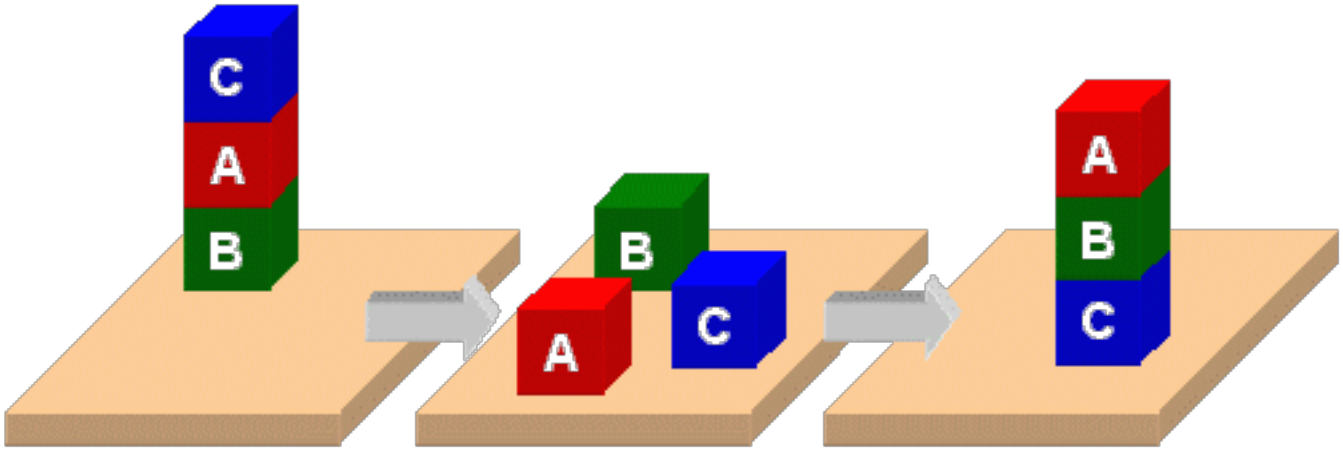
Classic Blocks World



Classic Blocks World

- We'll look at the classic blocks world domain
- Starting with
  - BW: a domain file
  - Several problem files
- We'll use planning.domains to demonstrate solving the problems
- And then show simple extensions to the domain by adding predicates and constants

bw.pddl 1

```
(define (domain bw)

 (:requirements :strips)

 (:predicates
    (on ?x ?y)        ; object ?x is on ?object ?y
    (on-table ?x)     ; ?x is directly on the table
    (clear ?x)        ; ?x has nothing on it
    (arm-empty)       ; robot isn't holding anything
    (holding ?x))     ; robot is holding ?x

 ;; 4 actions to manipulate objects: pickup, putdown, stack, unstack
 … actions in next four slides …
```

Allows basic add and delete effects in actions

List all the predicates with their arguments

Variables begin with a ?

bw.pddl 2

```
(:action pick-up
    :parameters (?ob)

    :precondition
        (and (clear ?ob)
             (on-table ?ob)
             (arm-empty))

    :effect
        (and (not (on-table ?ob))
             (not (clear ?ob))
             (not (arm-empty))
             (holding ?ob)))
```

Variable for the argument of a pick-up action

These three statements must be True before we can do a pick-up action

After doing a pick-up action, these become True

Adding (not ?X) removes ?X if it's in the KB and vice versa

bw.pddl 3

```
(:action put-down
   :parameters (?ob)
   :precondition (holding ?ob)
   :effect
      (and (not (holding ?ob))
           (clear ?ob)
           (arm-empty)
           (on-table ?ob)))
```

put-down means put the thing you're holding on the table

```
(:action stack
   :parameters (?ob1 ?ob2)
   :precondition (and (holding ?ob) (clear ?ob2))
   :effect
     (and (not (holding ?ob))
          (not (clear ?ob2))
          (clear ?ob)
          (arm-empty)
          (on ?ob ?ob2)))
```

stack means put the thing you are holding on another object

bw.pddl 5

```
(:action unstack
   :parameters (?ob1 ?ob2)
   :precondition
      (and (on ?ob1 ?ob2)
           (clear ?ob1)
           (arm-empty))
   :effect
      (and (holding ?ob1)
           (clear ?ob2)
           (not (clear ?ob1))
           (not (arm-empty))
           (not (on ?ob1 ?ob2)))

) ; this closes the domain definition
```

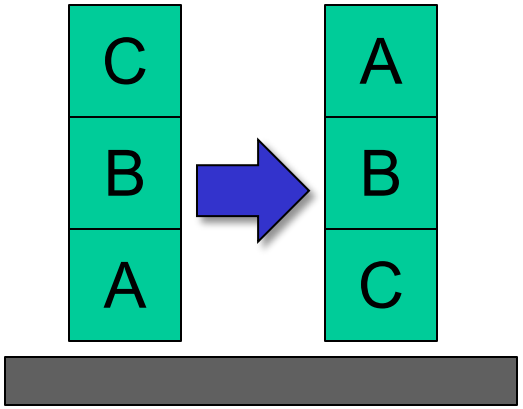
unstack means take the first arg off the second arg

First arg can't have anything on it & the robot can't be holding anything

Updates to KB describing new state of the world

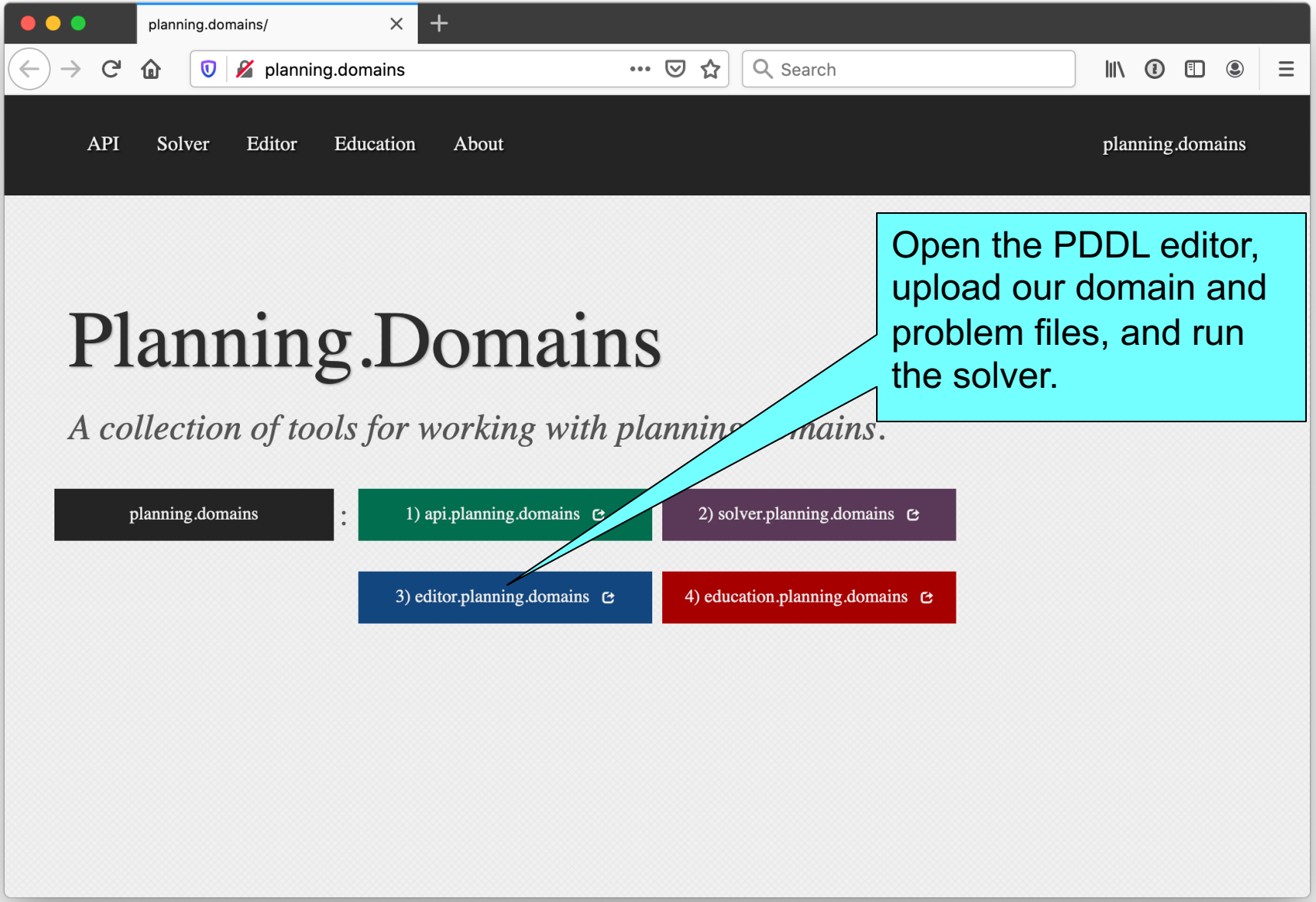
```
;; The arm is empty and there is a stack of three blocks: C is on B which is on A
;;  which is on the table.  The goal is to reverse the stack, i.e., have A on B and B
;;  on C.  No need to mention C is on the table, since domain constraints will enforce it.

(define (problem p03)
  (:domain bw)
  (:objects A B C)
  (:init (arm-empty)
         (on-table A)
         (on B A)
         (on C B)
         (clear C))
  (:goal (and (on A B)
              (on B C))))
```



p03.pddl

http://planning.domains/



Online Demonstration

Using planning.domains and files in the planning directory of our 2020 671 code repo

- Blocks world
  - bw.pddl
  - p01.pddl
  - p02.pddl …
- Air Cargo
  - ac_domain.pddl
  - Ac_p0.pddl

Fin