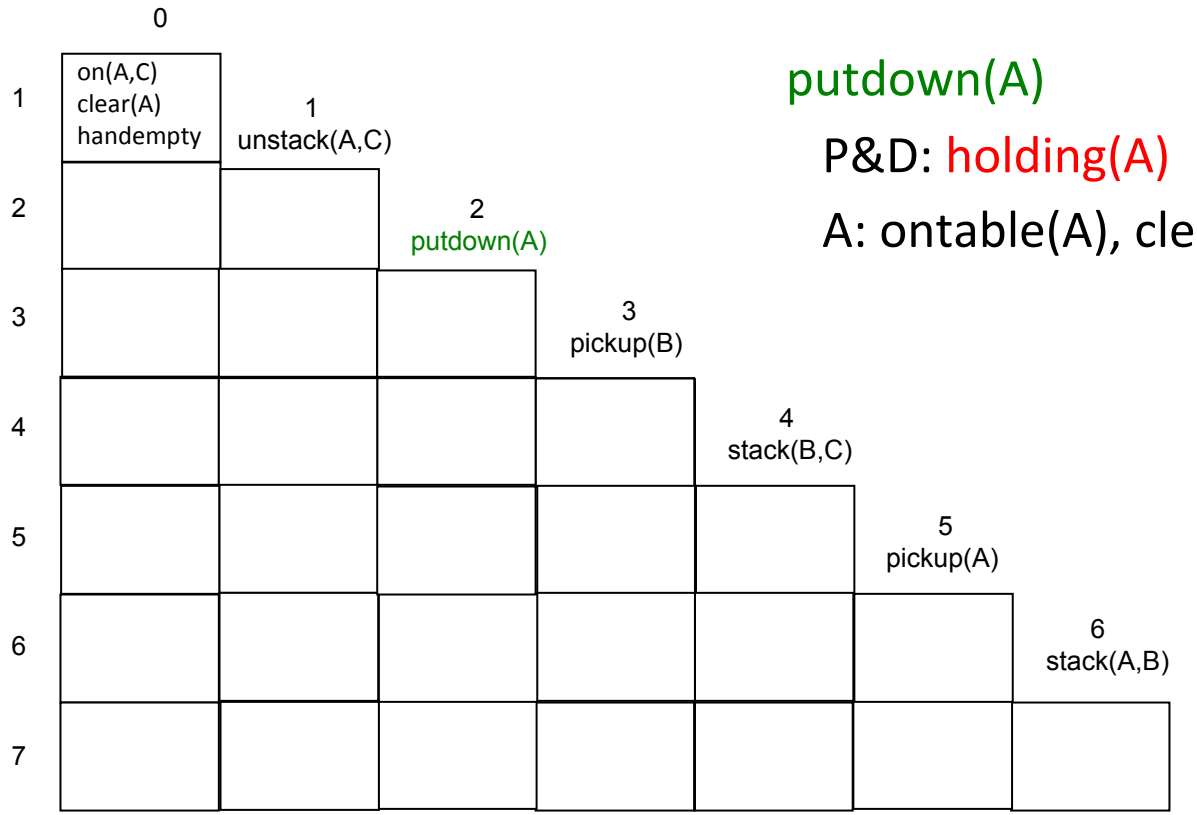
unstack(A,C)

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

A: holding(A), clear(C)

Colect here all predicates of list A of rule 1.

Plan execution: step 2



putdown(A)

P&D: holding(A)

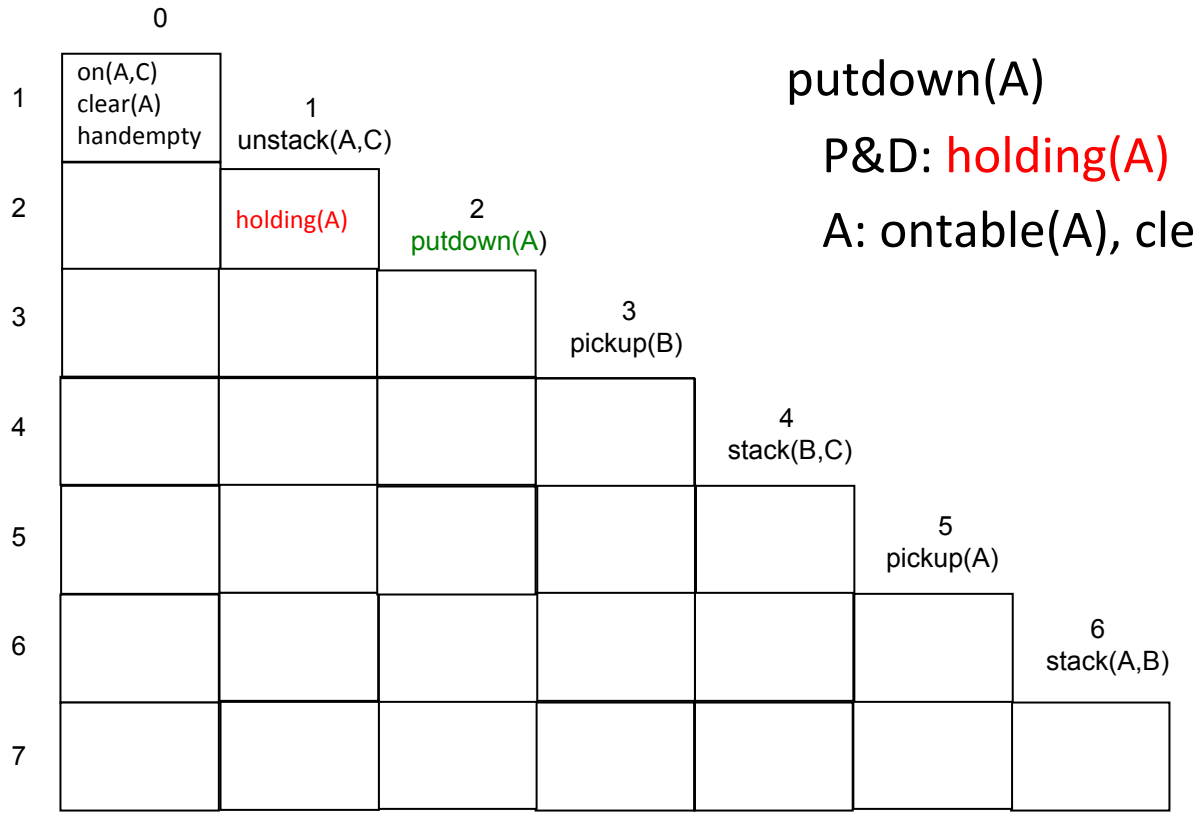
A: ontable(A), clear(A), handempty

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

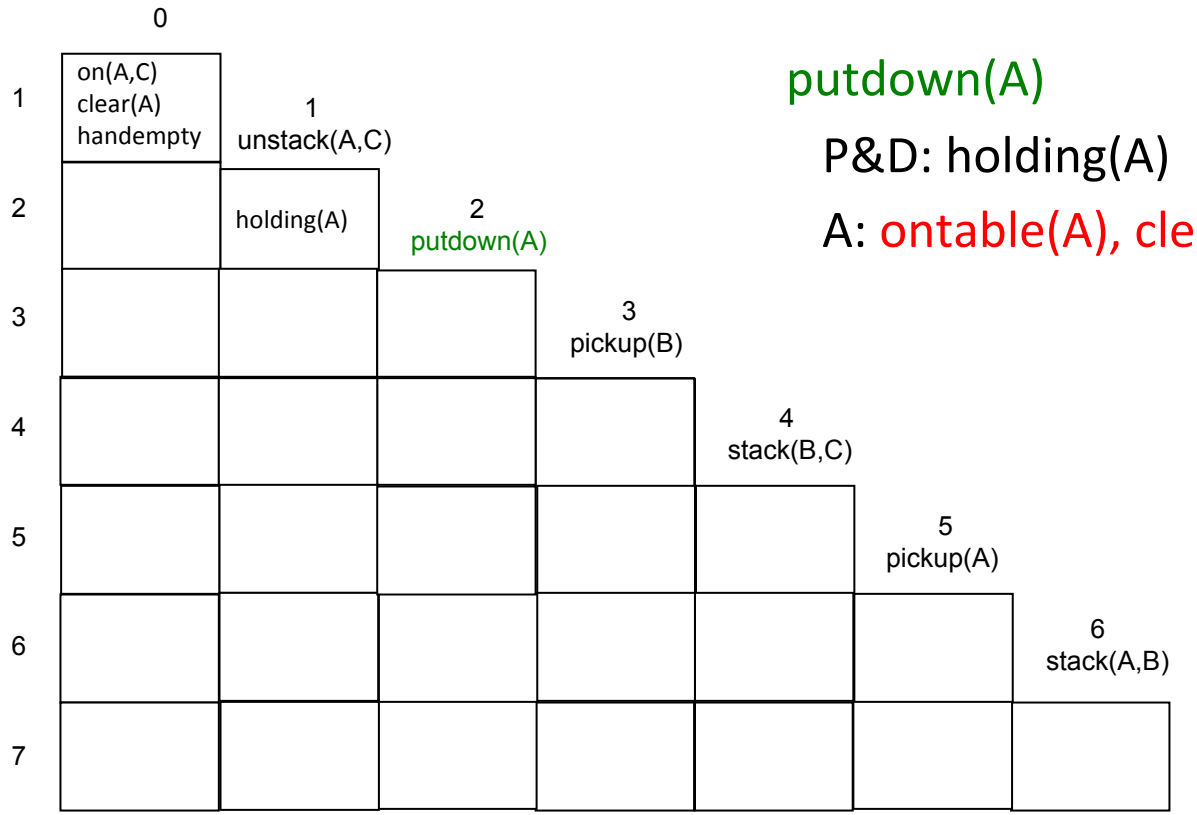


putdown(A)

P&D: holding(A)

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

Plan execution: step 2



putdown(A)

P&D: holding(A)

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

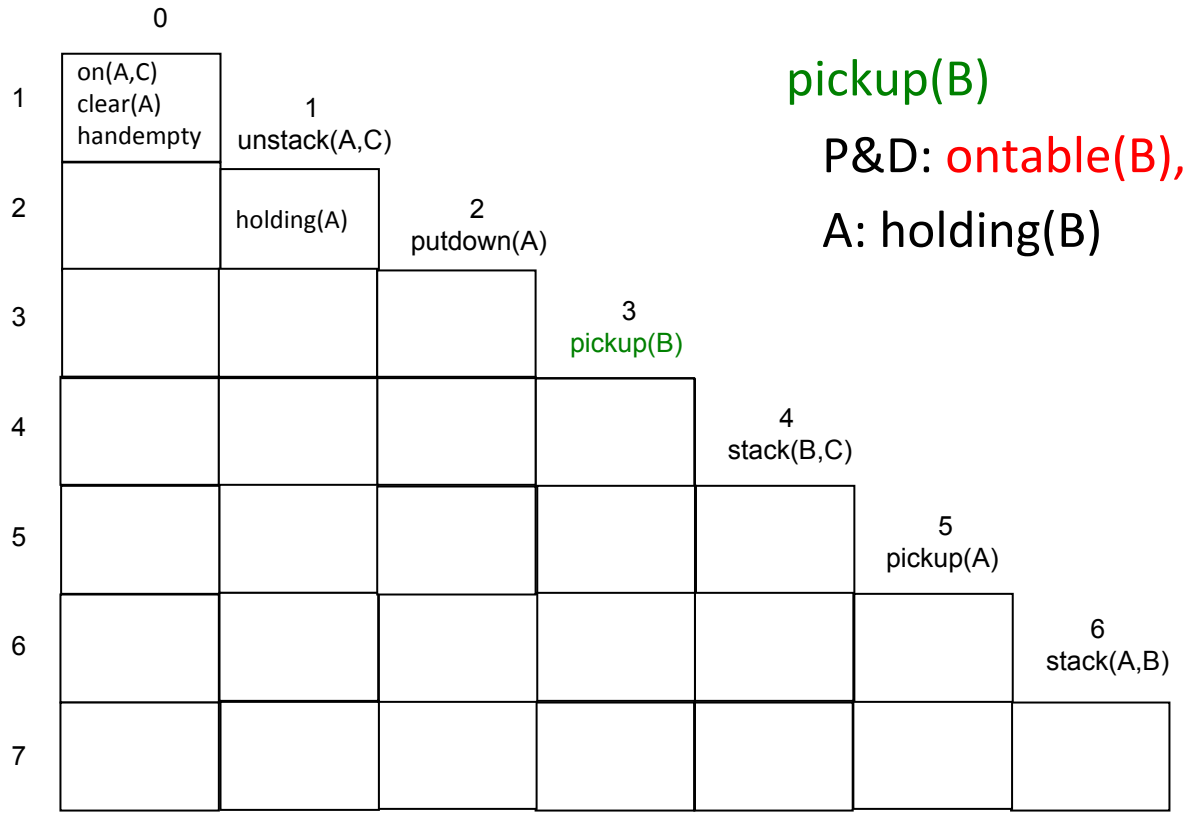
ontable(C)
ontable(B)
clear(B)

clear(C)

ontable(A)
clear(A)
handempty

Colect 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)

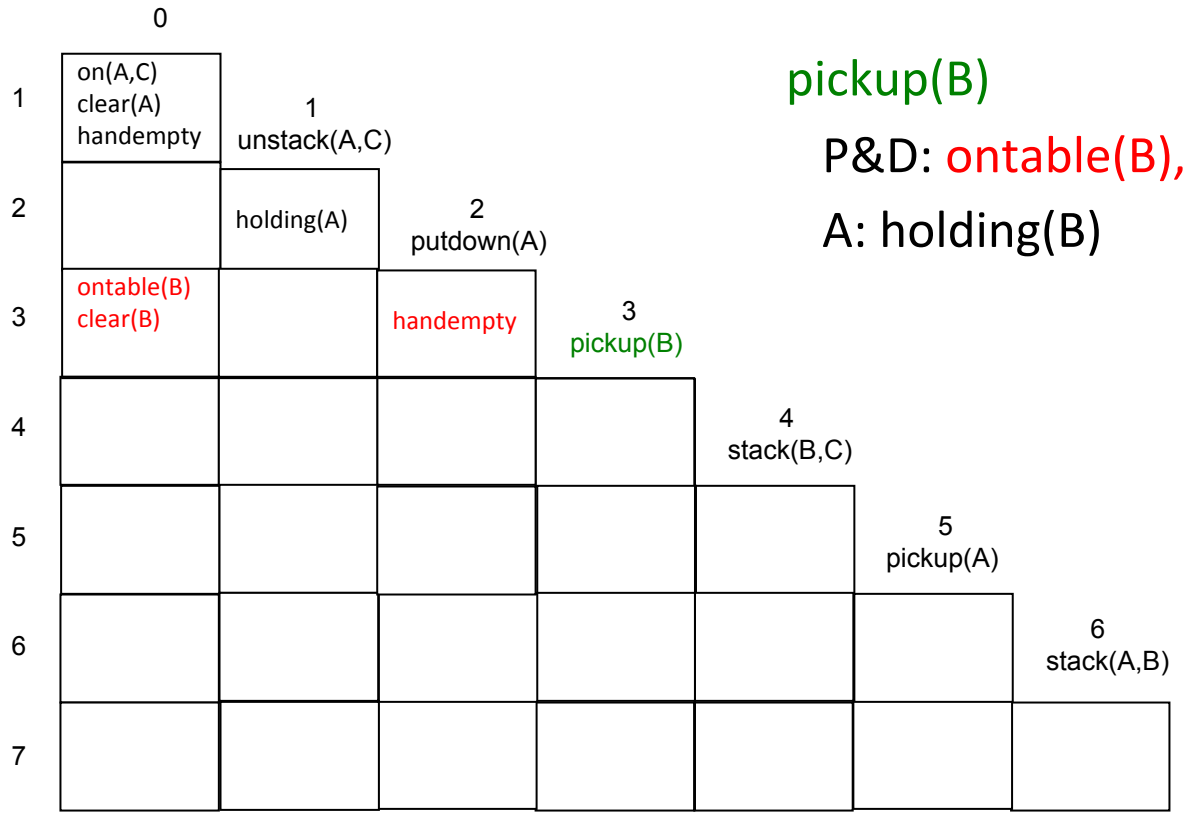
ontable(C)
ontable(B)
clear(B)

clear(C)

ontable(A)
clear(A)
handempty

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

Plan execution: step 3



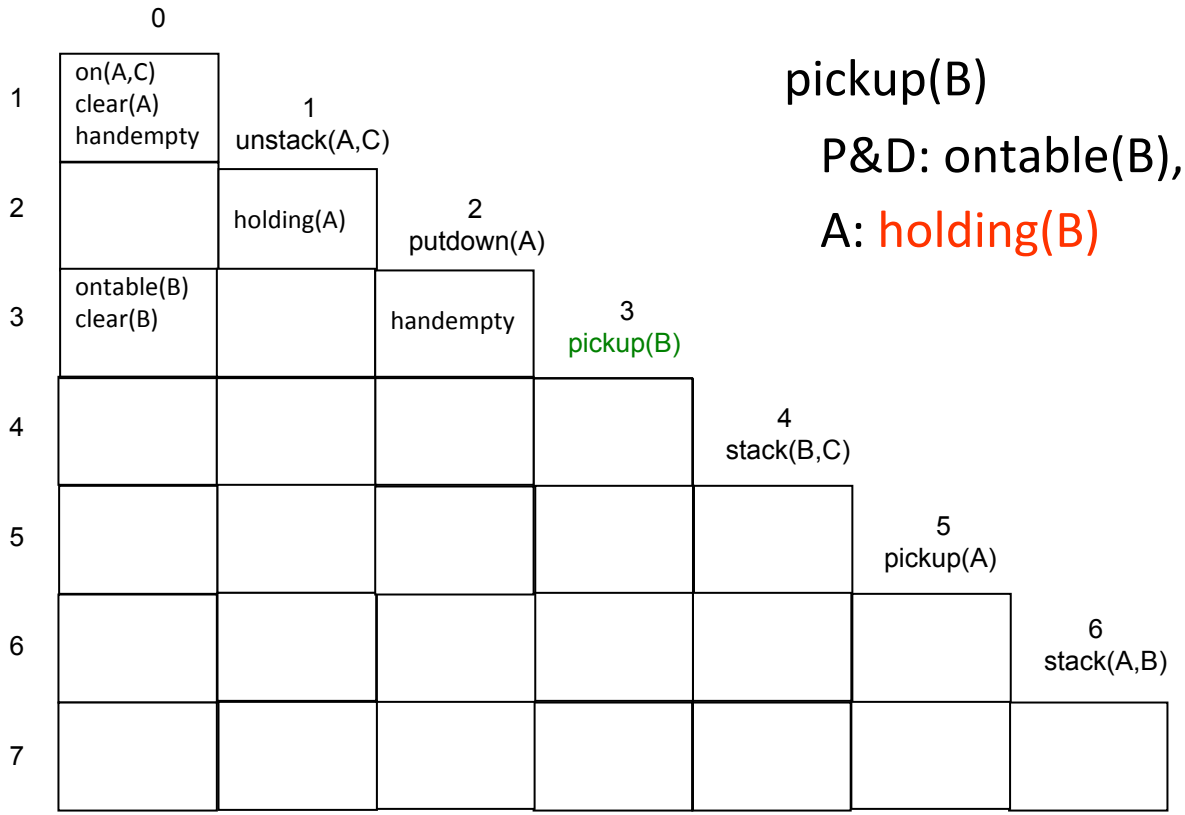
pickup(B)

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

A: holding(B)

… and bring them up in the line of rule 3

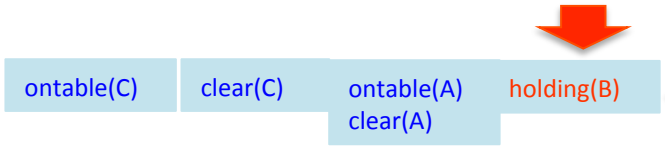
Plan execution: step 3



pickup(B)

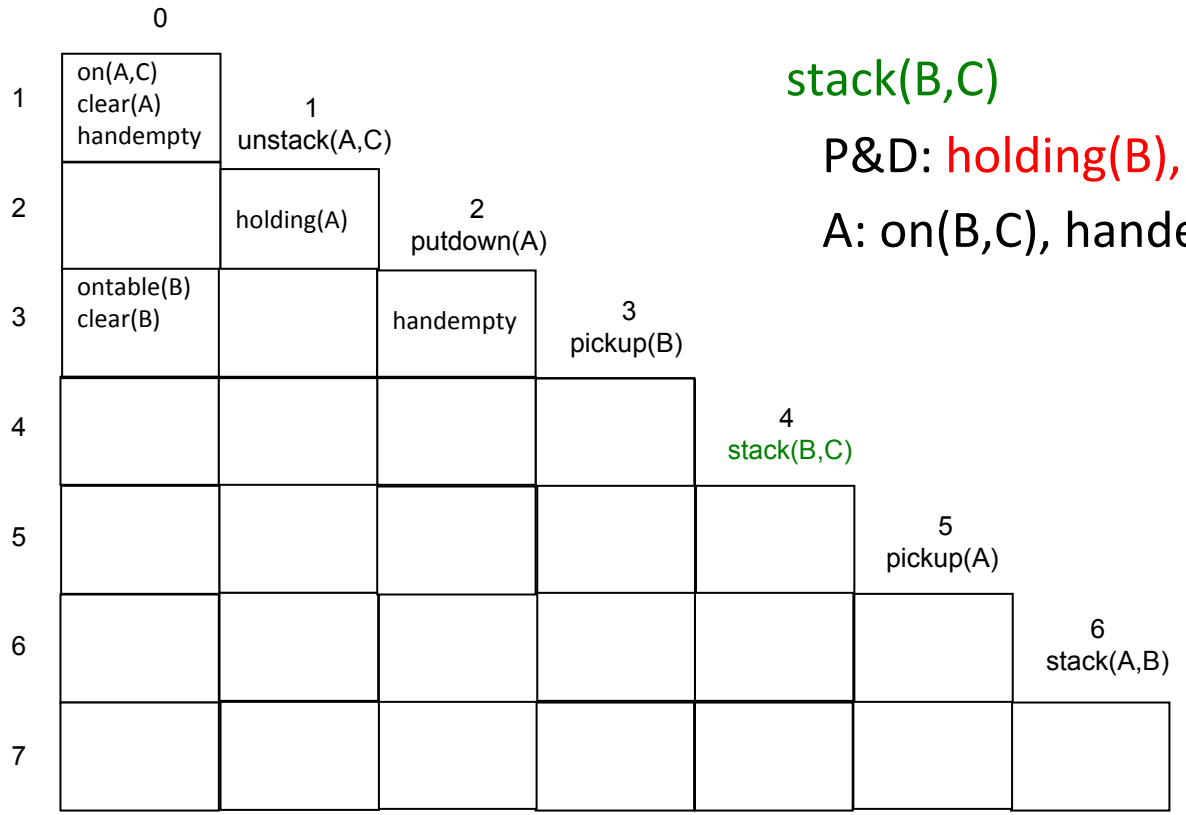
P&D: ontable(B), clear(B), handempty

A: holding(B)



Collect list A predicates of rule 3.

Plan execution: step 4



stack(B,C)

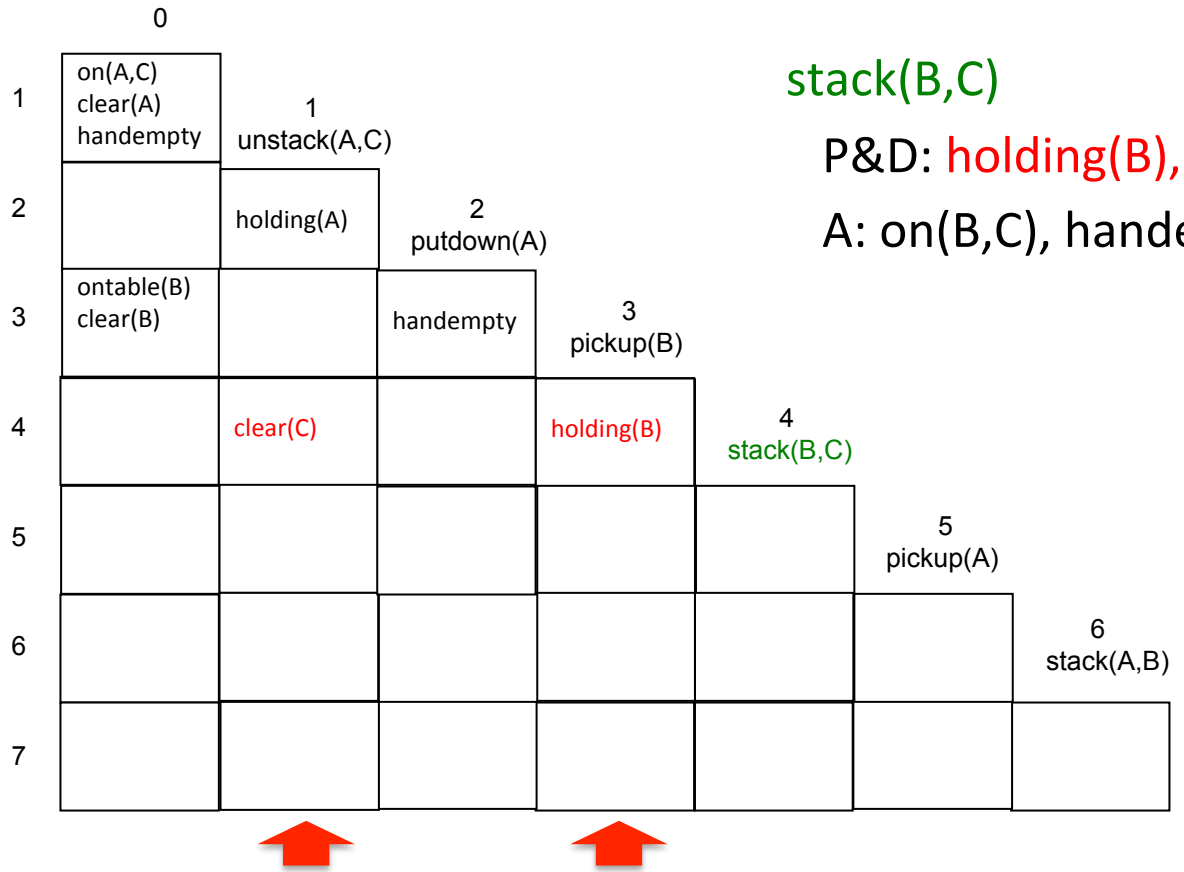
P&D: holding(B), clear(C)

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

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

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

Plan execution: step 4



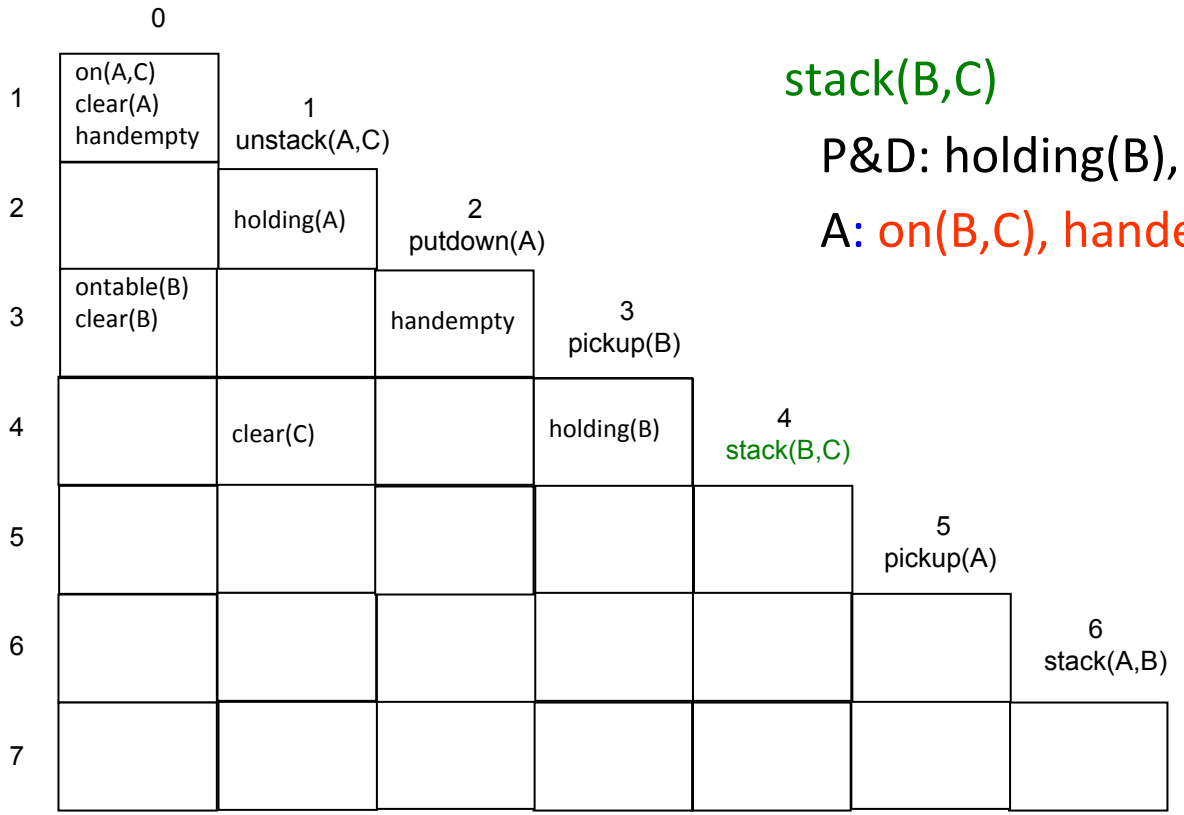
stack(B,C)

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

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

… and bring them up in the line of rule 4

Plan execution: step 4



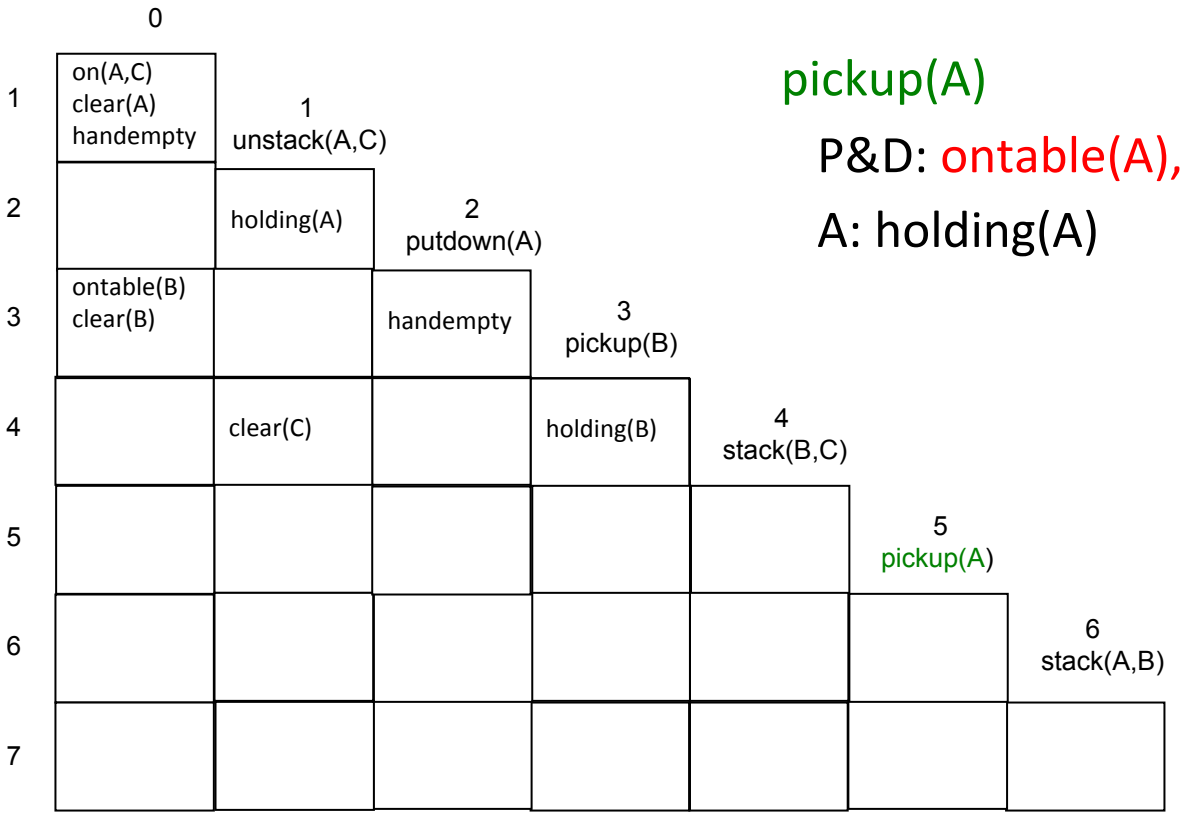
stack(B,C)

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

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

Collect here down all list A predicates of rule 4.

Plan execution: step 5



pickup(A)

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

A: holding(A)

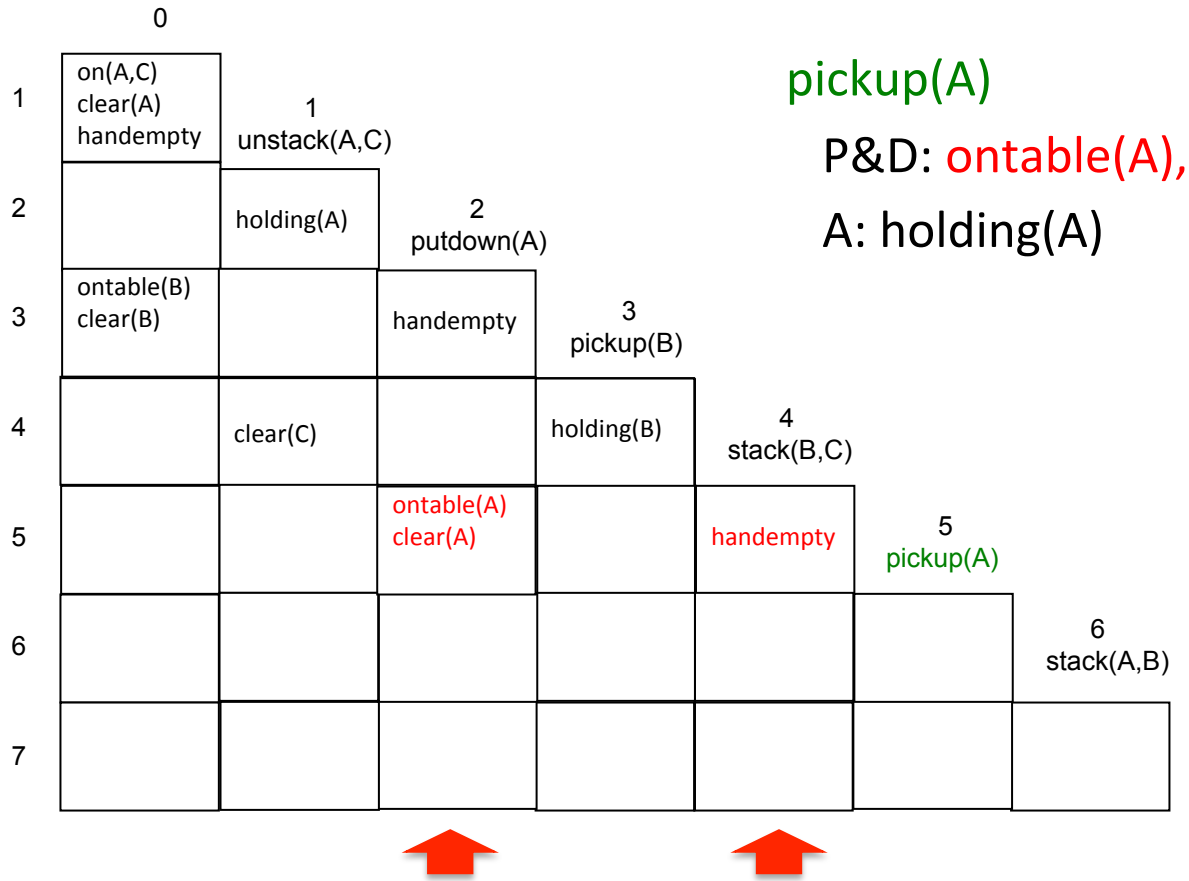
ontable(C)

ontable(A)
clear(A)

on(B,C)
handempty
clear(B)

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

Plan execution: step 5



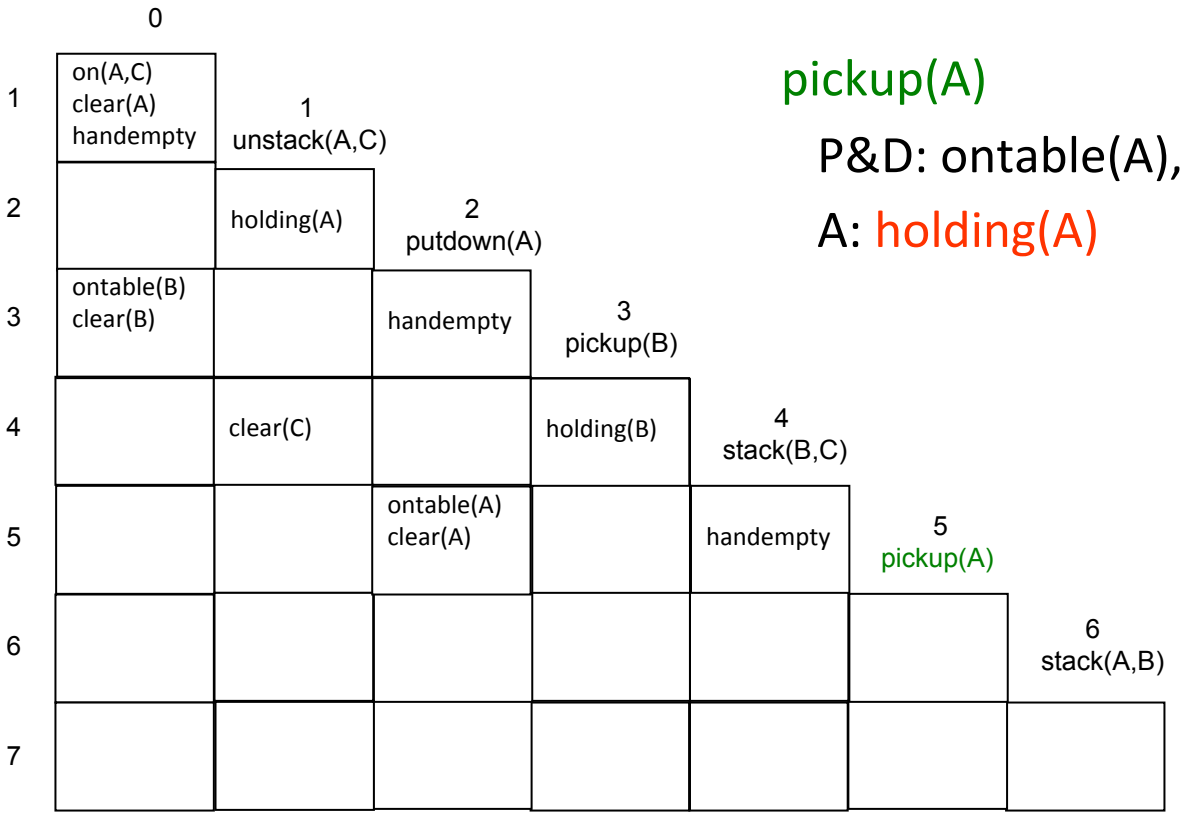
pickup(A)

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

A: holding(A)

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

Plan execution: step 5



pickup(A)

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

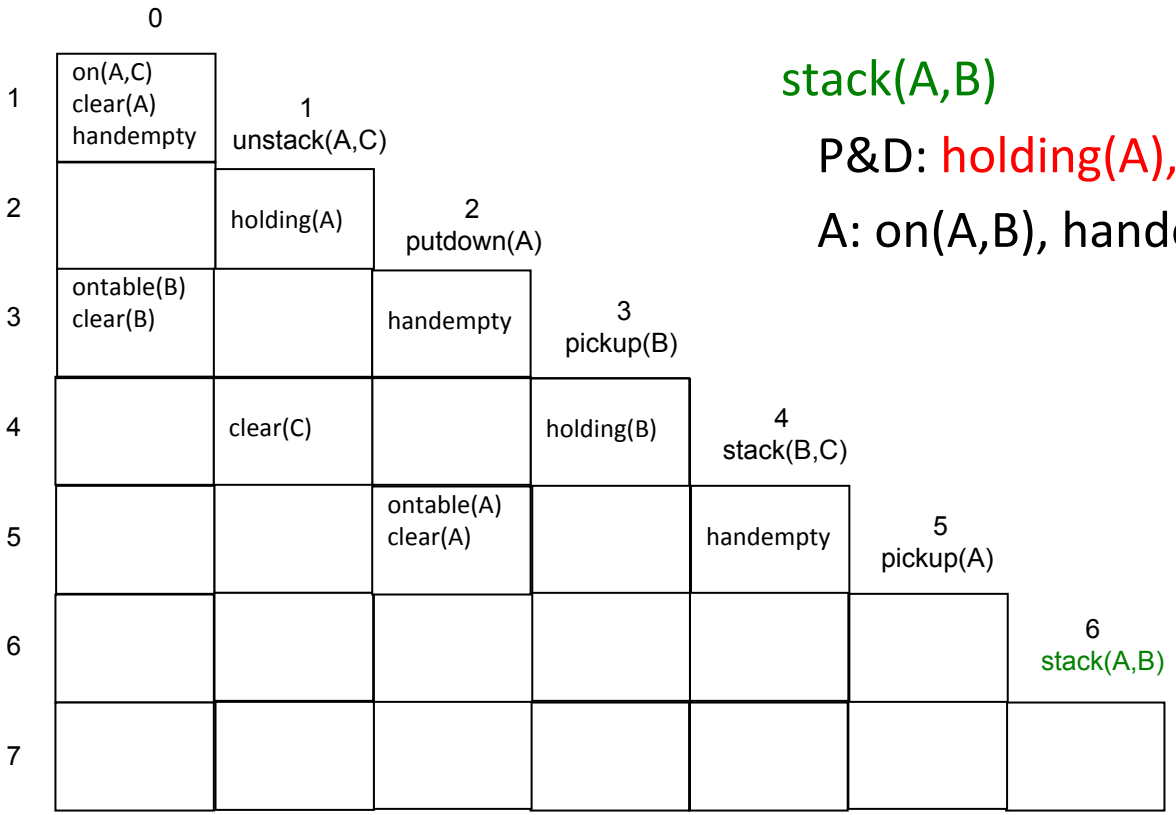
A: holding(A)

ontable(C)

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

Collect down all list A predicates of rule 5.

Plan execution: step 6



stack(A,B)

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

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

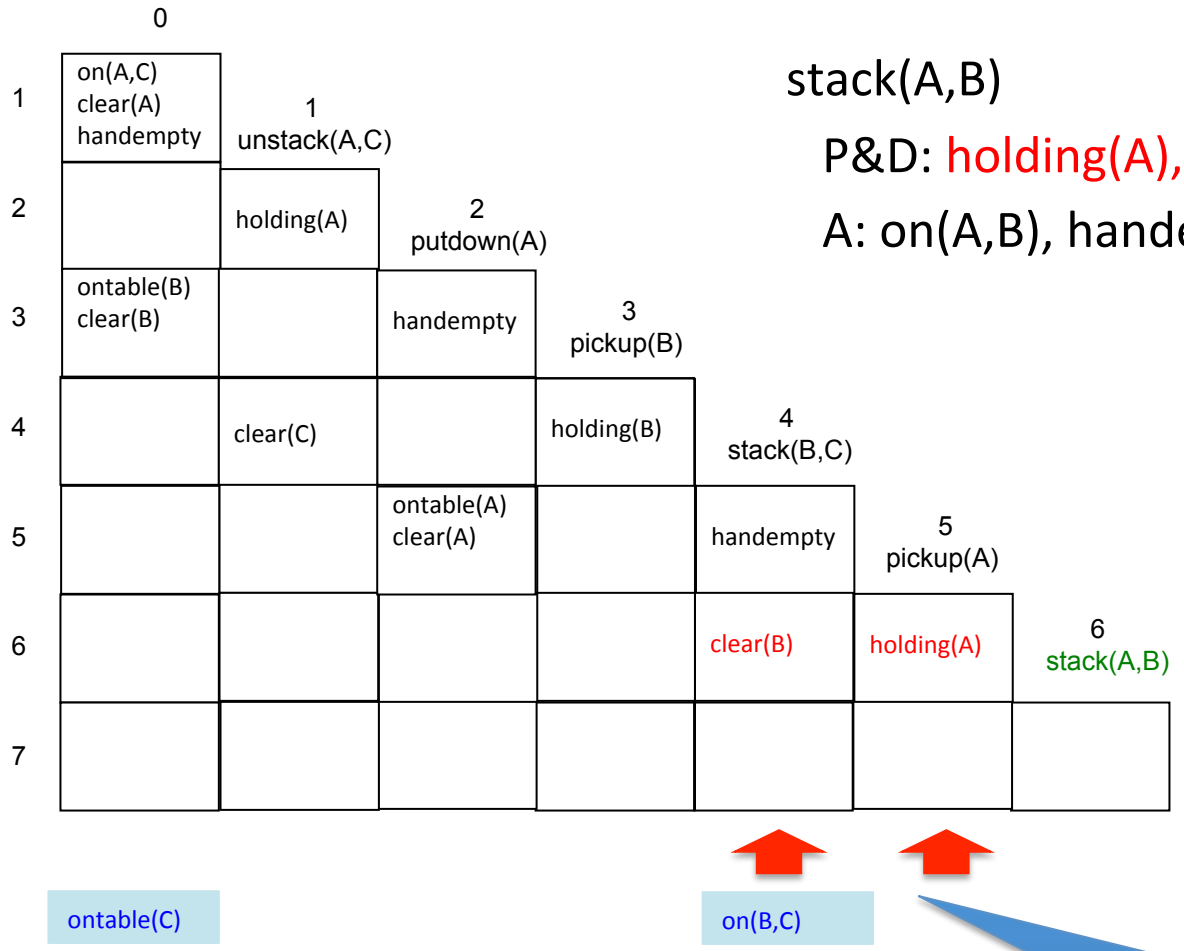
ontable(C)

on(B,C)
clear(B)

holding(A)

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

Plan execution: step 6



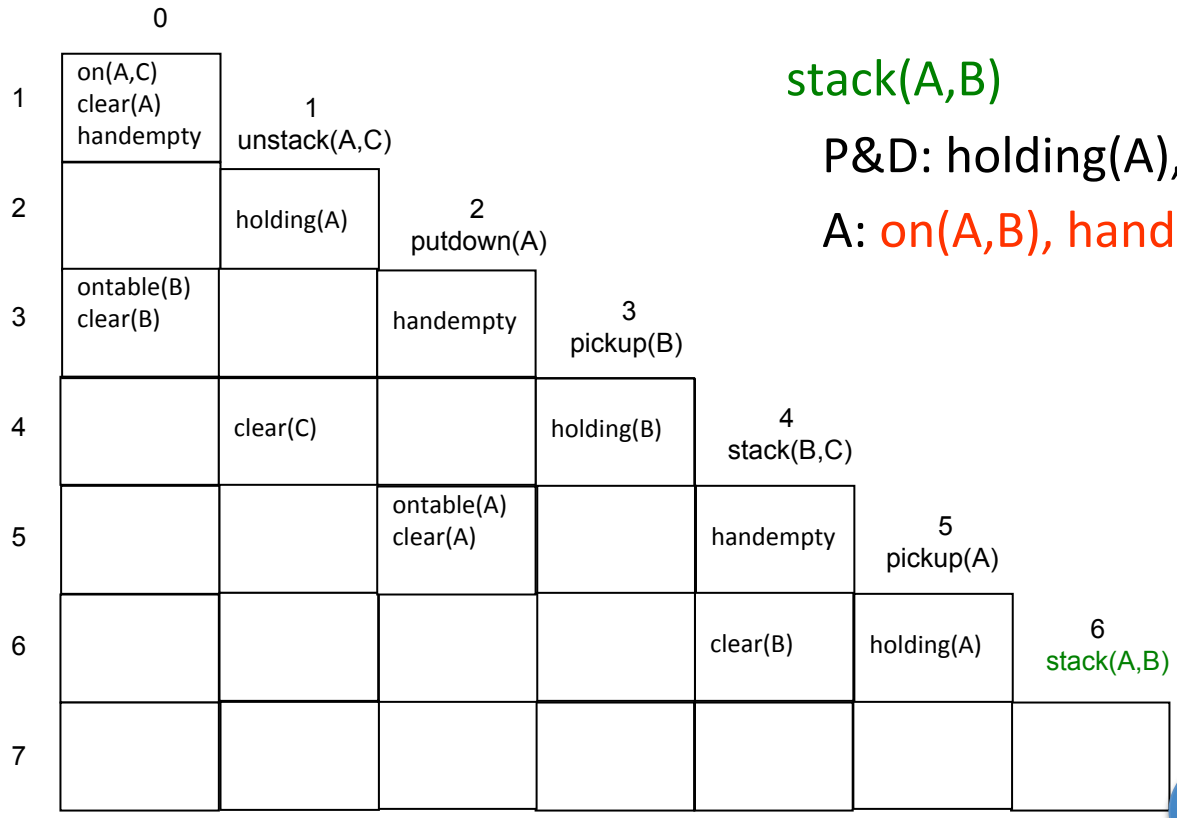
stack(A,B)

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

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

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

Plan execution: step 6



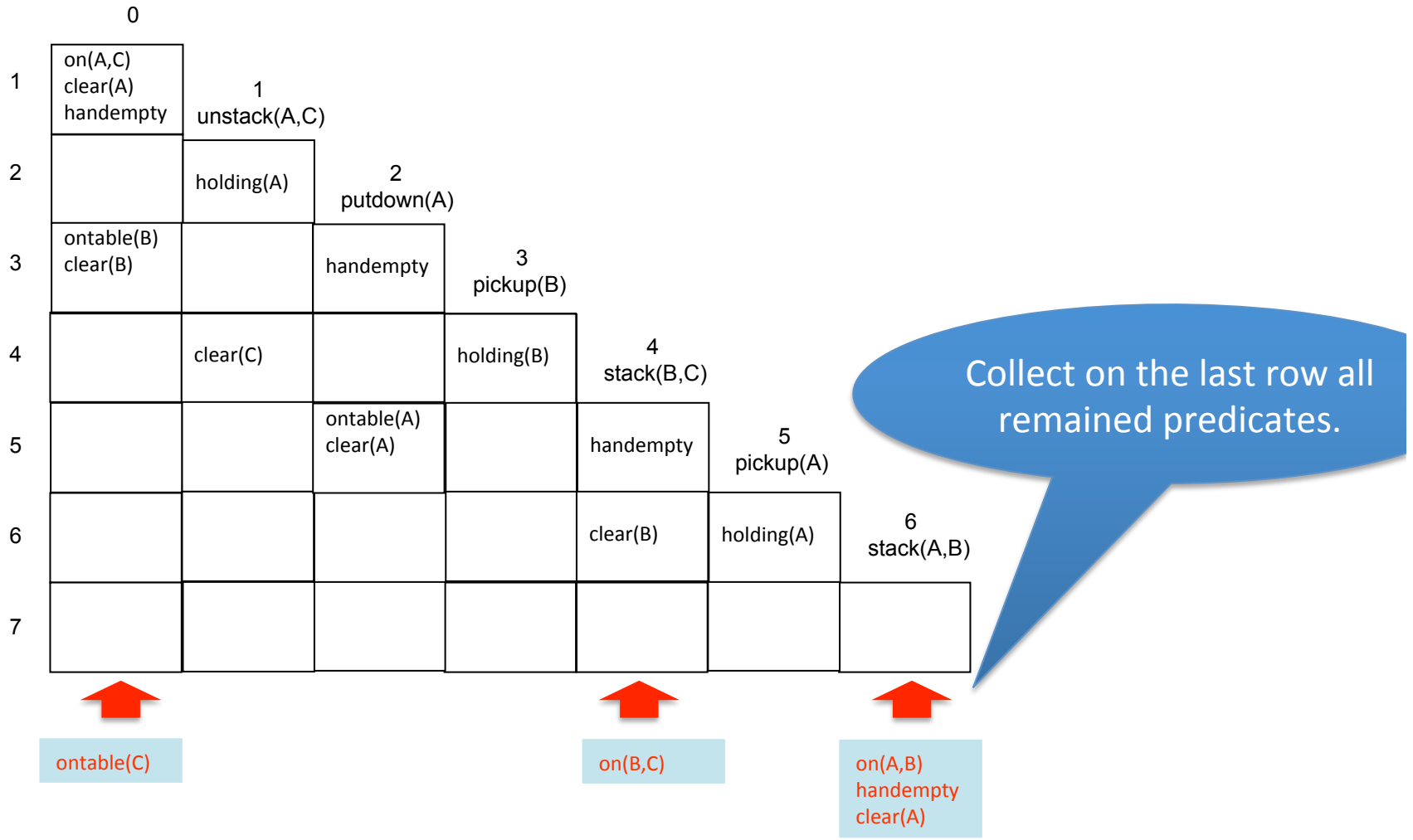
stack(A,B)

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

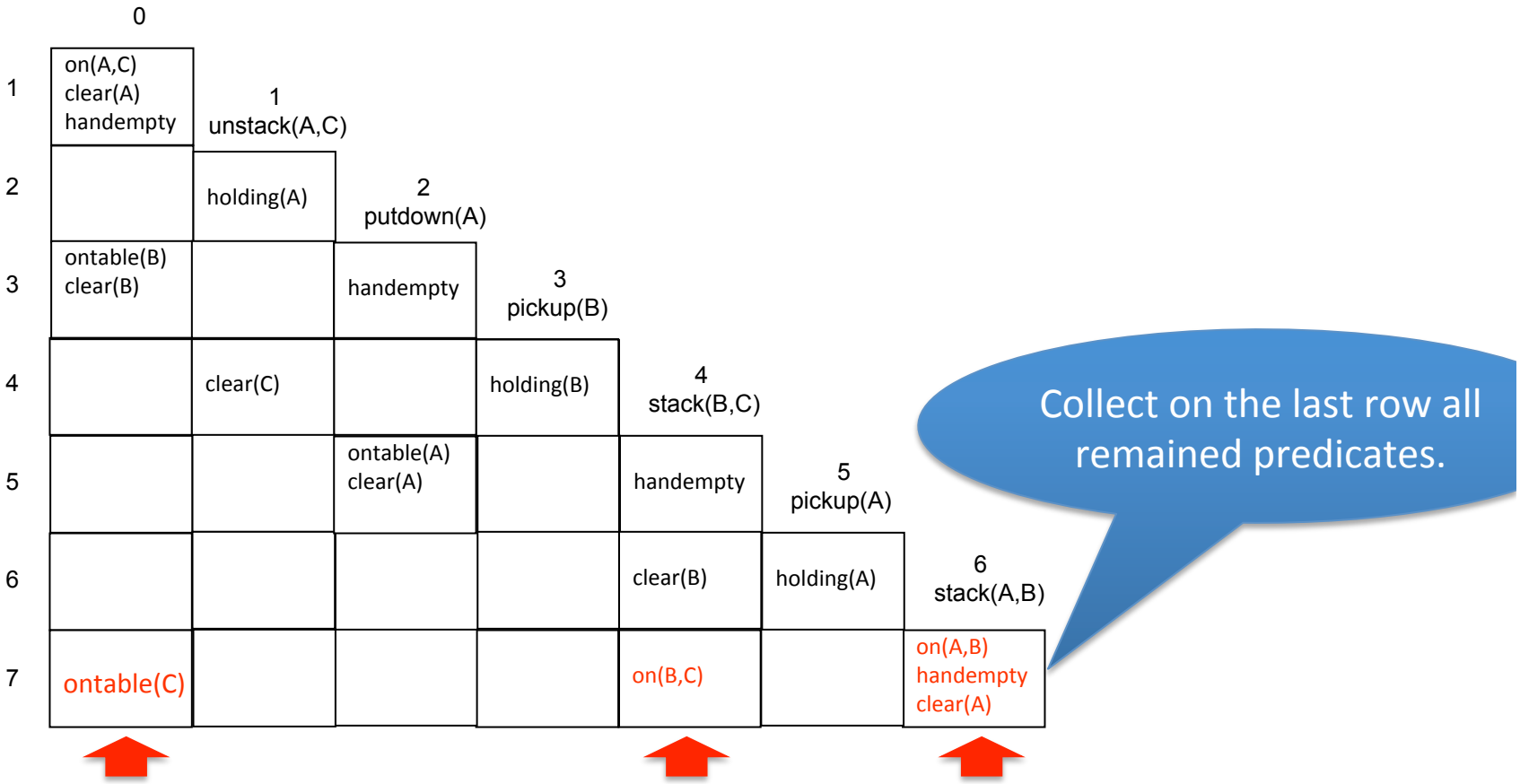
A: on(A,B), handempty, clear(A)

Collect down all list A predicates of rule 6.

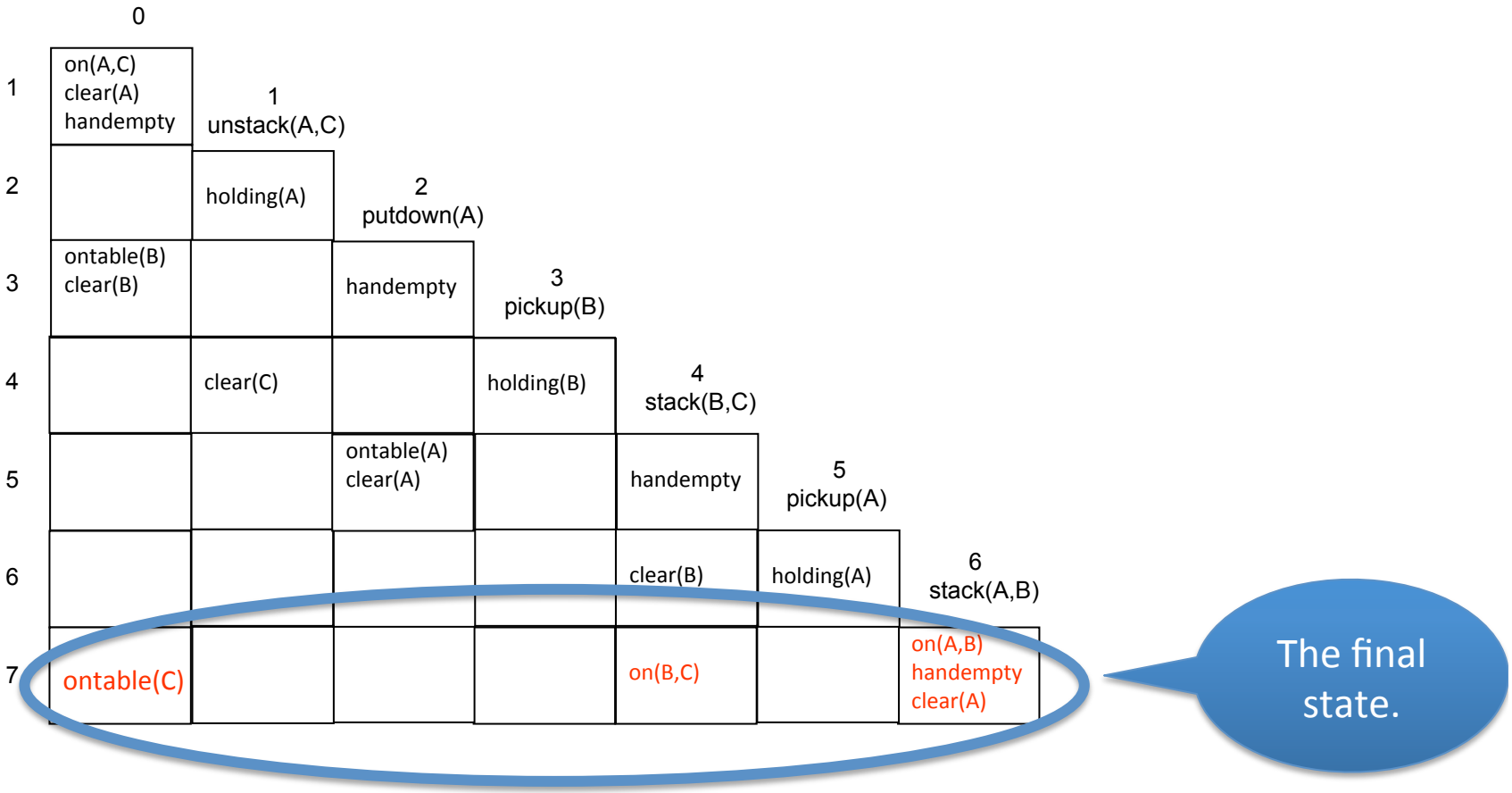
Plan execution: catching up



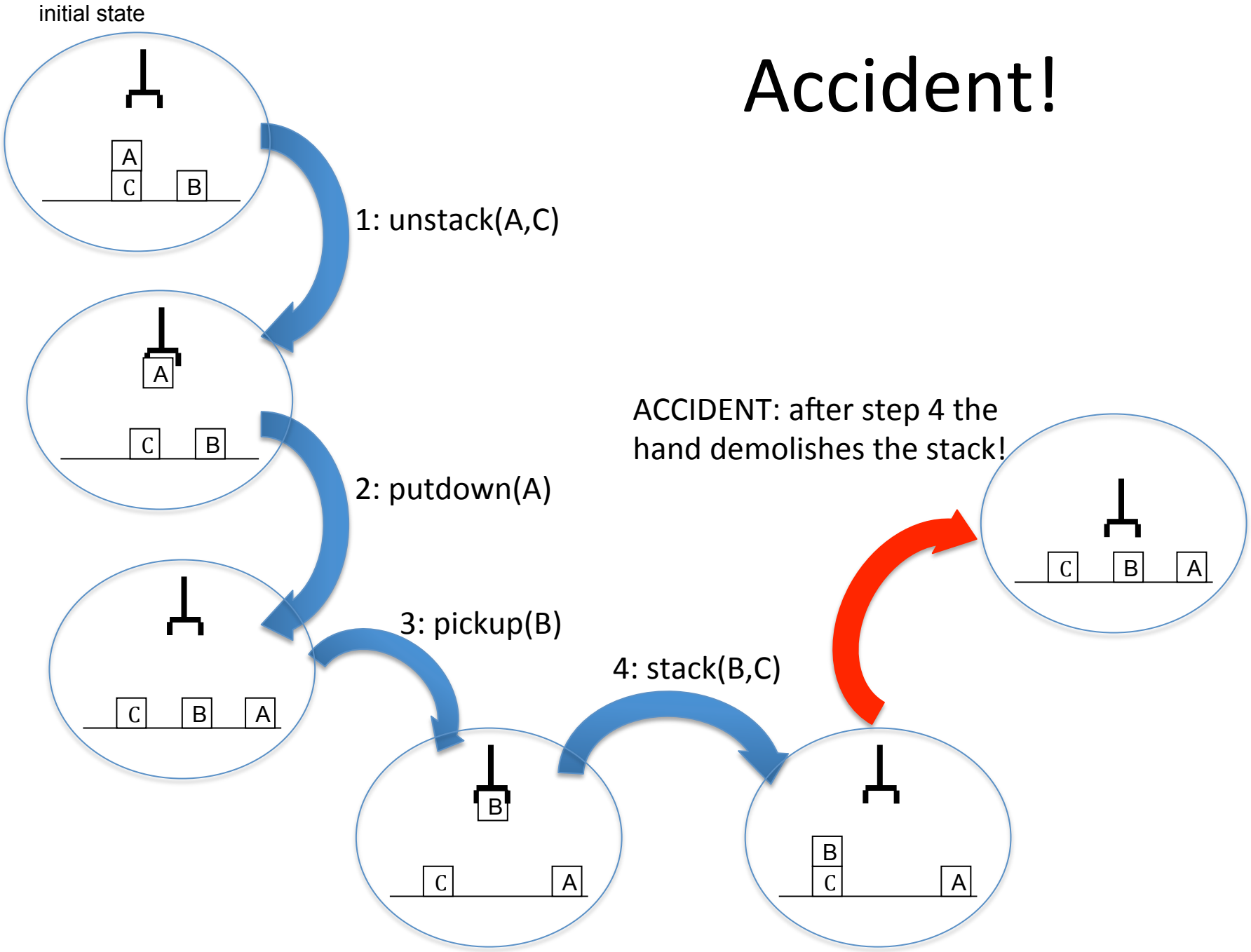
Plan execution: catching up

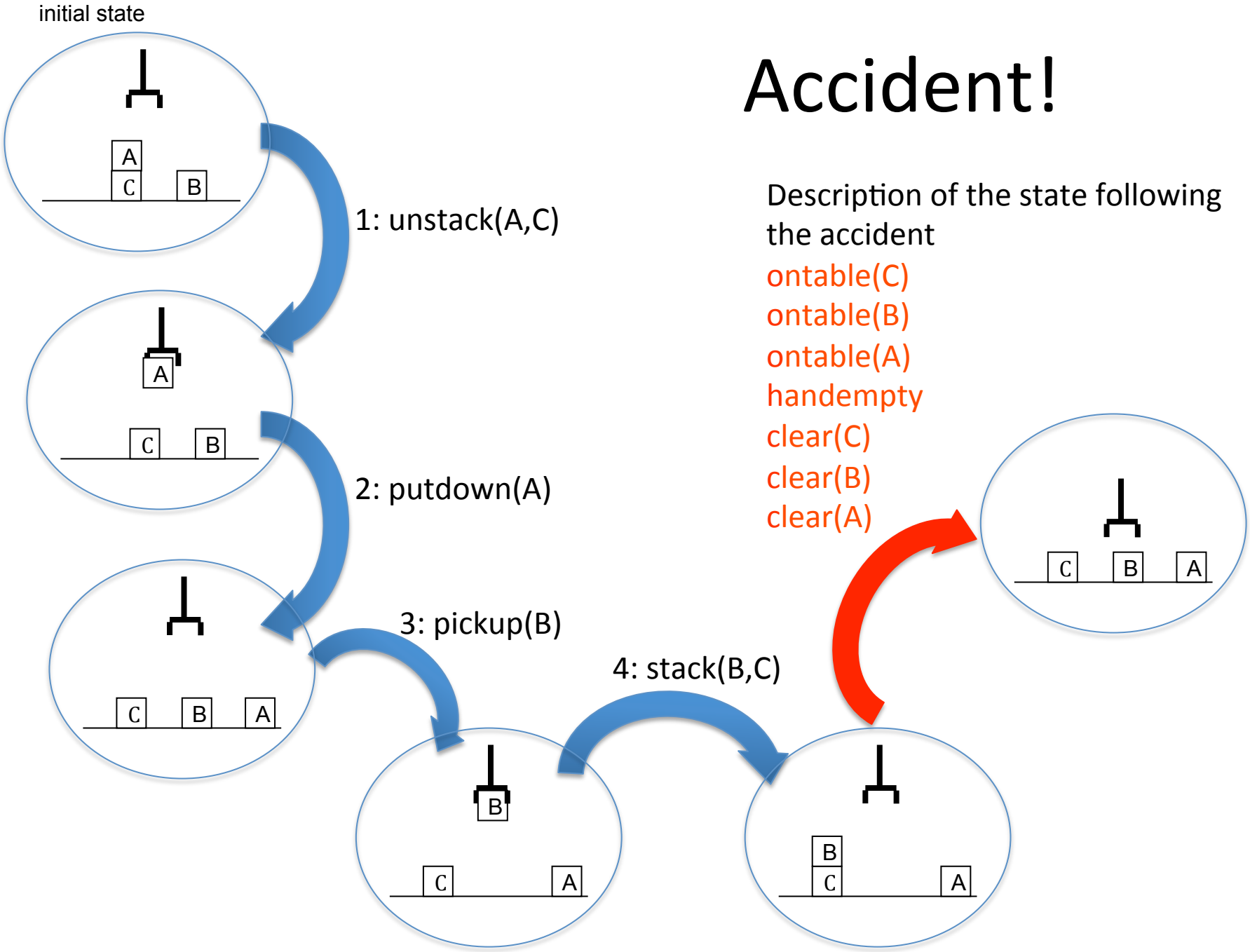


Plan execution: catching up



Accident!

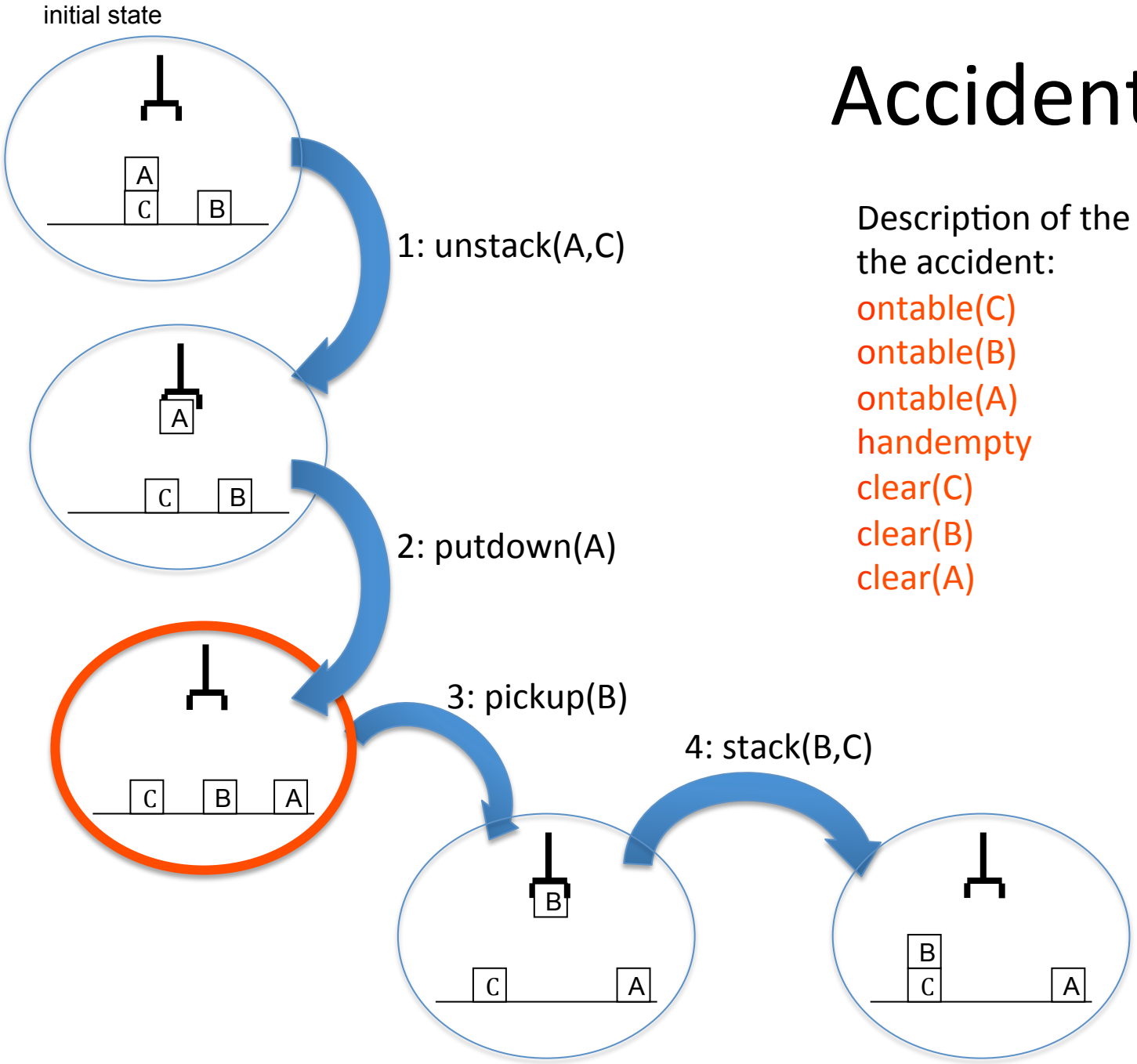

initial state
A
C
B
1: unstack(A,C)
A
C
B
2: putdown(A)
C
B
A
3: pickup(B)
B
C
A
4: stack(B,C)
B
C
A
ACCIDENT: after step 4 the hand demolishes the stack!
C
B
A



Accident!

Description of the state following the accident

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

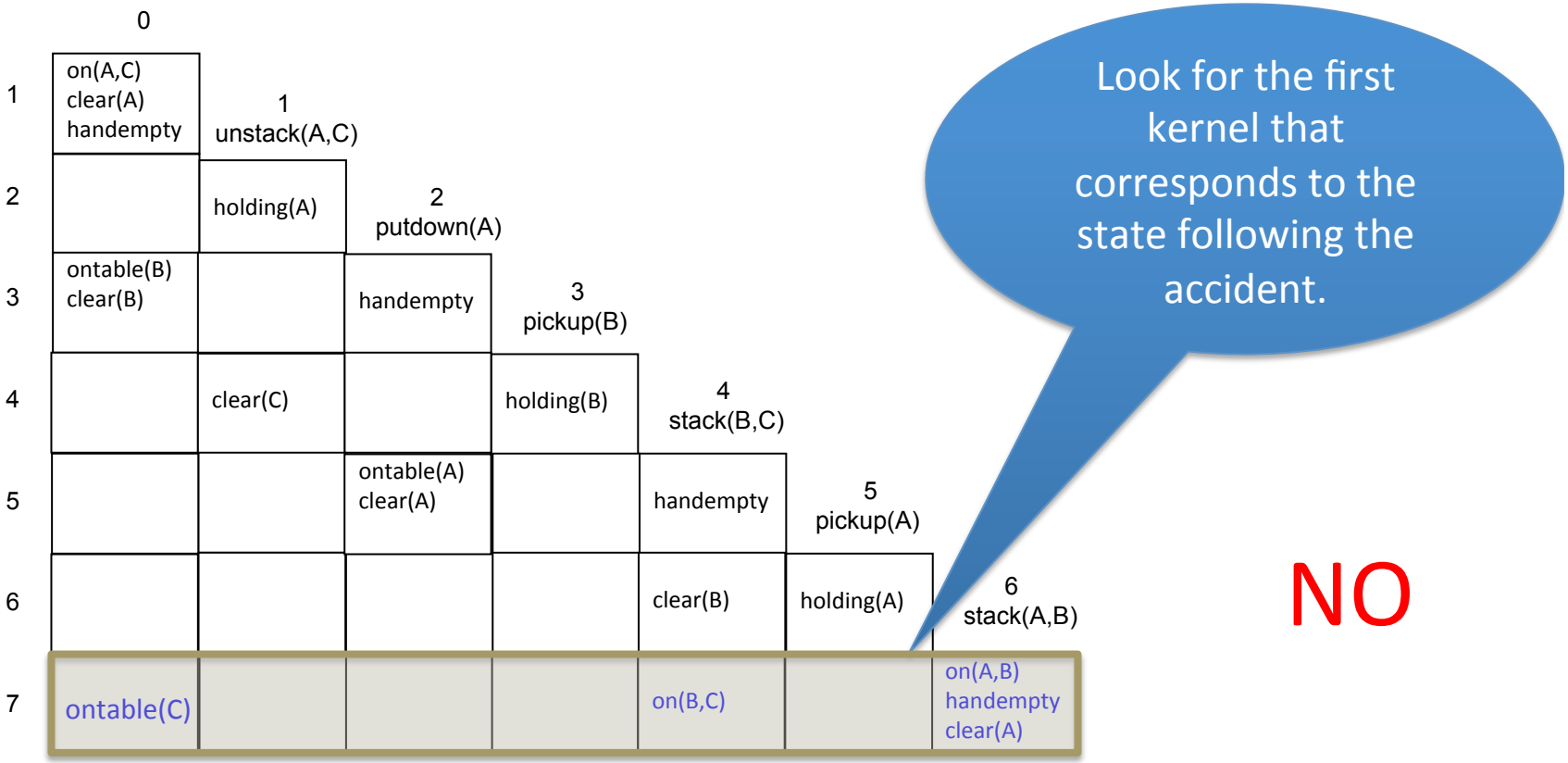


Accident!

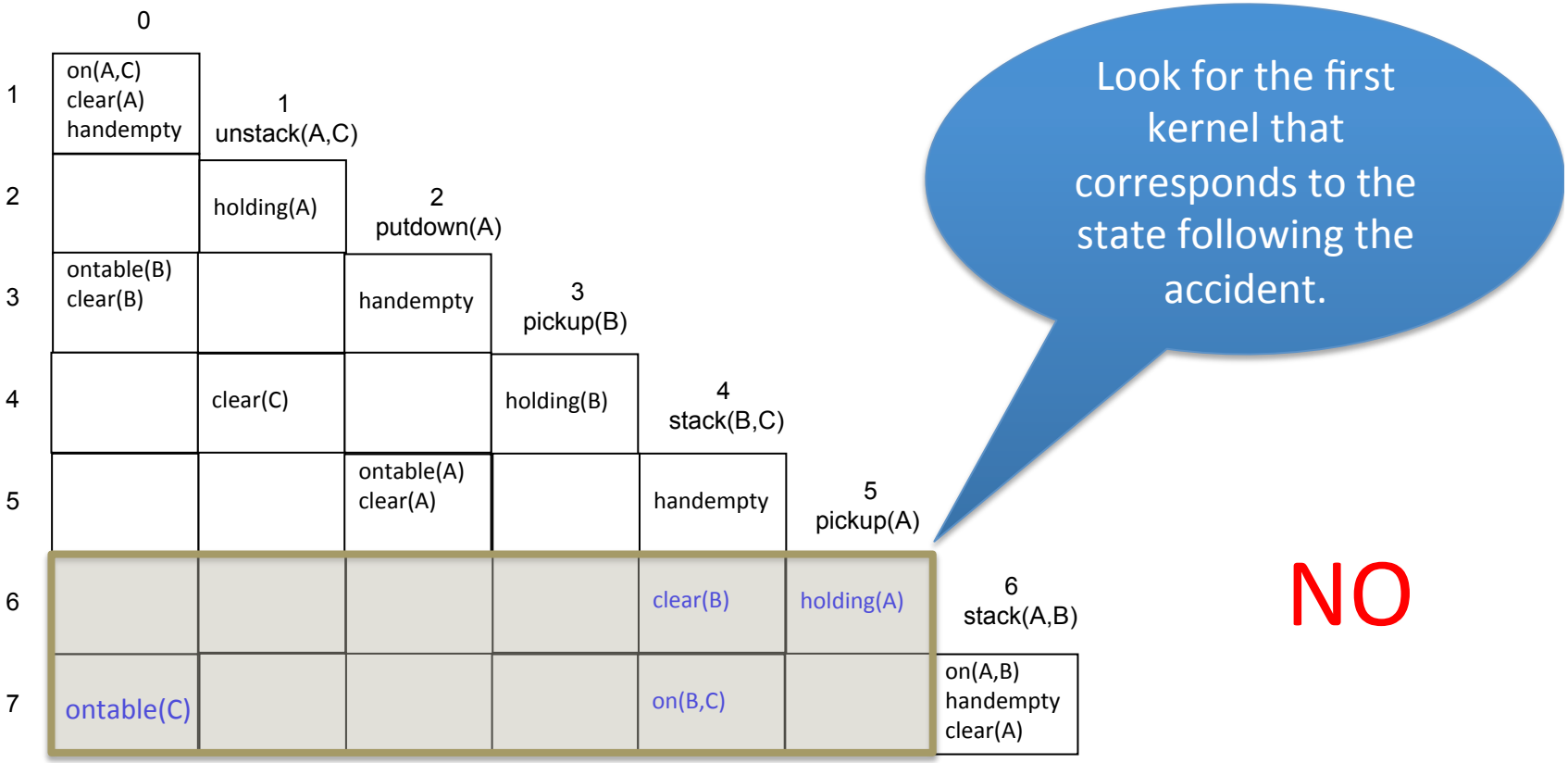
Description of the state following the accident:

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

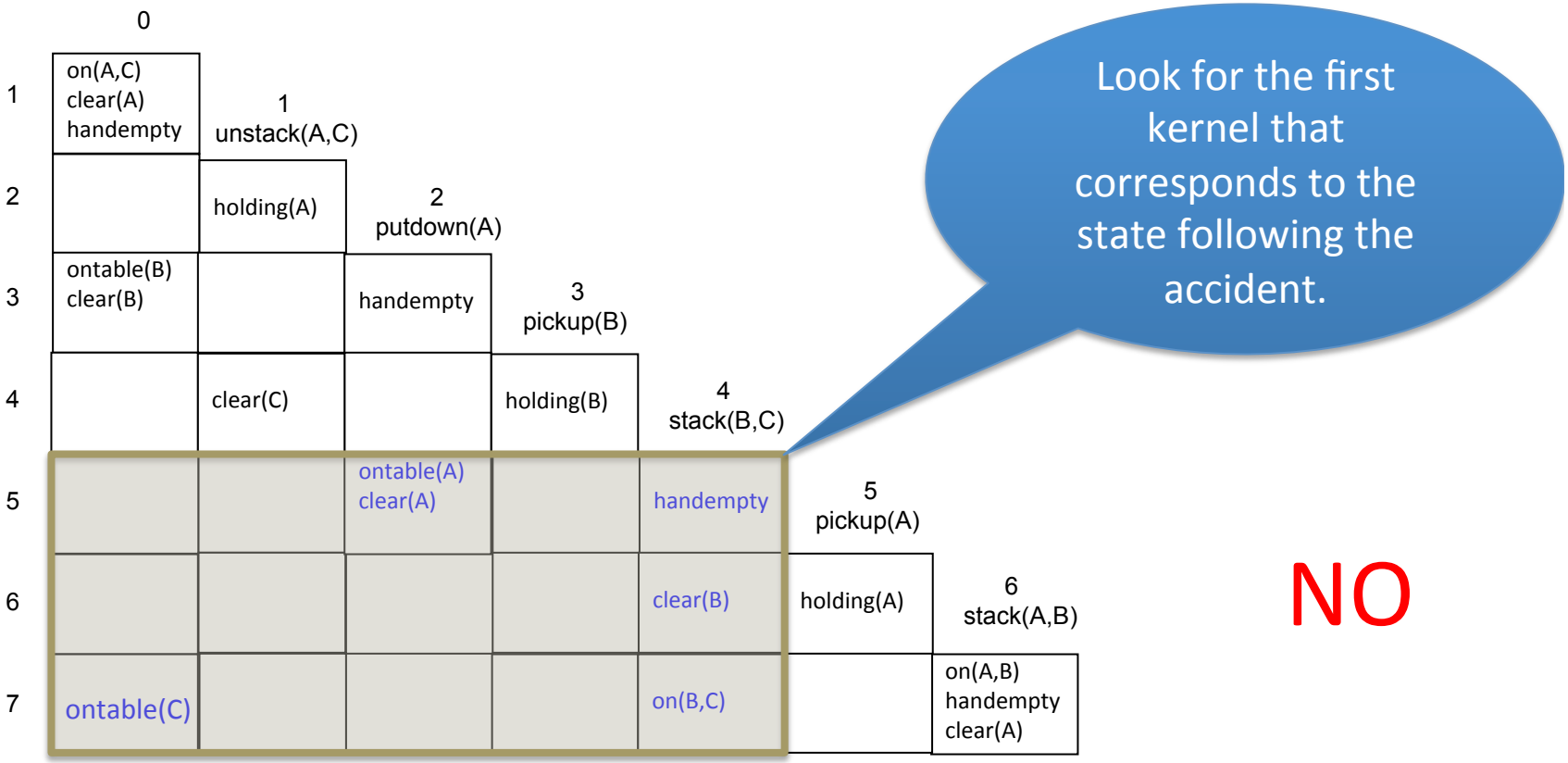
Kernel of order 7



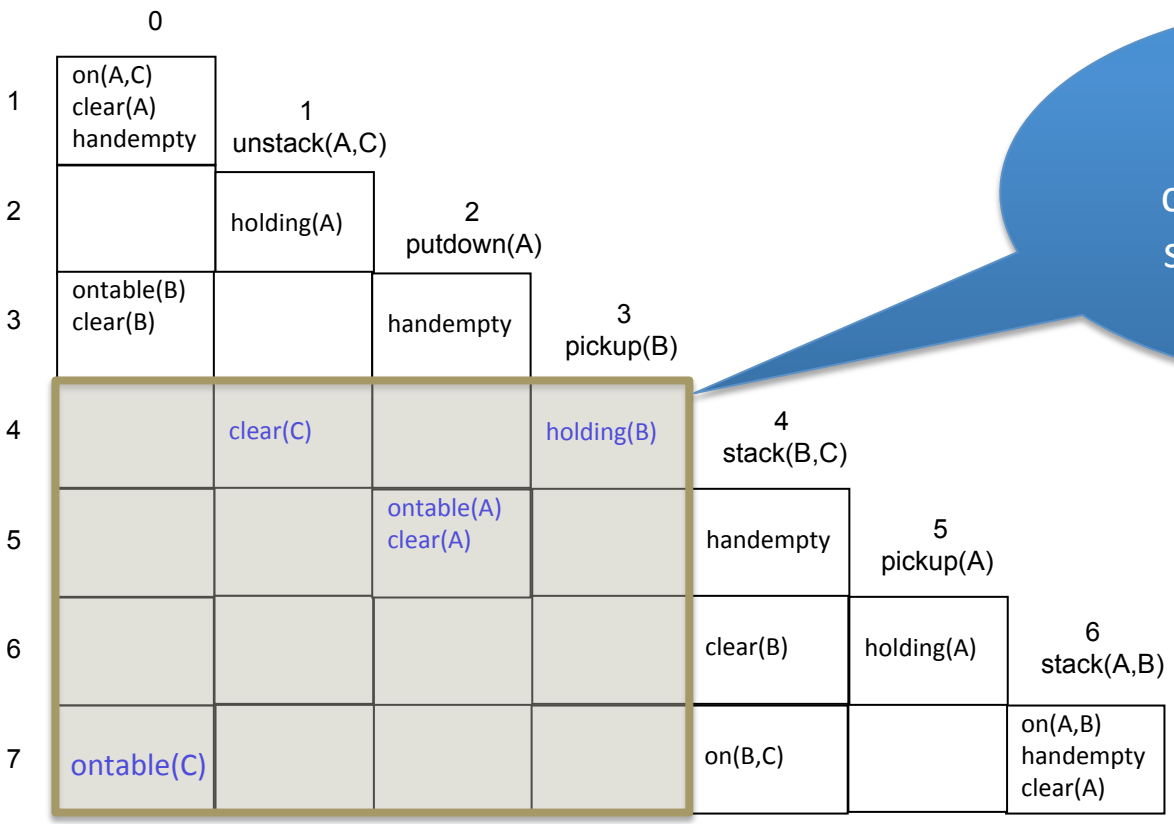
Kernel of order 6



Kernel or order 5



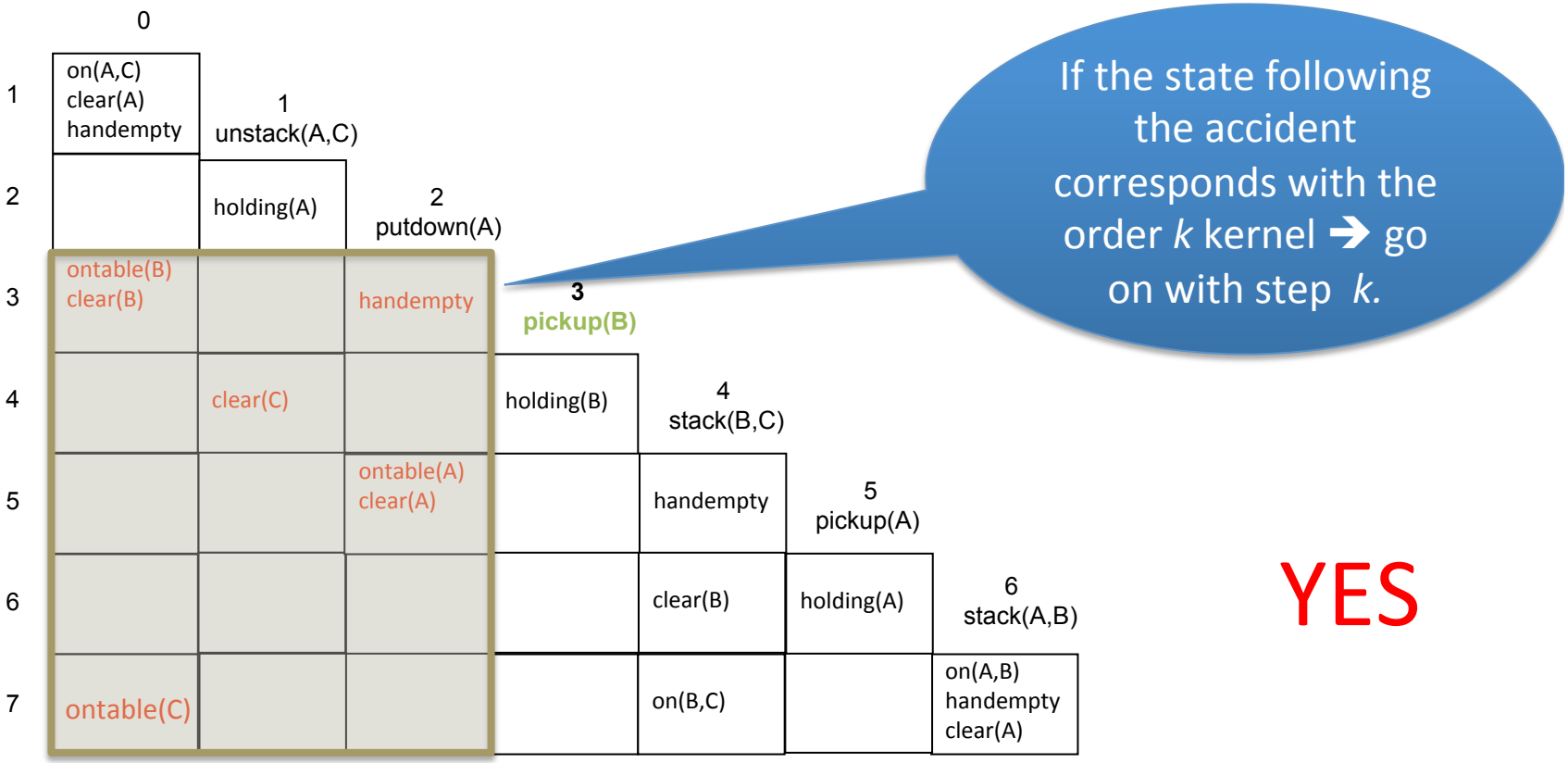
Kernel of order 4

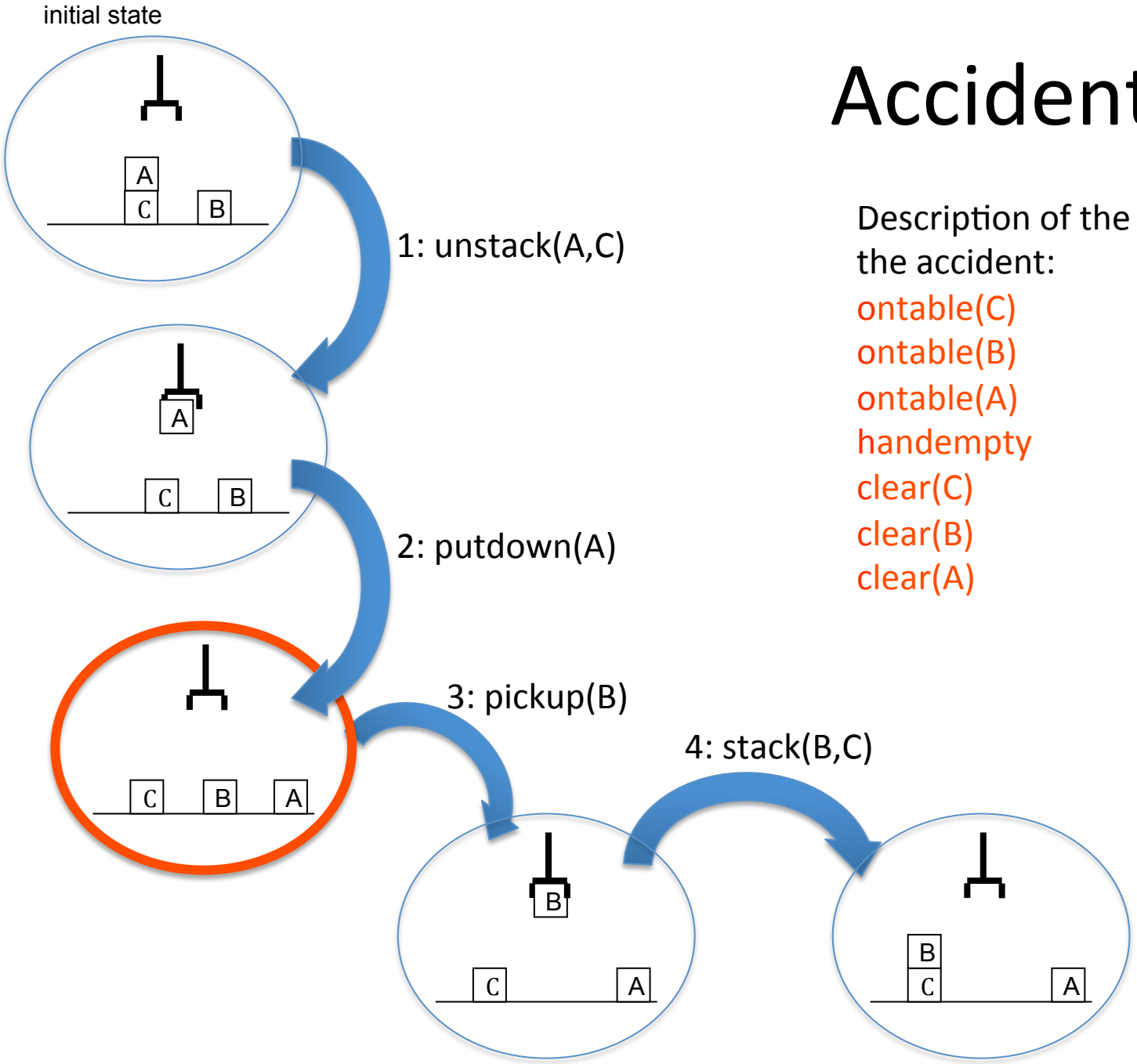


Look for the first kernel that corresponds to the state following the accident.

NO

Kernel of order 3





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!