RE - Final Presentation Report

Leveraging classical planners to empower large language models with optimal planning capabilities.

Git Hub Repository Link:

https://github.com/jainavsanghvi10/LLM-Common-Sense-and-Planning/

Project Introduction

- In the context of classical planning, traditional approaches seem to dominate against modern AI, statistical machine learning-based approaches.
- Effective planning involves two key aspects:
    - acquiring the necessary planning domain knowledge and

    - assembling that knowledge into executable plans.
- LLMs serve as a valuable tool for knowledge extraction.
- This experiment explores the synergy between large language models (LLMs) and classical planners to enhance planning capabilities.
- For any classical planning problems, their exists algorithms and AI Planners which when given the problem in the desired formatted way guaranteed to produce correct plans.
- Explore the approach to connect LLMs to such tools

Project Description

- The project aims to bridge the gap between LLMs and classical planners by combining them. We want to incorporates the strengths of classical planners into LLMs.
- LLMs are bad at planning (or long-horizon reasoning) but they are good at describing and translating textual inputs, including re-writing planning prompts in the PDDL format. (view PDDL as a different language than English, hence just a machine translation task)
- The main feature that motivates us to use such classical planning systems is that most of these planners are sound and complete, meaning that they are guaranteed to be logically correct and will output a plan if one exists.
- The model will takes in a natural language description of a planning problem, then returns a correct (or optimal) plan for solving that problem.
- This involves extracting planning knowledge from LLMs and utilizing classical planners to find optimal plans which solve the given task.

By providing the model with both a natural language problem description and its corresponding problem PDDL, LLMs learns to infer the problem PDDL file for a given problem. This enables the model to perform unseen downstream tasks without the need for parameter fine-tuning, showcasing its versatility in handling diverse problem-solving scenarios.

Project Objectives

- Using LLMs to translate the natural-language problem context and goal into a format called PDDL (planning domain definition language), then feeding the result into a separate program/AI planners that generates a plan based on the context and goal.
- With that in mind, I am interested in testing here is **how well LLMs can translate the natural-language prompt into PDDL**. Evaluated on the basis of whether the generated PDDL can actually solve the problem and how long the resulting solution takes.

Benchmark Problems:

BLOCKSWORLD: Given a set of piles of blocks on a table, a robot is tasked with rearranging them into a specified target configuration.

Design of the Experiment

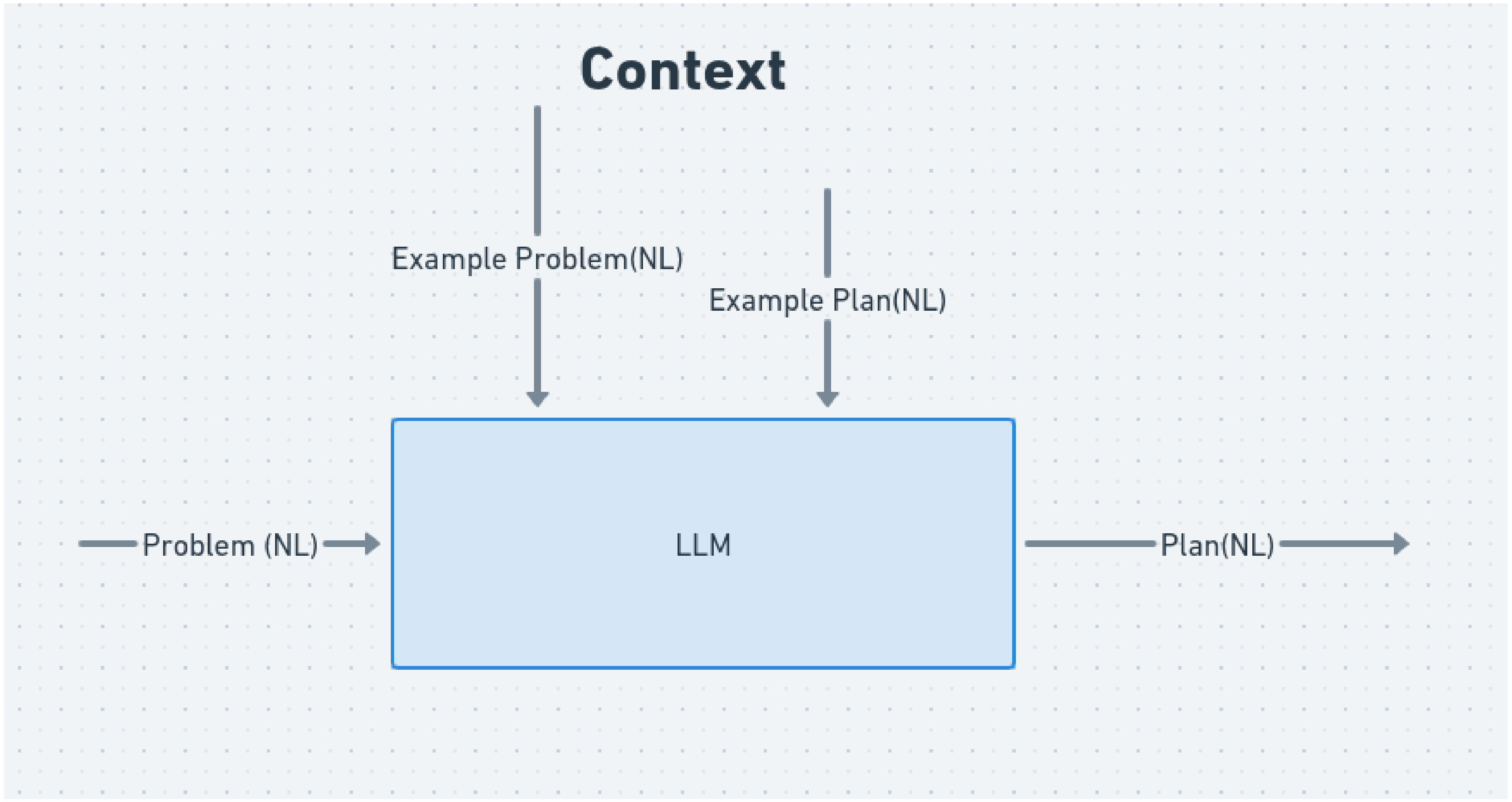
1. **Knowledge Acquisition:** Utilize LLMs to translate natural language descriptions into PDDL format.
2. **Reasoning/Planning:** Employ classical planners to generate solutions based on the acquired knowledge.

Methodology:

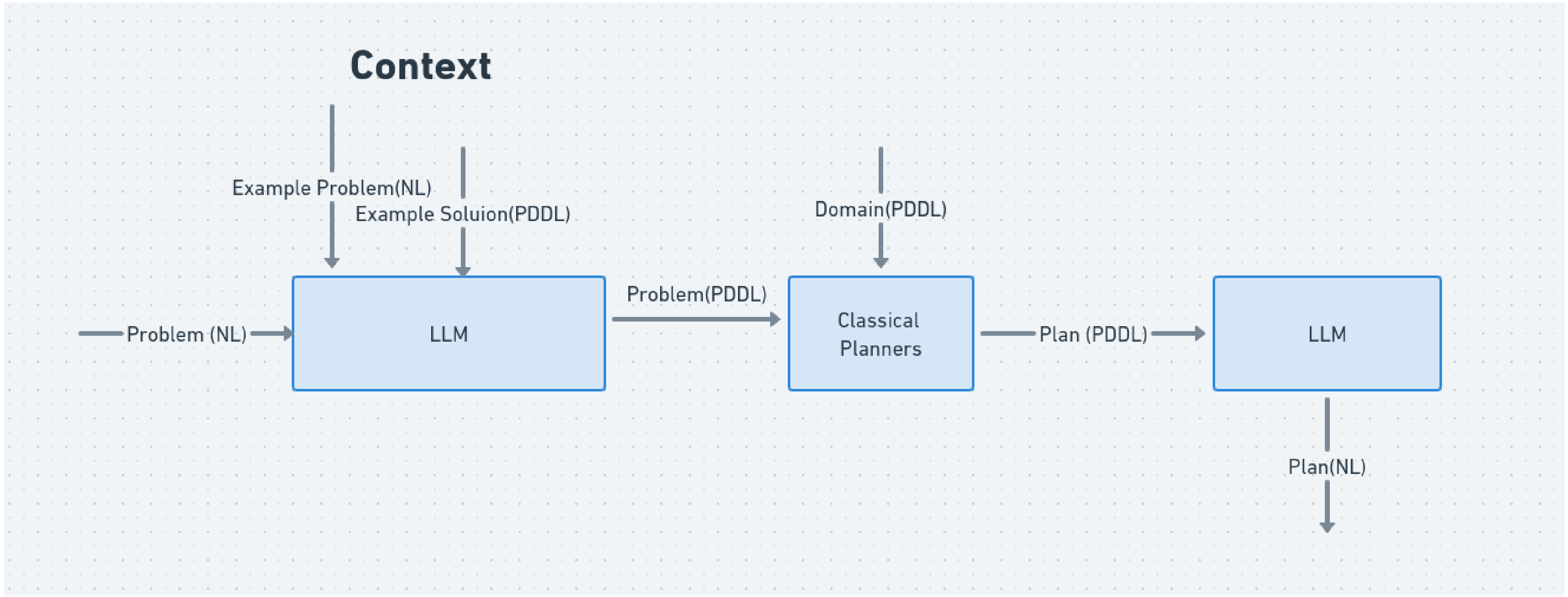
- Provide a natural language description of a planning problem to the LLM
- Outputs a problem description suitable as input to a AI planner/program
- Use the PDDL problem description to solves the problem using the AI planner/program, and
- Converts the output of the planner back to natural language

Total Experiments - 4

- **LLM (*w/o Context*)** : *directly ask the LLM for plan*
- **LLM (*with Context*)** : *directly ask the LLM for plan*

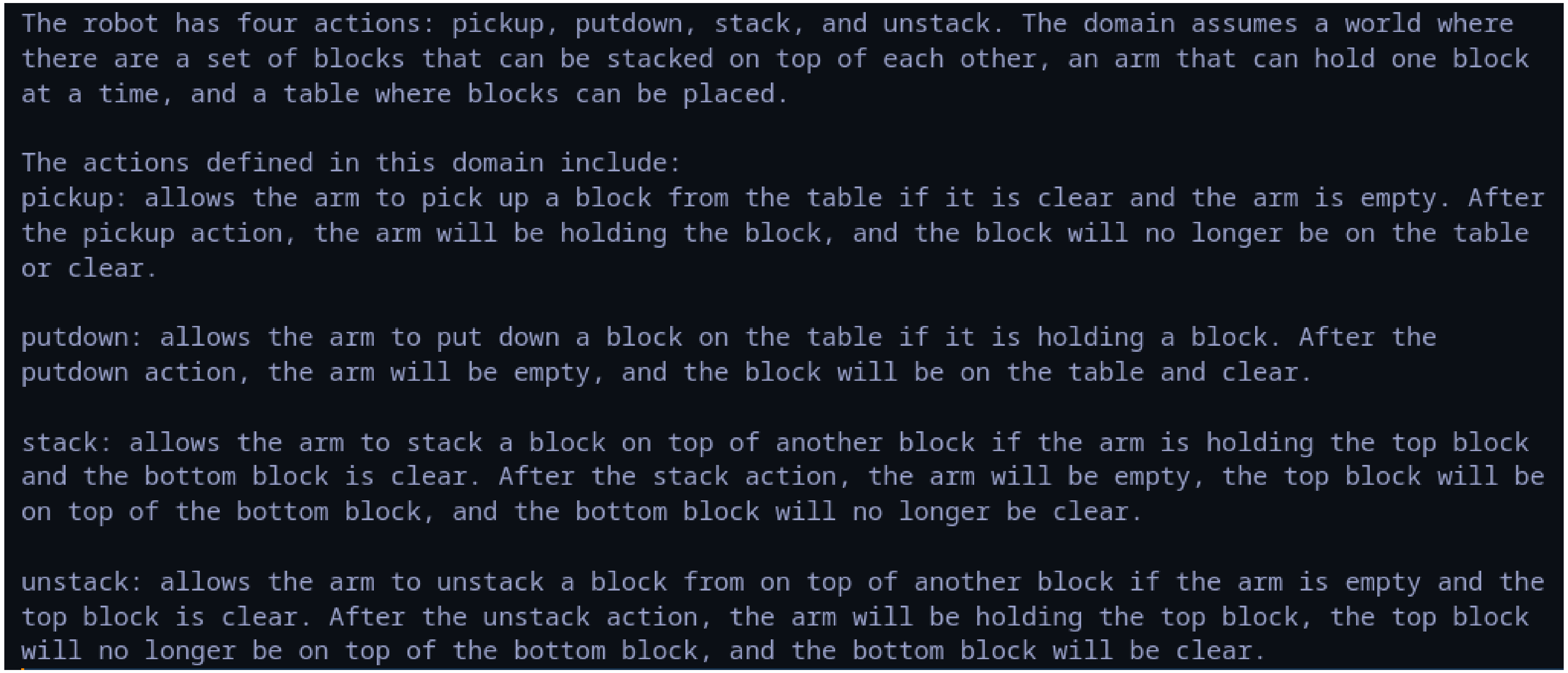


- **LLM + Planner (*w/o Context*)** : *no context, create the problem PDDL*
- **LLM + Planner (*with Context*)** : *create the problem PDDL given the context*



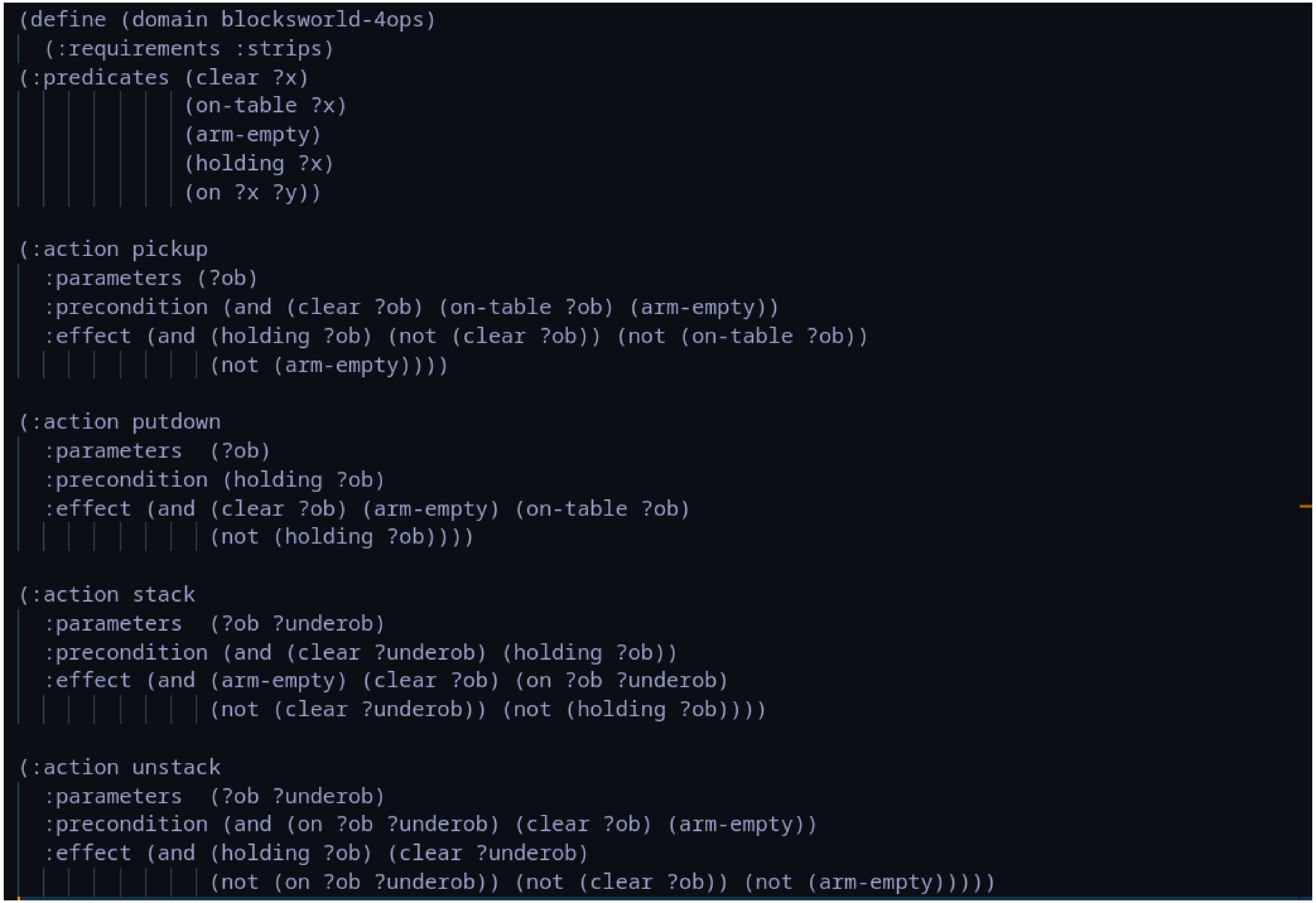
Domain:

▼ **Domain (Natural Language)**



```
The robot has four actions: pickup, putdown, stack, and unstack. The domain assumes a world where
there are a set of blocks that can be stacked on top of each other, an arm that can hold one block
at a time, and a table where blocks can be placed.

The actions defined in this domain include:
pickup: allows the arm to pick up a block from the table if it is clear and the arm is empty. After
the pickup action, the arm will be holding the block, and the block will no longer be on the table
or clear.

putdown: allows the arm to put down a block on the table if it is holding a block. After the
putdown action, the arm will be empty, and the block will be on the table and clear.

stack: allows the arm to stack a block on top of another block if the arm is holding the top block
and the bottom block is clear. After the stack action, the arm will be empty, the top block will be
on top of the bottom block, and the bottom block will no longer be clear.

unstack: allows the arm to unstack a block from on top of another block if the arm is empty and the
top block is clear. After the unstack action, the arm will be holding the top block, the top block
will no longer be on top of the bottom block, and the bottom block will be clear.
```

▼ **Domain (PDDL Format)**



```
(define (domain blocksworld-4ops)
  (:requirements :strips)
(:predicates (clear ?x)
             (on-table ?x)
             (arm-empty)
             (holding ?x)
             (on ?x ?y))

(:action pickup
  :parameters (?ob)
  :precondition (and (clear ?ob) (on-table ?ob) (arm-empty))
  :effect (and (holding ?ob) (not (clear ?ob)) (not (on-table ?ob))
               (not (arm-empty))))

(:action putdown
  :parameters  (?ob)
  :precondition (holding ?ob)
  :effect (and (clear ?ob) (arm-empty) (on-table ?ob)
               (not (holding ?ob))))

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

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

1. **Pickup:**
    - Condition: The arm must be empty, and the block to be picked up should be clear (not stacked on or under another block).
    - Effect: The arm picks up the block from the table, and the block is no longer on the table or clear.
2. **Putdown:**
    - Condition: The arm must be holding a block.
    - Effect: The arm puts down the held block onto the table, making the arm empty. The block is now on the table and clear.
3. **Stack:**
    - Condition: The arm must be holding the top block, and the bottom block should be clear (not having any block on top of it).
    - Effect: The arm stacks the block it's holding on top of another block, making the arm empty. The top block is now on top of the bottom block, and the bottom block is no longer clear.
4. **Unstack:**
    - Condition: The arm must be empty, and the top block should be clear (not having any block on top of it).
    - Effect: The arm unstacks the top block from another block, making the arm hold the top block. The top block is no longer on top of the bottom block, and the bottom block is clear.

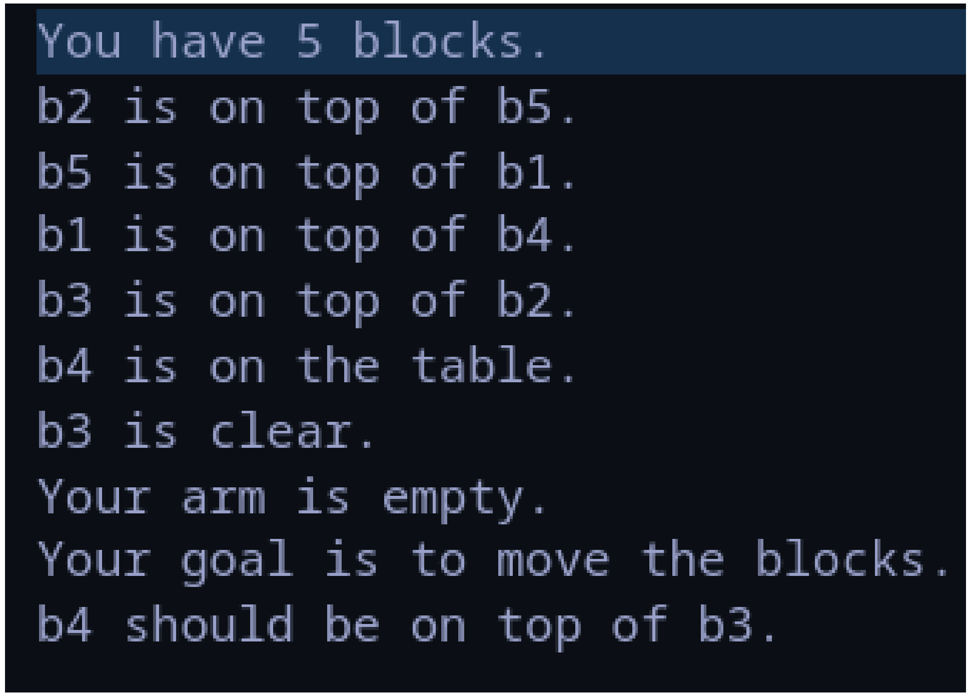
Context:

LLMs exhibit the capability of contextual learning, allowing them to handle new tasks without fine-tuning by relying on input-label pairs (demonstrations). GPT-3.5, when given problem descriptions in both natural language and PDDL as context, can directly solve the corresponding PDDL problem file with the included prompt.

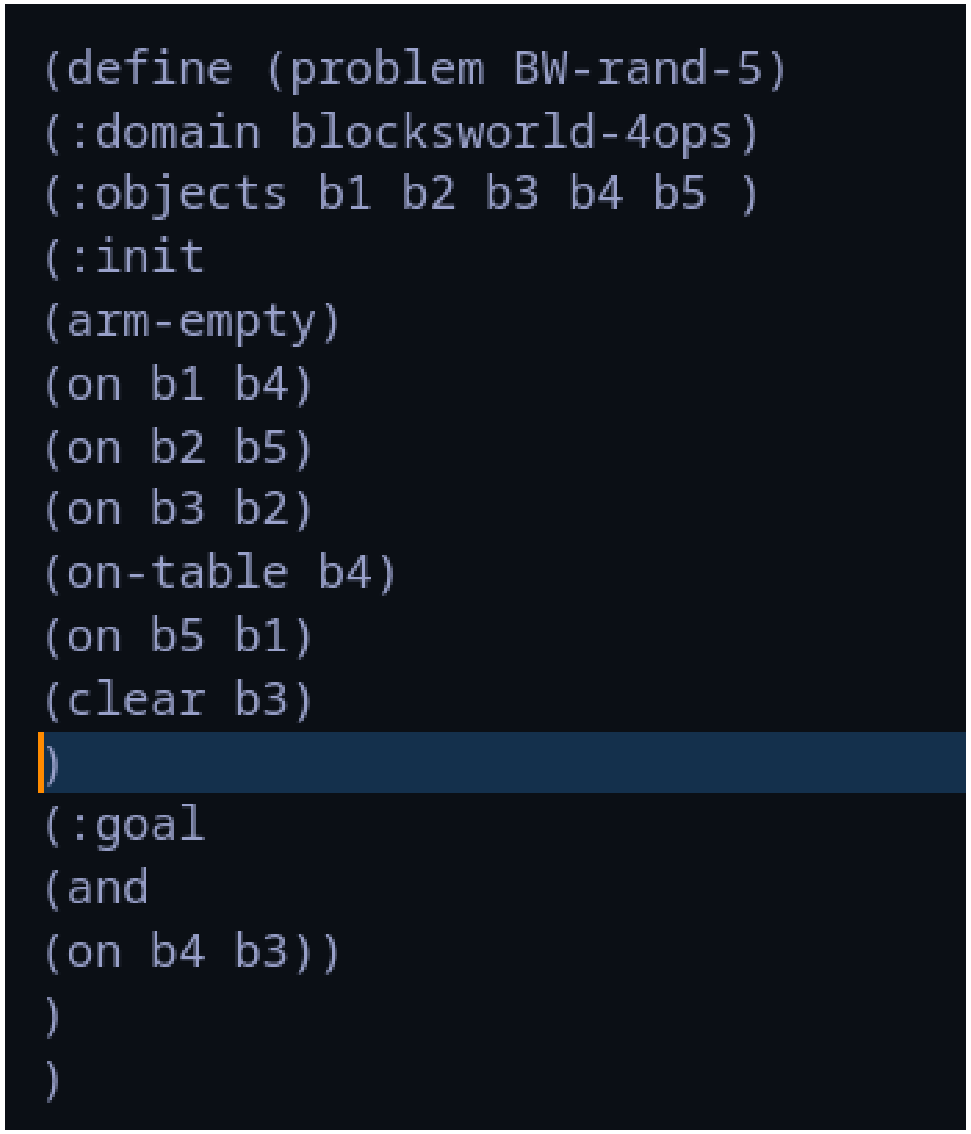
LLMs are known to be capable of in-context learning without finetuning their parameters. By in-context learning, we mean LLMs' ability to perform unseen downstream tasks by simply conditioning on a few input-label pairs(demonstrations).

In our case, a pair of problem descriptions in natural language and in PDDL (Solution in NL incase of LLM as Planner) provided to GPT-4 as context. When the context is included with the prompt from the example above, the resulting PDDL problem file is directly solvable by the planner.

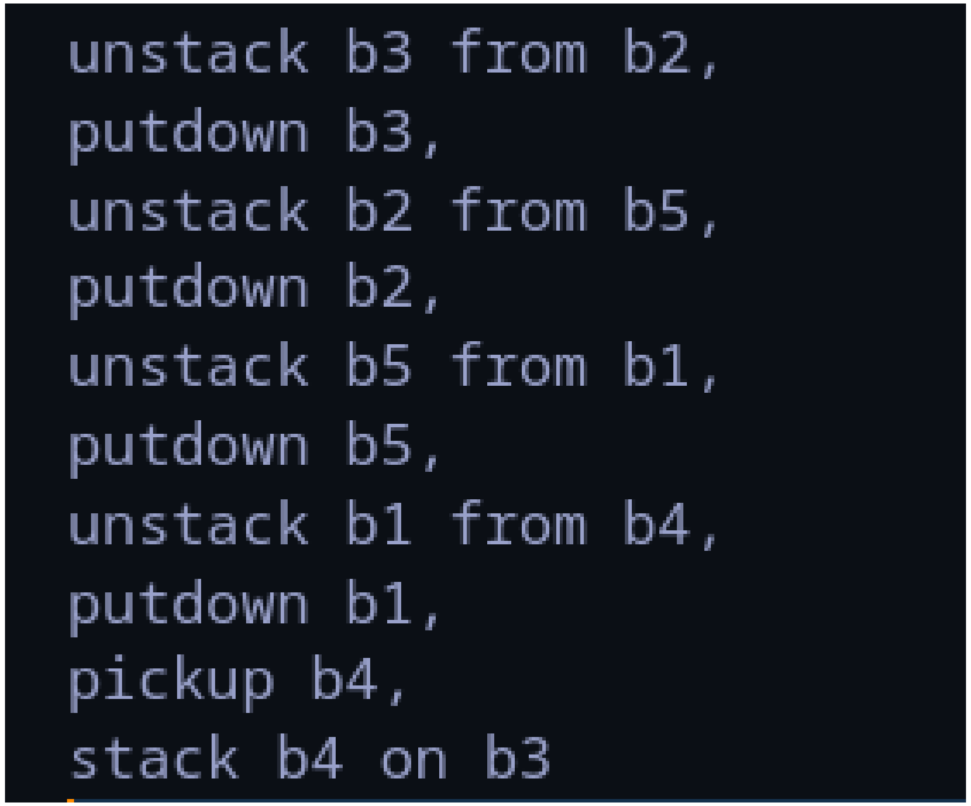
▼ **Example Problem**



```
You have 5 blocks.
b2 is on top of b5.
b5 is on top of b1.
b1 is on top of b4.
b3 is on top of b2.
b4 is on the table.
b3 is clear.
Your arm is empty.
Your goal is to move the blocks.
b4 should be on top of b3.
```



```
(define (problem BW-rand-5)
(:domain blocksworld-4ops)
(:objects b1 b2 b3 b4 b5 )
(:init
(arm-empty)
(on b1 b4)
(on b2 b5)
(on b3 b2)
(on-table b4)
(on b5 b1)
(clear b3)
)
(:goal
(and
(on b4 b3))
)
)
```



```
unstack b3 from b2,
putdown b3,
unstack b2 from b5,
putdown b2,
unstack b5 from b1,
putdown b5,
unstack b1 from b4,
putdown b1,
pickup b4,
stack b4 on b3
```

Edge Cases I have considered:

- Unstack b5 from b5 : For eg: P9
- Pick up b5 (when b5 is at the top of a stack of blocks) : For eg: P16

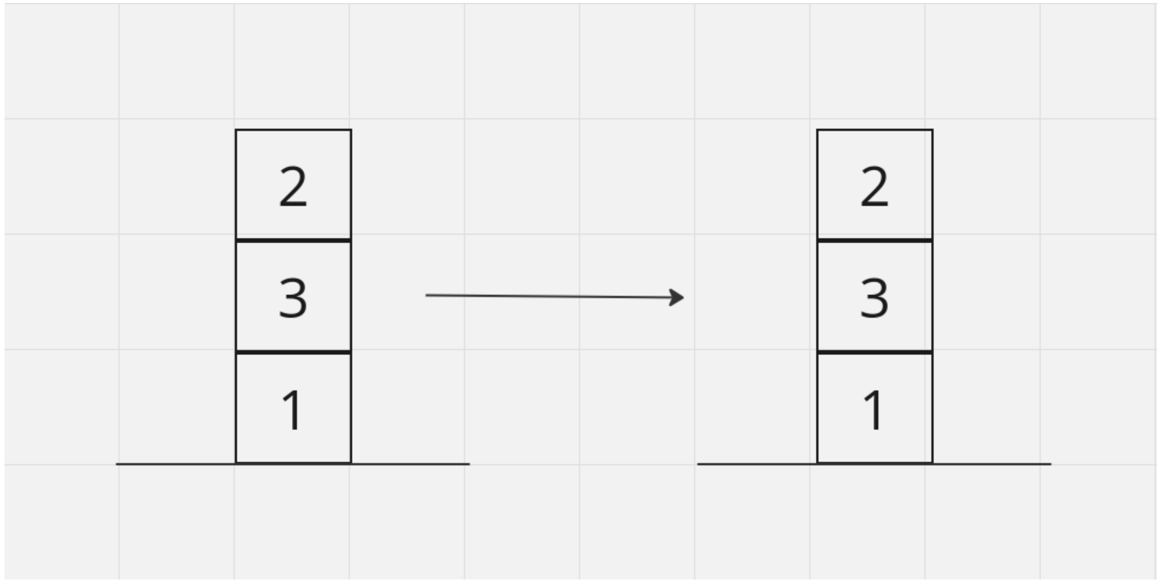
Results and Analysis Table:

Problem	LLM (*w/o Context*)	LLM (*with Context*)	LLM + Planner (*w/o Context*)	LLM + Planner (*with Context*)
1	❌	❌	✔️	✔️
2	❌	❌	❌	✔️
3	❌	❌	✔️	✔️
4	❌	❌	✔️	✔️
5	❌	❌	❌	✔️
6	❌	❌	❌	✔️
7	❌	❌	❌	✔️
8	❌	✔️	❌	❌
9	❌	❌	✔️	✔️
10	❌	❌	✔️	✔️
11	❌	❌	❌	✔️
12	❌	❌	❌	✔️
13	❌	❌	❌	❌
14	❌	❌	❌	✔️
15	❌	❌	❌	❌

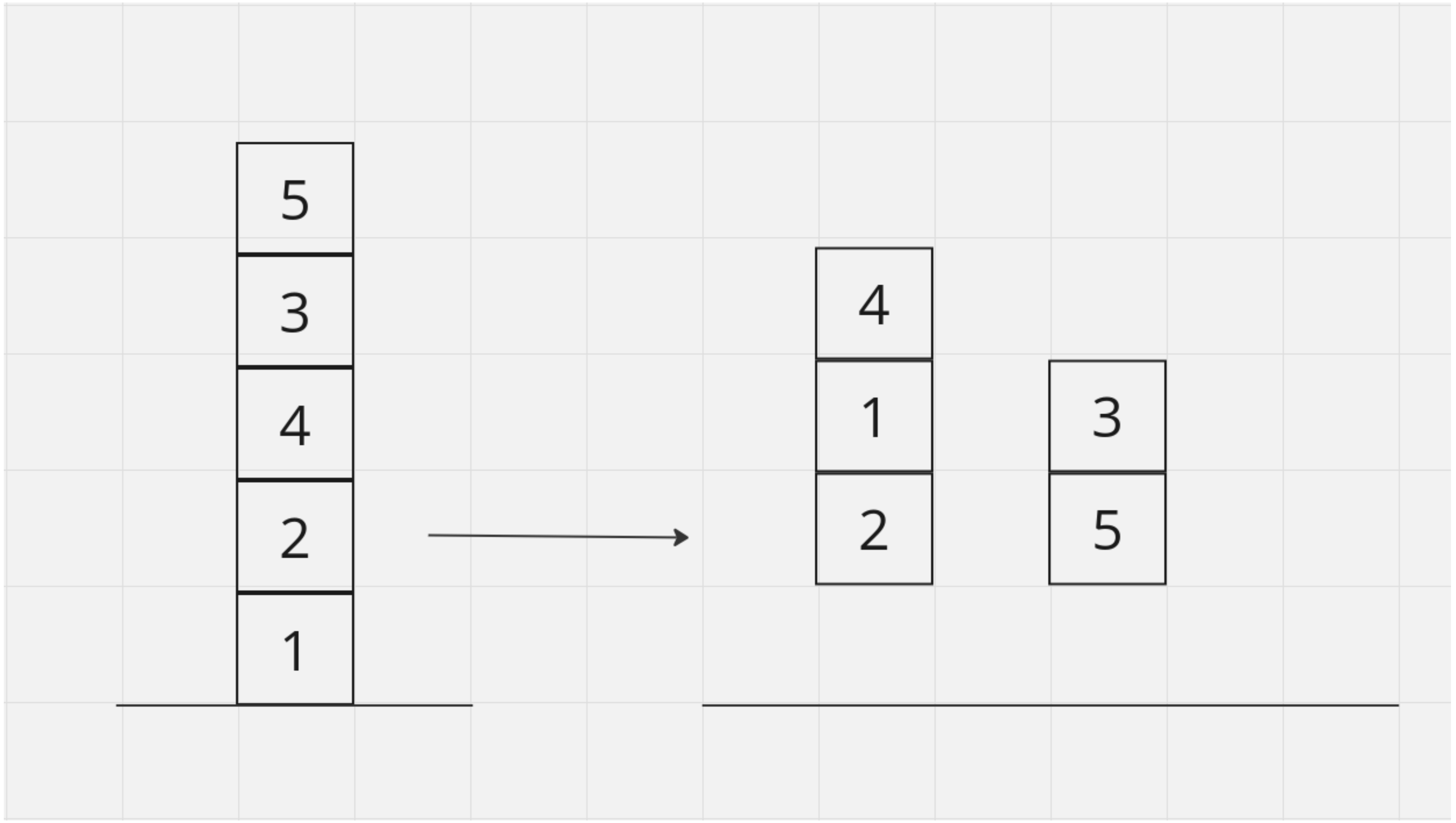
Problem	LLM (*w/o Context*)	LLM (*with Context*)	LLM + Planner (*w/o Context*)	LLM + Planner (*with Context*)
16	❌	❌	❌	✔️
17	❌	❌	❌	✔️
18	❌	❌	❌	✔️
19	❌	❌	❌	✔️
20	❌	❌	❌	✔️
Total Valid Plans (Success Rate)	0 (0%)	1 (5%)	6 (30%)	17 (85%)

Some Examples:

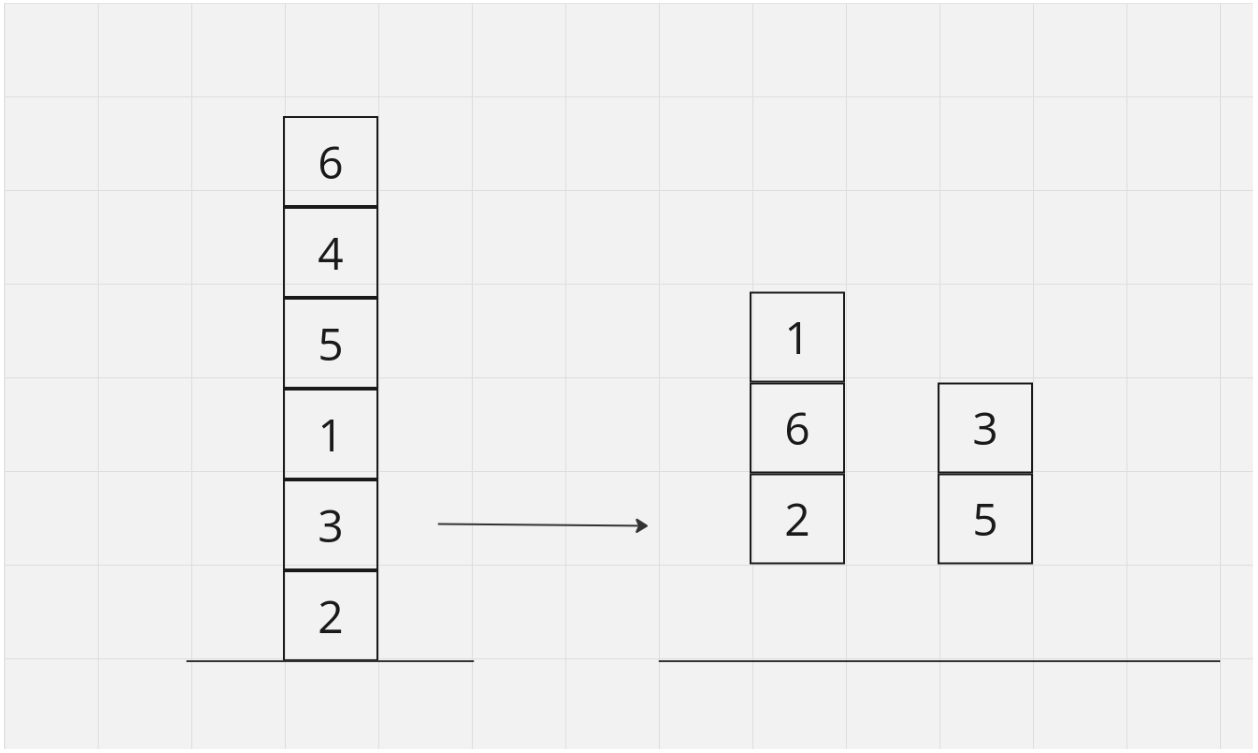
1



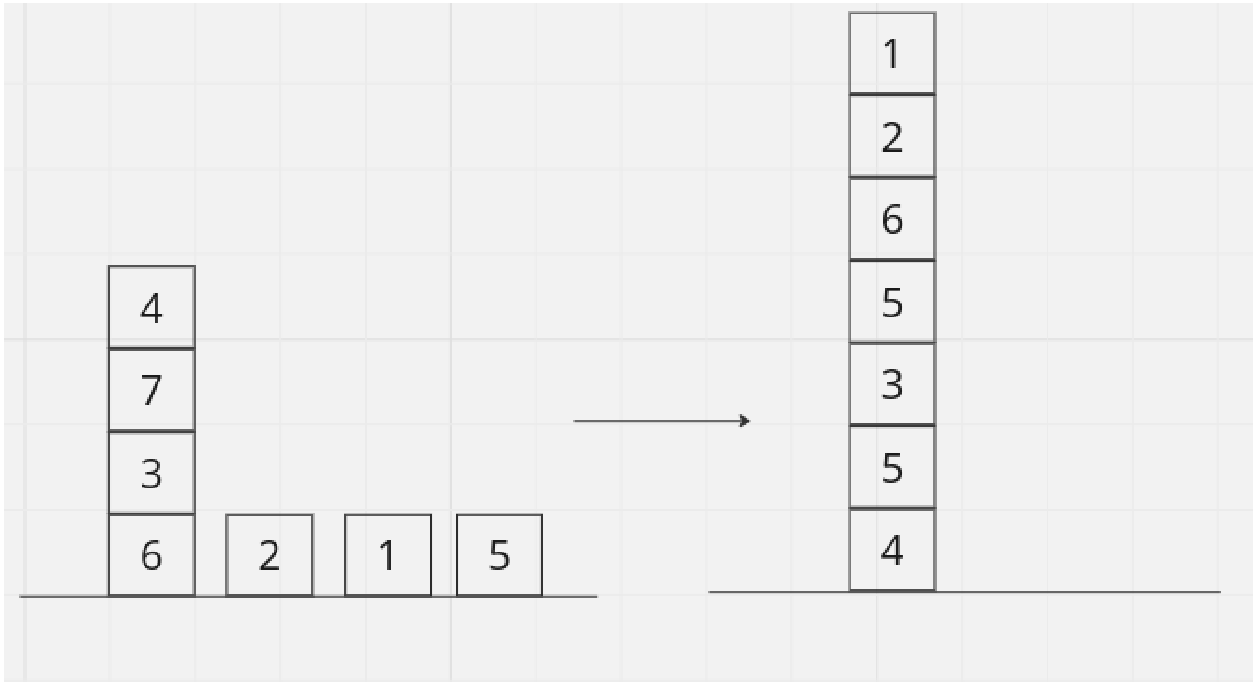
▼ **6**



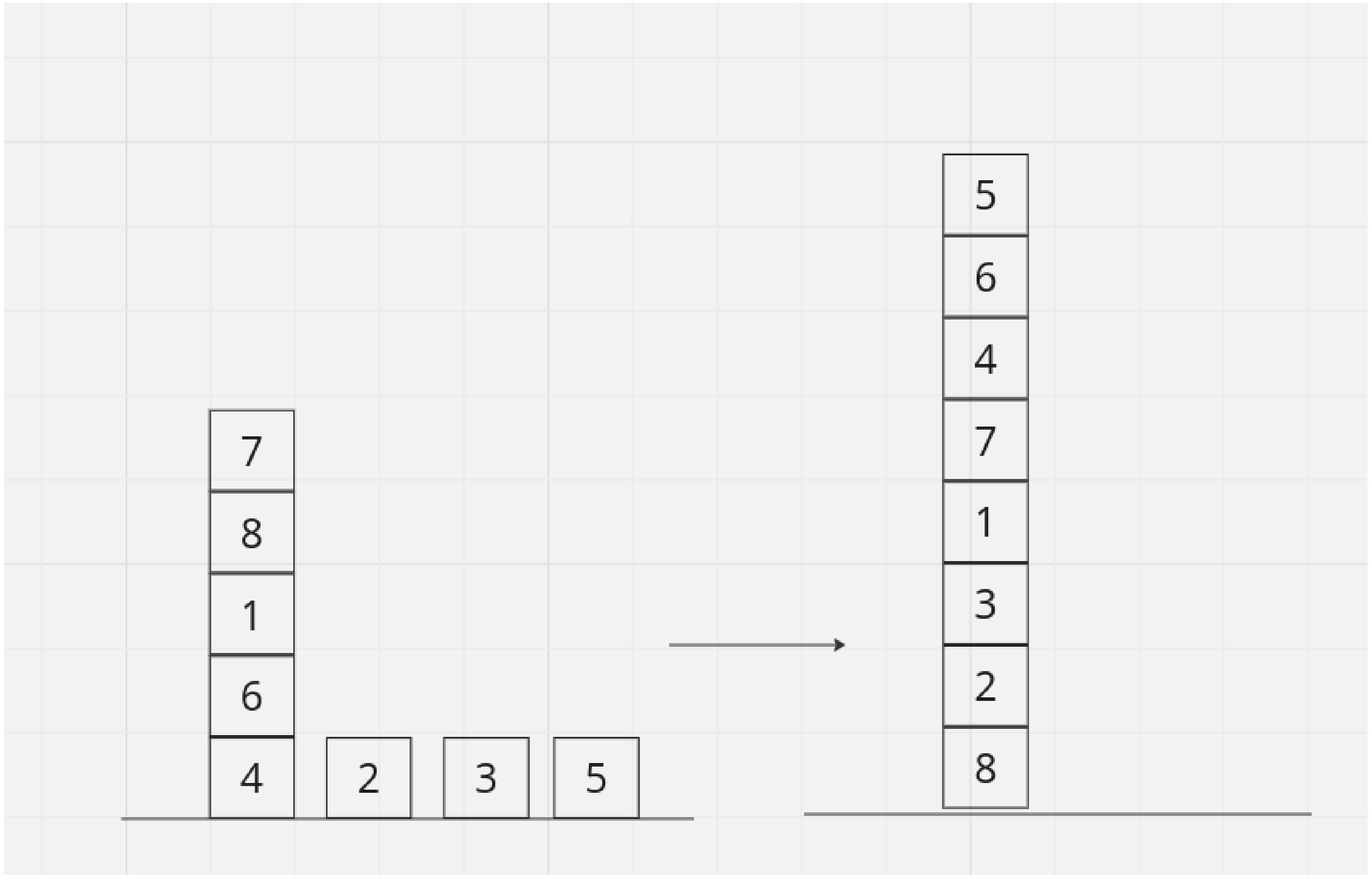
▼ **8**



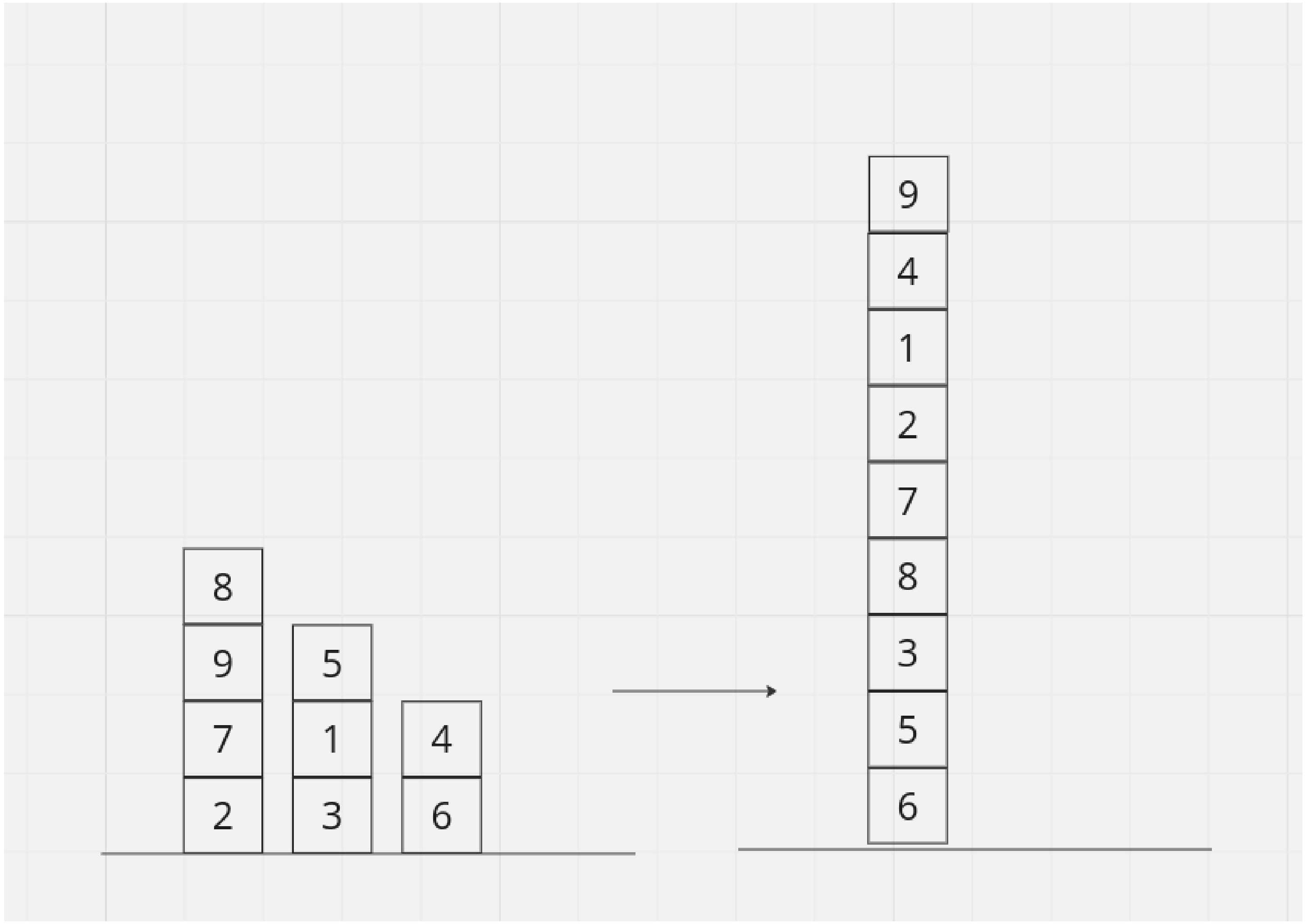
▼ **9**



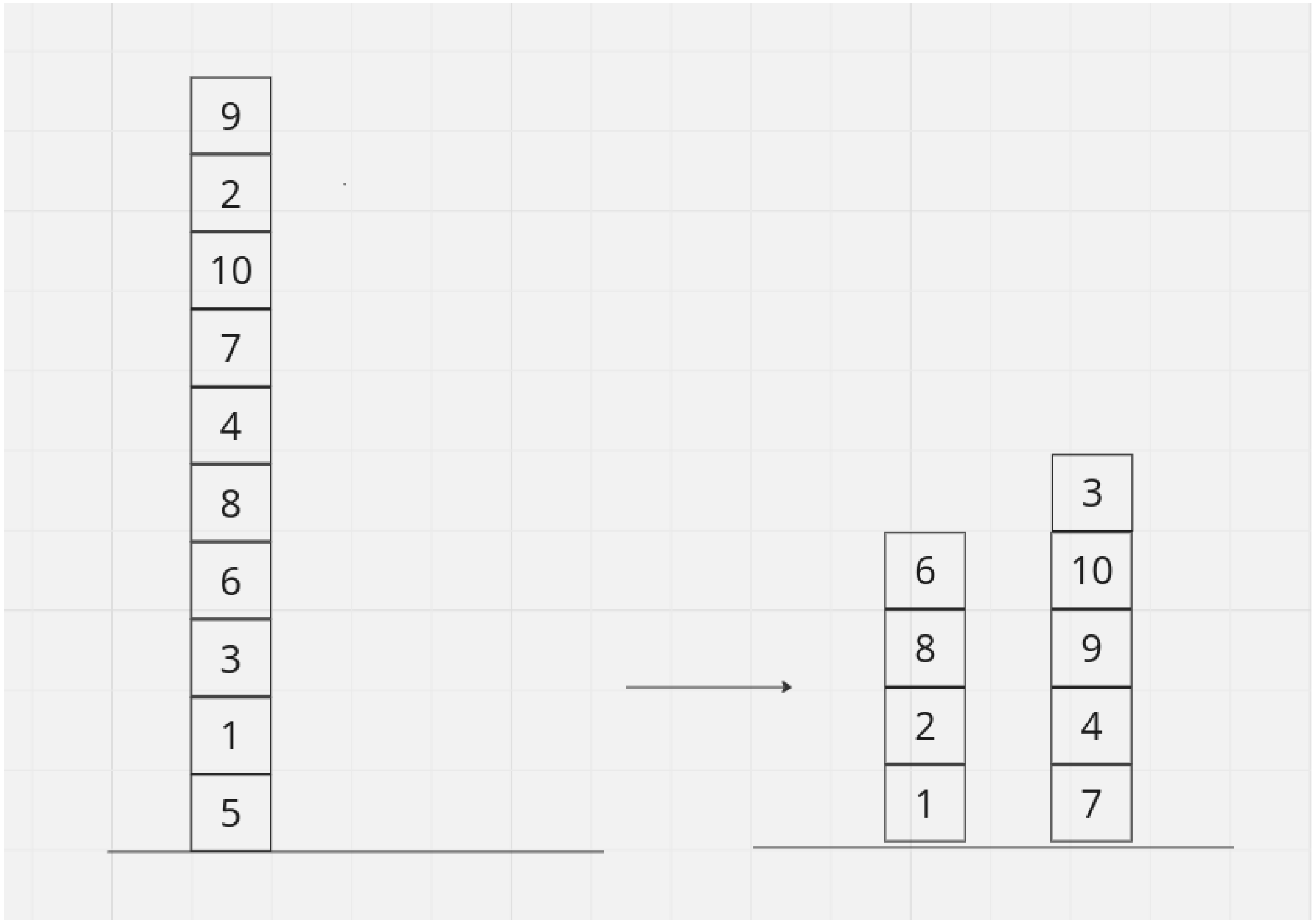
▼ 12



▼ **14**



▼ 15


9
2
10
7
4
8
6
3
1
5
6
8
2
1
3
10
9
4
7

Results:

- Even though LLM-AS-Planner provides a plan in natural language for every problem, most of these plans are not feasible. The main reason is that LLM AS-Planner lacks the ability to reason about precondition.
- The LLM+Planner combination produces an optimal plan for the majority of problems.
- Without the context (i.e., an example problem and its corresponding problem PDDL), we observe that LLMs fail to produce correct problem PDDL files. Therefore, the context is important for LLM + Planner combination to work.