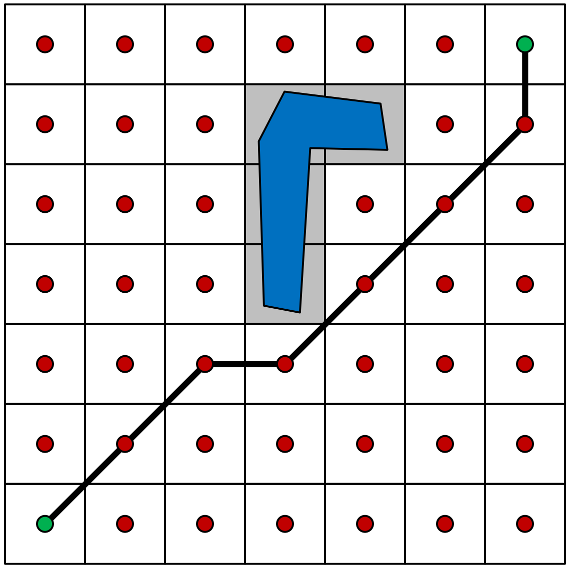
CMU 15-381

Lecture 5: Planning I

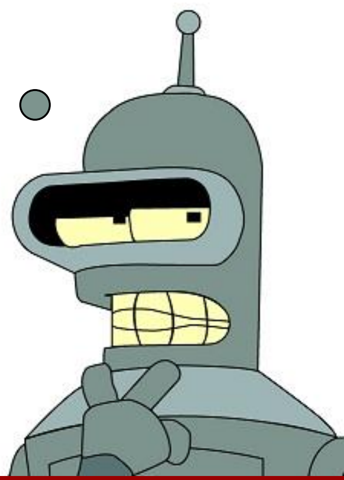
Teachers:
Emma Brunskill
Ariel Procaccia (this time)

Motion Planning

- Navigating between two points while avoiding obstacles
- A first approach: define a discrete grid
- Mark cells that intersect obstacles as blocked
- Find path through centers of remaining cells

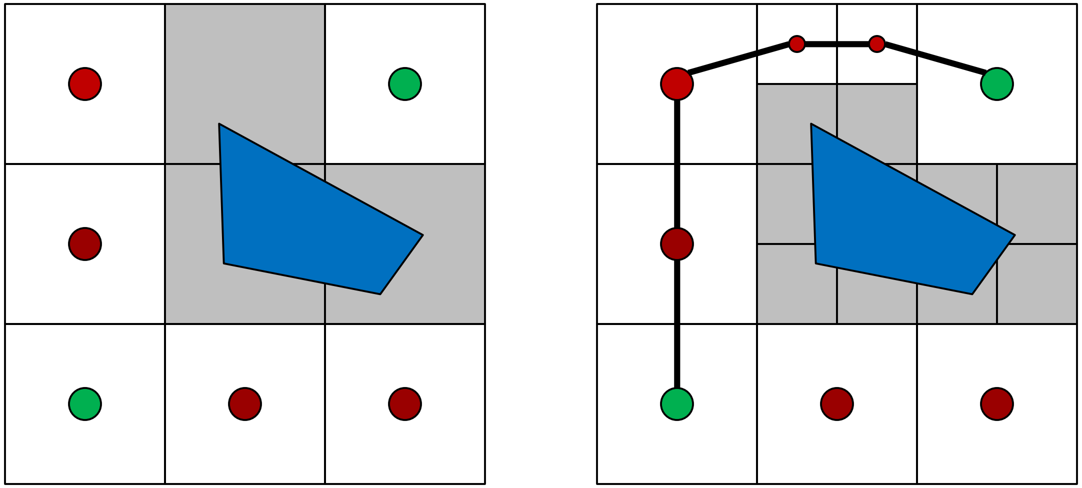


Is this approach optimal?
Complete?



Cell decomposition

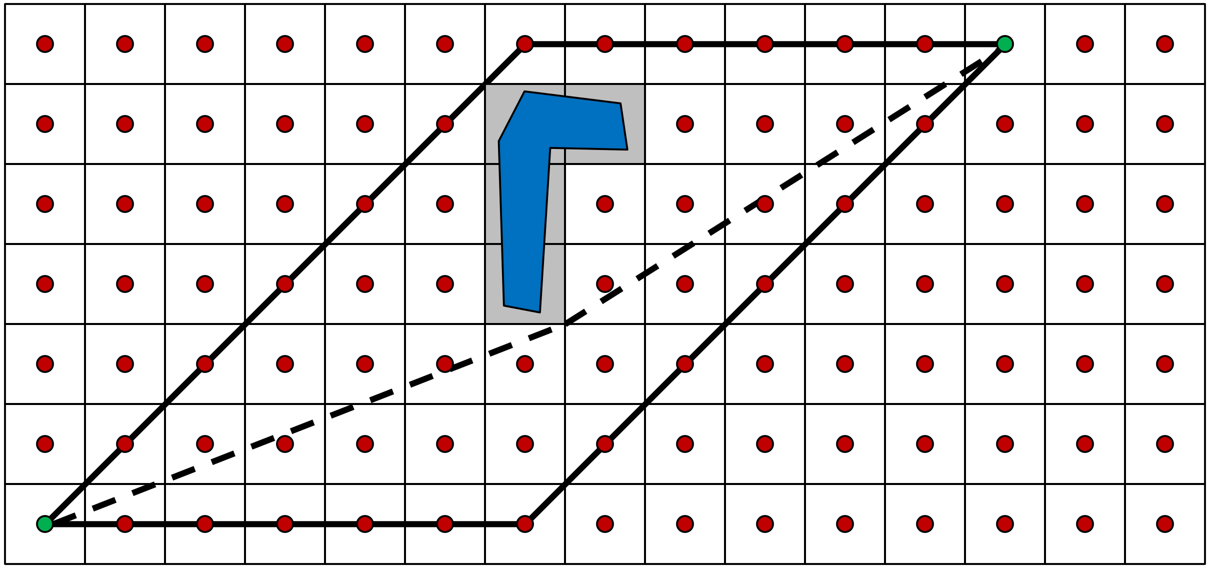
- Distinguish between
  - Cells that are contained in obstacles
  - Cells that intersect obstacles
- If no path found, subdivide the mixed cells



Is it complete now?

- An algorithm is resolution complete when:
  a. If a path exists, it finds it in finite time
  b. If a path does not exist, it returns in finite time
- Poll 1: Cell decomposition satisfies:
  1. a but not b
  2. b but not a
  3. Both a and b
  4. Neither a nor b

Cell decomposition

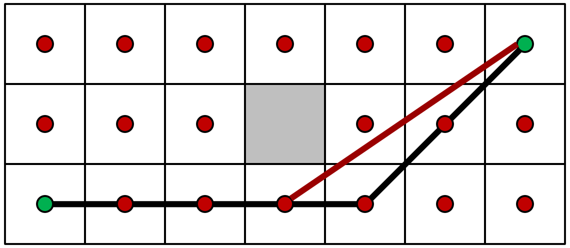


Shortest paths through cell centers

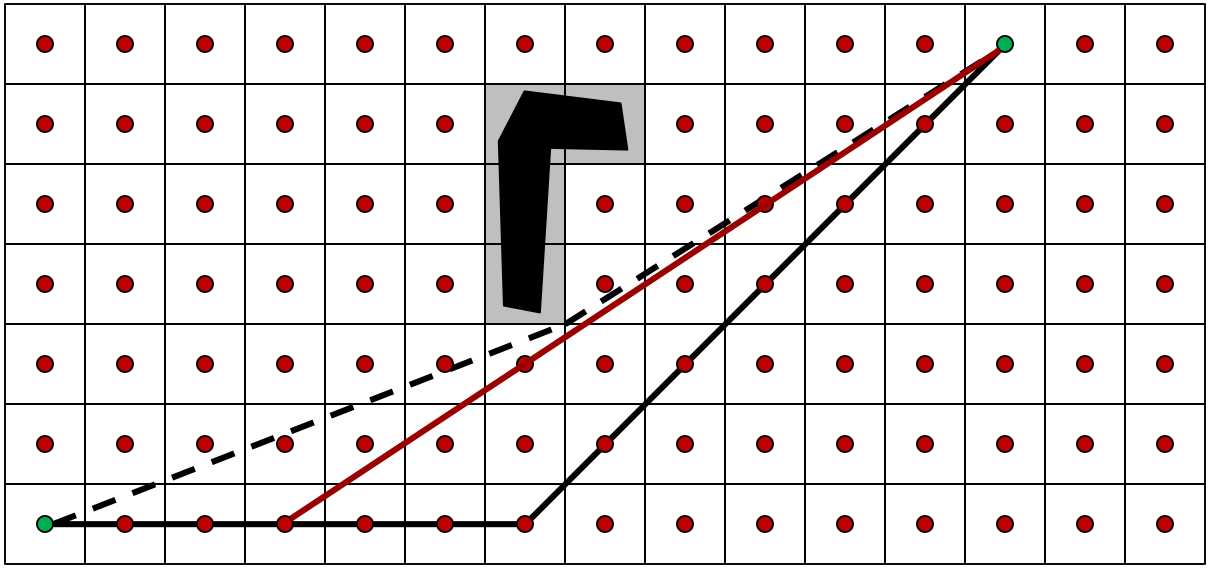
Shortest path

Solution 1: A* smoothing

- Allows connection to further states than neighbors on the grid
- Key observation:
  - If $x_1, \dots, x_n$ is valid path
  - And x_k is visible from x_j
  - Then $x_1, \dots, x_j, x_k, \dots, x_n$ is a valid path



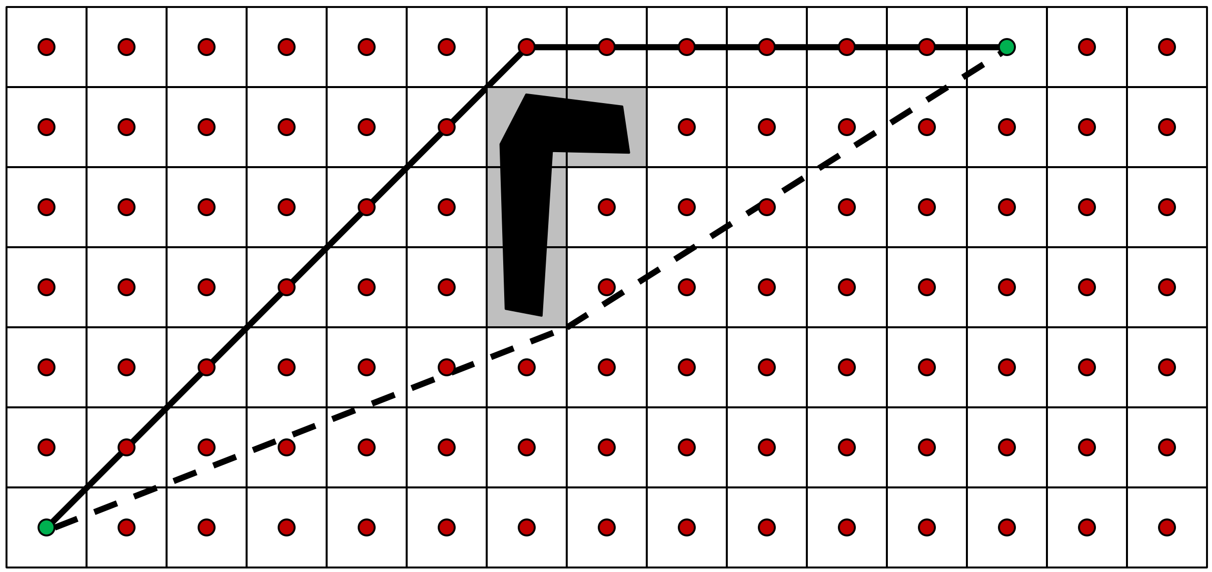
Smoothing works!



A shortest path through cell centers

Shortest path

Smoothing doesn't work!



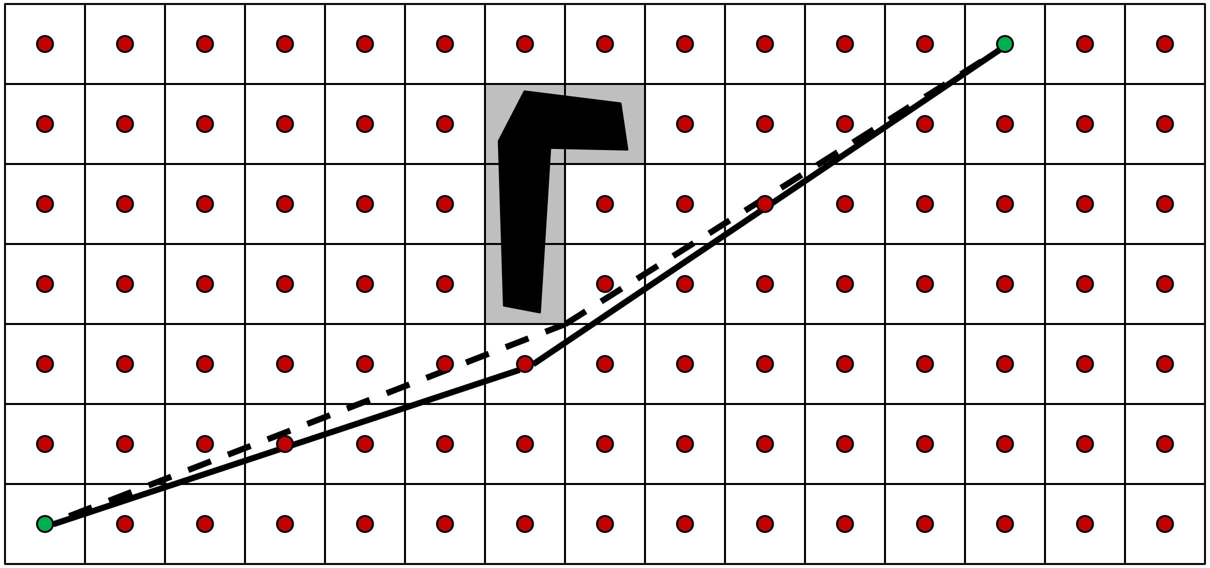
A shortest path through cell centers

Shortest path

Solution 2: Theta*

- Allow parents that are non-neighbors in the grid to be used during search
- Standard A*
  - $g(y) = g(x) + c(x, y)$
  - Insert y with estimate
    $$f(y) = g(x) + c(x, y) + h(y)$$
- Theta*
  - If parent(x) is visible from y, insert y with estimate $f(y) = g(\text{parent}(x)) + c(\text{parent}(x), y) + h(y)$

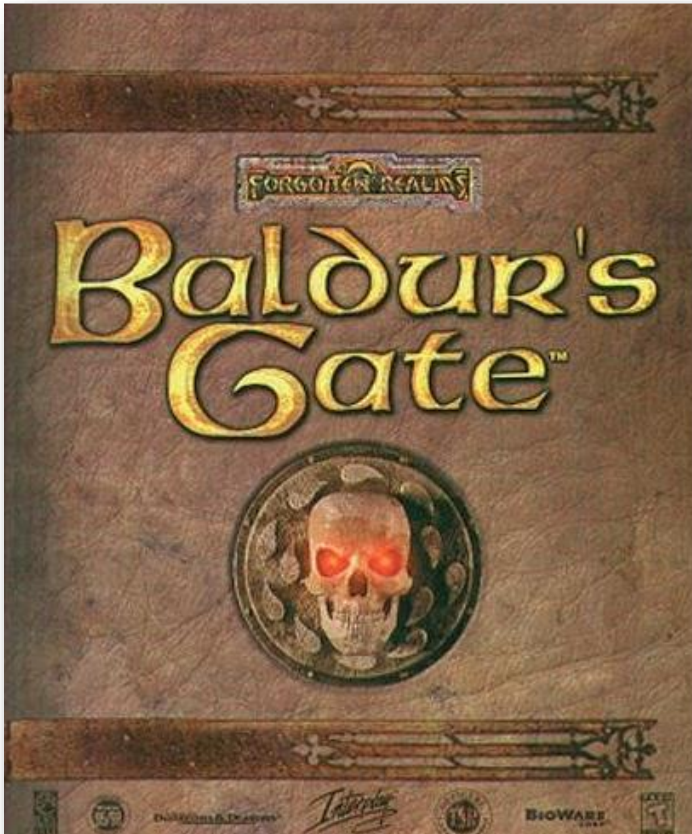
Theta* works!

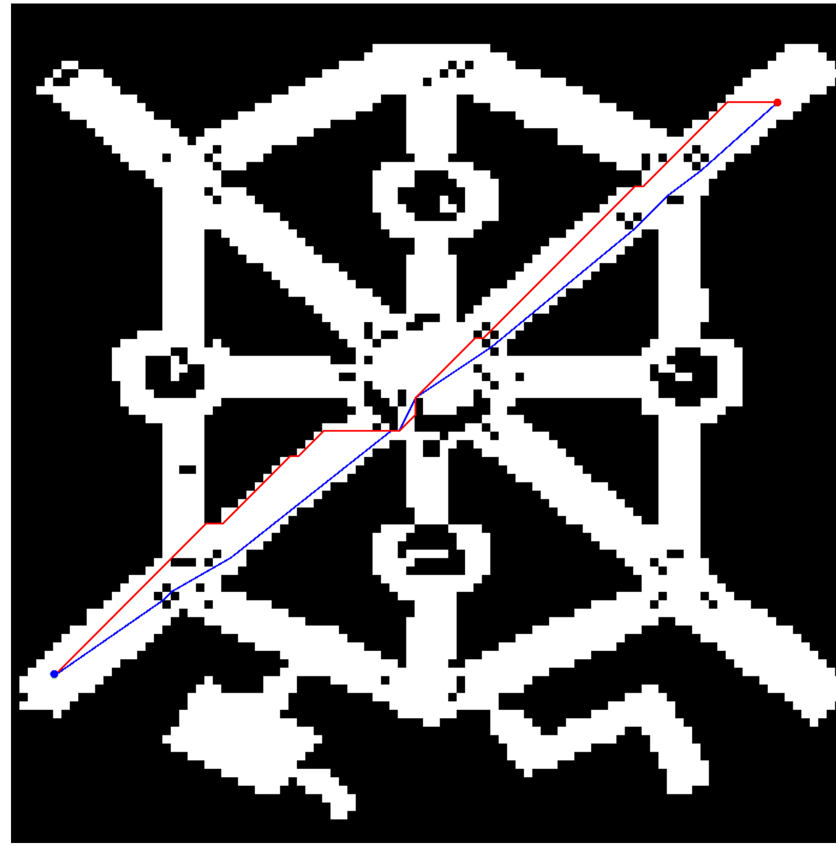


Theta* path, I think ☺

Shortest path

Theta* works!

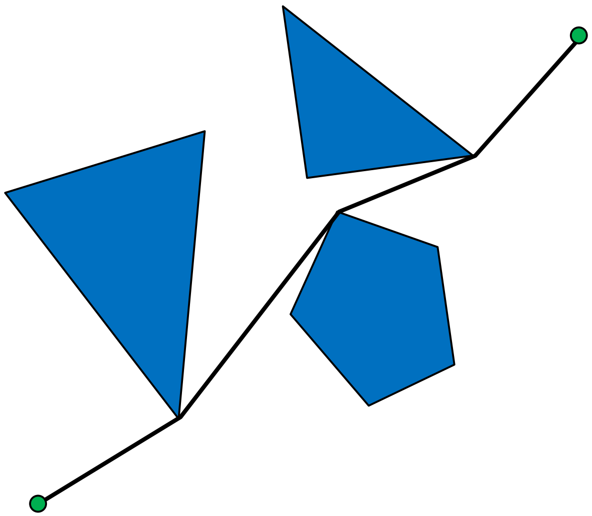




[Nash, AIGameDev 2010]

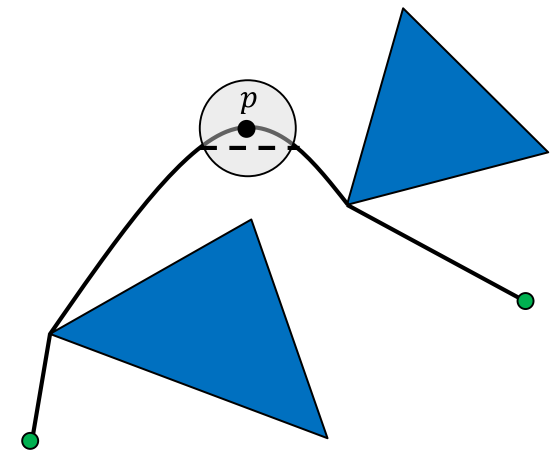
THE OPTIMAL PATH

- Polygonal path: sequence of connected straight lines
- Inner vertex of polygonal path: vertex that is not beginning or end
- Theorem: assuming polygonal obstacles, shortest path is a polygonal path whose inner vertices are vertices of obstacles



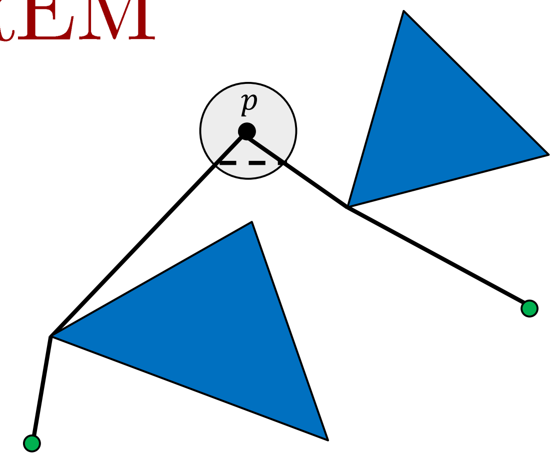
Proof of theorem

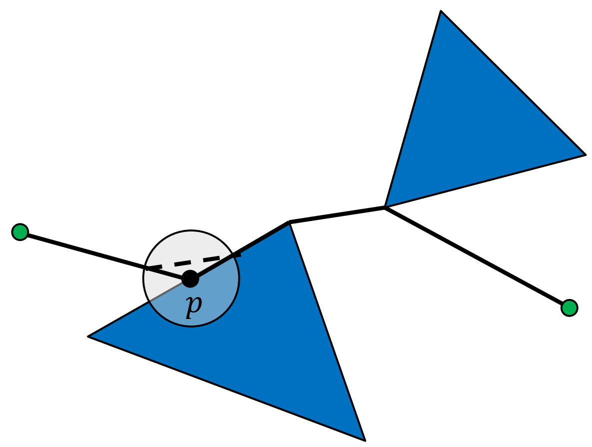
- Suppose for contradiction that shortest path is not polygonal
- Obstacles are polygonal $\Rightarrow$ $\exists$point p in interior of free space such that "path through p is curved"
- $\exists$disc of free space around p
- Path through disc can be shortened by connecting points of entry and exit



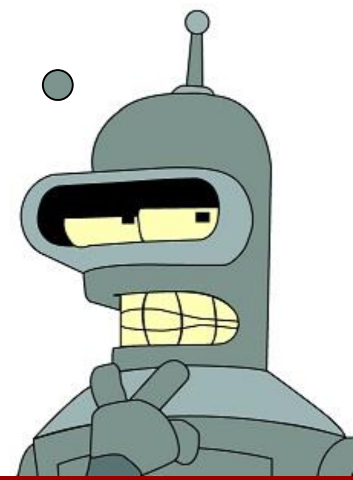
Proof of theorem

- Path is polygonal!
- Vertex cannot lie in interior of free space, otherwise we can do the same trick
- Vertex cannot lie on a the interior of an edge, otherwise we can do the same trick ∎





How would we define a graph on which A* would be optimal?



Planning, More Generally

- AI (also) studies rational action
- Devising a plan of action to achieve one's goal is a critical part of AI
- In fact, planning is glorified search
- We will consider a structured representation of states

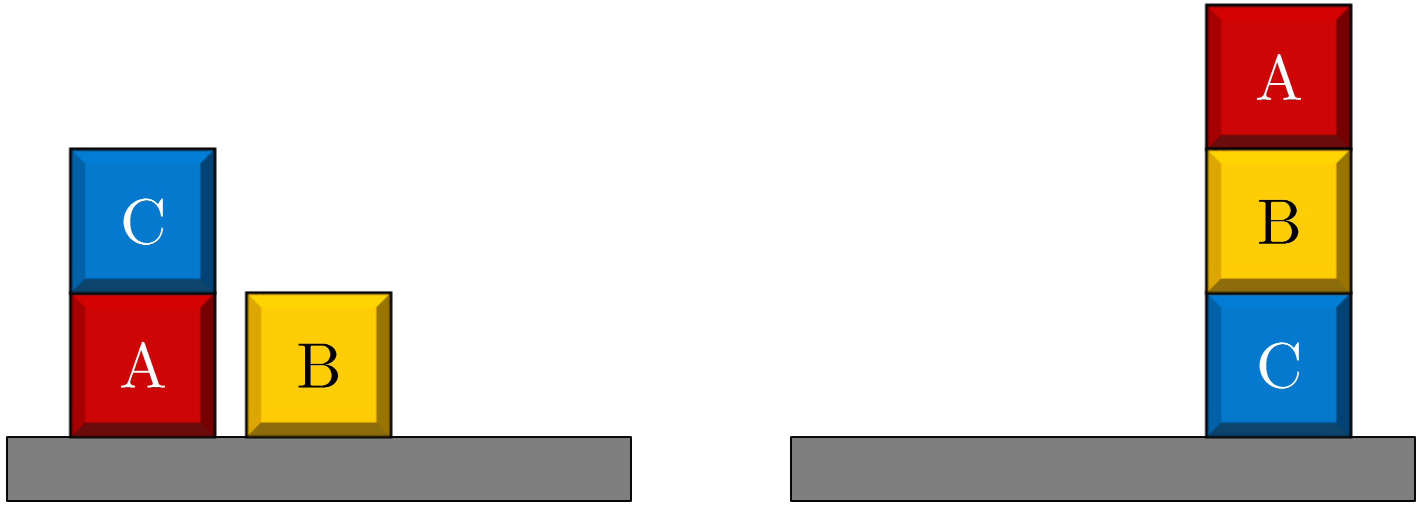
Propositional STRIPS planning

- STRIPS = Stanford Research Institute Problem Solver (1971)
- State is a conjunction of conditions, e.g., $\mathrm{at}(\mathrm{Truck}_1,\mathrm{Shadyside}) \wedge \mathrm{at}(\mathrm{Truck}_2,\mathrm{Oakland})$
- States are transformed via operators that have the form
  Preconditions $\Rightarrow$ Postconditions

Propositional STRIPS planning

- Pre is a conjunction of positive and negative conditions that must be satisfied to apply the operation
- Post is a conjunction of positive and negative conditions that become true when the operation is applied
- We are given the initial state
- We are also given the goals, a conjunction of positive and negative conditions
- We think of a state as a set of positive conditions, hence an operation has an "add list" and a "delete list"

Blocks world



Start

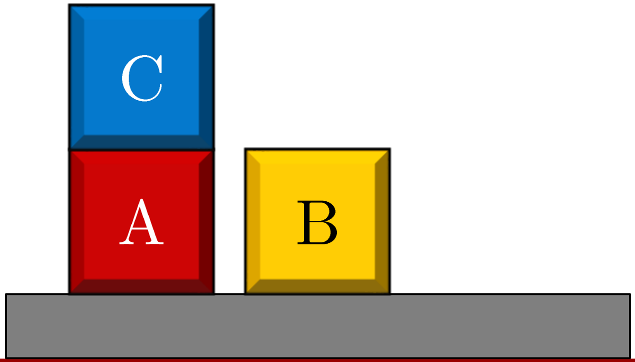
Goal

Blocks world

- Conditions: on(A,B), on(A,C), on(B,A), on(B,C), on(C,A), on(C,B), clear(A), clear(B), clear(C), on(A,Table), on(B,Table), on(C,Table)
- Operators for moving blocks
  - Move C from A to the table:
    clear(C) $\wedge$ on(C,A)
    $\Rightarrow$ on(C,Table) $\wedge$ clear(A) $\wedge$ $\neg$on(C,A)
  - Move A from the table to B
    clear(A) $\wedge$ on(A,Table) $\wedge$ clear(B)
    $\Rightarrow$ on(A,B) $\wedge$ $\neg$clear(B) and $\neg$on(A,Table)

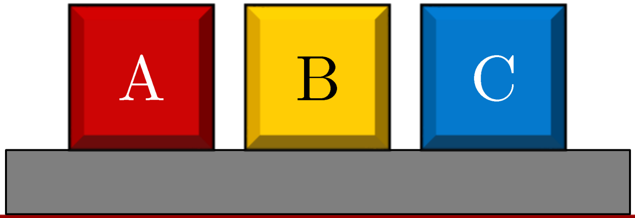
The plan

- State: on(C,A), on(A,Table), on(B,Table), clear(B), clear(C)
- Action:
  $\text{clear}(C) \wedge \text{on}(C,A)$
  $\Rightarrow \text{on}(C,\text{Table}) \wedge \text{clear}(A) \wedge \neg\text{on}(C,A)$



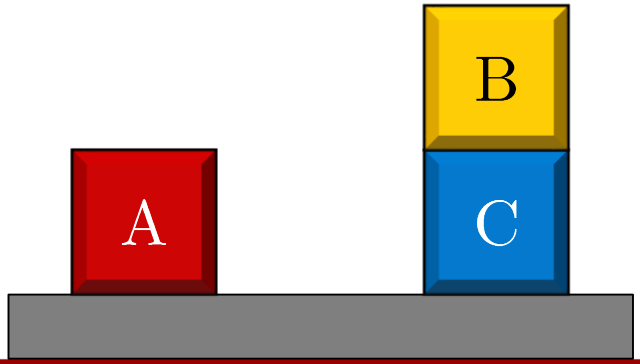
The plan

- State: on(A,Table), on(B,Table), clear(B), clear(C), on(C,Table), clear(A)
- Action:
  clear(C) $\wedge$ on(B,Table) $\wedge$ clear(B)
  $\Rightarrow$ on(B,C) $\wedge$ $\neg$clear(C) and $\neg$on(B,Table)



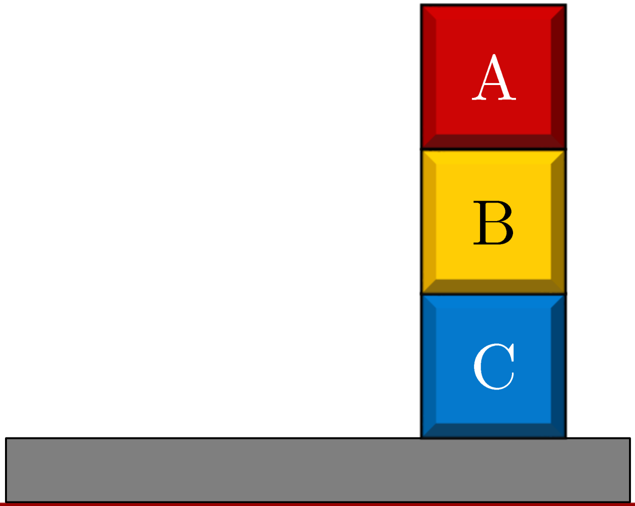
The plan

- State: on(A,Table), clear(B), on(C,Table), clear(A), on(B,C)
- Action:
  clear(B) $\wedge$ on(A,Table) $\wedge$ clear(A)
  $\Rightarrow$ on(A,B) $\wedge$ $\neg$clear(B) and $\neg$on(A,Table)



The plan

- State: on(C,Table), clear(A), on(B,C), on(A,B)
- Goals: on(A,B), on(B,C)



Complexity of planning

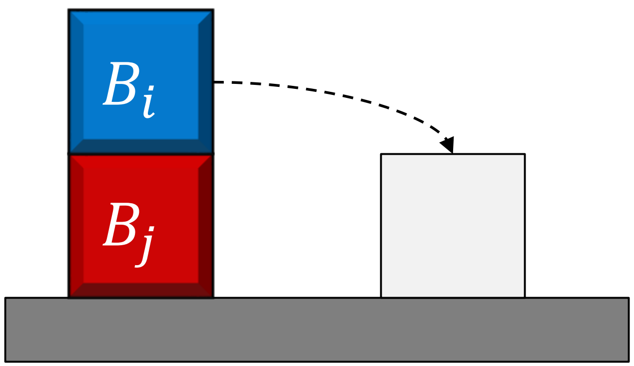
- Plansat is the problem of determining whether a given planning problem is satisfiable
- In general Plansat is **PSPACE**-complete
- We will look at some special cases

Complexity of Planning

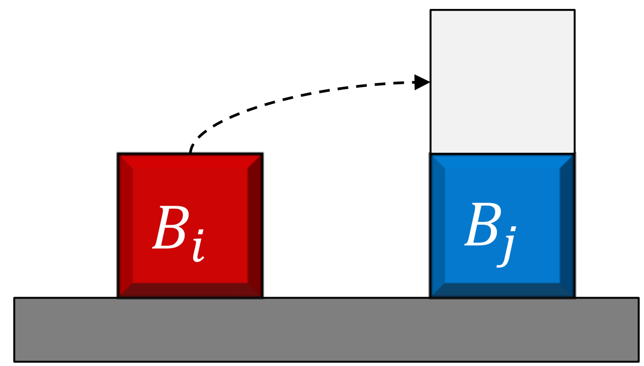
- Theorem 1: Assume that actions have only positive preconditions and a single postcondition. Then PLANSAT is in **P**
- Theorem 2: Blocks world problems can be encoded as above
- Silly corollary: Blocks world problems can be solved in polynomial time (Duh)

Proof of Theorem 2

- We will convert blocks world operators to operators that have only positive preconditions and a single postcondition
- Let the blocks be $B_1, \dots, B_n$
- Conditions: $\mathrm{off}(i, j)$ means B_i is not on top of B_j



$$\bigwedge_k \mathrm{off}(k, i) \wedge \bigwedge_{k \neq i} \mathrm{off}(k, j) \Rightarrow \mathrm{off}(i, j)$$



$$\bigwedge_k \mathrm{off}(k, i) \wedge \bigwedge_k \mathrm{off}(i, k) \wedge \bigwedge_k \mathrm{off}(k, j) \Rightarrow \neg\mathrm{off}(i, j)$$

Proof of Theorem 1*

- Lemma: It is sufficient to consider plans that first make conditions true, then make conditions false
- Proof:
  - Suppose that o_i and o_{i+1} are adjacent operators s.t. the postcondition p of o_i is negative and the postcondition q of o_{i+1} is positive
  - If $p = q$ then we can delete o_i because its effect is reversed
  - Otherwise, we can switch o_i and o_{i+1} $\blacksquare$

* Just for fun

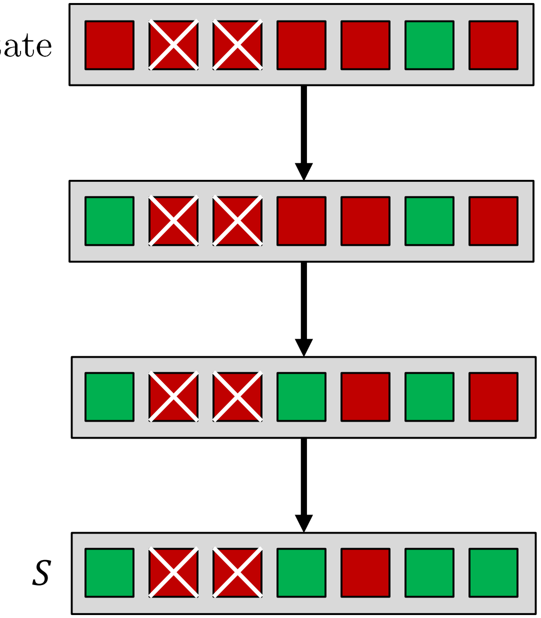
Proof of Theorem 1*

- By the lemma, if there is a solution, there is an intermediate state S such that
  - S can be reached from the initial state using operations with positive postconditions
  - The positive goals are a subset of S
  - Negative goals can be achieved via operations with negative postconditions
- Search for an intermediate state S with these properties

* Just for fun

Proof of Theorem 1*

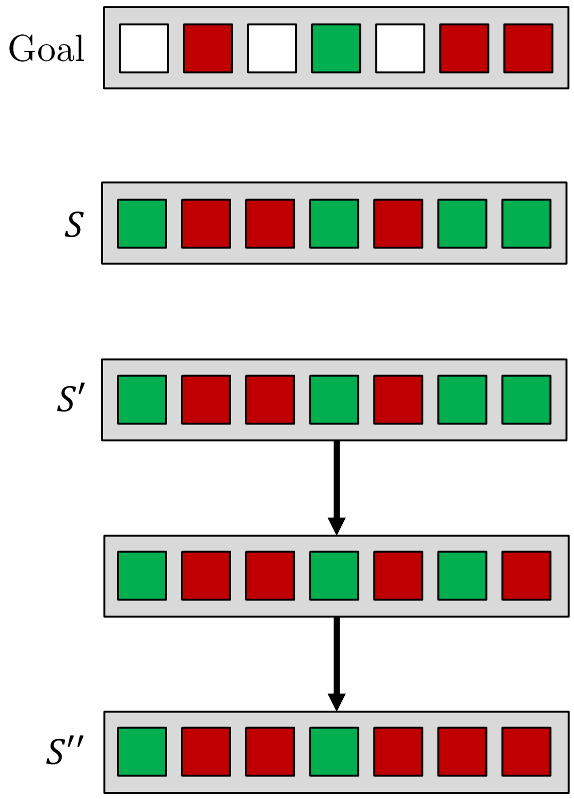
- Implement procedure **TurnOn**(X): given set of conditions X, find maximal state S such that $S \cap X = \emptyset$ that can be reached from initial state using operators with positive postconditions
  - Preconditions are positive, so:
  - Simply apply all such operators until it makes no difference



* Just for fun

Proof of Theorem 1*

- Denote S'' the state resulting from removing negative goals from S
- Implement procedure TurnOff(S): find the maximal S' such that S'' is reachable from S' using operators with negative postconditions in S
  - Simply search backwards from S'' and reverse operators with (i) negative postconditions in S (ii) preconditions satisfied



* Just for fun

Proof of Theorem 1*

- In the first iteration, if positive goals are not satisfied by S, there is no way to achieve them
- If $S \setminus S' \neq \emptyset$, it is impossible to remove these conditions; must be added to X
- X grows monotonically $\Rightarrow$ polynomial time ■

$X = \emptyset$
loop
    $S = \mathrm{TurnOn}(X)$
    if S does not contain positive goals then return reject
    $S' = \mathrm{TurnOff}(S)$
    if $S = S'$ then return accept
    $X = X \cup (S \setminus S')$
    if X intersects with initial state then return reject

* Just for fun

Summary

- Terminology:
  - Cell decomposition
  - Resolution completeness
  - Theta*
  - STRIPS
- Big ideas:
  - A* can be modified to work well in continuous spaces