Planning Problem Representation

PDDL + Assignment #1-1

Michaela Urbanovská

PUI Tutorial
Week 2

Lecture Check

- Any questions regarding the lecture or organization?

Organization Recap

- One assignment for every part of the class
- Assignment #1 has 3 parts
- We collect feedback every week to improve the course as we go
- The assignment for classical planning is new $\rightarrow$ feedback is appreciated

Planning Motivation

- **General problem solving**
- Basically can solve all your problems

Planning Motivation

- **General problem solving**
- Basically can solve all your problems
- Problem

Planning Motivation

- **General problem solving**
- Basically can solve all your problems
- Problem + representation

Planning Motivation

- **General problem solving**
- Basically can solve all your problems
- Problem + representation + solver

Planning Motivation

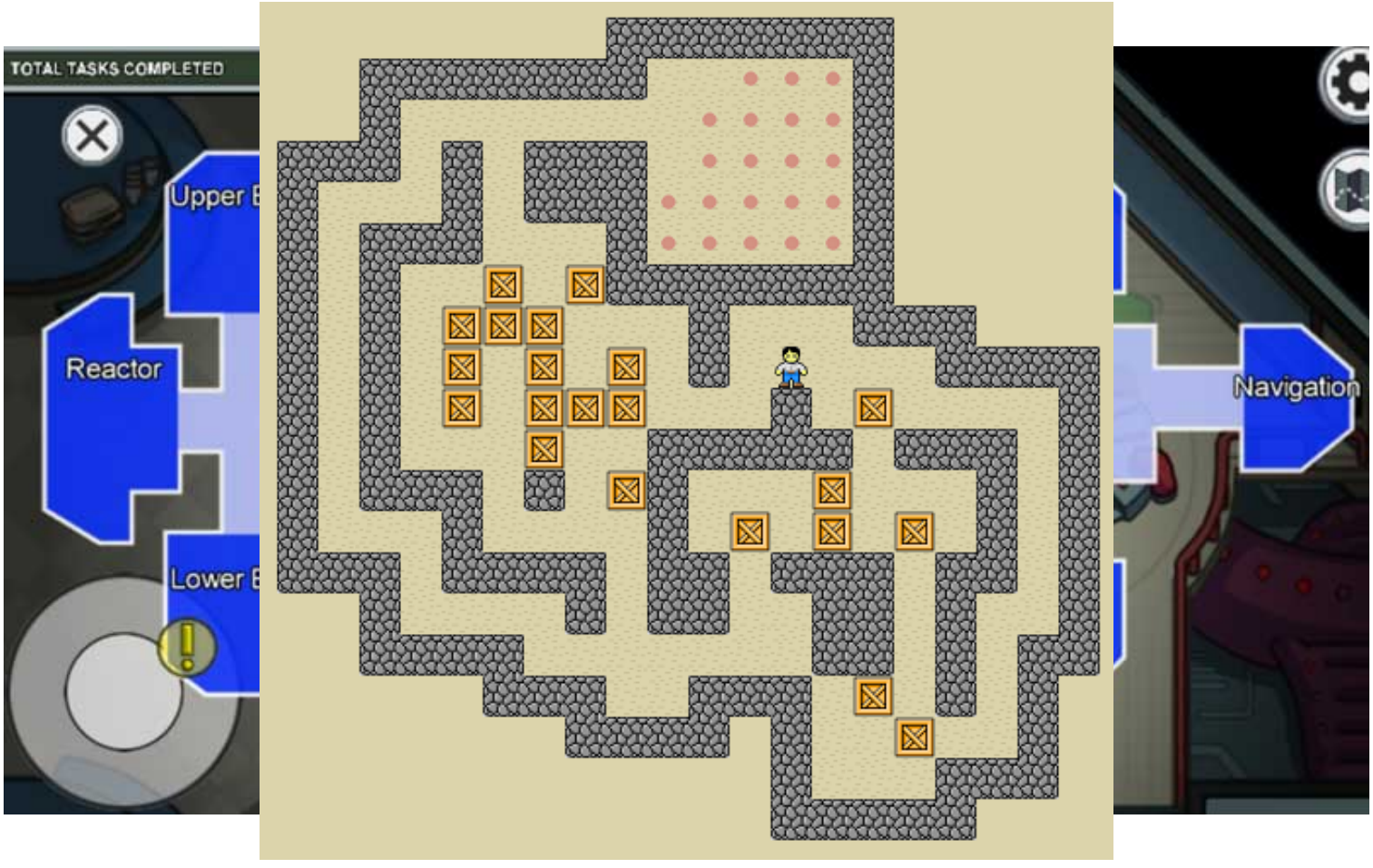
- **General problem solving**
- Basically can solve all your problems
- Problem + representation + solver = **solution**

Planning Motivation

- **General problem solving**
- Basically can solve all your problems
- Problem + representation + solver = **solution**

Planning Motivation

- **General problem solving**
- Basically can solve all your problems
- Problem + representation + solver = **solution**

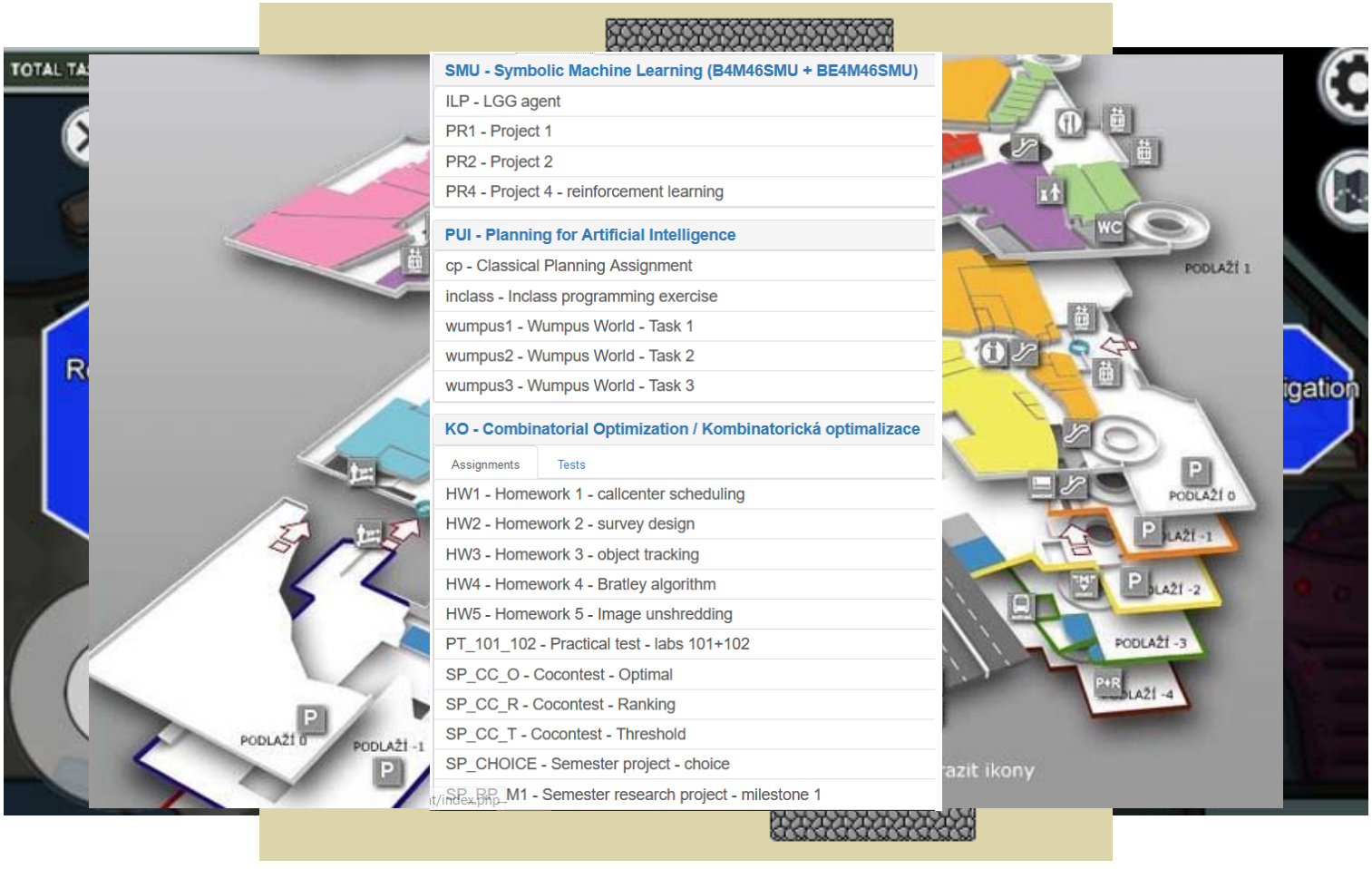


Planning Motivation

- **General problem solving**
- Basically can solve all your problems
- Problem + representation + solver = **solution**

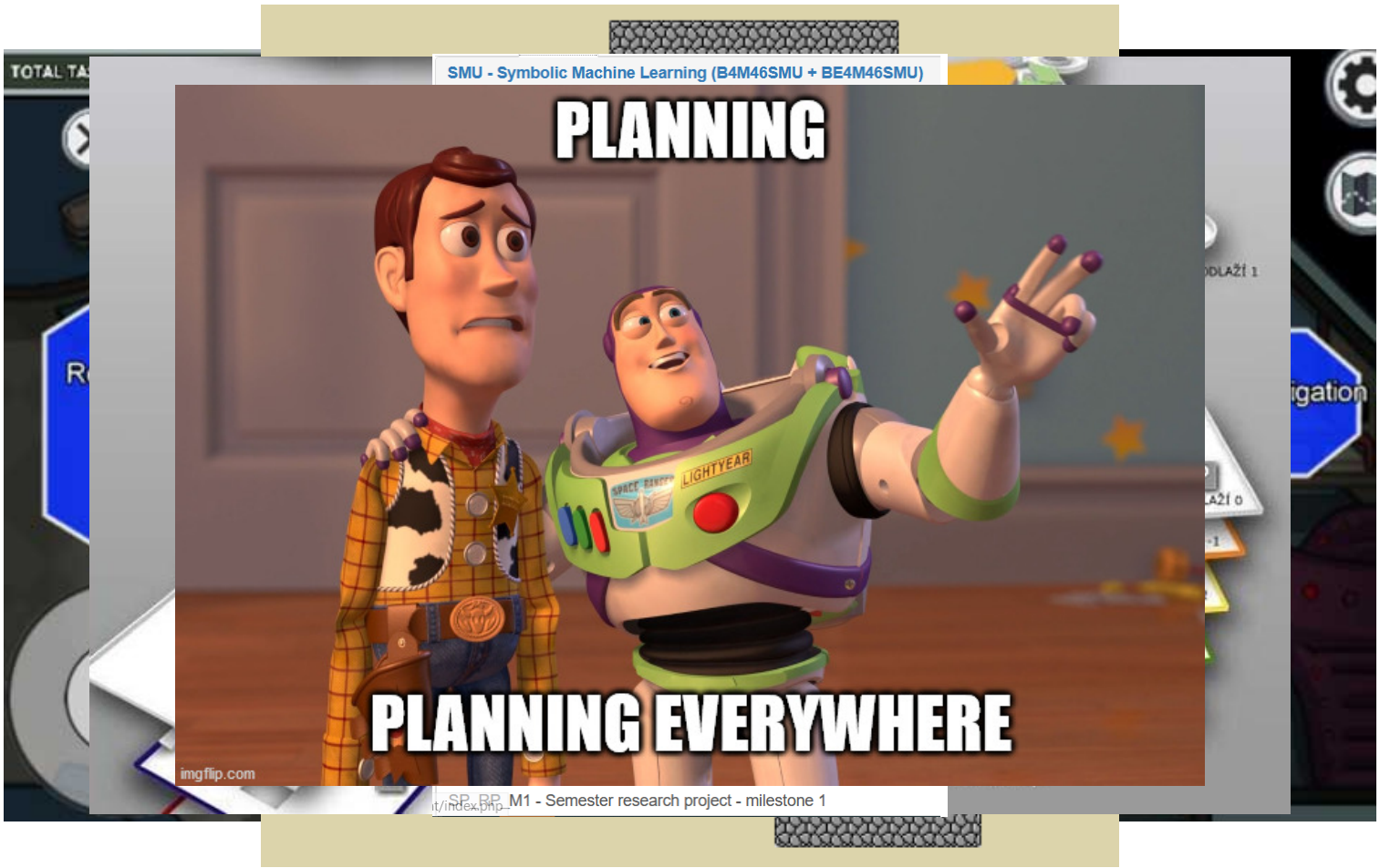
Planning Motivation

- **General problem solving**
- Basically can solve all your problems
- Problem + representation + solver = **solution**



Planning Motivation

- **General problem solving**
- Basically can solve all your problems
- Problem + representation + solver = **solution**

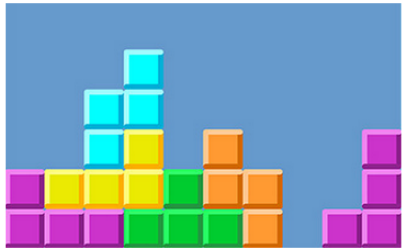


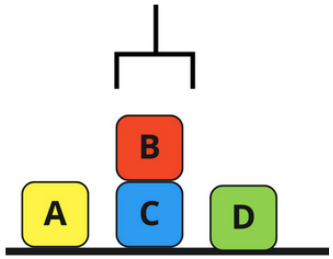
Planning Benchmarks

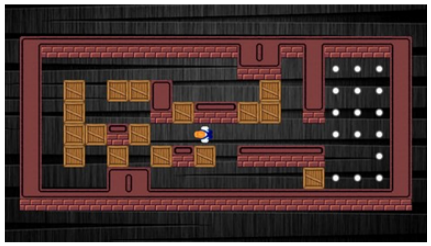
For example...

- Airport
- Depot
- Sokoban
- Blocksworld
- Elevators
- Floortile
- Parking
- Pipesworld
- Tetris
- Tidybot
- Woodworking









Benchmarks

- Often inspired by real-world problems
- Sometimes even modeled real-world problems
- Problems with interesting properties to test performance of algorithms / heuristics
- Doesn't have to correlate with real-world performance

Benchmarks

- Often inspired by real-world problems
- Sometimes even modeled real-world problems
- Problems with interesting properties to test performance of algorithms / heuristics
- Doesn't have to correlate with real-world performance

...this is not the class with robots.

Problem Definitions

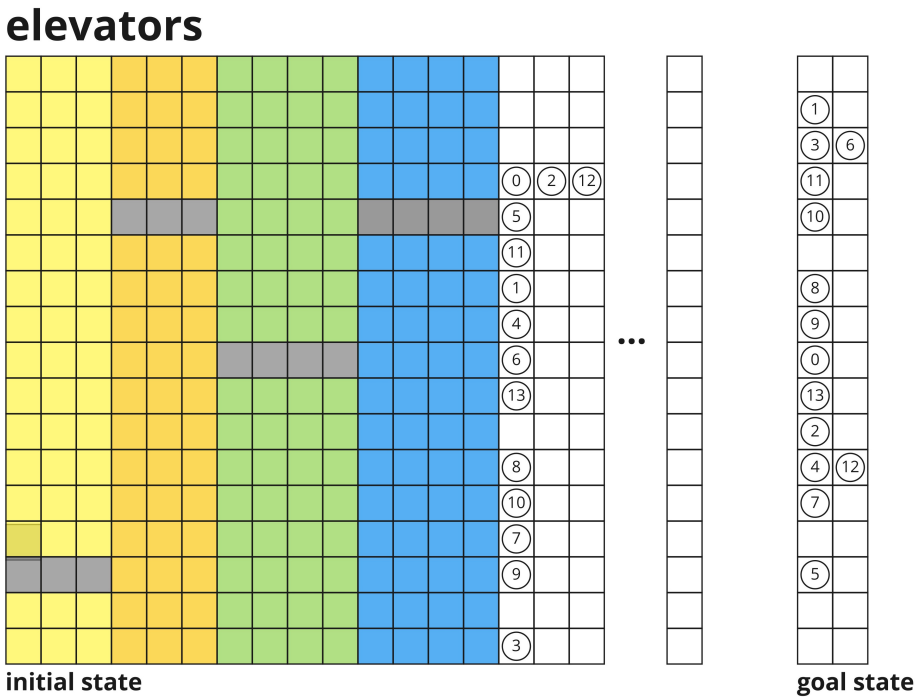
There are many different types of planning...
In this part of the course $\rightarrow$ classical planning

- Fully defined environment
- Deterministic actions
- Domain-independent approaches

Problem Modeling

How to model a problem?

- Natural language description?
- Video footage?
- Drawing?
- Standardized language?



(sat11) The scenario is the following: There is a building with N+1 floors, numbered from 0 to N. The building can be separated in blocks of size M+1, where M divides N. Adjacent blocks have a common floor. For example, suppose N=12 and M=4, then we have 13 floors in total (ranging from 0 to 12), which form 3 blocks of 5 floors each, being 0 to 4, 4 to 8 and 8 to 12. The building has K fast (accelarating) elevators that stop only in floors that are multiple of M/2 (so M has to be an even number). Each fast elevator has a capacity of X persons. Furthermore, within each block, there are L slow elevators, that stop at every floor of the block. Each slow elevator has a capacity of Y persons (usually Y

PDDL Language

- **P**lanning **D**omain **D**efinition **L**anguage
- General language to describe planning problems
- Based on first-order logic
- Syntax similar to LISP
- **Domain definition**
  - defines the world / environment
- **Problem definition**
  - defines problem instance in the modeled world
- Online editor: *http://editor.planning.domains*

PDDL Language - Domain

Domain definition

- Defines the environment / world
  - Hierarchy of types
  - Predicates (properties, relations)
  - Action schemas

PDDL Language - Domain

Domain definition

- Defines the environment / world
  - Hierarchy of types
  - Predicates (properties, relations)
  - Action schemas

Domain file structure

(define (domain *name*)
    PDDL definition of types
    PDDL definition of predicates
    PDDL definition of actions
)

PDDL Language - Problem

Problem definition

- Defines the problem instance we want to solve
  - Objects and their types
  - Initial state of the world
  - Goal condition

PDDL Language - Problem

Problem definition

- Defines the problem instance we want to solve
  - Objects and their types
  - Initial state of the world
  - Goal condition

Problem file structure

(define (problem *name*) (:domain *name*)
    PDDL definition of objects
    PDDL definition of initial state
    PDDL definition of goal
)

PDDL Language

Domain file structure

(define (domain *name*)
    PDDL definition of types
    PDDL definition of predicates
    PDDL definition of actions
)

Problem file structure

(define (problem *name*) (:domain *name*)
    PDDL definition of objects
    PDDL definition of initial state
    PDDL definition of goal
)

PDDL Language

Domain file structure

(define (domain *name*)
    PDDL definition of types
    PDDL definition of predicates
    PDDL definition of actions
)

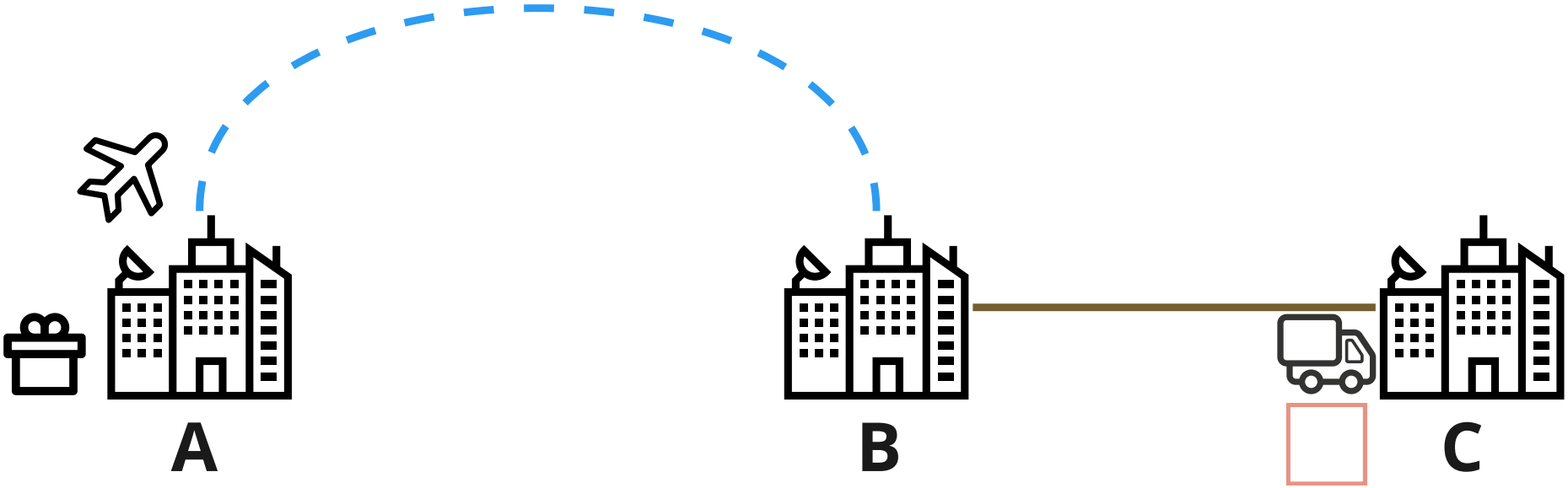
Problem file structure

(define (problem *name*) (:domain *name*)
    PDDL definition of objects
    PDDL definition of initial state
    PDDL definition of goal
)

...now what?

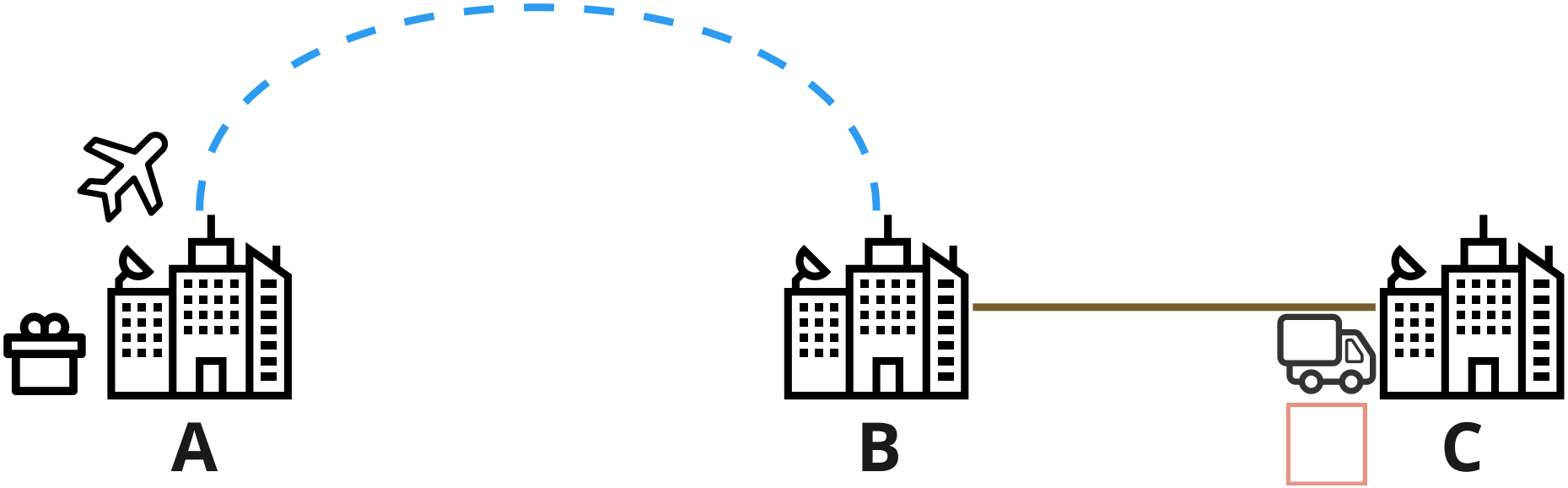
PDDL Example

Logistics example



Let's try it out (http://editor.planning.domains)

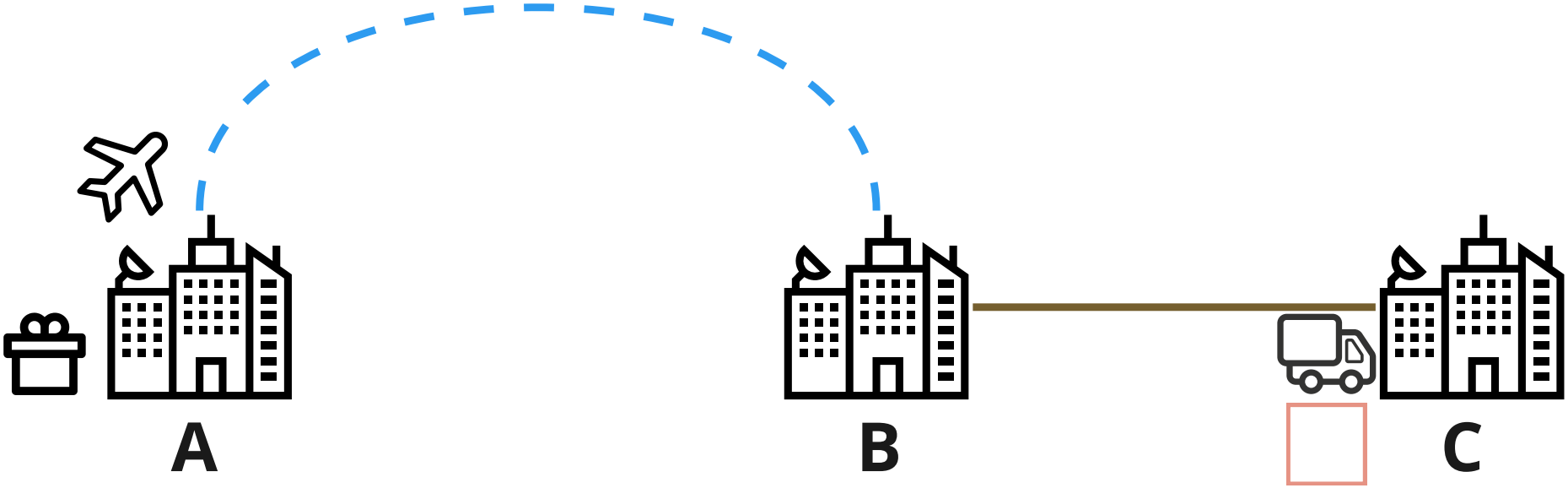
PDDL Example



Domain definition

- *types:*
- *predicates:*
- *actions:*

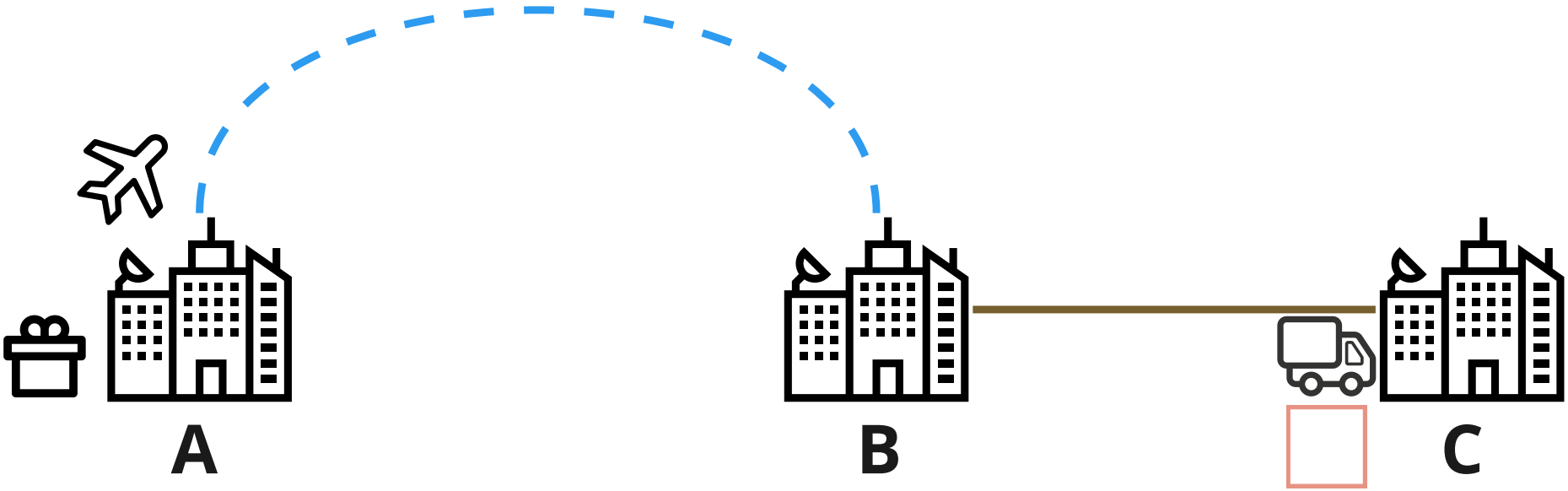
PDDL Example



Domain definition

- *types:* location, vehicle, package, ...
- *predicates:*
- *actions:*

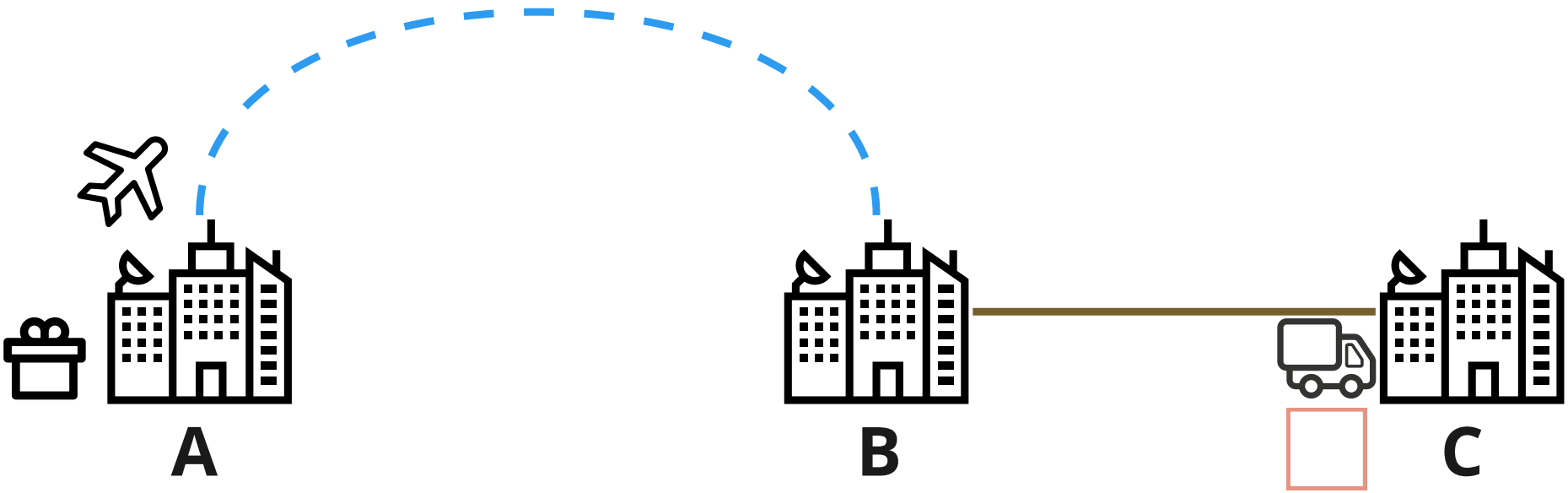
PDDL Example



Domain definition

- *types:* location, vehicle, package, ...
- *predicates:* where is truck, where is package, is there a route between locations, ...
- *actions:*

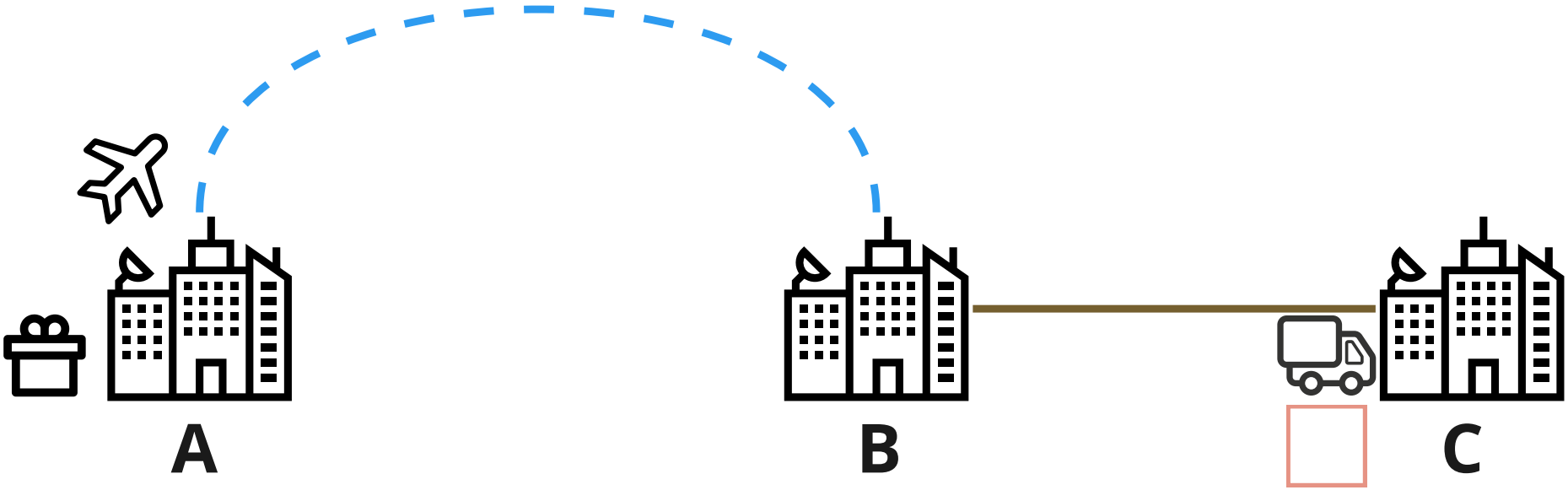
PDDL Example



Domain definition

- *types:* location, vehicle, package, ...
- *predicates:* where is truck, where is package, is there a route between location, ...
- *actions:* load package, drive truck, ...

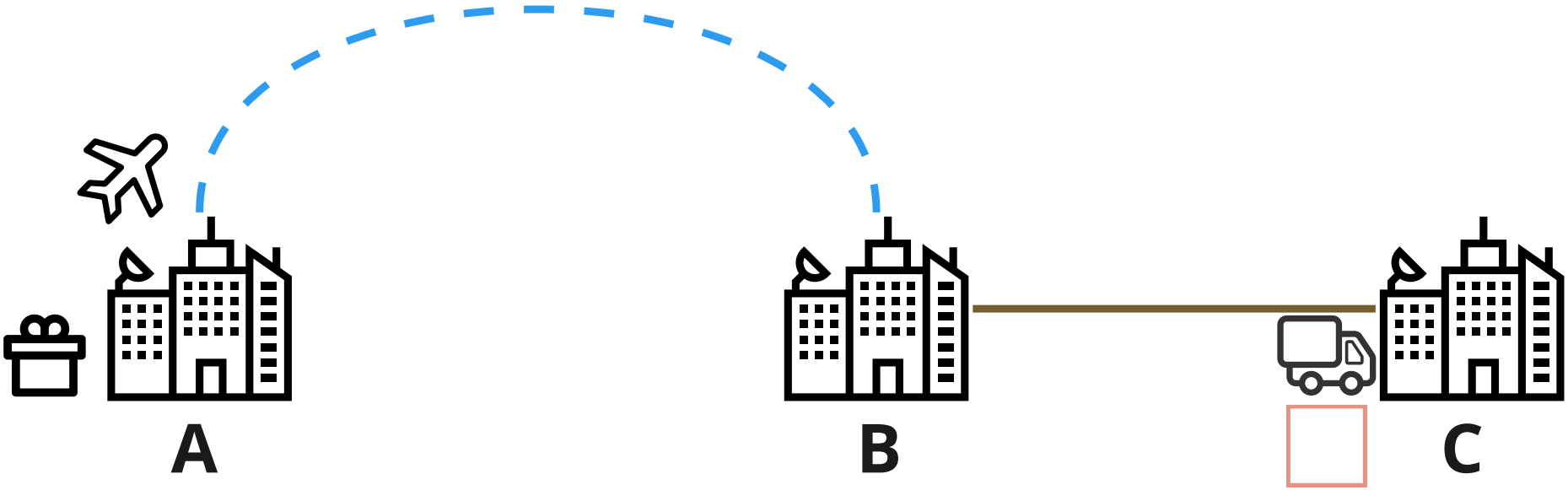
PDDL Example



Action schema

- *parameters:* objects and their types
- *preconditions:* what has to hold so that action can be applied
- *effects:* what happens after we apply the action

PDDL Example

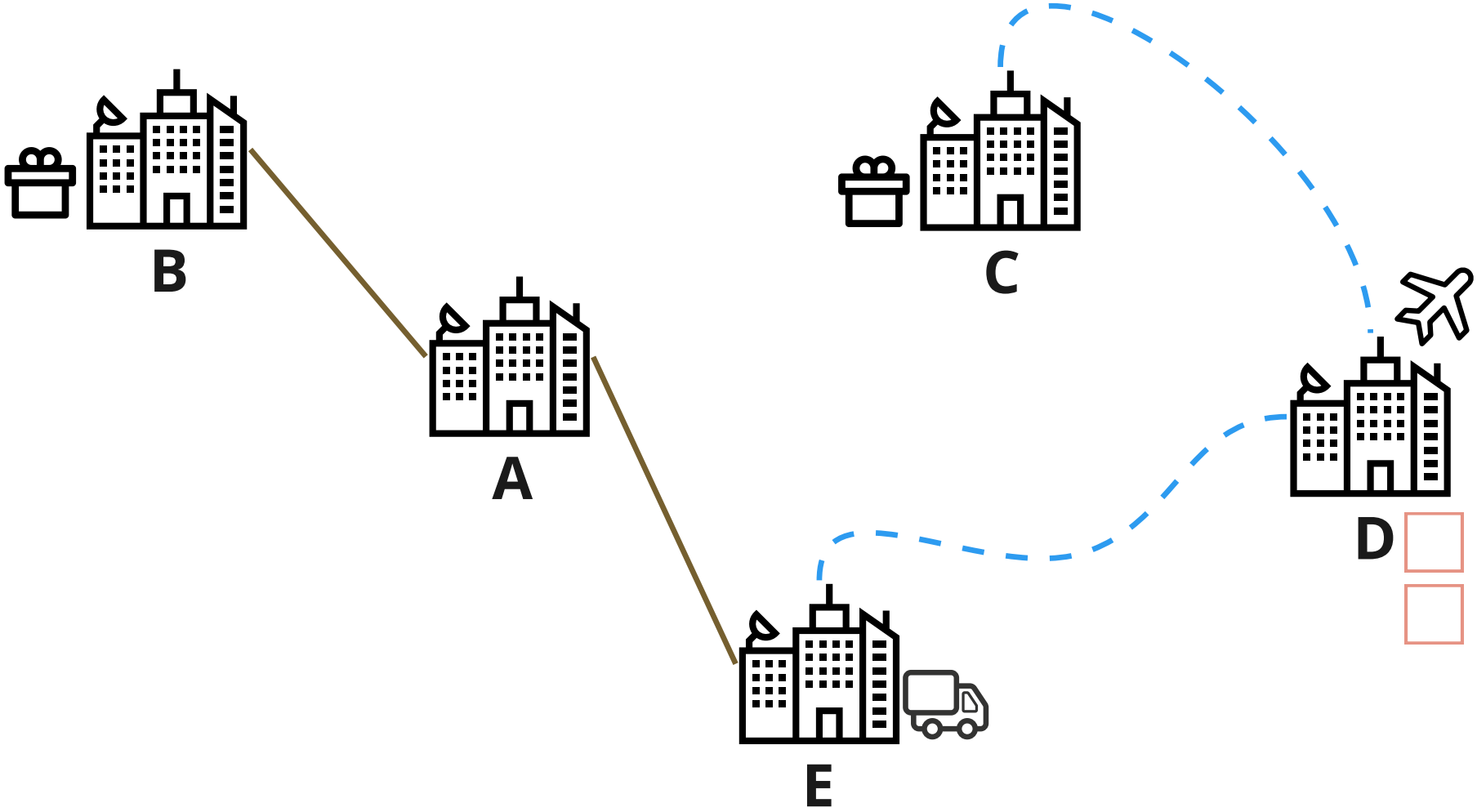


Problem definition

- *objects:* airplane, package, location A, ...
- *initial state:* package is at A, truck is at C, B is connected to C, ...
- *goal condition:* package is at C

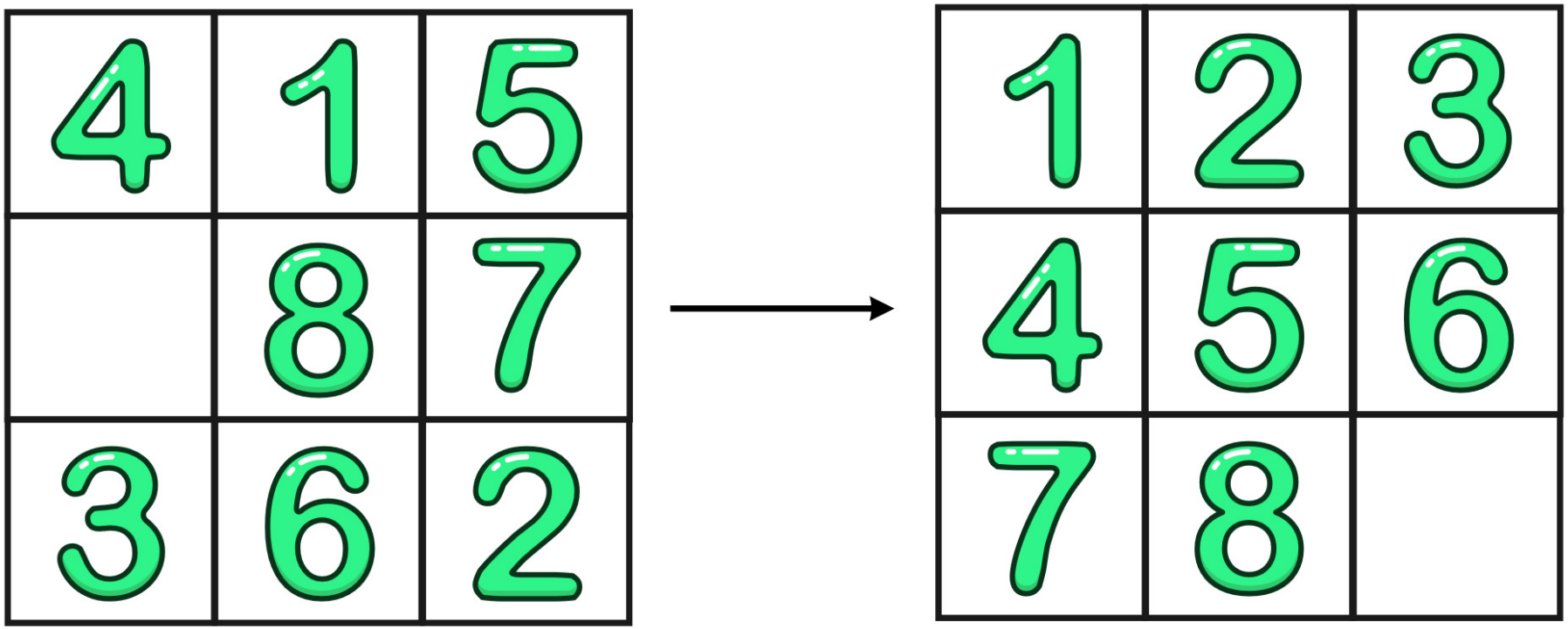
PDDL Example

Let's reuse the domain definition and solve another logistics problem.

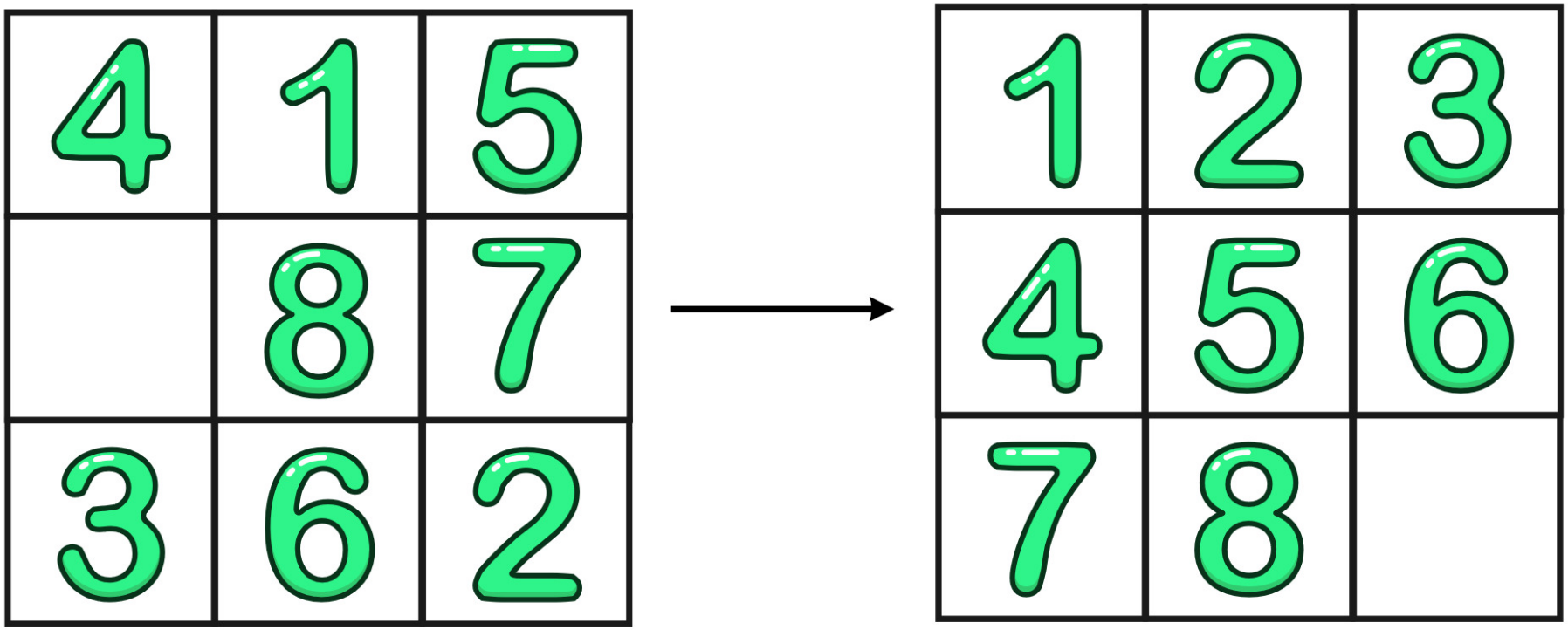


PDDL Example

Sliding puzzle example



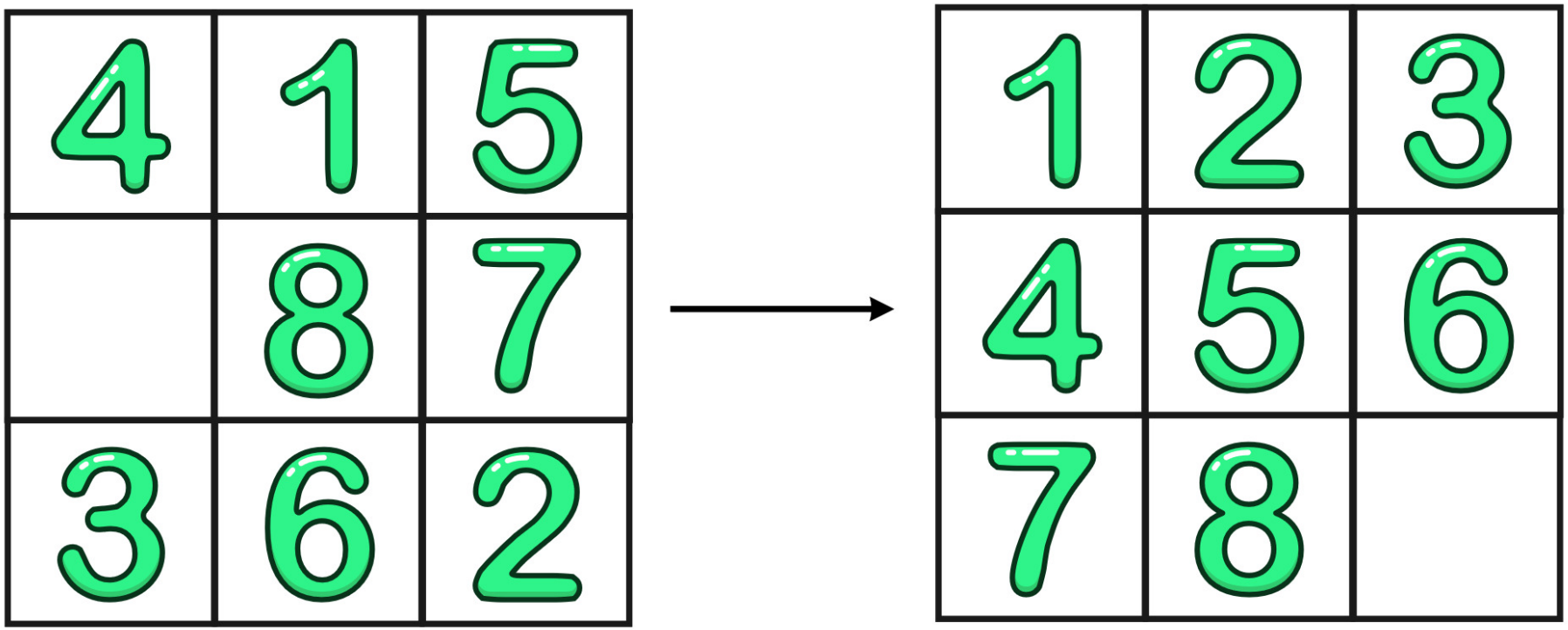
PDDL Example



Domain definition

- *types:*
- *predicates:*
- *actions:*

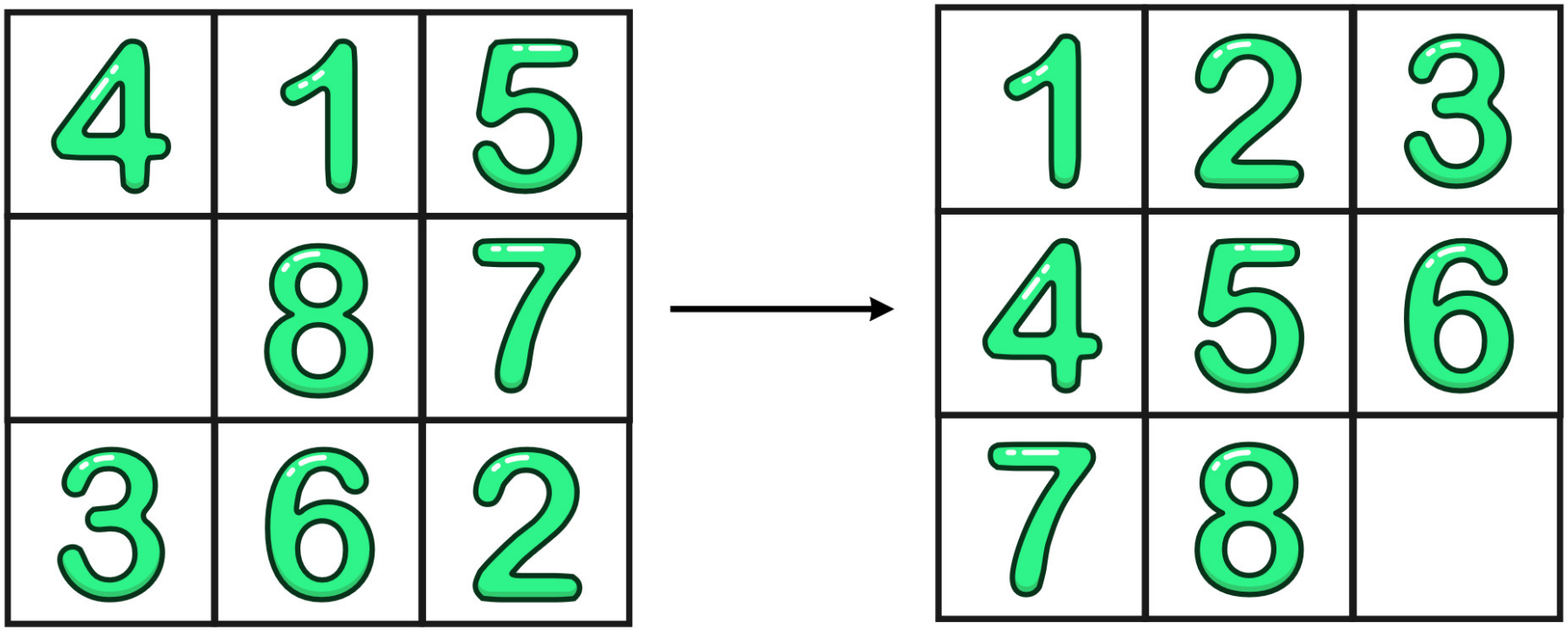
PDDL Example



Domain definition

- *types:* tiles and numbers
- *predicates:*
- *actions:*

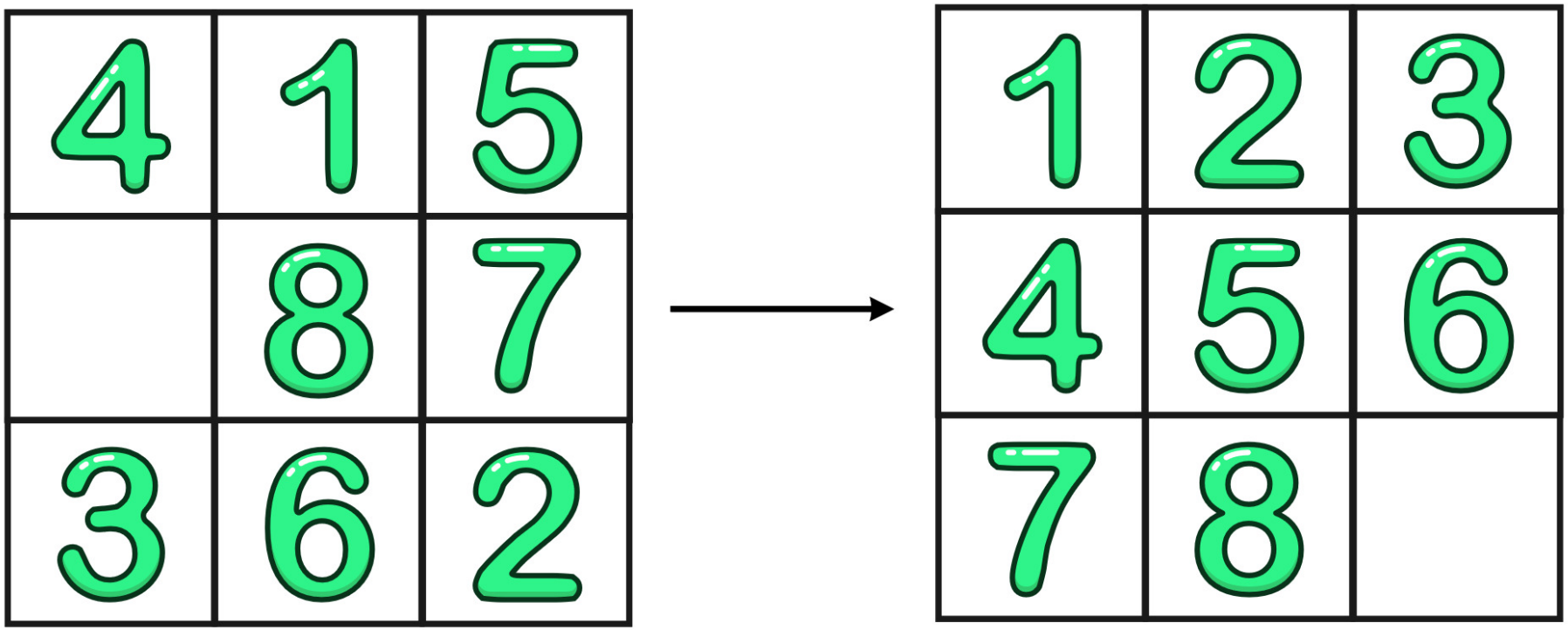
PDDL Example



Domain definition

- *types:* tiles and numbers
- *predicates:* connection between tiles, location of numbers, ...
- *actions:*

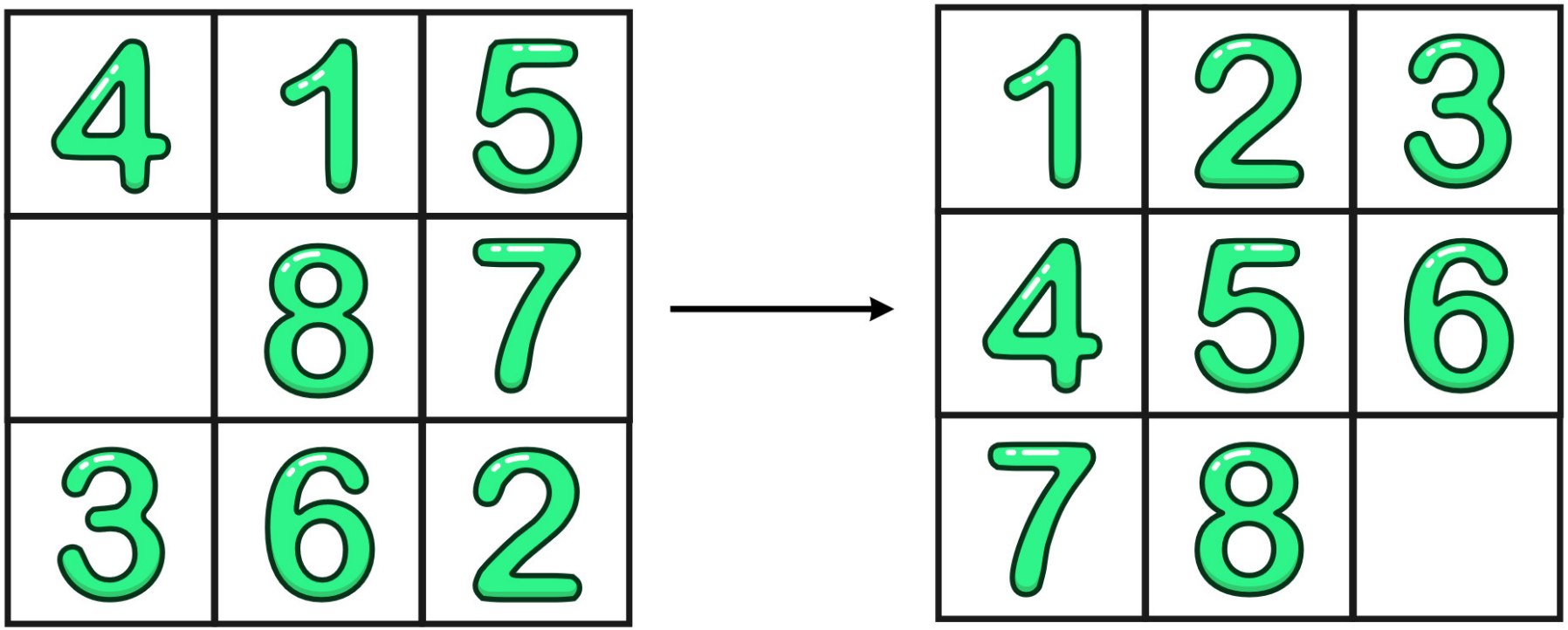
PDDL Example



Domain definition

- *types:* tiles and numbers
- *predicates:* connection between tiles, location of numbers, ...
- *actions:* slide tile

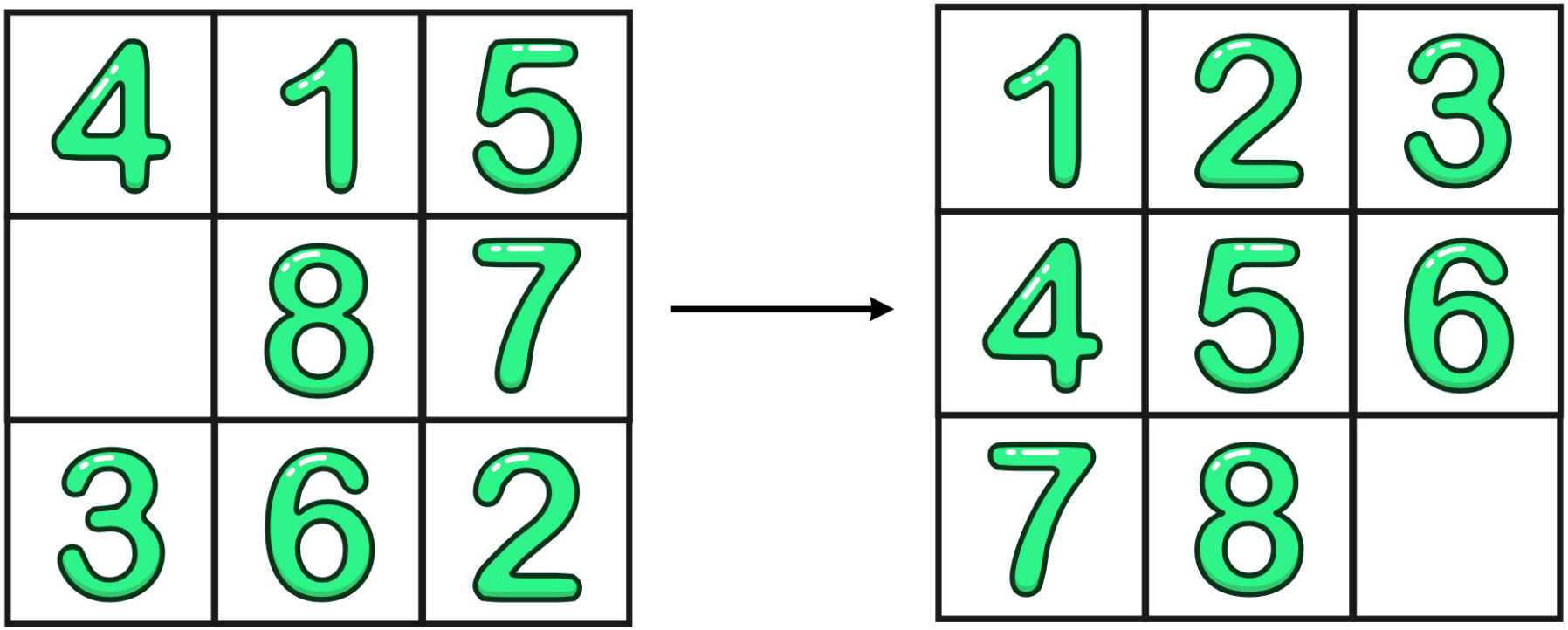
PDDL Example



Problem definition

- *objects:* number 4, number 1, ...
- *initial state:*
- *goal condition:*

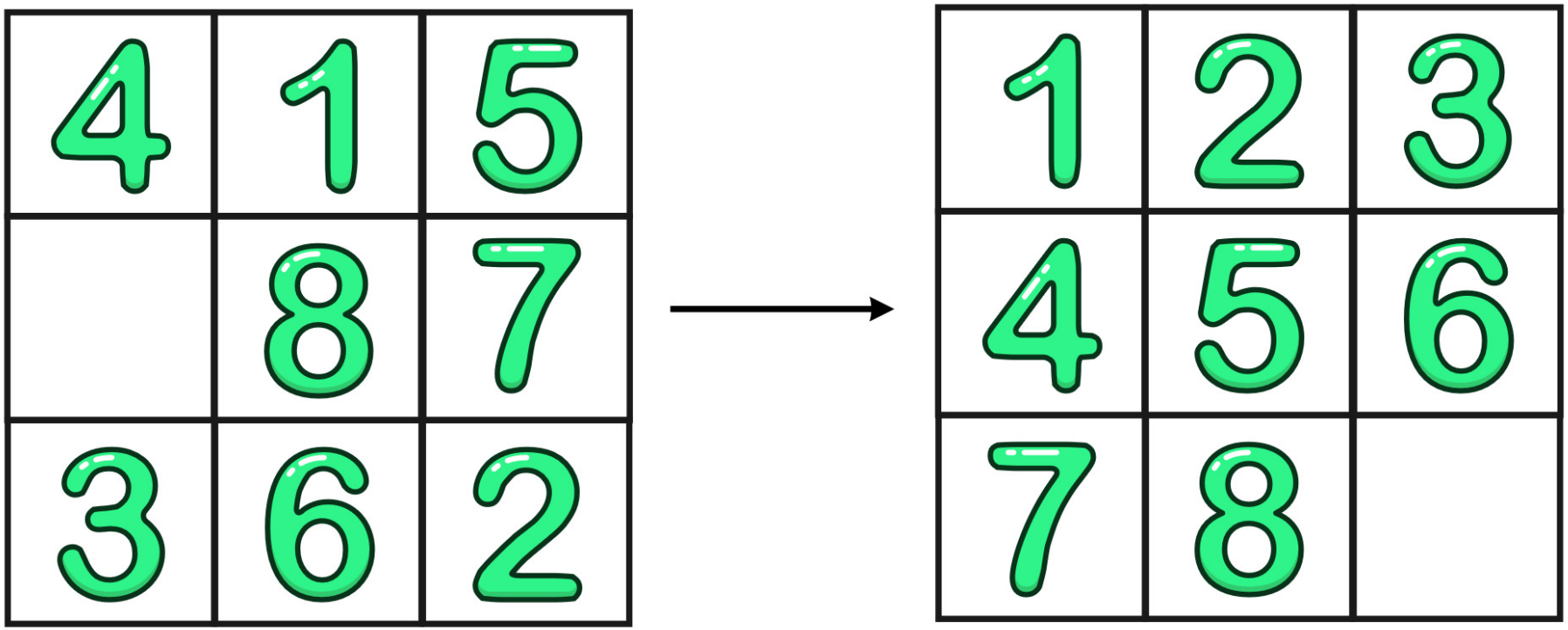
PDDL Example



Problem definition

- *objects:* number 4, number 1, ...
- *initial state:* number 4 at top left tile, ...
- *goal condition:*

PDDL Example



Problem definition

- *objects:* number 4, number 1, ...
- *initial state:* number 4 at top left tile, ...
- *goal condition:* number 1 at top left tile, bottom right tile empty, ...

PDDL Language

- Many other things possible in PDDL
  - negative preconditions
    (not (at ?p ?loc))
  - conditional effects
    (when CONDITION EFFECT)
  - universally quantified formula
    (forall (?a1 - type1 ?a2 - type2 ...) EFFECT)
  - existentially quantified formula
    (exists (?a1 - type1 ?a2 - type2 ...) EFFECT)
  - action costs
    (:functions (total-cost) - number); (increase (total-cost) 5) in effects

Assignment #1-1 - PDDL

Links to information on CW

- General information about the whole Assignment #1 here
- Information about Assignment #1-1 - PDDL here

Assignment #1-1 - PDDL

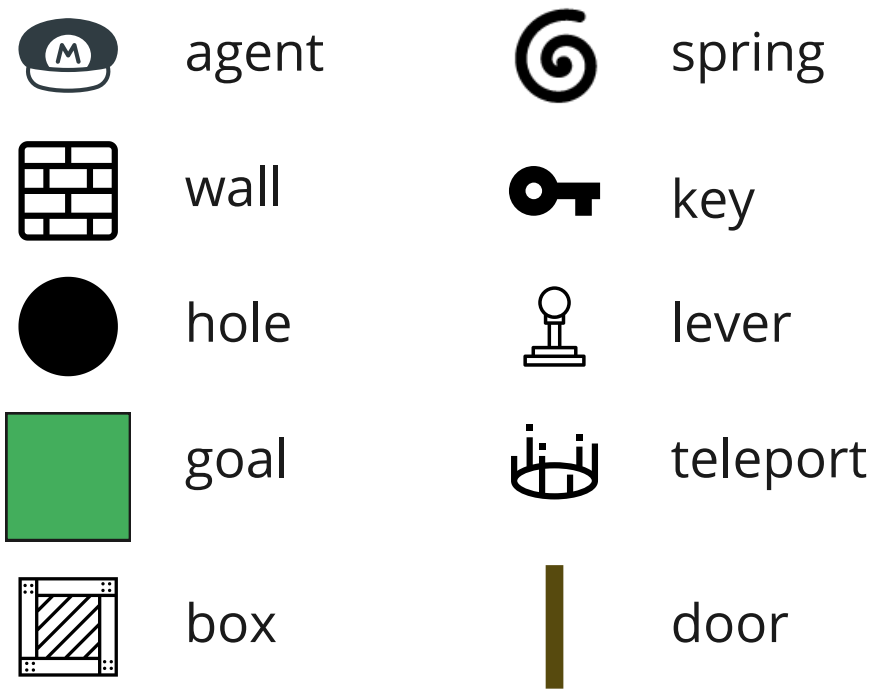
- First part of the Assignment #1
- **Task:** model given problem domain and problems in PDDL based on given descriptions and images
- **Points:** maximum 5
- **Deadlines**
  - 6.3.2023 - 23:59 (Monday)
  - 8.3.2023 - 23:59 (Wednesday)

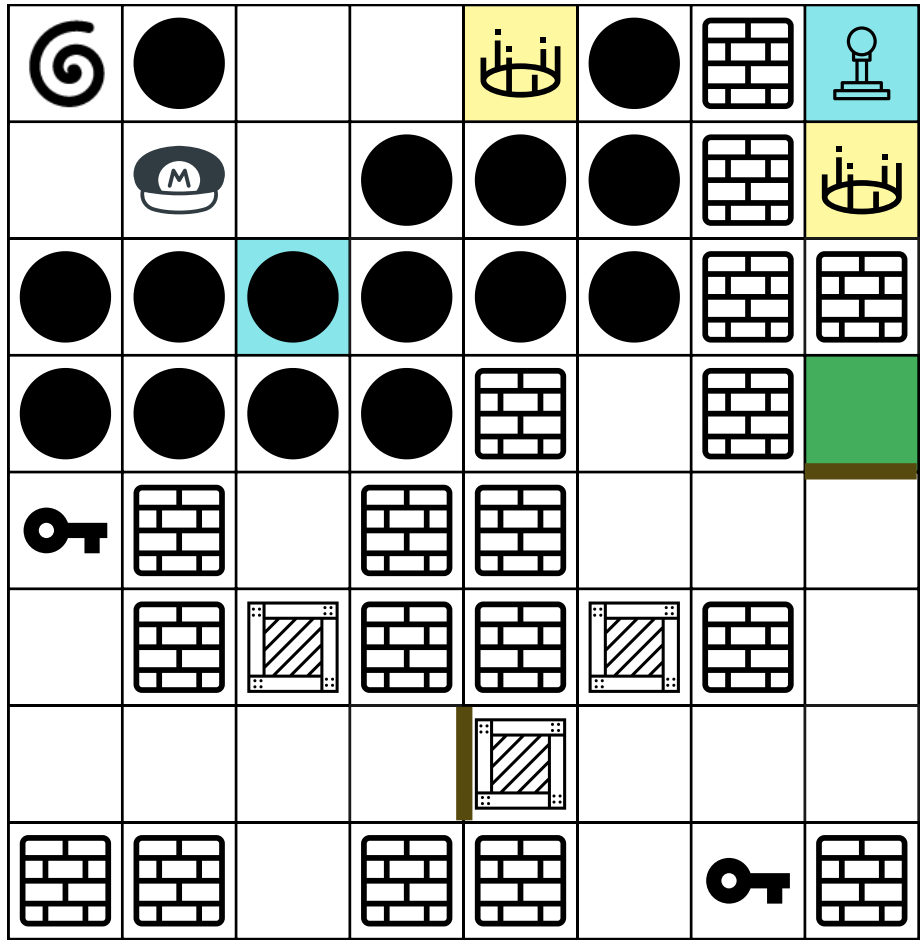
Assignment #1-1 - PDDL

Grid-Mario

- Grid-based game with one agent
- Mechanics
  - Movement on the grid (4-neighborhood)
  - Pushing boxes
  - Using teleports
  - Pulling levers
  - Unlocking doors
  - Jumping on springs
- Everyone models the movement + 2 mechanics

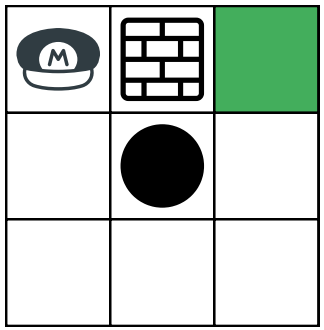
Grid-Mario



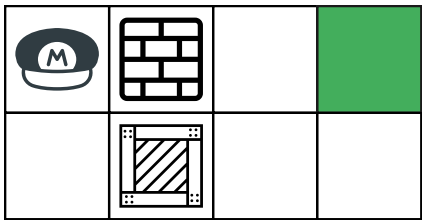


Grid-Mario

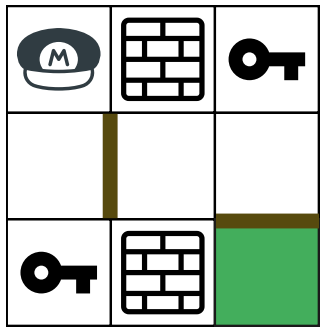
Movement



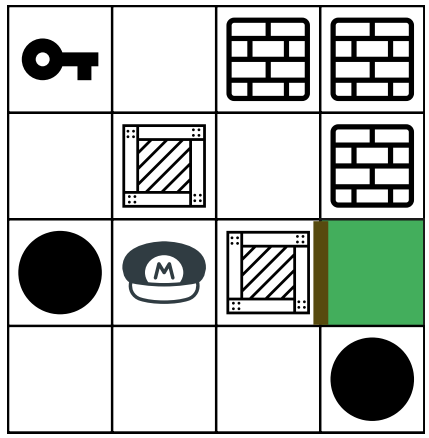
Boxes



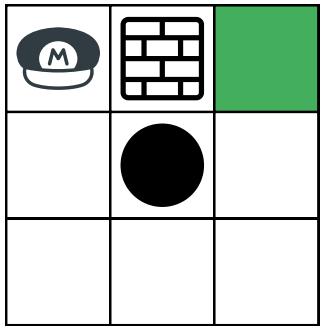
Keys



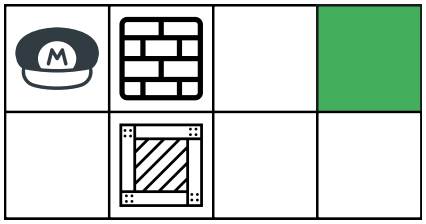
Boxes + keys



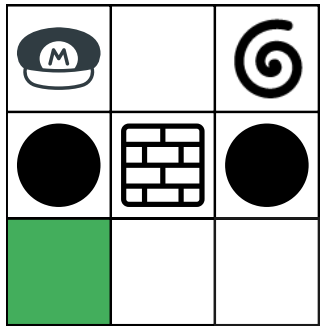
Movement



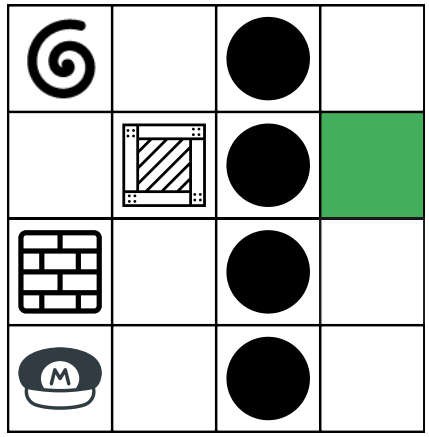
Boxes



Springs

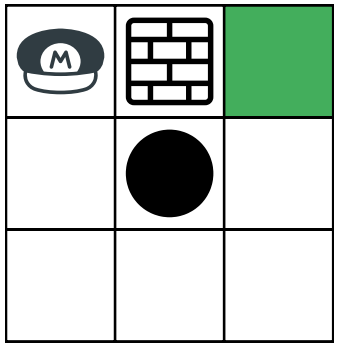


Boxes + springs

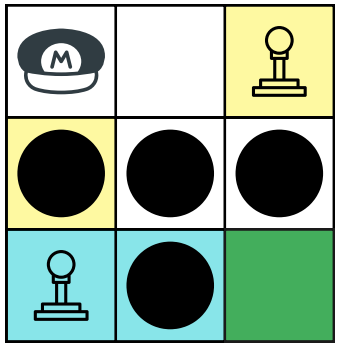


Grid-Mario

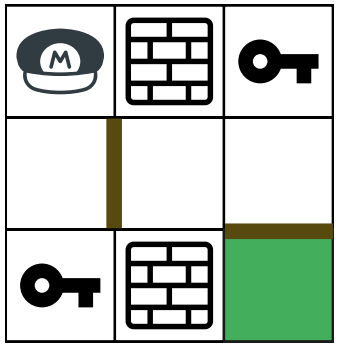
Movement



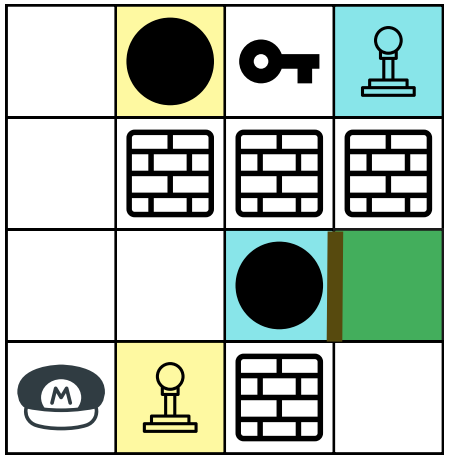
Levers



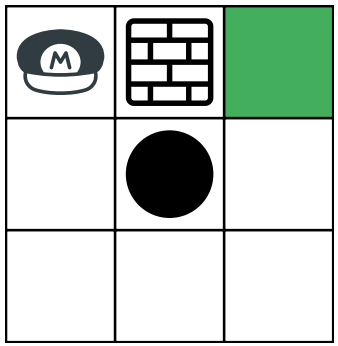
Keys



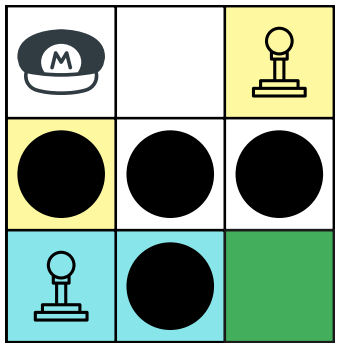
Levers + keys



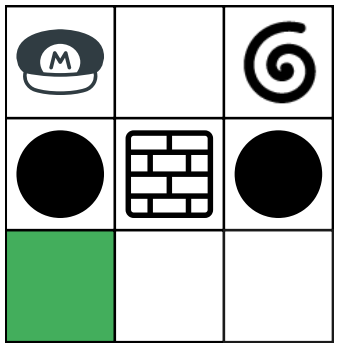
Movement



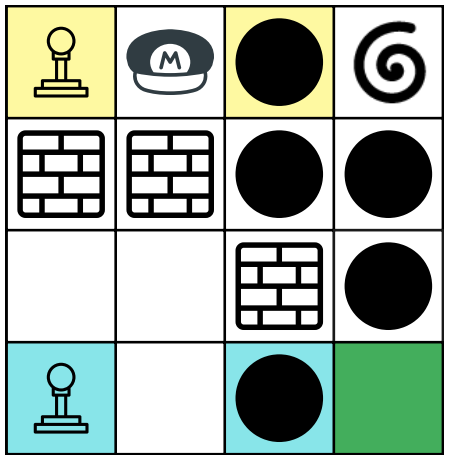
Levers



Springs

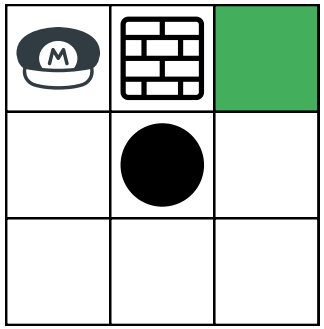


Levers + springs

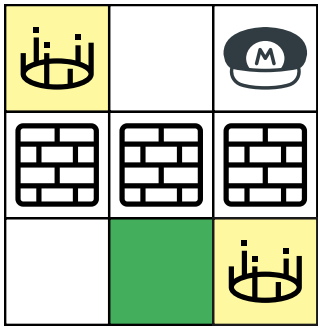


Grid-Mario

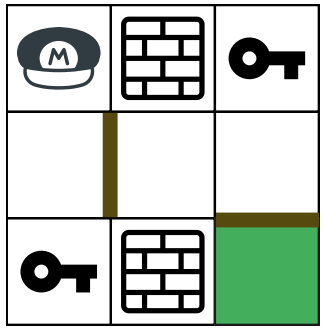
Movement



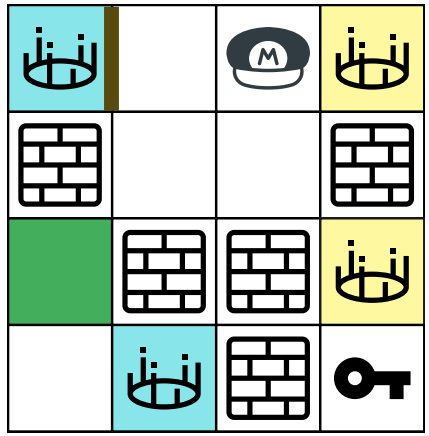
Teleports



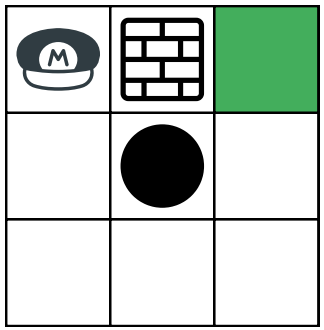
Keys



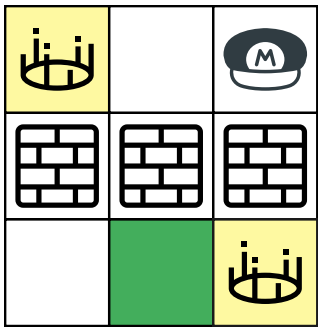
Teleports + keys



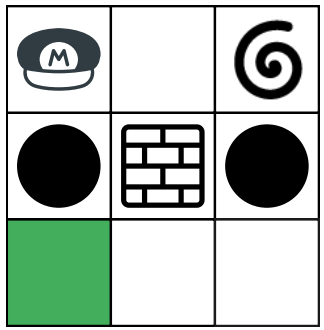
Movement



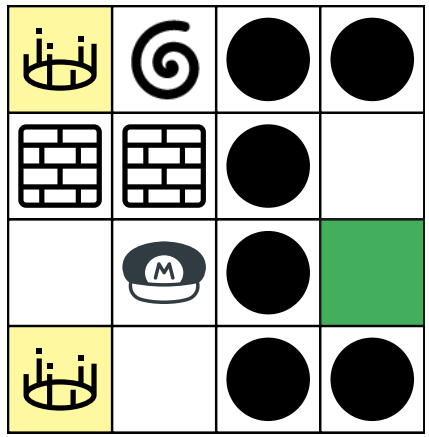
Teleports



Springs



Teleports + springs



Assignment #1-1 - PDDL

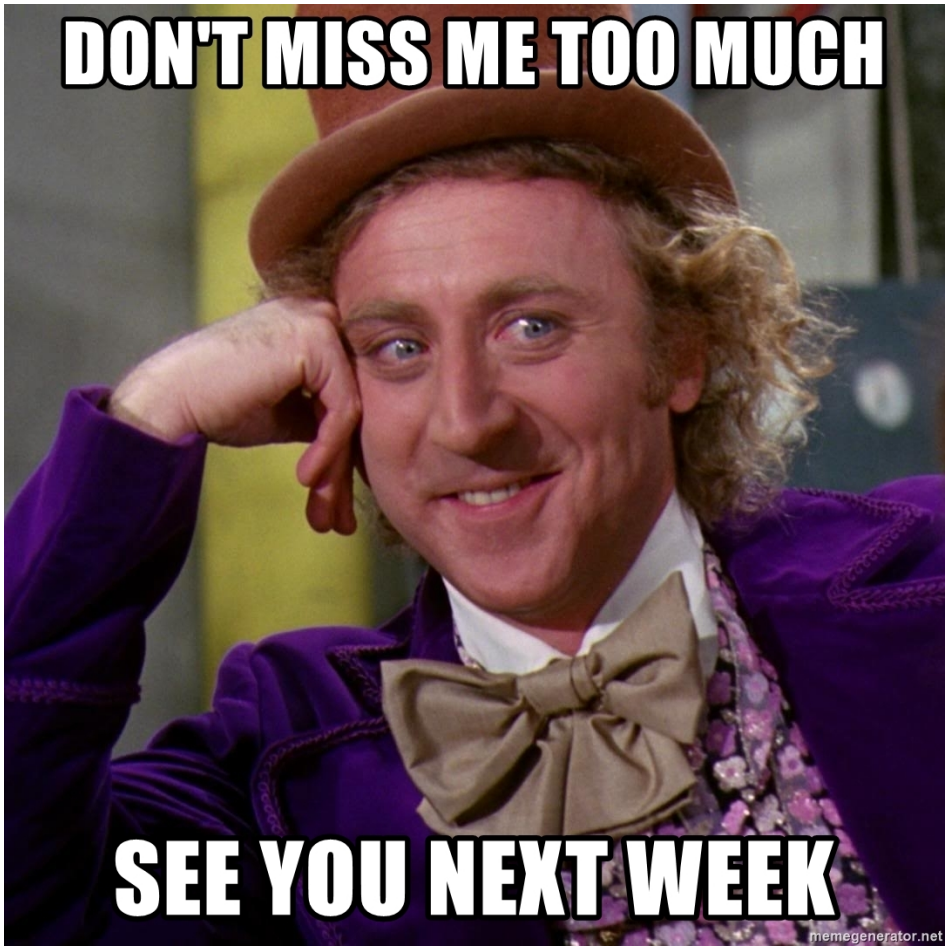
Submission

- No automatic evaluation
- Upload .zip archive to BRUTE with all PDDL files
  - domain.pddl - modeled domain
  - p01.pddl - problem instance that tests movement
  - p02.pddl - second problem instance (lever / box / teleport)
  - p03.pddl - third problem instance (keys / springs)
  - p04.pddl - last given problem instance testing all modeled mechanics
- Make sure all problems are solvable by the online editor/solver
- If anything seems unclear, please contact me

Recap

- You know motivation behind planning
- You are able to model problem and domains in PDDL
- You are able to read PDDL
- Assignment #1-1 - PDDL is assigned

The End



Feedback form