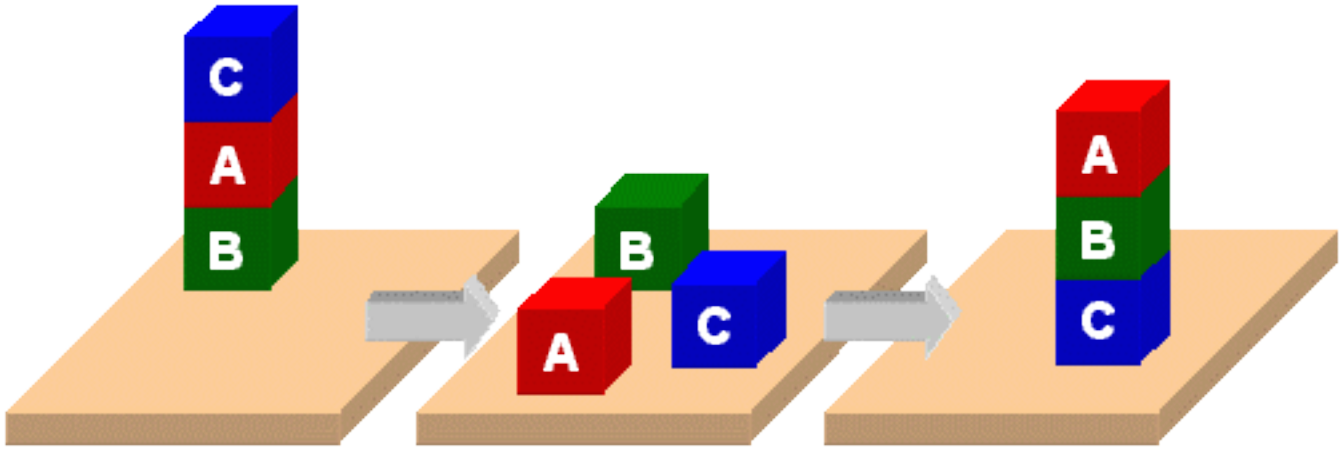
Classic Blocks World



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

 ;; the four classic actions for manipulating objects
```

… actions in next four slides …

Allows basic add and delete effects in actions

List all the predicates with their arguments

bw.pddl 2

```
(:action pick-up
   :parameters (?ob1)

   :precondition
      (and (clear ?ob1)
           (on-table ?ob1)
           (arm-empty))

   :effect
      (and (not (on-table ?ob1))
           (not (clear ?ob1))
           (not (arm-empty))
           (holding ?ob1)))
```

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

bw.pddl 3

```
(:action pick-up
  :parameters (?ob1)

  :precondition
    (and (clear ?ob1)
         (on-table ?ob1)
         (arm-empty))

  :effect
    (and (not (on-table ?ob1))
         (not (clear ?ob1))
         (not (arm-empty))
         (holding ?ob1)))
```

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

bw.pddl 4

```
(: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 think you are holding on the table

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

stack means put the thing you are holding on another object

bw.pddl 5

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

) ; this closes the domain definition
```

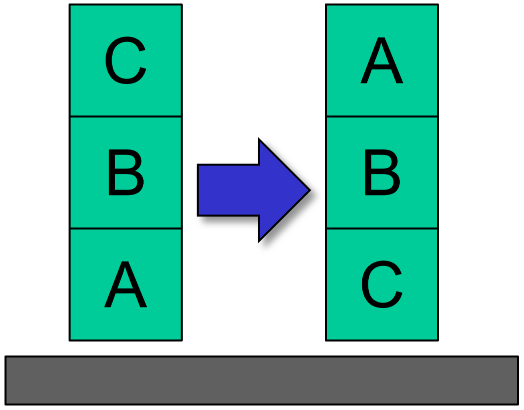
unstack means take the first arg off the second arg

First arg can’t have anything on it and the robt cannot be holding anything

Here are the updates to our knowledge base describing the state of the world

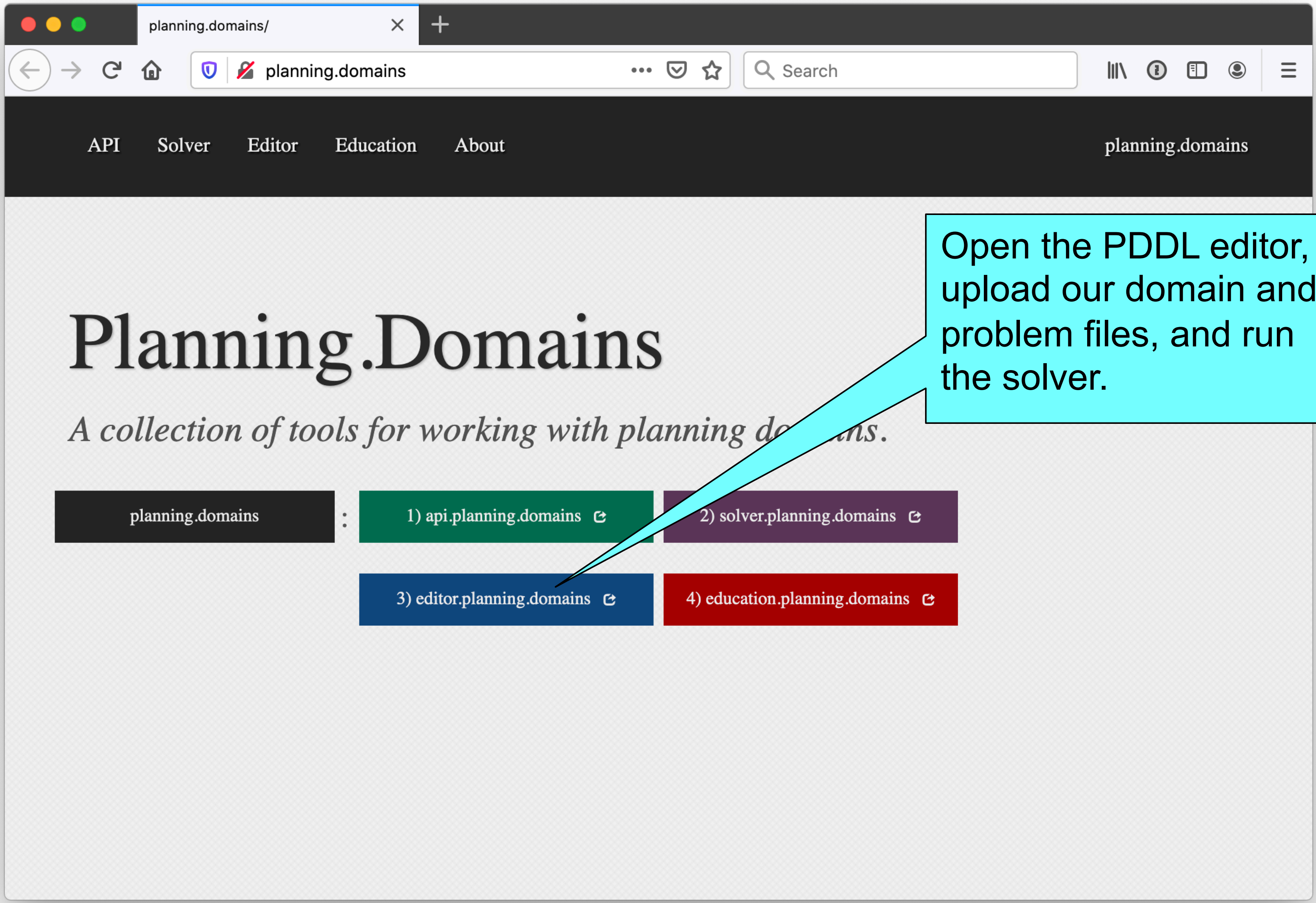
```
;; The arm is empty and there is a stack of three blocks: C is on B which is on A
;;  which is on the table.  The goal is to reverse the stack, i.e., have A on B and B
;;  on C.  No need to mention C is on the table, since domain constraints will enforce it.

(define (problem 00)
  (: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))))
```



p00.pddl

http://planning.domains/



Planning.Domains

A collection of tools for working with planning domains.

Open the PDDL editor, upload our domain and problem files, and run the solver.

Online Demonstration

We'll try an online demonstration, using planning.domains and the files in the planning subdirectory of our 471 code repository

- bw.pddl
- p01.pddl
- p02.pddl
- p03.pddl
- p12.pddl
- p36.pddl

Fín