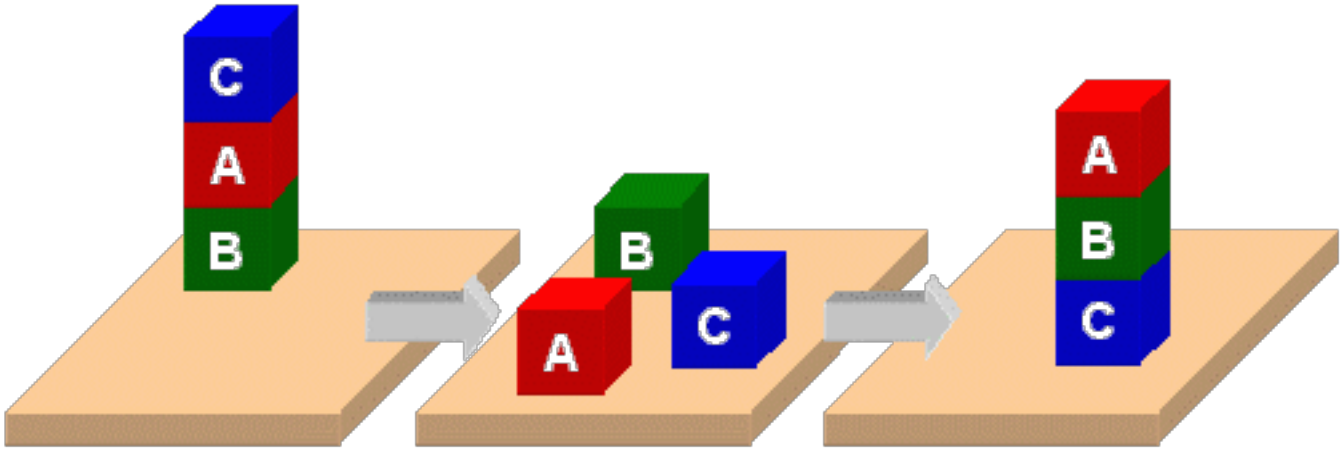
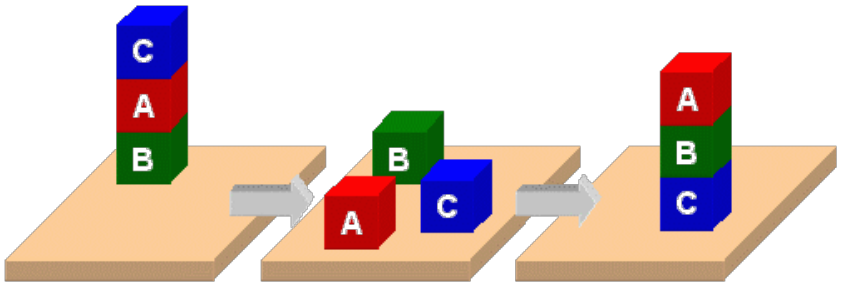
# PDDL and the Blocks World



# Knowledge for Planning

- We'll describe PDDL, a standard for representing planning problems
- We'll look at the classic blocks world in PDDL via:
  - 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

# PDDL




- **Planning Domain Description Language**
- Based on STRIPS with various extensions
- First defined by Drew McDermott (Yale) et al.
  - Classic spec: PDDL 1.2; good reference guide
- Used in biennial International Planning Competition (IPC) series (1998-2022)
- Many planners use it as a standard input
- Latest version is 3.1 and newer variants exist

# PDDL is still widely used

- After 24 years, PDDL still used in many planning systems and domains as a standard for input and output
- Its representation was updated, e.g., adding
  - fluents (facts that change over time)
  - preferences (aka soft constraints)
- New variants support multiple agents, ontologies, and more
- It still retains is traditional Lisp syntax

# PDDL Representation

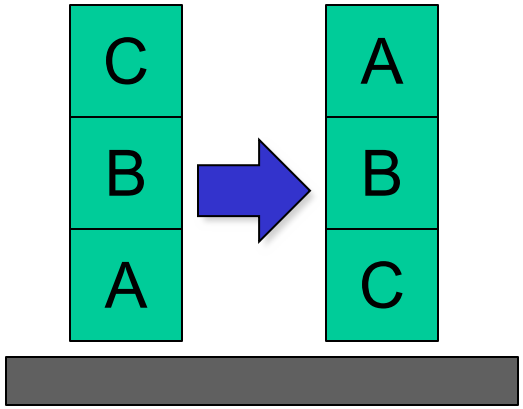
- Task specified via two files: **domain file** and **problem file**
  - Both use a logic-oriented notation with Lisp syntax
- **Domain file** defines a domain via *requirements*, *predicates*, *constants*, and *actions*
  - Used for many different problem files
- **Problem file:** defines problem by describing its *domain*, specific *objects*, *initial state,* and *goal state*
- **Planner:** domain + problem → a plan

# Blocks Word Domain File


```
(define (domain BW)
  (:requirements :strips)
  (:constants red green blue yellow small large)
  (:predicates (on ?x ?y) (on-table ?x) (color  ?x ?y) … (clear ?x))
  (:action pick-up
     :parameters (?obj1)
     :precondition (and (clear ?obj1) (on-table ?obj1)
                                (arm-empty))
     :effect (and (not (on-table ?obj1))
                  (not (clear ?obj1))
                  (not (arm-empty))
                  (holding ?obj1)))
  … more actions ...)
```

# Blocks Word Problem File


```
(define (problem 00)
    (:domain BW)
    (:objects A B C)
    (:init (arm-empty)
            (ontable A)
           (on B A)
           (on C B)
           (clear C))
    (:goal (and (on A B)
                (on B C)
                (ontable C)))
```

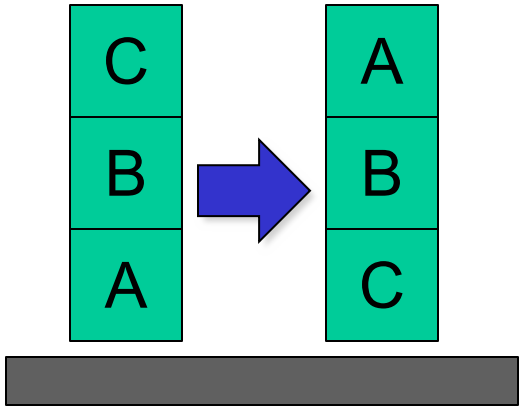


# What's a Plan?

- For simple planning problems…
- A planner takes a problem that identifies the problem domain (e.g. BW)
- And produces an ordered set of actions with references to objects in the problem
- Which when executed in order achieves the goal

# Planner: Domain + Problem => Plan


```
(define (problem 00)
   (:domain BW)
   (:objects A B C)
   (:init (arm-empty)
          (on B A)
          (on C B)
          (clear C))
   (:goal (and (on A B)
               (on B C))))
```




domain + problem ➔ planner ➔

Begin plan
1 (unstack c b)
2 (put-down c)
3 (unstack b a)
4 (stack b c)
5 (pick-up a)
6 (stack a b)
End plan

# 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

% starts a one-line comment

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.

# 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
```

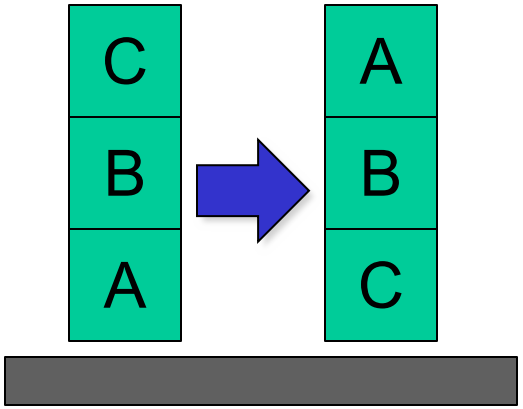
unstack means take the first arg off the second arg

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

Updates to KB describing new state of the world

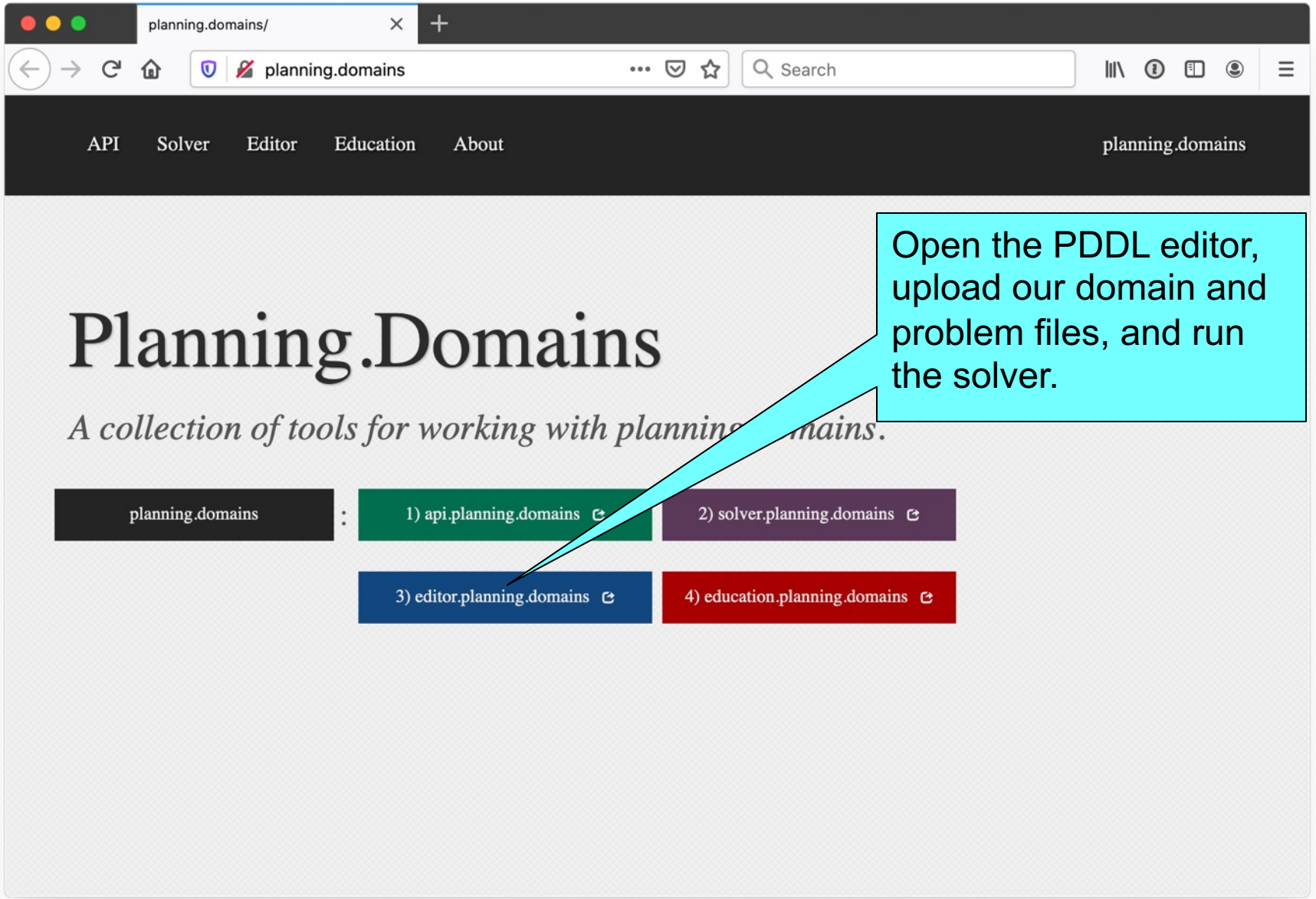
```
;; The arm is empty and there is a stack of three blocks: C is on B which is on A
;;  which is on the table.  The goal is to reverse the stack, i.e., have A on B and B
;;  on C.  No need to mention C is on the table, since domain constraints will enforce it.

(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/



# Planning.domains

- Open source environment for providing planning services using PDDL (GitHub)
- Default planner is ff (aka, fastForward)
  - very successful forward-chaining heuristic search planner producing sequential plans
  - Can be configured to work with other planners
- Use interactively or call via web-based API
- We've used it for to extend blocks world domain in homework

# Online Demonstration

Using planning.domains and files in the planning directory of our 2022 code and data repo

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

*Fin*