NL2Plan

Robust LLM-Driven Planning from Minimal Text Descriptions

Elliot Gestrin, Marco Kuhlmann, Jendrik Seipp

June 3, 2024

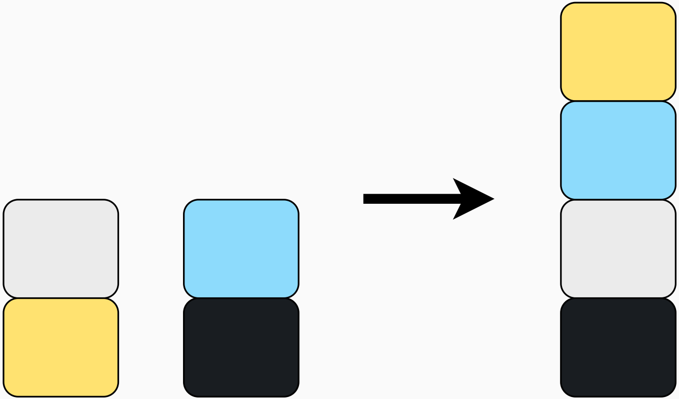
Linköping University

NL2Plan

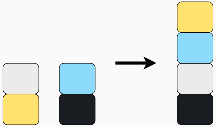
Plans from natural language descriptions via PDDL

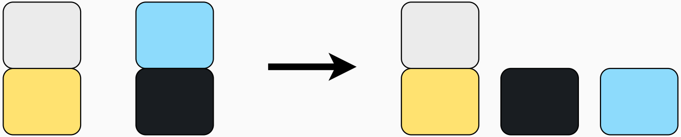
Introduction

Planning with Large Language Models

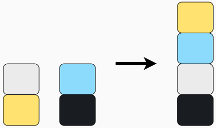


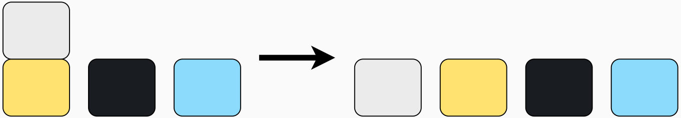
Planning with Large Language Models



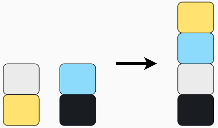


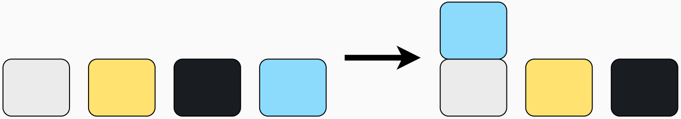
Planning with Large Language Models



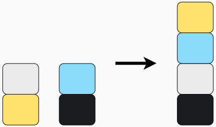


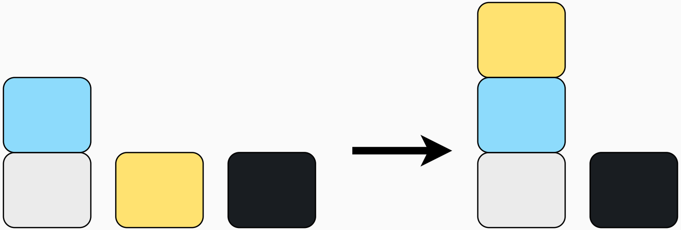
Planning with Large Language Models



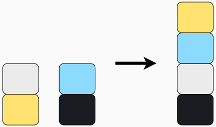


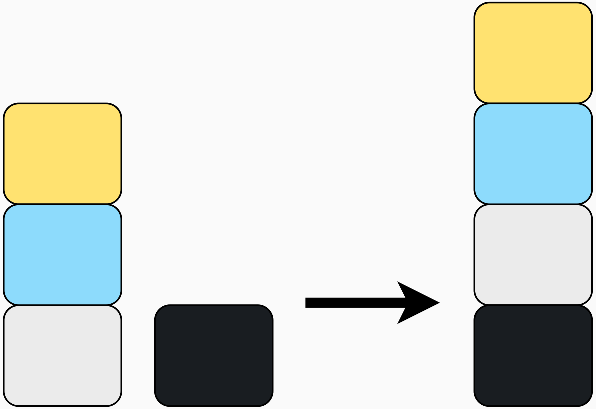
Planning with Large Language Models





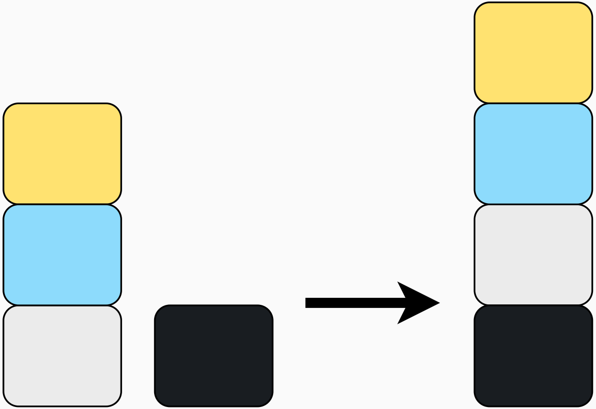
Planning with Large Language Models





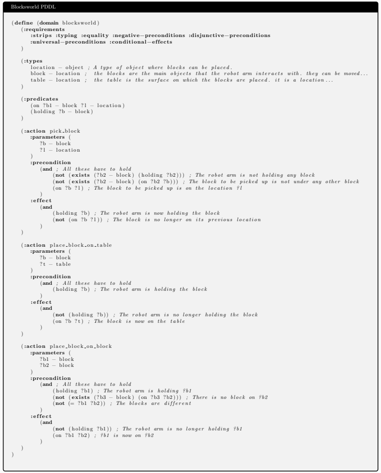
Planning with Large Language Models

- Invalid actions
- Convenient



Planning with Classical Tools

- Powerful
- Requires experience



Blocksworld PDDL

```
(define (domain blocksworld)
   (:requirements
      :strips :typing :equality :negative-preconditions :disjunctive-preconditions
      :universal-preconditions :conditional-effects
   )

   (:types
      location - object ; A type of object where blocks can be placed.
      block - location ;  the blocks are the main objects that the robot arm interacts with. they can be moved...
      table - location ;  the table is the surface on which the blocks are placed. it is a location...
   )

   (:predicates
      (on ?b1 - block ?l - location)
      (holding ?b - block)
   )

   (:action pick_block
      :parameters (
         ?b - block
         ?l - location
      )
      :precondition
         (and ; All these have to hold
             (not (exists (?b2 - block) (holding ?b2))) ; The robot arm is not holding any block
             (not (exists (?b2 - block) (on ?b2 ?b))) ; The block to be picked up is not under any other block
             (on ?b ?l) ; The block to be picked up is on the location ?l
         )
      :effect
         (and
             (holding ?b) ; The robot arm is now holding the block
             (not (on ?b ?l)) ; The block is no longer on its previous location
         )
   )

   (:action place_block_on_table
      :parameters (
         ?b - block
         ?t - table
      )
      :precondition
         (and ; All these have to hold
             (holding ?b) ; The robot arm is holding the block
         )
      :effect
         (and
             (not (holding ?b)) ; The robot arm is no longer holding the block
             (on ?b ?t) ; The block is now on the table
         )
   )

   (:action place_block_on_block
      :parameters (
         ?b1 - block
         ?b2 - block
      )
      :precondition
         (and ; All these have to hold
             (holding ?b1) ; The robot arm is holding ?b1
             (not (exists (?b3 - block) (on ?b3 ?b2))) ; There is no block on ?b2
             (not (= ?b1 ?b2)) ; The blocks are different
         )
      :effect
         (and
             (not (holding ?b1)) ; The robot arm is no longer holding ?b1
             (on ?b1 ?b2) ; ?b1 is now on ?b2
         )
   )
)
```

LLMs → PDDL

- Parse-and-Solve: Initial and goal states[1]
- LLM+P: Initial and goal states[2]
- Guan et al.: Actions and predicates[3]

[1] Collins et al., *Structured, flexible, and robust: benchmarking and improving large language models towards more human-like behavior in out-of-distribution reasoning tasks.*

[2] Liu et al., *LLM+P: Empowering Large Language Models with Optimal Planning Proficiency.*

[3] Guan et al., *Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning.*

NL2Plan

NL2Plan - Idea

1. Extract Information
2. Formalize PDDL
3. Apply Planner

NL2Plan - Idea

1. Extract Information
    - Type Extraction
    - Hierarchy Construction
    - Action Extraction
2. Formalize PDDL
    - Action Construction
    - Task Extraction
3. Apply Planner
    - Planning

NL2Plan - Input

The AI agent here is a mechanical robot arm that can pick and place the blocks. Only one block may be moved at a time: it may either be placed on the table or placed atop another block. Because of this, any blocks that are, at a given time, under another block cannot be moved.

There are four blocks currently. The blue block is on the red which is on the yellow. The yellow and the green are on the table. I want the red on top of the green.

NL2Plan - Type Extraction

- LLM-driven
- Extract types

Input

The AI agent here is a mechanical robot arm that can pick and place the blocks. Only one block may be moved at a time: it may either be placed on the table or placed atop another block. Because of this, any blocks that are, at a given time, under another block cannot be moved. Currently, ...

⇓

Output

- block: The blocks are the main objects that the robot arm interacts with. They can be moved and placed on top of each other or on the table.
- table: The table is the surface on which the blocks are placed. It is a location where blocks can be placed.

NL2Plan - Hierarchy Construction

- LLM-driven
- Organize types

Input

The AI agent here is a mechanical robot arm that can pick and place the blocks. Only one block may be moved at a time: it may either be placed on the table or placed atop another block. Because of this, ...

Types:

- block: ...
- table: The table is the surface on which the blocks are placed. It is a location where blocks can be placed.

⇓

Output

- location: An object where blocks can be placed
  - block: The blocks are...
  - table: The table is...

NL2Plan - Action Extraction

- LLM-driven
- Decide actions

Input

The AI agent here is a mechanical robot arm that can pick and place the blocks. Only one block may be moved at a time: it may either be placed on the table or placed atop another block. Because of this, ...

Types:

- location: An object...
    - block: The blocks are...
    - table: The table is...

⇓

Output

- pick: The robot arm picks up a block from its current location. The block must not have any other block on top of it. Example: The robot arm picks up the blue block from the red block.
- place-on-table: ...
- place-on-block: ...

NL2Plan - Action Construction

- LLM-driven
- Based on Guan et al.[a]
- Iterative PDDL
  - Actions
  - Predicates

Input

The AI agent here is...

Types:

- location: An object...
  - block: The blocks are...
  - table: The table is...

Predicates: No predicates defined.

Action: pick - The robot arm picks up a block from its current location. The block must not have any other block on top of it. Example: ...

⇓

Output

Inputs: ...

Precondition: ...

Effect: ...

New Predicates: ...

[a] Guan et al., *Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning*.

NL2Plan - Action Construction

Output

```
Inputs: (?b - block ?l - location)
Precondition: (and
 (on ?b ?l)
 (not (exists (?b2 - block) (on ?b2 ?b)))
 (not (exists (?b2 - block) (holding ?b2)))
)
Effect: (and
 (holding ?b)
 (not (on ?b ?l))
)
New Predicates:
          (on ?b1 - block ?l - location): true if the block ?b1 is on the
location ?l
 (holding ?b - block): true if the robot arm is holding the block ?b.
```

NL2Plan - Task Extraction

- LLM-driven
- Define the task

Input

The AI agent...There are four blocks currently. The blue block is on the red which is on the yellow ...I want the red on top of the green.

Types: ...

Predicates:

- on: ...
- holding: ...

⇓

Output

Objects:

- blue, red, yellow, green - block
- table1 - table

State:

- (on blue red)
- (on red yellow) ...

Goal:

- (on red green)

NL2Plan - Planning

- Planner-driven
- Solve PDDL

Input

Generated PDDL

⇓

Output

```
(pick blue red)
(place-table blue table1)
(pick red yellow)
(place-block red green)
; cost = 4 (unit cost)
```

NL2Plan - Feedback & Validation

- LLM Feedback
- Automatic Validation

Input

The AI agent here is a mechanical robot arm that can pick and place the blocks. Only one block may be moved at a time: it may either be placed on the table or placed atop another block. Because of this, any blocks that are, at a given time, under another block cannot be moved. Currently, ...

Types:

- blocks: The blocks are...
- pick: The robot arm picks up...
- table: The table is...
- red-block: The red block is...

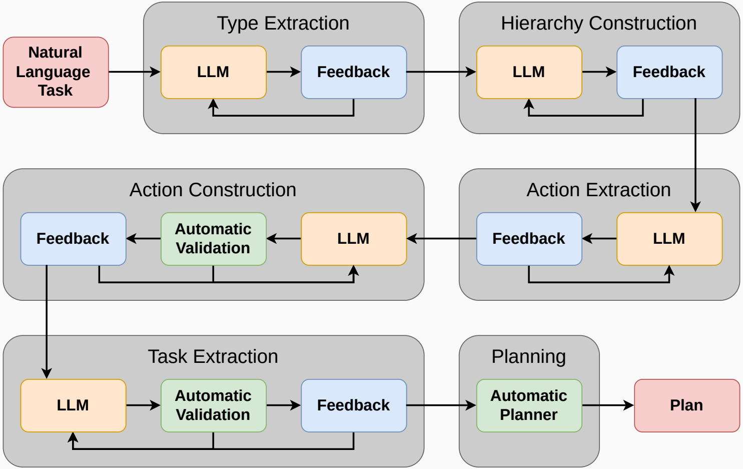
⇓

Output

I suggest the following:

- Remove the ”red-block” type, it is an instance.
- Remove the ”pick” type, as it is an action.
- Change the ”blocks” type to ”block”.

NL2Plan


Natural Language Task
Type Extraction
LLM
Feedback
Hierarchy Construction
LLM
Feedback
Action Extraction
LLM
Feedback
Action Construction
LLM
Automatic Validation
Feedback
Task Extraction
LLM
Automatic Validation
Feedback
Planning
Automatic Planner
Plan

Results

Experiments

- 5 domains:
  - Blocksworld
  - Tyreworld
  - Household
  - ISR
  - ISR Assisted
- 3 difficulties:
  - Easy
  - Medium
  - Hard
- Zero-Shot CoT
- Cost savings

Results

- Coverage
- Domain modelling
- Explainability
- Token Usage

Results

- Coverage
  - NL2Plan > Zero-Shot CoT
- Domain modelling
- Explainability
- Token Usage

Results

- Coverage
    - NL2Plan > Zero-Shot CoT
- Domain modelling
    - NL2Plan > Zero-Shot CoT
- Explainability
- Token Usage

Results

- Coverage
    - NL2Plan > Zero-Shot CoT
- Domain modelling
    - NL2Plan > Zero-Shot CoT
- Explainability
    - Interpretable
    - Faithful
- Token Usage

Results

- Coverage
  - NL2Plan > Zero-Shot CoT
- Domain modelling
  - NL2Plan > Zero-Shot CoT
- Explainability
  - Interpretable
  - Faithful
- Token Usage
  - NL2Plan » Zero-Shot CoT

Usage

Usage

- Planning

Usage

- Planning
- PDDL assistance

NL2Plan

Robust LLM-Driven Planning from Minimal Text Descriptions

- NL2Plan plans better
- NL2Plan is interpretable
- NL2Plan is expensive
- NL2Plan could assist

Elliot Gestrin
Marco Kuhlmann
Jendrik Seipp

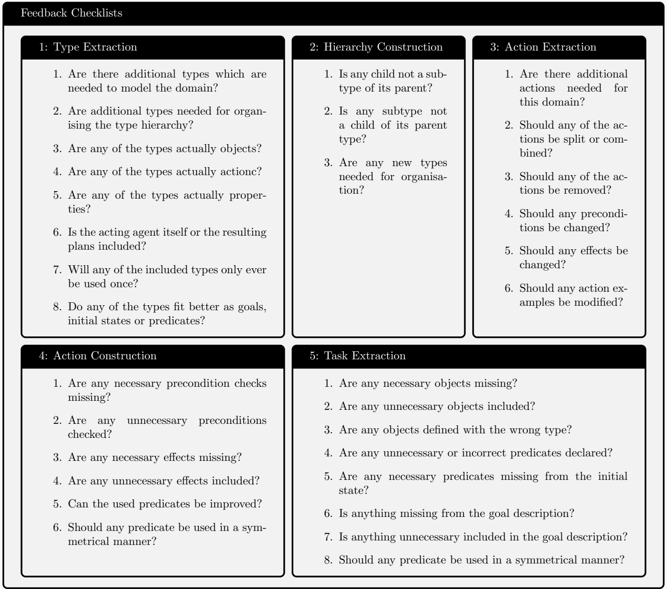
Results

		Zero-Shot CoT		NL2Plan	
Blocksworld	Easy	✓		✓	
	Med.	✗	Moves stack of blocks.	✓	
	Hard	✗	Moves lower block.	✓	
Tyreworld	Easy	∼	Loosens already loose nut.	✓	
	Med.	✓		✓	
	Hard	✗	Fetches non-existent jack.	✗	Incorrectly specifies boot as open. Also domain flaw.
Household	Easy	∼	Places mug on closed cabinet.	✓	
	Med.	✗	Opens container inside fridge.	✗	Fails to use involved predicates. No plan found.
	Hard	✗	Picks up non-pickupable pizza.	✗	Refers to previous iteration. No plan found.
ISR	Easy	✗	Invalid node replacement.	✓	
	Med.	✗	Invalid node replacement.	✗	Defined directed graph. Also domain flaw.
	Hard	✗	Invalid node replacement.	✗	Defined directed graph. Also domain flaw.
ISR Assisted	Easy	✗	Invalid node placement.	✓	
	Med.	✗	Invalid node placement.	✓	
	Hard	✗	Invalid node placement.	✓	

Token Usage

		NL2Plan				
	Zero-Shot CoT	Step 1	Step 2	Step 3	Step 4	Step 5
Blocksworld	314	5669	2640	5261	50866	7409
Tyreworld	612	3475	3012	5549	88738	12315
Household	841	4424	4012	7089	160205	23334
ISR	647	5657	4638	8412	29793	11676
ISR Assisted	735	6160	4884	5411	30567	17443
Average	630	5077	3837	6344	72034	14435

Feedback Checklist



Feedback Checklists

1: Type Extraction

1. Are there additional types which are needed to model the domain?
2. Are additional types needed for organising the type hierarchy?
3. Are any of the types actually objects?
4. Are any of the types actually actionc?
5. Are any of the types actually properties?
6. Is the acting agent itself or the resulting plans included?
7. Will any of the included types only ever be used once?
8. Do any of the types fit better as goals, initial states or predicates?

2: Hierarchy Construction

1. Is any child not a subtype of its parent?
2. Is any subtype not a child of its parent type?
3. Are any new types needed for organisation?

3: Action Extraction

1. Are there additional actions needed for this domain?
2. Should any of the actions be split or combined?
3. Should any of the actions be removed?
4. Should any preconditions be changed?
5. Should any effects be changed?
6. Should any action examples be modified?

4: Action Construction

1. Are any necessary precondition checks missing?
2. Are any unnecessary preconditions checked?
3. Are any necessary effects missing?
4. Are any unnecessary effects included?
5. Can the used predicates be improved?
6. Should any predicate be used in a symmetrical manner?

5: Task Extraction

1. Are any necessary objects missing?
2. Are any unnecessary objects included?
3. Are any objects defined with the wrong type?
4. Are any unnecessary or incorrect predicates declared?
5. Are any necessary predicates missing from the initial state?
6. Is anything missing from the goal description?
7. Is anything unnecessary included in the goal description?
8. Should any predicate be used in a symmetrical manner?