Problem Domain Description Language

willie

(some slides adapted from Stavros Vassos, University of Athens)

Announcements

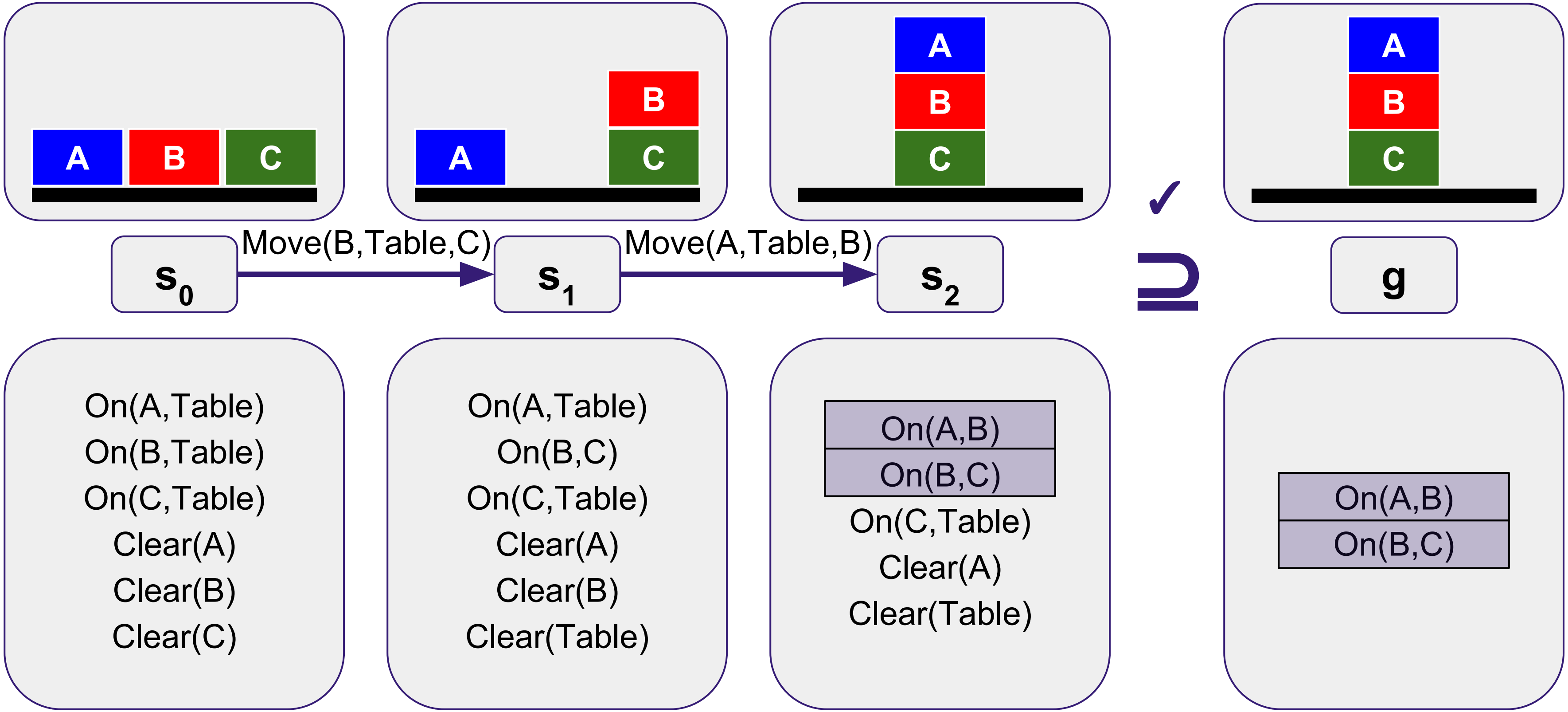
Assignment 4

- Issues with Windows

Practice midterm

This week: Chapter 10

Blocks World



$$s_0 \xrightarrow{\text{Move(B,Table,C)}} s_1 \xrightarrow{\text{Move(A,Table,B)}} s_2 \overset{\checkmark}{\supseteq} g$$

s_0:

On(A,Table)
On(B,Table)
On(C,Table)
Clear(A)
Clear(B)
Clear(C)

s_1:

On(A,Table)
On(B,C)
On(C,Table)
Clear(A)
Clear(B)
Clear(Table)

s_2:

On(A,B)
On(B,C)
On(C,Table)
Clear(A)
Clear(Table)

g:

On(A,B)
On(B,C)

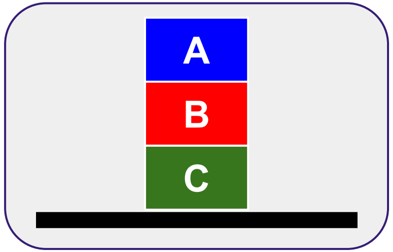
STRIPS planning

Init(On(A,Table) ∧ On(B,Table) ∧ On(C,Table) ∧ Clear(A) ∧
Clear(B) ∧ Clear(C))

Goal(On(A,B) ∧ On(B,C))

Action(Move(b,x,y),
PRECONDITIONS: On(b,x) ∧ Clear(b) ∧ Clear(y)
EFFECTS: On(b,y) ∧ Clear(x) ∧ ¬On(b,x) ∧ ¬Clear(y))

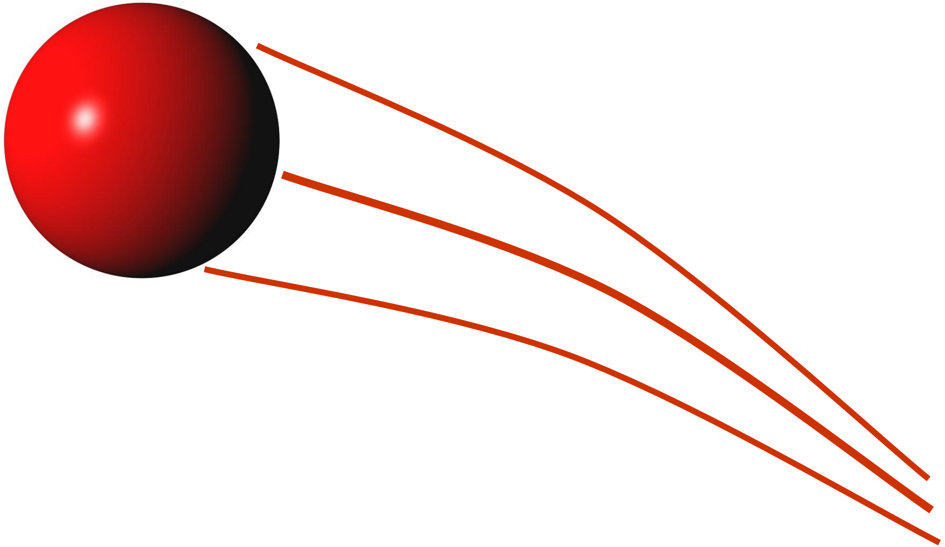
Action(MoveToTable(b,x),
PRECONDITIONS: On(b,x) ∧ Clear(b)
EFFECTS: On(b,Table) ∧ Clear(x) ∧ ¬On(b,x))



s_2

On(A,B)
On(B,C)
On(C,Table)
Clear(A)
Clear(Table)

Pass the ball - STRIPS



Init(HasBall(Ethan) ∧
    LastLetter(Ethan, n))

Goal(HasBall(Willie) ∧
    FirstLetter(Willie,w))

Action(PassBall(x,l,y,f)
    PRECONDITIONS: LastLetter(x,l) ∧ FirstLetter(y,f) ∧
        Same(l,f) ∧ HasBall(x)
    EFFECTS: HasBall(y) ∧ ¬HasBall(x)

And now…

Problem

Domain

Description

Language

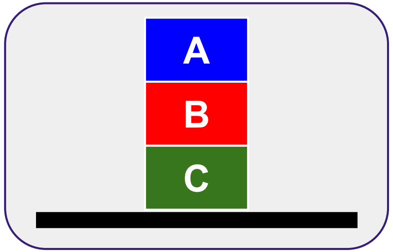
STRIPS planning

Init(On(A,Table) ∧ On(B,Table) ∧ On(C,Table) ∧ Clear(A) ∧
Clear(B) ∧ Clear(C))

Goal(On(A,B) ∧ On(B,C))

Action(Move(b,x,y),
PRECONDITIONS: On(b,x) ∧ Clear(b) ∧ Clear(y)
EFFECTS: On(b,y) ∧ Clear(x) ∧ ¬On(b,x) ∧ ¬Clear(y))

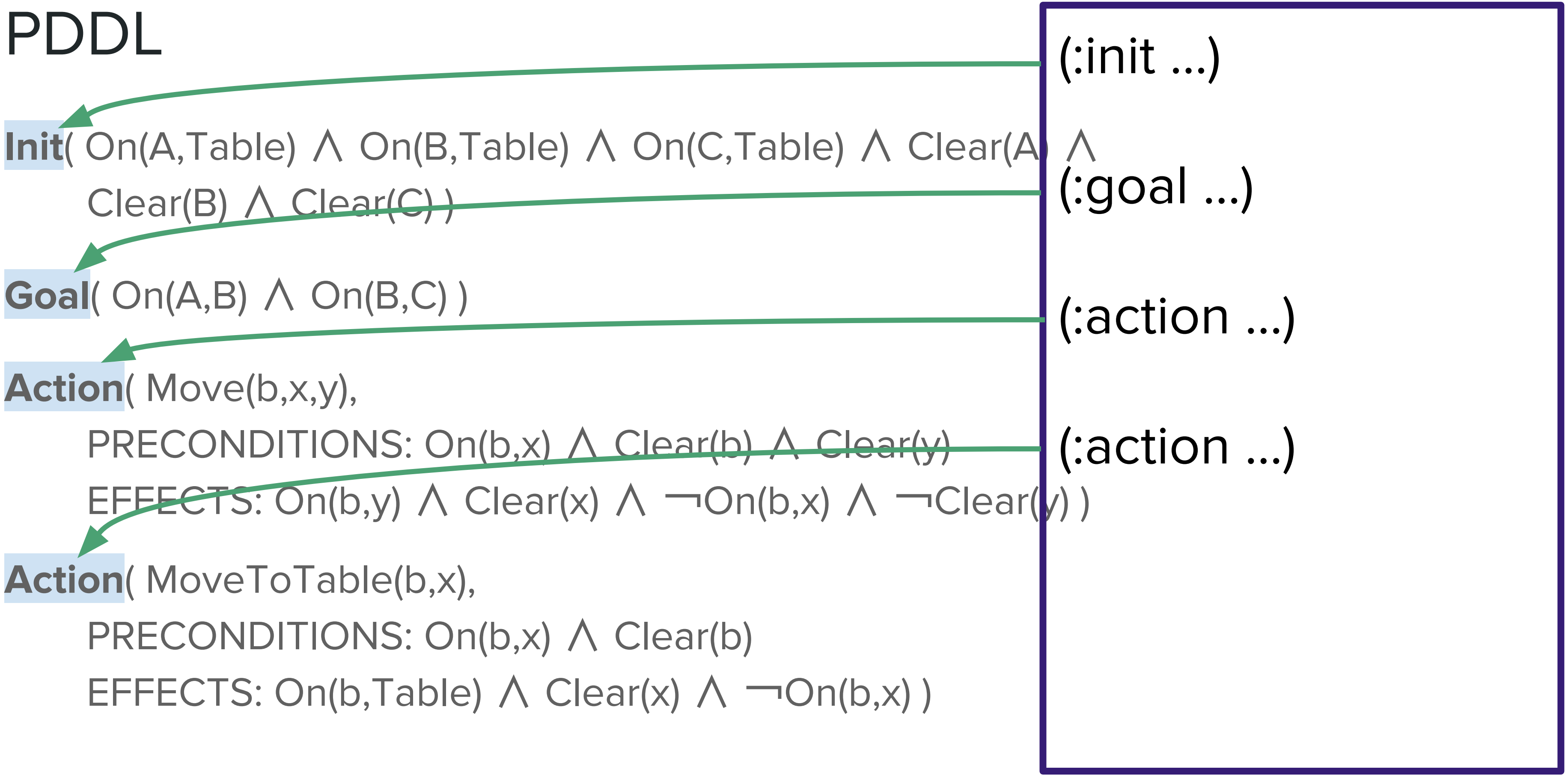
Action(MoveToTable(b,x),
PRECONDITIONS: On(b,x) ∧ Clear(b)
EFFECTS: On(b,Table) ∧ Clear(x) ∧ ¬On(b,x))



s_2

On(A,B)
On(B,C)
On(C,Table)
Clear(A)
Clear(Table)

PDDL



Init(On(A,Table) ∧ On(B,Table) ∧ On(C,Table) ∧ Clear(A) ∧
Clear(B) ∧ Clear(C))

Goal(On(A,B) ∧ On(B,C))

Action(Move(b,x,y),
PRECONDITIONS: On(b,x) ∧ Clear(b) ∧ Clear(y)
EFFECTS: On(b,y) ∧ Clear(x) ∧ ¬On(b,x) ∧ ¬Clear(y))

Action(MoveToTable(b,x),
PRECONDITIONS: On(b,x) ∧ Clear(b)
EFFECTS: On(b,Table) ∧ Clear(x) ∧ ¬On(b,x))

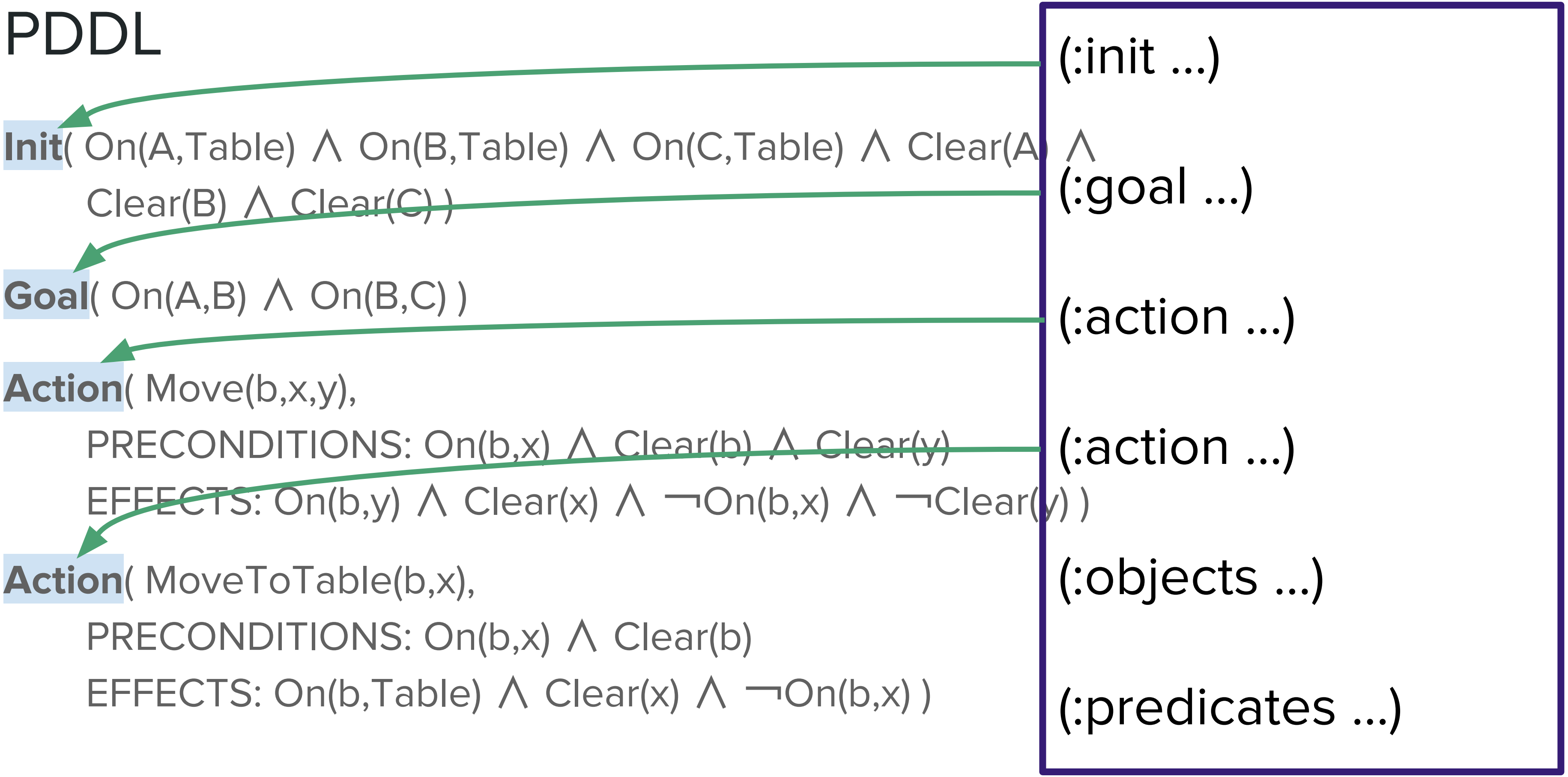
(:init ...)

(:goal ...)

(:action ...)

(:action ...)

PDDL



Init(On(A,Table) ∧ On(B,Table) ∧ On(C,Table) ∧ Clear(A) ∧
Clear(B) ∧ Clear(C))

Goal(On(A,B) ∧ On(B,C))

Action(Move(b,x,y),
PRECONDITIONS: On(b,x) ∧ Clear(b) ∧ Clear(y)
EFFECTS: On(b,y) ∧ Clear(x) ∧ ¬On(b,x) ∧ ¬Clear(y))

Action(MoveToTable(b,x),
PRECONDITIONS: On(b,x) ∧ Clear(b)
EFFECTS: On(b,Table) ∧ Clear(x) ∧ ¬On(b,x))

(:init ...)

(:goal ...)

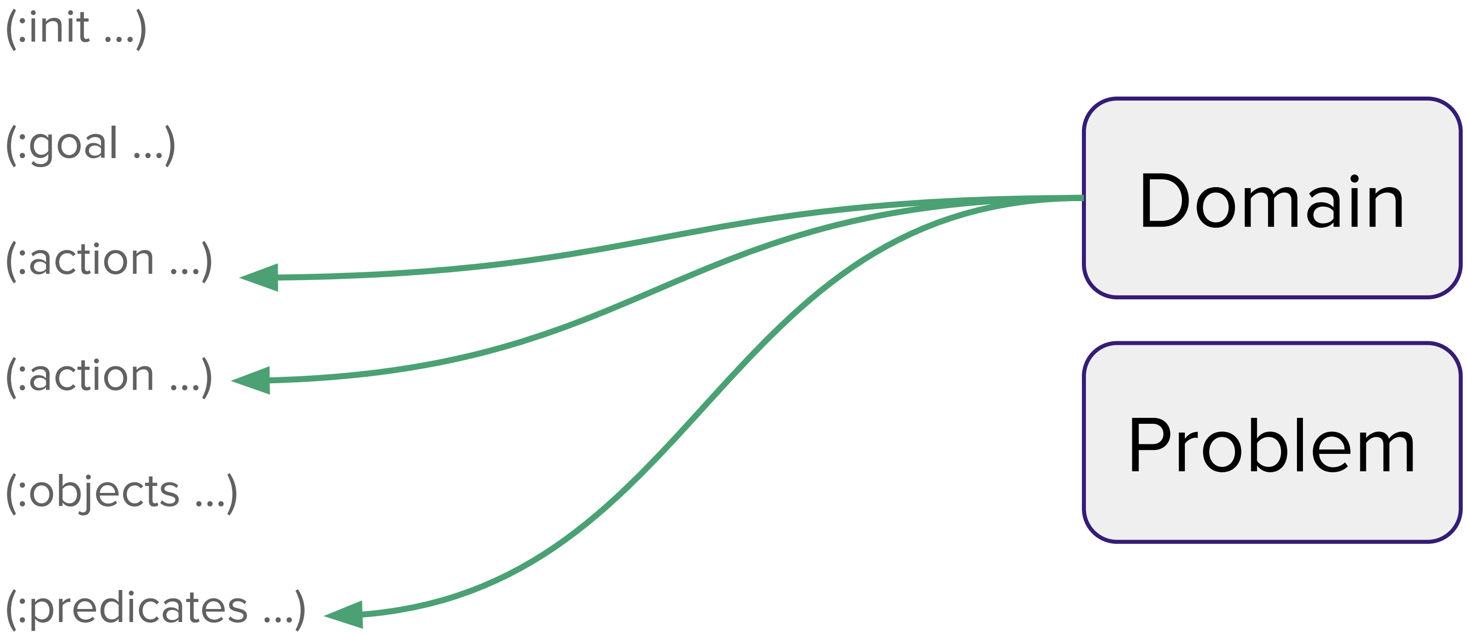
(:action ...)

(:action ...)

(:objects ...)

(:predicates ...)

Problem Domain Description Language



(:init ...)

(:goal ...)

(:action ...)

(:action ...)

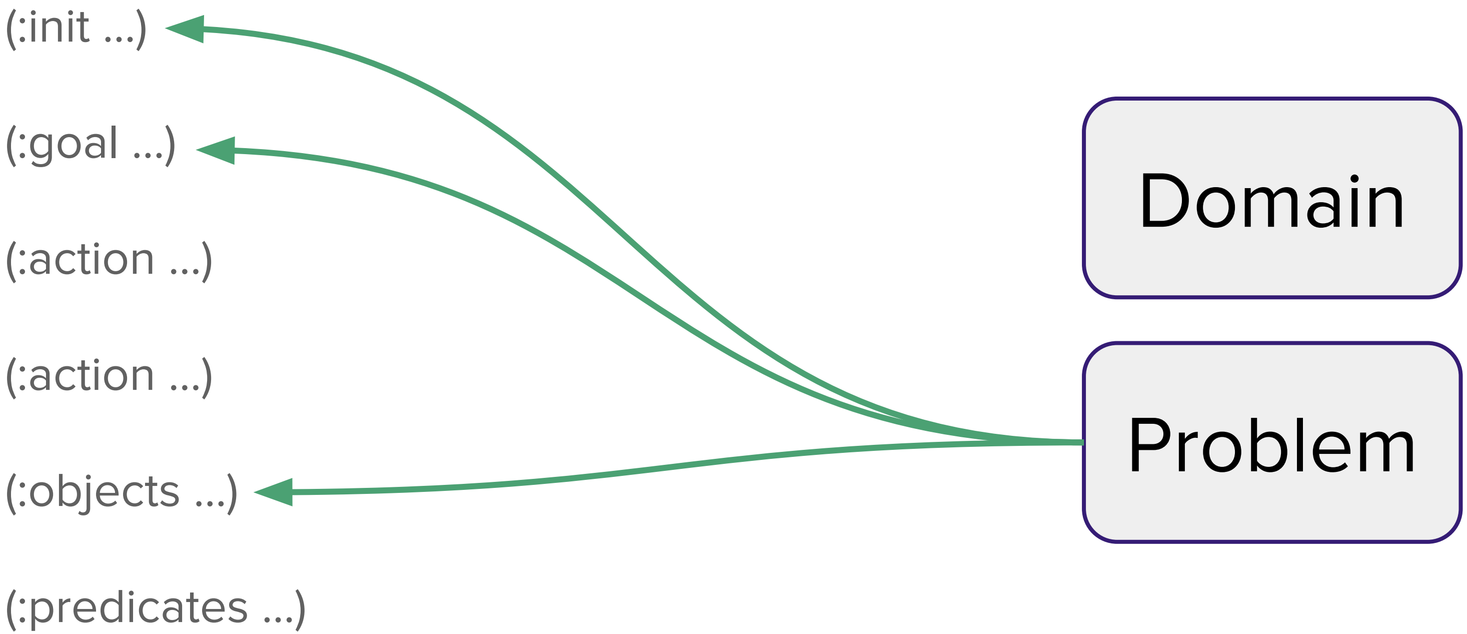
(:objects ...)

(:predicates ...)

Domain

Problem

Problem Domain Description Language



(:init ...)

(:goal ...)

(:action ...)

(:action ...)

(:objects ...)

(:predicates ...)

Domain

Problem

Problem Domain Description Language

(:predicates ...)

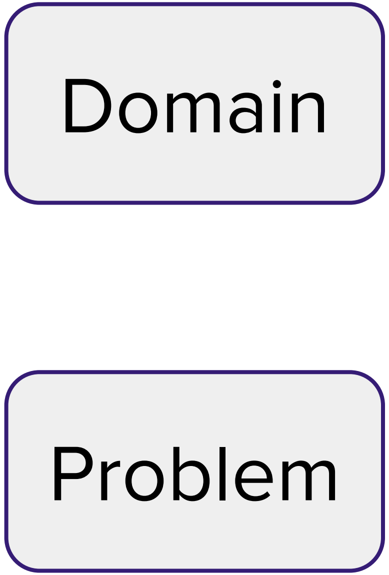
(:action ...)

(:action ...)

(:objects ...)

(:init ...)

(:goal ...)



Domain

Problem

Problem Domain Description Language

On(A,B)	→	(on a b)
¬On(A,B)	→	(not (on a b))
On(A,B) ∧ On(B,C)	→	(and (on a b) (on b c))
On(x,y)	→	(on ?x ?y)

Blocks World in PDDL

Blocks world **domain**

Available predicates	(:predicate ...)
Available actions	(:action ...)

Blocks World in PDDL

Blocks world **domain**

Available predicates

```
(:predicate
        (on ?x ?y)
        (clear ?x)
)
```

Blocks World in PDDL

Blocks world **domain**

Available action

```
(:action move
          :parameters (?b ?x ?y)
          :precondition (and (on ?b ?x)  (clear ?b)
                             (clear ?y))
          :effect ( ... )
)
```

Blocks World in PDDL

Blocks world **domain**

Available action

```
(:action move-to-table
          :parameters (?b ?x)
          :precondition ( ... )
          :effect ( ... )
)
```

Blocks World in PDDL

Blocks world **problem**

Available objects	(:objects …)
Initial state	(:init …)
Goal	(:goal …)

Blocks World in PDDL

Blocks world **problem**

Available objects

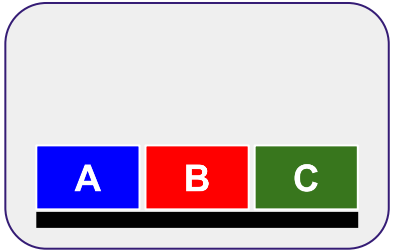
```
(:objects
    a b c table
)
```

Blocks World in PDDL

Blocks world **problem**

Initial state

```
(:init
        (on a table)    (clear a)
        (on b table)    (clear b)
        (on c table)    (clear c)
)
```

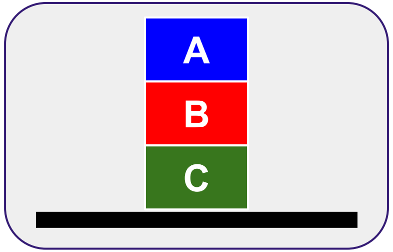


Blocks World in PDDL

Blocks world **problem**

Goal

```
(:goal
        (and (on a b) (on b c) )
)
```



Blocks World in PDDL

blocks-domain.txt

```
(define (domain gripper)
  (:requirements :strips)
  (:predicates (on ?x ?y) (clear ?x))
  (:action move
     :parameters (?b ?x ?y)
     :precondition (and (on ?b ?x)
                         (clear ?b) (clear ?y))
     :effect (and (not (on ?b ?x))
                  (not (clear ?y))
                  (on ?b ?y)
                  (clear ?x)))
...
)
```

blocks-problem1.txt:

```
(define (problem gripper1)
  (:domain gripper)
  (:objects a b c table)

  (:init
         (on a table)  (on b table)  (on c table)
         (clear a)  (clear b)  (clear c)
  )

(:goal (and (on a b) (on b c)))
)
```

Types in PDDL

blocks-domain.txt

```
(define (domain gripper)
  (:requirements :typing)
  (:types block
  (:predicates (on ?x ?y - block) (clear ?x - block))
  (:action move
     :parameters (?b ?x ?y - block)
     :precondition (and (on ?b ?x - block)
                        (clear ?b - block) (clear ?y - block))
     :effect (and (not (on ?b ?x - block))
                  (not (clear ?y - block))
                  (on ?b ?y - block)
                  (clear ?x - block)))
...
)
```

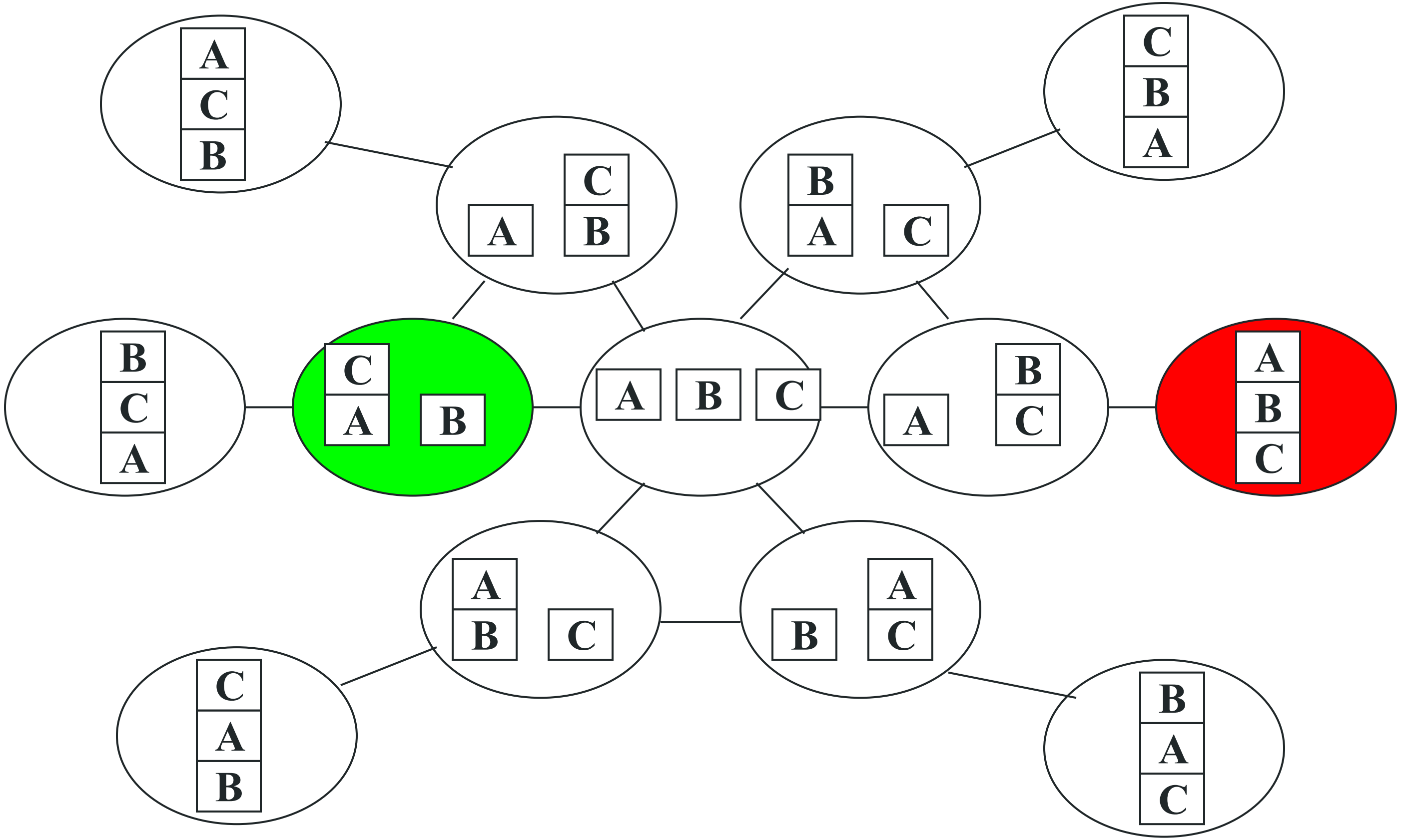
Solving Problems with PDDL

PDDL - used in International Planning Competition 1998 - today

Planners

- SAT Plan
- TL Plan
- FF
- Blackbox
- SHOP2
- TALPlanner
- many more...

Solving with Graph Search



Search for a Solution

Forward Chaining

Start State: Initial State

Applicable actions in a given state:

Those for which the *preconditions* are true in the current state

Goal test: are all literals in the goal state included in the current state?

Backward Chaining

Start State: Goal

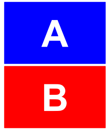
Relevant actions in a given state:

Those for which the *efffects* are true in the current (goal) state

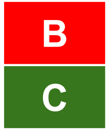
Goal test: are all literals in the current state included in the initial state?

Decompose the problem

Problem 1



Problem 2



Search for a solution to problem 1

Then, search for a solution to problem 2

Forward chain, looking considering each applicable action

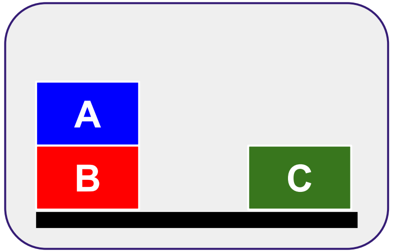
Choose the action that minimizes the distance between the current state and the goal

What might be a problem with this?

Sussman Anomaly

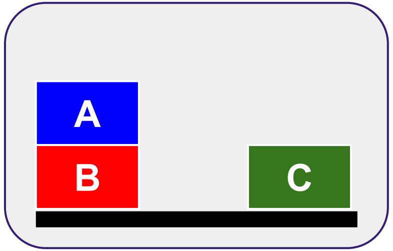
Planner finds a straightforward solution to Problem 1, placing A on B

Planner cannot pursue solve Problem 2 without undoing the solution to Problem 1



Sussman Anomaly

Planner finds a straightforward solution to Problem 1, placing A on B



Planner cannot pursue solve Problem 2 without undoing the solution to Problem 1

Heuristic Search

Relax the Problem

Remove preconditions

```
(:action move
    :parameters (?b ?x ?y)
    :precondition (and (on ?b ?x)
                                (clear ?b) (clear ?y))
    :effect (and (not (on ?b ?x))
                    (not (clear ?y))
                    (on ?b ?y)
                    (clear ?x)))
(:action move-to-table
    :parameters (?b ?x)
    :precondition (and (on ?b ?x)
                                (clear ?b)))
    :effect (and (not (on ?b ?x))
                    (on ?b Table)
                    (clear ?x)))
```

Relax the Problem

if you remove too many constraints,
you receive too many solutions to the problem

Remove preconditions

And negative effects

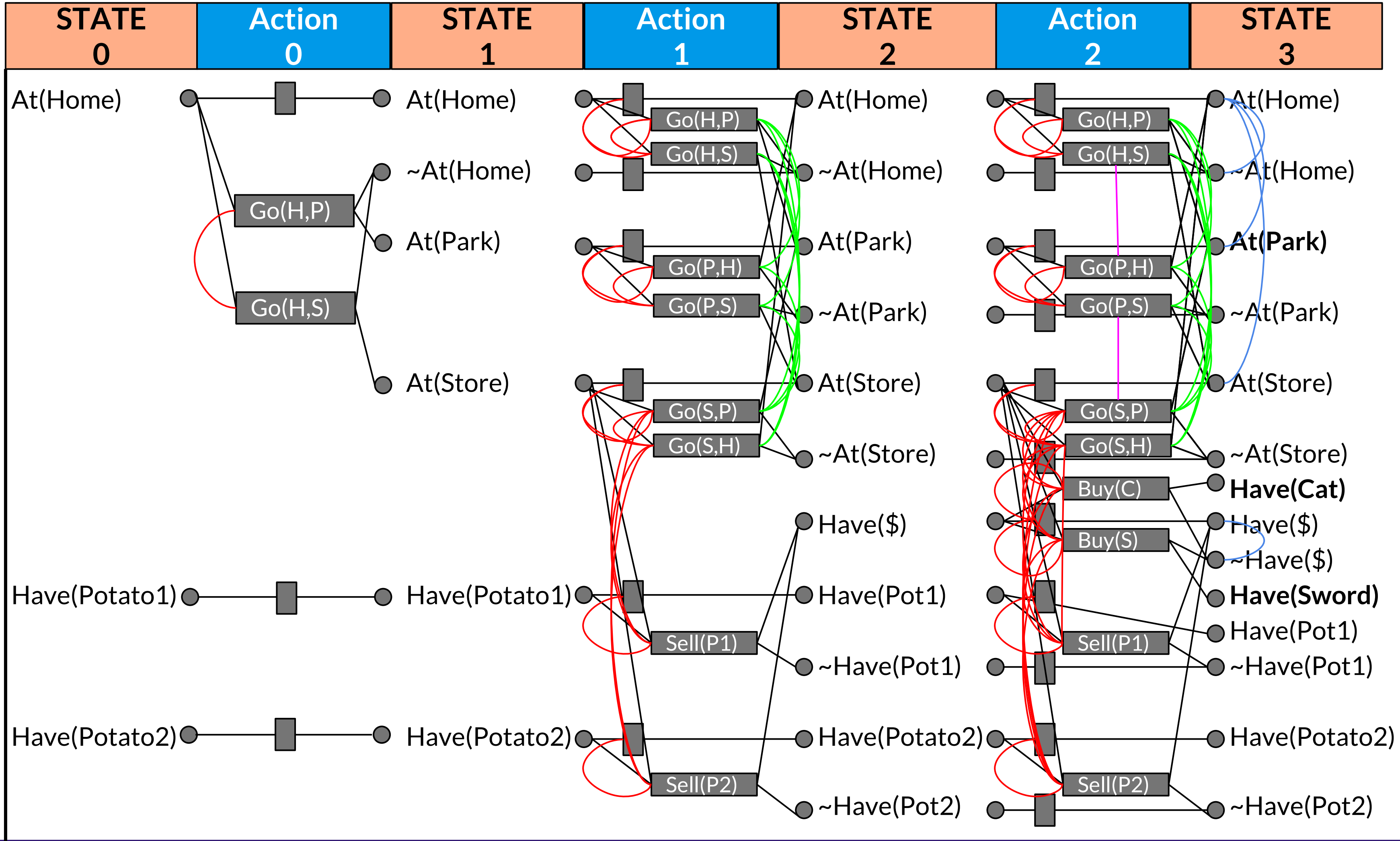
```
(:action move
    :parameters (?b ?x ?y)
    :precondition (and (on ?b ?x)
                       (clear ?b) (clear ?y))
    :effect (and (not (on ?b ?x))
                 (not (clear ?y))
                 (on ?b ?y)
                 (clear ?x)))
(:action move-to-table
    :parameters (?b ?x)
    :precondition (and (on ?b ?x)
                       (clear ?b)))
    :effect (and (not (on ?b ?x))
                 (on ?b Table)
                 (clear ?x)))
```

Relax the Problem

Just negative effects

```
(:action move
    :parameters (?b ?x ?y)
    :precondition (and (on ?b ?x)
                       (clear ?b) (clear ?y))
    :effect (and (not (on ?b ?x))
                 (not (clear ?y))
                 (on ?b ?y)
                 (clear ?x)))
(:action move-to-table
    :parameters (?b ?x)
    :precondition (and (on ?b ?x)
                       (clear ?b)))
    :effect (and (not (on ?b ?x))
                 (on ?b Table)
                 (clear ?x)))
```

GraphPlan



STATE 0

At(Home)

Have(Potato1)

Have(Potato2)

Action 0

Go(H,P)

Go(H,S)

STATE 1

At(Home)

~At(Home)

At(Park)

At(Store)

Have(Potato1)

Have(Potato2)

Action 1

Go(H,P)

Go(H,S)

Go(P,H)

Go(P,S)

Go(S,P)

Go(S,H)

Sell(P1)

Sell(P2)

STATE 2

At(Home)

~At(Home)

At(Park)

~At(Park)

At(Store)

~At(Store)

Have($)

Have(Pot1)

~Have(Pot1)

Have(Potato2)

~Have(Pot2)

Action 2

Go(H,P)

Go(H,S)

Go(P,H)

Go(P,S)

Go(S,P)

Go(S,H)

Buy(C)

Buy(S)

Sell(P1)

Sell(P2)

STATE 3

At(Home)

~At(Home)

At(Park)

~At(Park)

At(Store)

~At(Store)

Have(Cat)

Have($)

~Have($)

Have(Sword)

Have(Pot1)

~Have(Pot1)

Have(Potato2)

~Have(Pot2)

Pyperplan

Simple PDDL Planner

https://bitbucket.org/malte/pyperplan

problem of breadth-first search : uses up a lot of space

Online PDDL Editor

Online tool for editing PDDL and solving problems defined in PDDL

Import domains and problems from various resources

- International Planning Competition

http://editor.planning.domains/

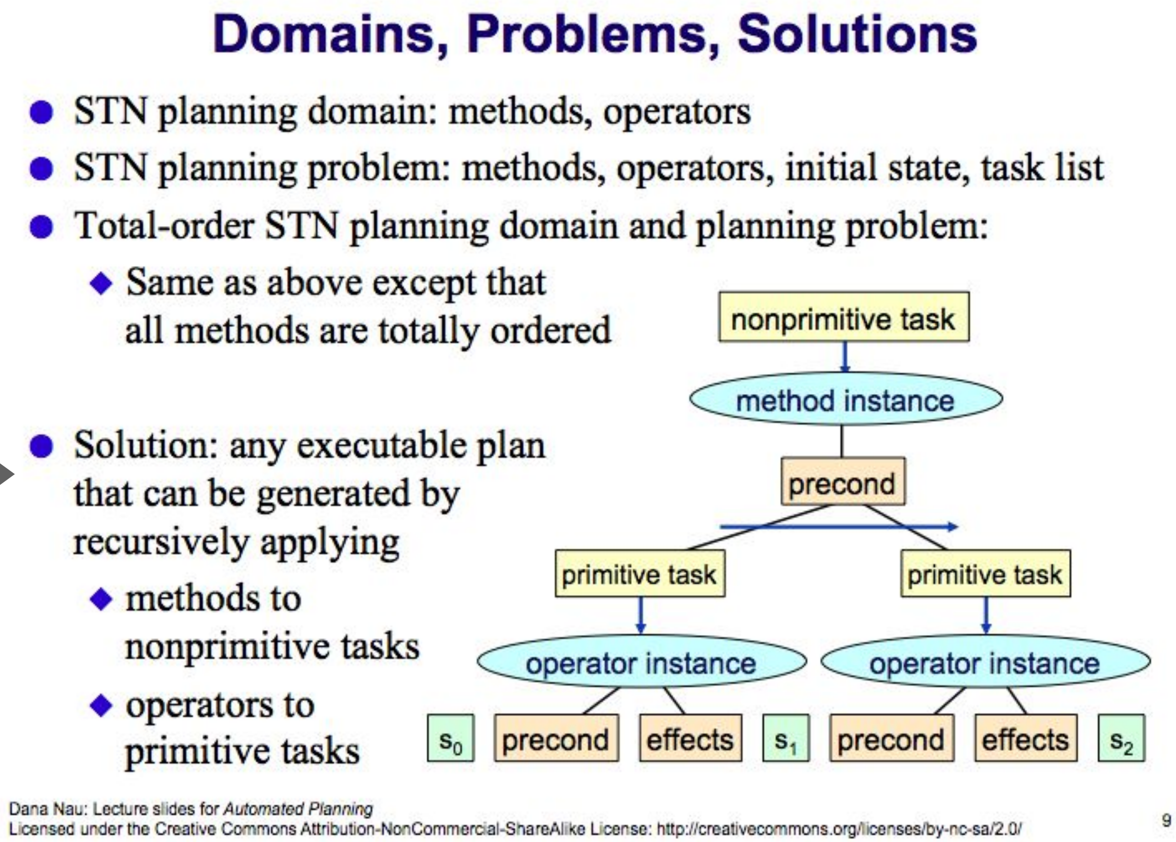
Other approaches

PlanSAT

GraphPlan

Case-based Planning

Hierarchical Task Network ➜
(HTN)



Domains, Problems, Solutions

- STN planning domain: methods, operators
- STN planning problem: methods, operators, initial state, task list
- Total-order STN planning domain and planning problem:
  - Same as above except that all methods are totally ordered
- Solution: any executable plan that can be generated by recursively applying
  - methods to nonprimitive tasks
  - operators to primitive tasks

Dana Nau: Lecture slides for *Automated Planning*