



THE UNIVERSITY OF  
MELBOURNE

# Towards Model-Based Reasoning in Large Language Models: A Planning Perspective

Nir Lipovetzky | [LM4Plan@ICAPS](mailto:LM4Plan@ICAPS) | 11 November 2025



OPTiMA

ARC TRAINING CENTRE IN  
OPTIMISATION TECHNOLOGIES  
INTEGRATED METHODOLOGIES  
AND APPLICATIONS

# My Research / Work



I've completed my PhD in Barcelona, moved Melbourne Uni since 12/12/12.

My main research interests are Automated planning, specifically:

- **Planning** algorithms and tools, e.g. LAPKT, planning.domains, planimation.
- **Goal** and Intention **Recognition**.
- **Integration** of **Planning**, **Learning** and **Control**.
- **Creating bridges** across communities solving **common benchmarks**: Atari, ARC, etc.

Love Surfing & Futbol, but not FC. Barcelona ;)

# Collaborators



Science is a collaborative effort, It can only be done with great people like Chao Lei (He's looking for PostDoc position), Dr. Sukai Huang (Monash), A/prof. Kris Ehinger (UoM), Prof. Trevor Cohn (UoM / Google), among others.

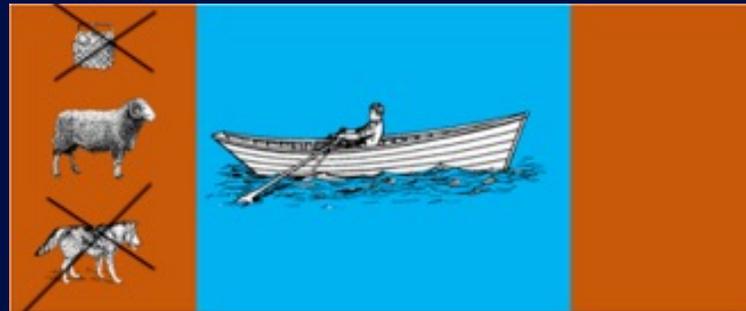
**Disclaimer:** There's a large body of work in our community that I won't cover.

This talk covers work primarily done in our research group.

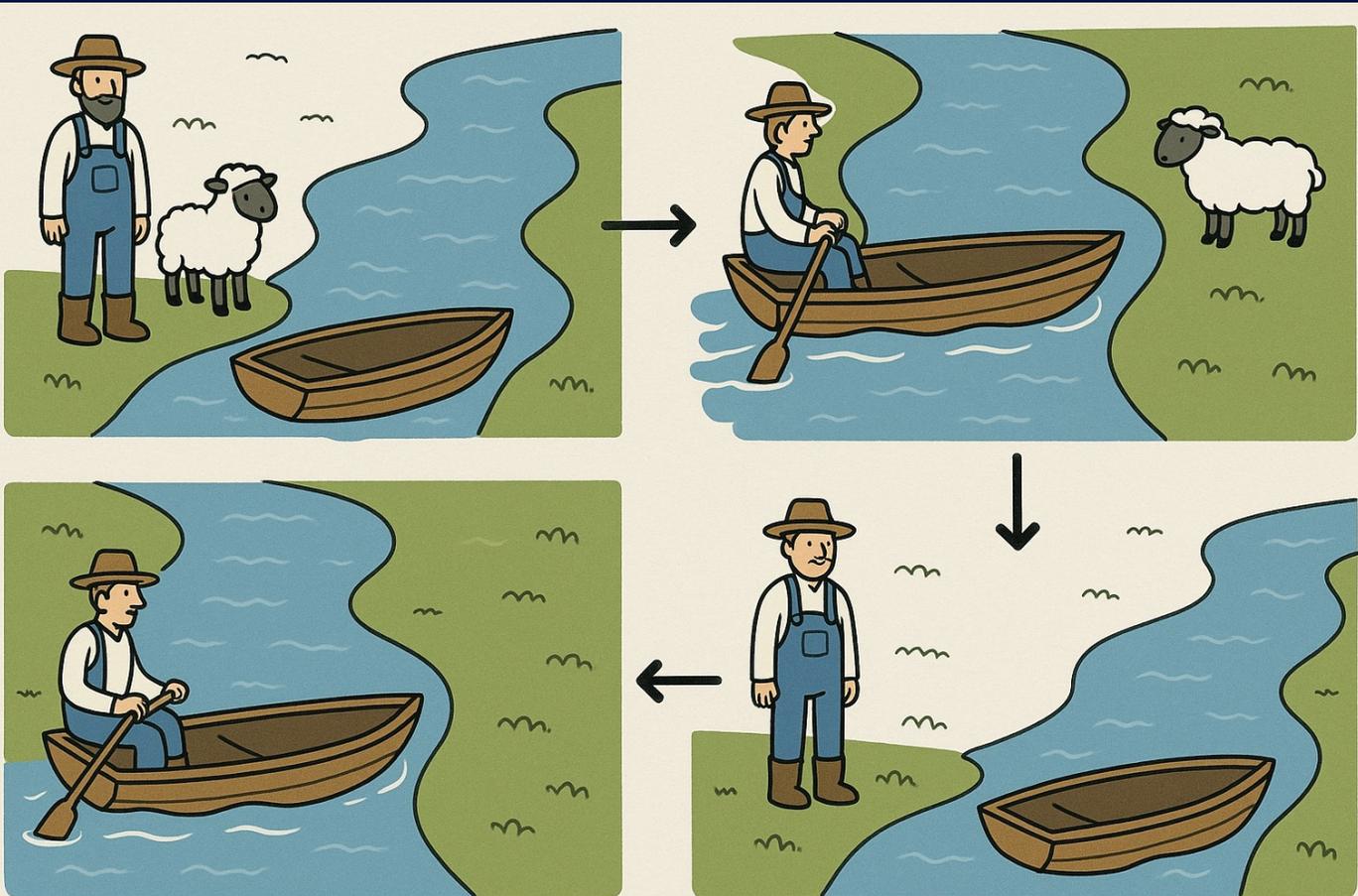


# LLM Plan robustness

A farmer stands at the side of a river with a sheep. There is a boat with enough room for one person and one animal. How can the **farmer get himself and the sheep to the other side** of the river using the boat in the smallest number of trips?



# LLM Plan robustness



← Post

Reply ↕

 Yann LeCun    
@ylecun

👤 ⋮



LLMs can plan, eh?

 Colin Fraser @colin\_fraser · Jun 21, 2024

Claude still can't solve the impossible one farmer one sheep one boat problem

**CF** A farmer stands at the side of a river with a sheep. There is a boat with enough room for one person and one animal. How can the farmer get himself and the sheep to the other side of the river using the boat in the smallest number of trips?

To solve this problem, let's think through it step by step:

1. Initial situation: The farmer and the sheep are on one side of the river, with a boat that can carry one person and one animal.
2. Possible moves:
  - The farmer can go alone
  - The farmer can take the sheep
3. The goal is to get both the farmer and the sheep to the other side in the fewest trips.
4. Solution: Step 1: The farmer takes the sheep across the river. Step 2: The farmer returns alone. Step 3: The farmer crosses the river alone.
5. Counting the trips:
  - Farmer with sheep (1 trip)
  - Farmer returns (2 trips)
  - Farmer crosses alone (3 trips)

Therefore, the farmer can get himself and the sheep to the other side of the river in a minimum of 3 trips.

This is the most efficient solution because:

- The farmer can't send the sheep alone (animals can't operate boats).
- The farmer can't go first and leave the sheep behind, as there would be no way to get the sheep across after.

Would you like me to explain any part of this solution in more detail?

📄 Copy 🔄 Retry 🗑️



Claude can make mistakes. Please double-check responses.

Reply to Claude...

Claude 3.5 Sonnet

# ChatGPT5 Plan robustness



ChatGPT >

A farmer stands at teh side of a river with a sheep. There is a boat with enough room for one person and one animal. Ho can the farmer get himself and the sheep to the other side of the river using the boat in the smallest number of trips?

Thought for 5s >

3 trips.

Steps:

1. Farmer takes the sheep across the river (trip 1).
2. Farmer returns alone to the original side (trip 2).
3. Farmer crosses alone to the far side (trip 3).

Now both farmer and sheep are on the other side.



ChatGPT >

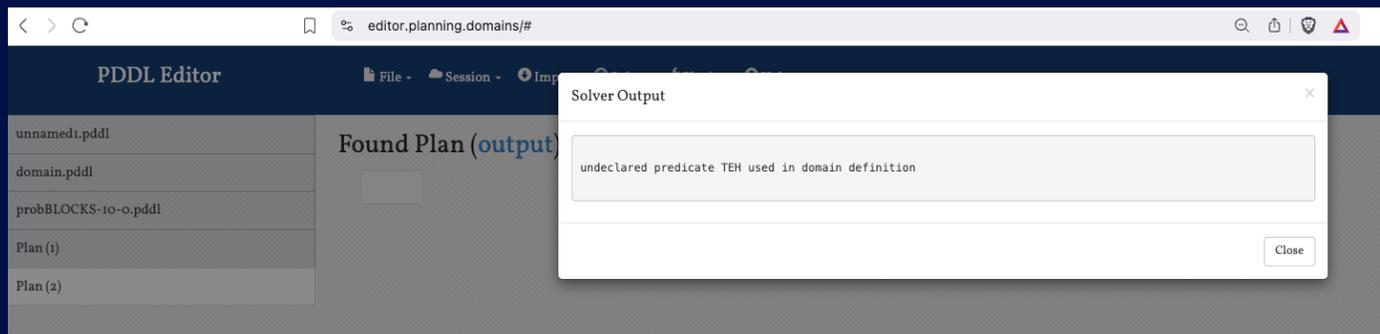
**Thinking longer**

Thought for 9s >

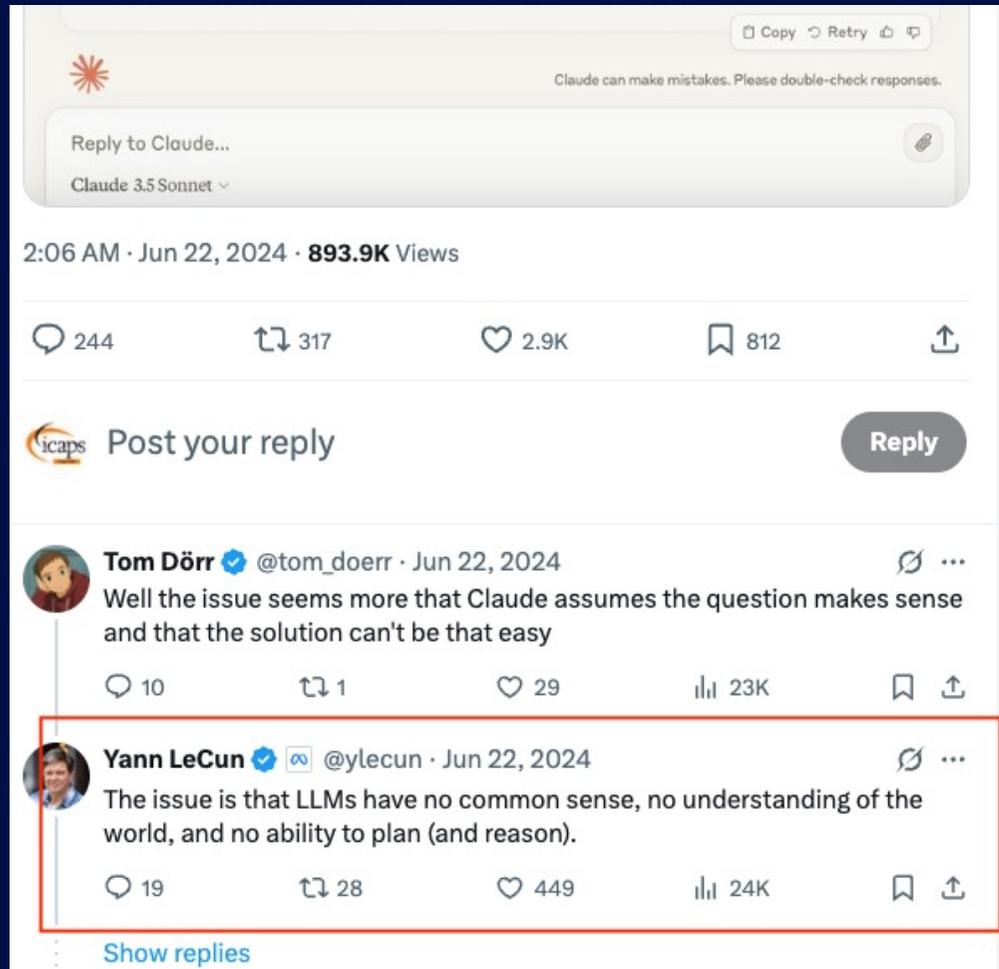
One trip. The farmer simply puts the sheep in the boat with him and rows straight across — both get to the other side in a single crossing.



A farmer stands at the side of a river with a sheep. There is a boat with enough room for one person and one animal. Ho can the farmer get himself and the sheep to the other side of the river using the boat in the smallest number of trips?



# LLM Plan robustness



Copy Retry

Claude can make mistakes. Please double-check responses.

Reply to Claude...

Claude 3.5 Sonnet

2:06 AM · Jun 22, 2024 · 893.9K Views

244 317 2.9K 812

icaps Post your reply Reply

**Tom Dörr** @tom\_doerr · Jun 22, 2024  
Well the issue seems more that Claude assumes the question makes sense and that the solution can't be that easy  
10 1 29 23K

**Yann LeCun** @ylecun · Jun 22, 2024  
The issue is that LLMs have no common sense, no understanding of the world, and no ability to plan (and reason).  
19 28 449 24K

Show replies

# LLMs and Planning: Setting the Scene

- LLMs **can't plan** (in the classical sense).
- LLMs **still can't plan reliably**, even with reasoning models.

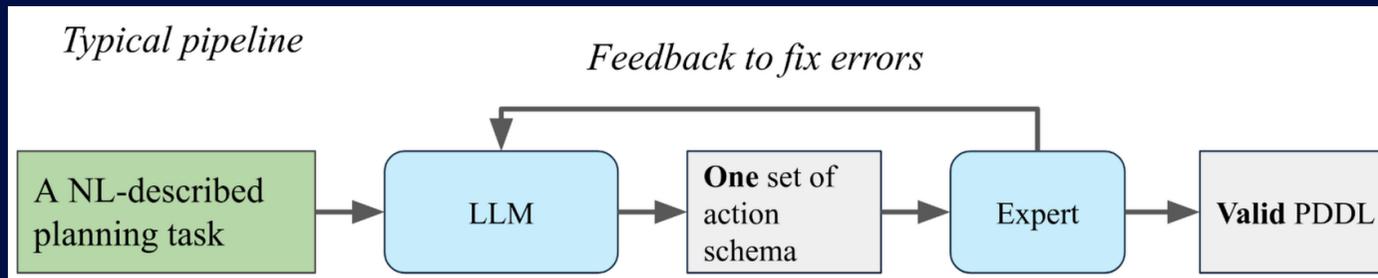
LLMs need **formal** reasoning models to plan robustly!



Bridge the gap between natural language  
and  
formal models.

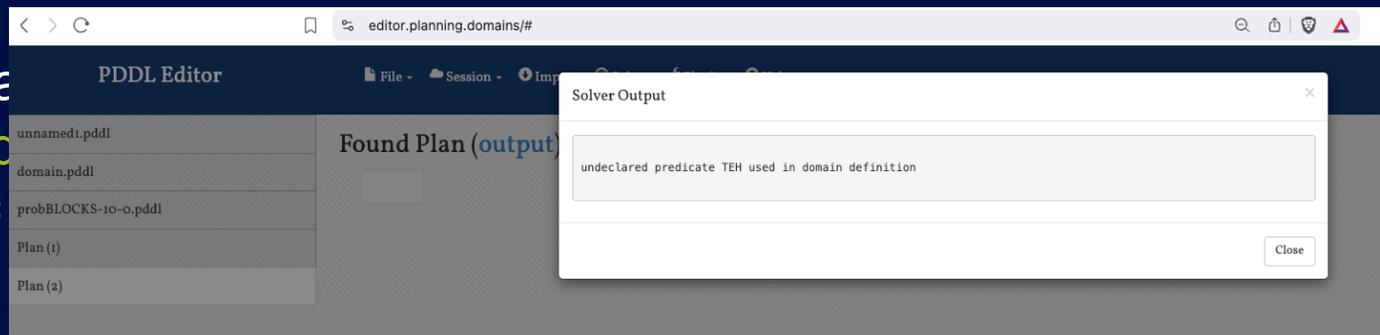
How Hard can it be?

# LLMs generating Formal Models: Experts needed! (Issue 1)



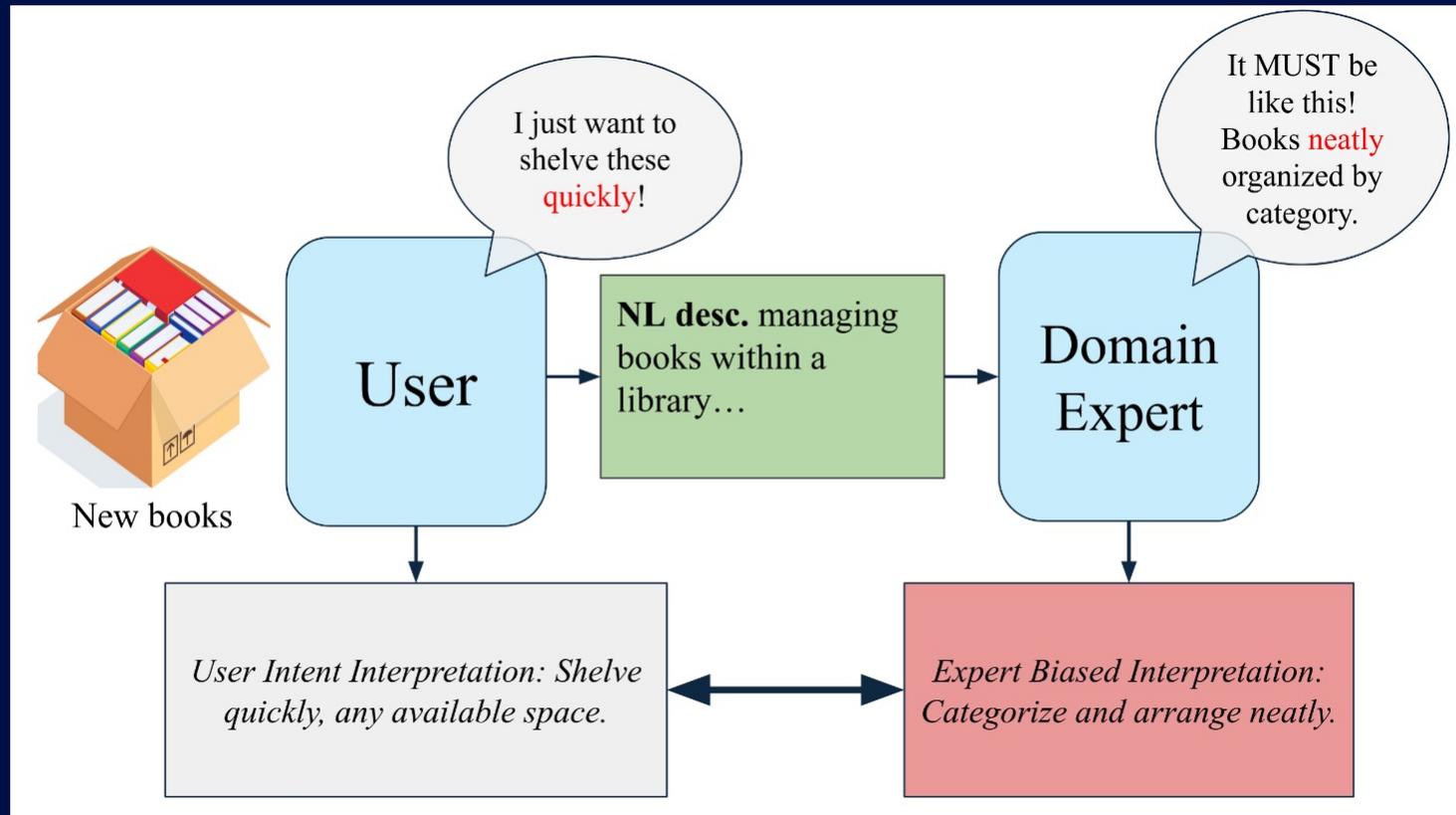
- 1 missing or contradictory predicate in an action schema can prevent the planner from finding a **valid** plan.

- Given an action schema being **valid** (i.e.  $p=0.9$ ,  $M=1$ ), the probability of finding a plan is **0.1** (i.e.  $p=0.1$ ,  $M=1$ ). E.g.

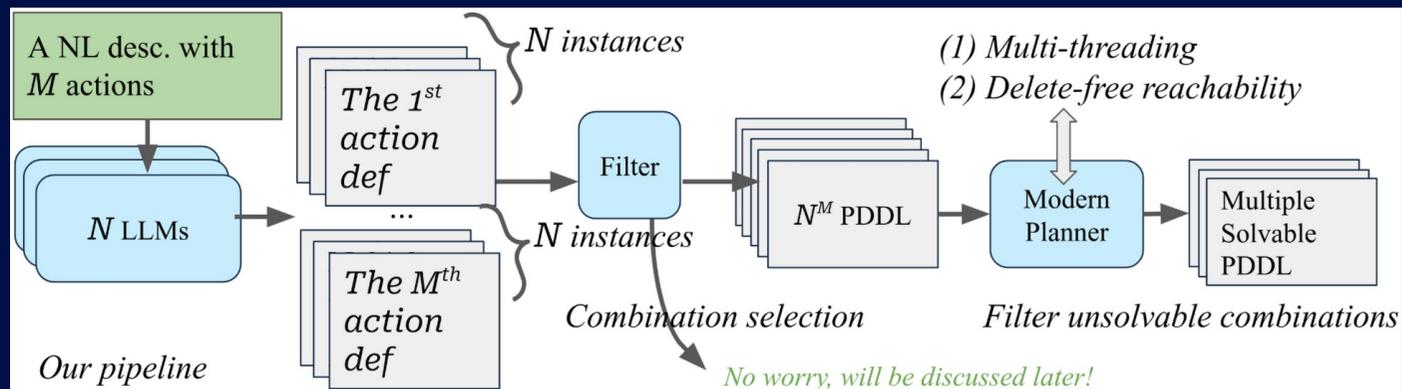


- However, actions are *correlated*, actual prob. is much lower.

# Subjective Bias: User / Human or LLM modeler (Issue 2)



# Proposed solution: Planning in the Dark



Sukai Huang et al. AAAI'25

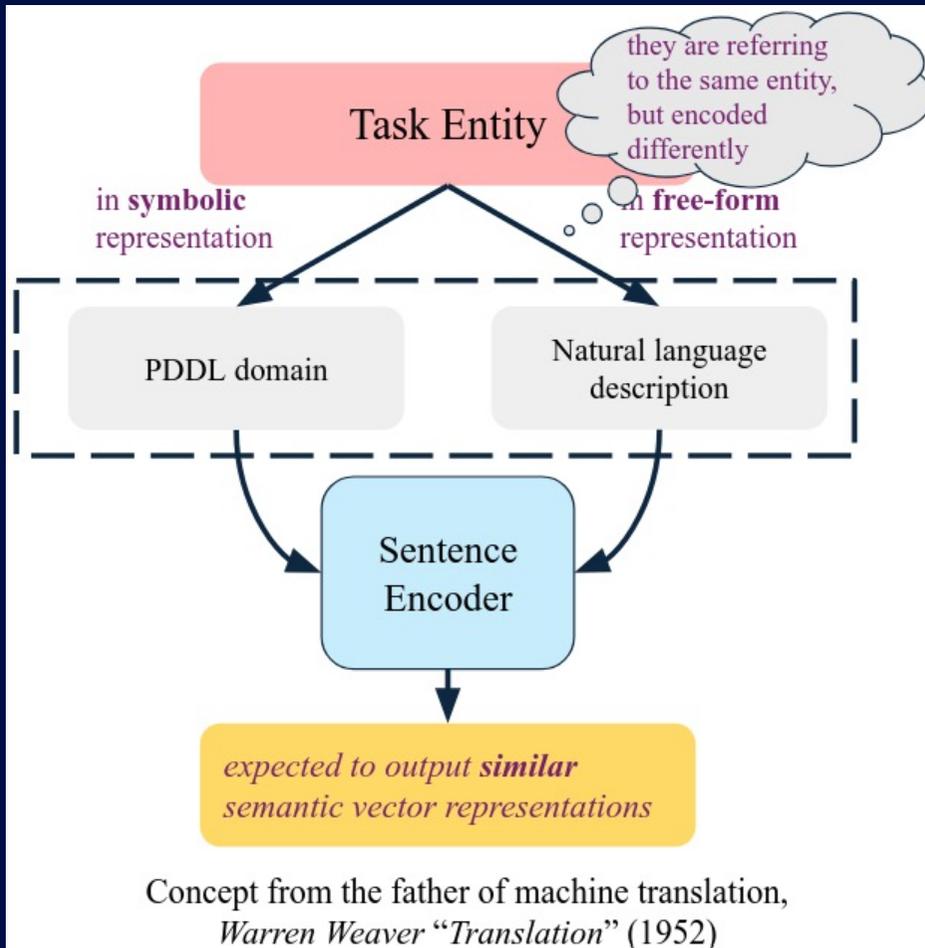
Validity - the probability that **at least one action schema is solvable** is  $1 - (1 - p^M)^{N^M}$

Subjective Bias - Generate **multiple** formal models

**New bottleneck:** verifying the solvability of each combination.

Prob. all schema wrong

# Semantic Coherence Filtering and Ranking



A vanilla **sentence encoder** can quantify semantic relatedness between NL input and generated PDDL snippets.

- **Compute** *cosine similarity* between the embeddings.
- **Filter** out *low-confidence* schemas.
- **Rank** plan candidates based on the  $\Sigma$  similarity score of its actions.

# Fine-tuning the Sentence Encoder is Convenient!



THE UNIVERSITY OF  
MELBOURNE

Manipulation Type	Description	Example
Swap	Exchanges a predicate between preconditions and effects	Precondition: (at ?x ?y) Effect: (not (at ?x ?z)) → Precondition: (not (at ?x ?z)) Effect: (at ?x ?y)
Negation	Negates a predicate in either preconditions or effects	Precondition: (clear ?x) → Precondition: (not (clear ?x))
Removal	Removes a predicate from either preconditions or effects	Precondition: (and (on ?x ?y) (clear ?x)) → Precondition: (on ?x ?y)
Addition	Adds mutually exclusive (mutex) predicates to preconditions or effects (Helmert 2009)	Effect: (on-table ?x) → Effect: (and (on-table ?x) (holding ?x))

Can further *fine-tune* the sentence encoder.

**Contrastive training** with **hard negatives** synthesized via precondition & effect manipulation

# Planning in the Dark - Discussion

Our pipeline enables a *fully automated* end-to-end LLM + formal model planning without expert intervention.

- **2 mins** avg time per task for a **32-thread** CPU, much **faster** than traditional expert-dependent pipelines (50K schemas evaluated).
- **10 LLMs** are adequate for **6-action schema** problems.
- **Better** plan quality than **Tree of Thought** direct LLM planning.



AAAI'25 + Appendix

By generating multiple candidate action schemas, the model reflects *different possible interpretations* of the **ambiguous** natural language input.

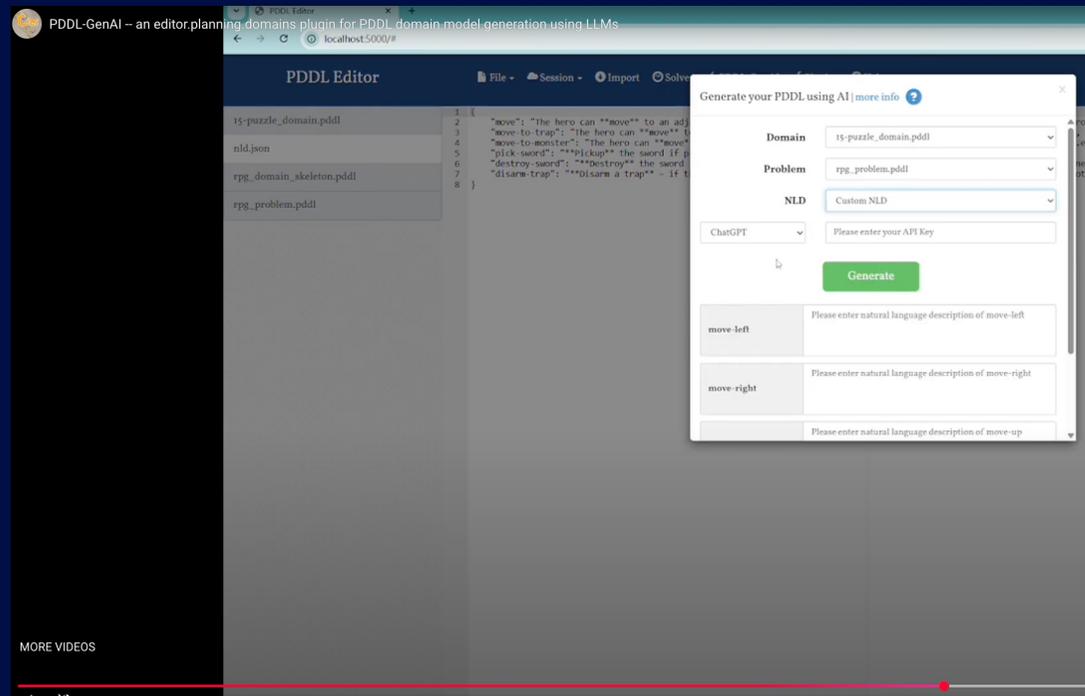
It lowers the barrier of novice users to model in PDDL

# Demo - Wednesday Evening

PDDL-GenAI -- A Plugin for LLM-Based Support in Planning Domain Modeling, ICAPS Demo.

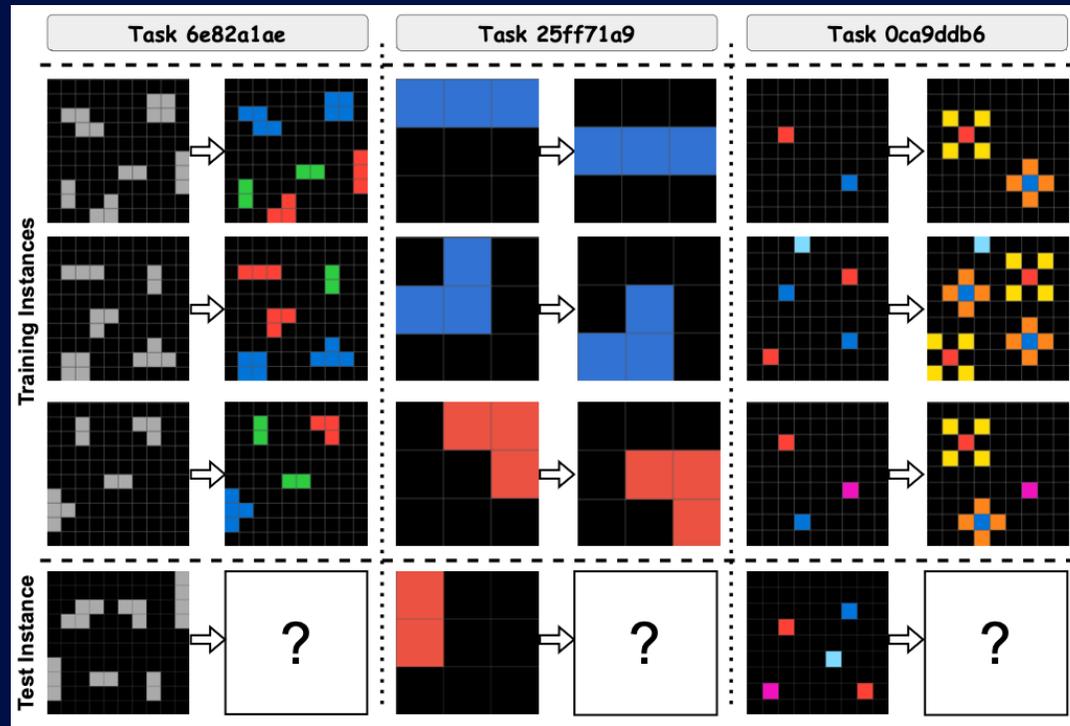


THE UNIVERSITY OF  
MELBOURNE



**Authors:** Jinbiao Wang, Rui Peng, Tong Chen, Chaowen Zeng, Weilin Wang, Sukai Huang, Nir Lipovetzky.

# Abstraction and Reasoning Corpus (ARC)



Abstraction and Reasoning Corpus (ARC) is a set of **abstract visual reasoning tasks** that measure the **gap** of **abstract reasoning** and **generalization** capacities between **humans** and **AI systems** (Chollet (2019)).

# ARC – Human and AI performance



**Human:** 88% of the ARC tasks can be successfully communicated (Acquaviva et al. (2022)).

VS

A **program synthesis** engine with a powerful enough **domain-specific language** is a **promising combination for solving** ARC tasks (Chollet (2019)).

**Kaggle** competition's first-place solution: 40% of the 160 object-centric ARC tasks have been successfully solved (2020).

**Abstract Reasoning with Graph Abstractions** (ARGA): 36% of the 160 object-centric ARC tasks have been successfully solved (Xu et al. (2023)).

# ARC as Generalised Planning



Our approach **Generalized Planning for Abstract Reasoning** (GPAR) achieved state-of-the-art over the Object centric ARC benchmark:

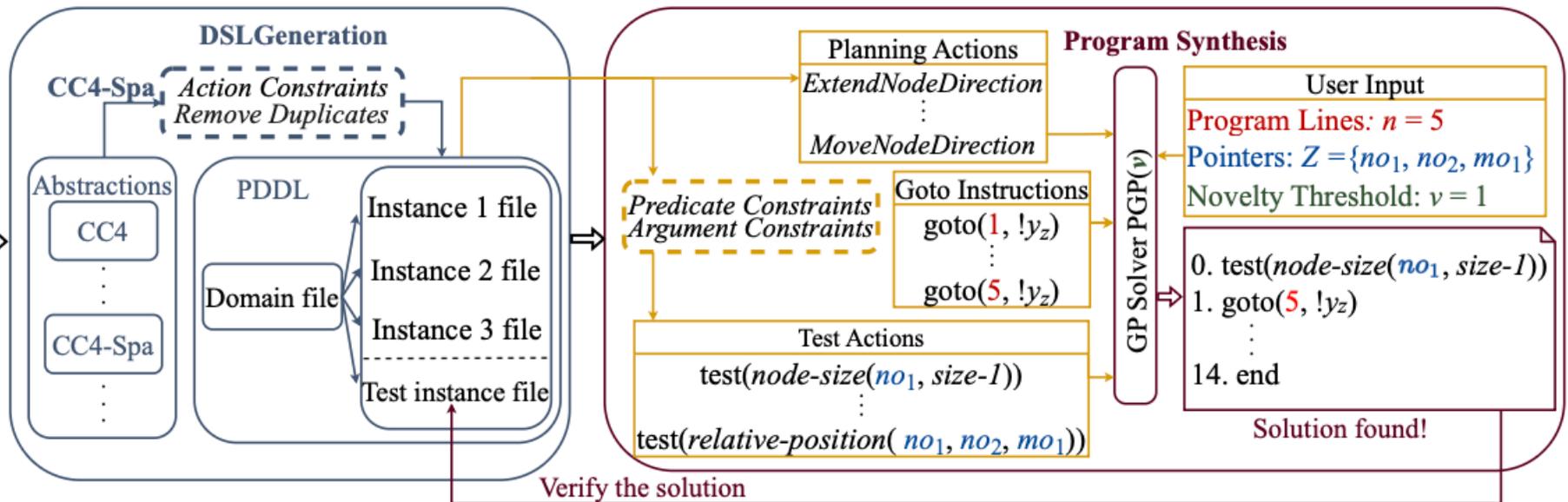
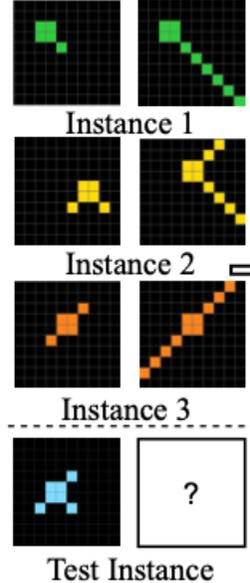
- A novel method to solve abstract reasoning tasks based on **generalized planning**.
- A **domain-specific language** encoding based on **PDDL**.
- The usage of novel **ARC domain knowledge** to reduce the **size of the solution space**.



Generalized Planning for the Abstraction and Reasoning Corpus, Chao Lei et al. AAAI'24

# ARC as Generalised Planning

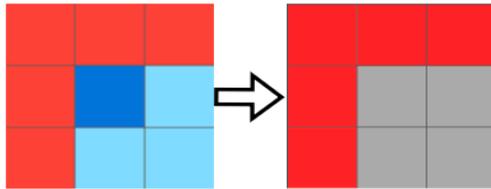
Task 7ddcd7ec



Generalized Planning for the Abstraction and Reasoning Corpus, Chao Lei et al. AAI'24

Task 9565186b

Instance 1



File for Domain



```
(define (domain ARC-9565186b)
  (:requirements :strips)
  (:types node pixel size color - object)
  (:predicates
    (node-color ?no - node ?co - color)
    (node-size ?no - node ?si - size)
    (contain-pixel ?no - node ?pi - pixel)
    (pixel-color ?pi - pixel ?co -color))

  (:action UpdateColor
    :parameters (?no - node, ?co1 - color ?co2 - color)
    :precondition (node-color ?no ?co1)
    :effect @PixelColorUpdate(?no, ?co2)))
```

File for Instance 1

```
(define (problem 9565186b-1)
  (:domain ARC-9565186b)
  (:objects pixel-0-0 pixel-0-1 pixel-0-2 pixel-1-0 pixel-1-1 pixel-1-2
    pixel-2-0 pixel-2-1 pixel-2-2 - pixel
    node-1 node-2 node-3 - node
    size-1 size-3 size-4 size-5 - size
    red blue grey cyan - color)

  (:INIT (node-size node-1 size-5) (node-color node-1 red) (node-size node-2 size-1)
    (node-color nod-2 blue) (node-size node-3 size-3) (node-color nod-3 cyan)
    (pixel-color pixel-0-0 red) (pixel-color pixel-0-1 red) (pixel-color pixel-0-2 red)
    (pixel-color pixel-1-0 red) (pixel-color pixel-1-1 blue) (pixel-color pixel-1-2 cyan)
    (pixel-color pixel-2-0 red) (pixel-color pixel-2-1 cyan) (pixel-color pixel-2-2 cyan)
    (contain-pixel node-1 pixel-0-0) (contain-pixel node-1 pixel-0-1) (contain-pixel node-1 pixel-0-2)
    (contain-pixel node-1 pixel-1-0) (contain-pixel node-2 pixel-1-1) (contain-pixel node-3 pixel-2-1)
    (contain-pixel node-1 pixel-2-0) (contain-pixel node-3 pixel-2-1) (contain-pixel node-3 pixel-2-2))

  (:Goal(AND (pixel-color pixel-0-0 red) (pixel-color pixel-0-1 red) (pixel-color pixel-0-2 red)
    (pixel-color pixel-1-0 red) (pixel-color pixel-1-1 grey) (pixel-color pixel-1-2 grey)
    (pixel-color pixel-2-0 red) (pixel-color pixel-2-1 grey) (pixel-color pixel-2-2 grey))))
```

# PDDL Fragment



@ denotes external functions

(Frances et al. AAI'17)

# ARC – Domain Knowledge & State Space reduction



**Abstraction Removal:** Unique representations kept.

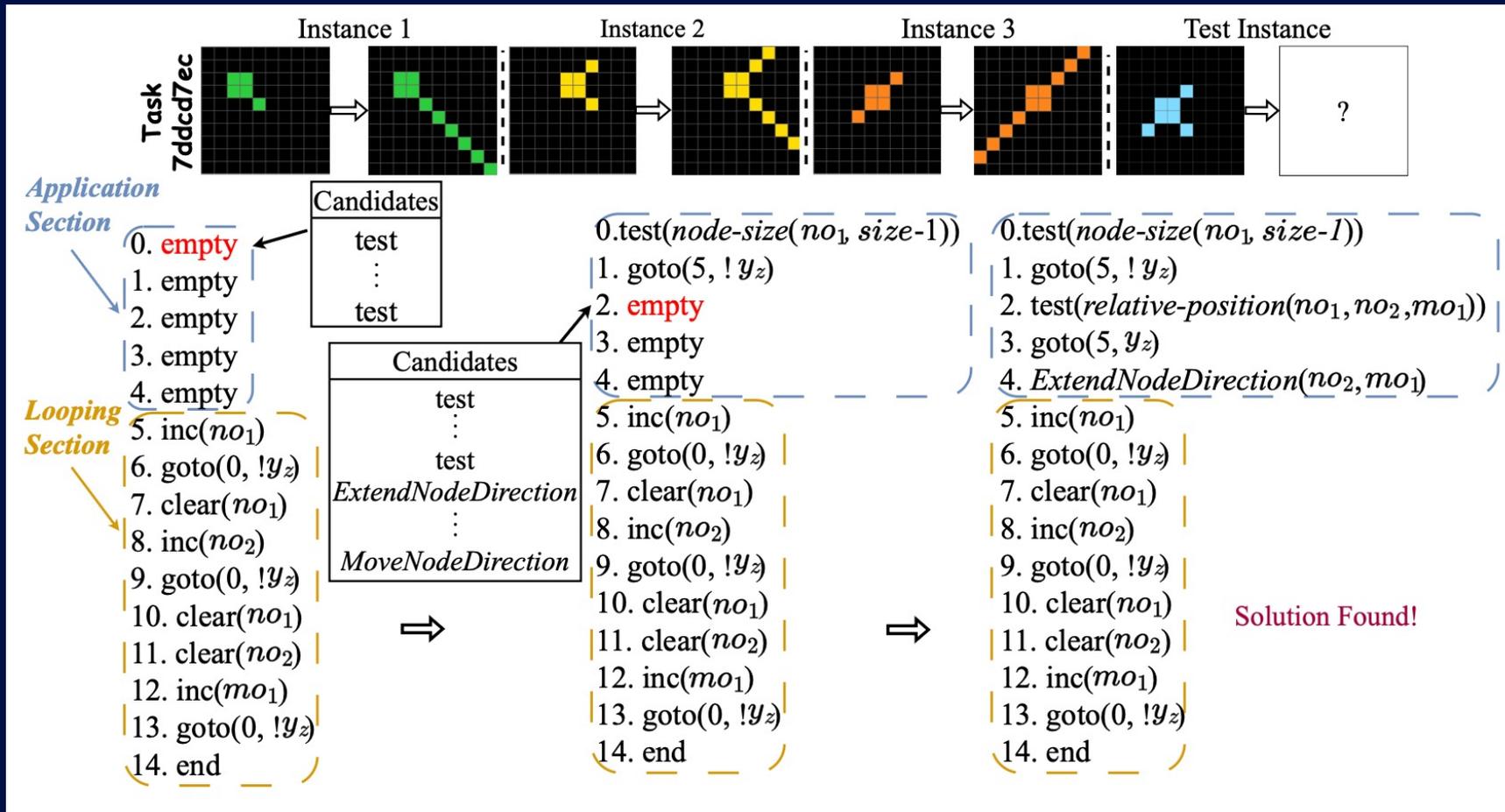
**Action Pruning:** Actions update positions, colors, or sizes in input instances.

**Predicate Constraint:** used as a condition iff it is not an invariant.

**Argument Constraint:** arguments in predicates describe attributes that exist in all inputs.

**Structural Restrictions:** Application of conditional executions + Looping over pointers.

# ARC - Program



# ARC - Summary

GPAR **solves more training instances** than other solvers and **generalises better to test instances** than Kaggle First place.

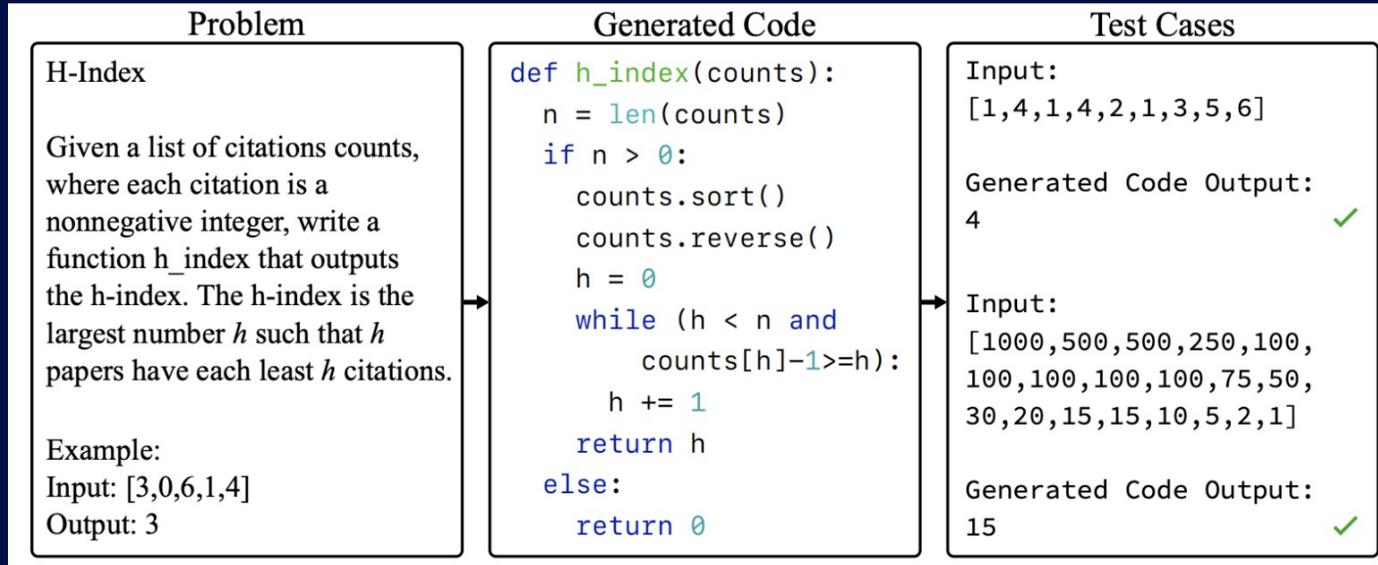
**Problem 1:** Identifying the **most useful abstractions** is still an **open problem**.

**Problem 2:** **New heuristics** can be defined to **guide the search** of programs, as finding a program is hard.

Can LLMs play a role to address Problems 1 and 2 ?



# Text-to-Code Generation



**Code generation**, also known as **program synthesis**, studies the automatic construction of a program that satisfies a specified high-level input requirement (Gulwani et al., 2017).

# Challenges in Code Generation



## Lexical Constraints

**Correct:**

```
def foo():
```

**Incorrect:**

```
df foo()
```

## Grammatical Constraints

**Correct:**

```
def greet(name):  
    print("Hello, " + name)
```

**Incorrect:**

```
def greet(name):  
print("Hello, " + name)
```

## Semantic Constraints

**Correct:**

```
def total(lst):  
    return sum(lst)
```

**Incorrect:**

```
def total(lst):  
    return len(lst)
```

**Lexical:** Invalid tokens, **Grammar:** Invalid syntax target language, **Semantic:** Invalid logic.

# LLM Code Generation



**Sampling:** Inefficient; does not follow human programming strategies.

**Planning-Aided:** Lacks code and plan refinement capabilities.

**Refinement:** Feedback messages lack precise correction instructions.

**Multi-Agent:** Hallucination degrades communication and increased token consumption.

# Planning-Driven Programming: A Large Language Model Programming Workflow (LPW)



We address limitations with a **two phase** for code generation:

**Solution generation:** For **plan creation** and **verification**.

**Code implementation:** For **code development** and **refinements**.



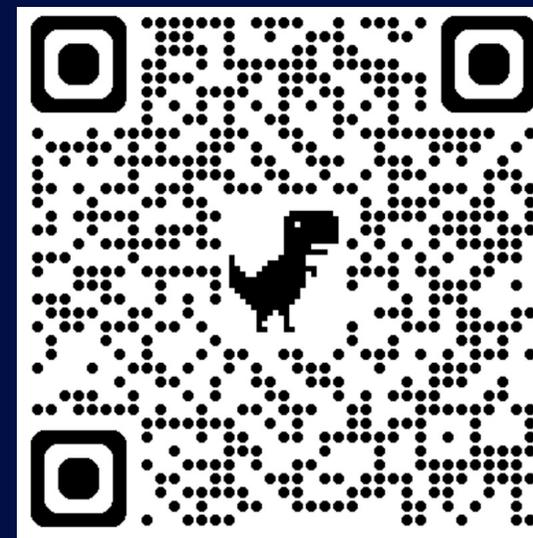
Planning-Driven Programming: A Large Language Model Programming Workflow, Chao Lei et al. ACL'25

# LPW - Summary

LPW outperforms all competing methods across all benchmarks and various LLM backbones. It establishes new state-of-the-art Pass@1 accuracy, notably achieving 98.2% on HumanEval.

	HumanEval		HumanEval-ET		MBPP		MBPP-ET	
	Acc $\uparrow$	$\Delta$ $\uparrow$						
Baseline	74.4 $\pm$ 0.8	–	66.5 $\pm$ 1.3	–	67.4 $\pm$ 0.5	–	52.8 $\pm$ 0.3	–
SP	77.4 $\pm$ 0.8	3.0	69.5 $\pm$ 0.8	3.0	69.2 $\pm$ 0.4	1.8	52.4 $\pm$ 0.2	-0.4
MapCoder	77.4 $\pm$ 1.0	3.0	66.5 $\pm$ 1.0	0.0	72.0 $\pm$ 0.5	4.6	56.6 $\pm$ 0.5	3.8
SD	81.1 $\pm$ 1.0	6.7	72.0 $\pm$ 1.0	5.5	71.2 $\pm$ 0.3	3.8	56.0 $\pm$ 0.1	3.2
LDB	82.9 $\pm$ 1.0	8.5	72.6 $\pm$ 1.0	6.1	72.4 $\pm$ 0.3	5.0	55.6 $\pm$ 0.2	2.8
LPW (ours)	<b>89.0 <math>\pm</math>0.8</b>	<b>14.6</b>	<b>77.4 <math>\pm</math>0.8</b>	<b>10.9</b>	<b>76.0 <math>\pm</math>0.2</b>	<b>8.6</b>	<b>57.6 <math>\pm</math>0.1</b>	<b>4.8</b>

	HumanEval	HumanEval-ET	MBPP	MBPP-ET	LiveCode	APPS	CodeContests
Baseline	91.5 $\pm$ 0.3	81.7 $\pm$ 0.3	78.4 $\pm$ 0.4	62.6 $\pm$ 0.2	45.7 $\pm$ 0.6	41.7 $\pm$ 0.9	28.0 $\pm$ 0.5
GPT-4o LDB	92.1 $\pm$ 0.0	81.7 $\pm$ 0.0	82.4 $\pm$ 0.3	65.4 $\pm$ 0.0	54.3 $\pm$ 0.3	53.2 $\pm$ 0.3	29.3 $\pm$ 0.3
LPW (ours)	<b>98.2 <math>\pm</math>0.3</b>	<b>84.8 <math>\pm</math>0.3</b>	<b>84.8 <math>\pm</math>0.2</b>	<b>65.8 <math>\pm</math>0.1</b>	<b>59.3 <math>\pm</math>0.6</b>	<b>62.6 <math>\pm</math>0.3</b>	<b>34.7 <math>\pm</math>0.3</b>



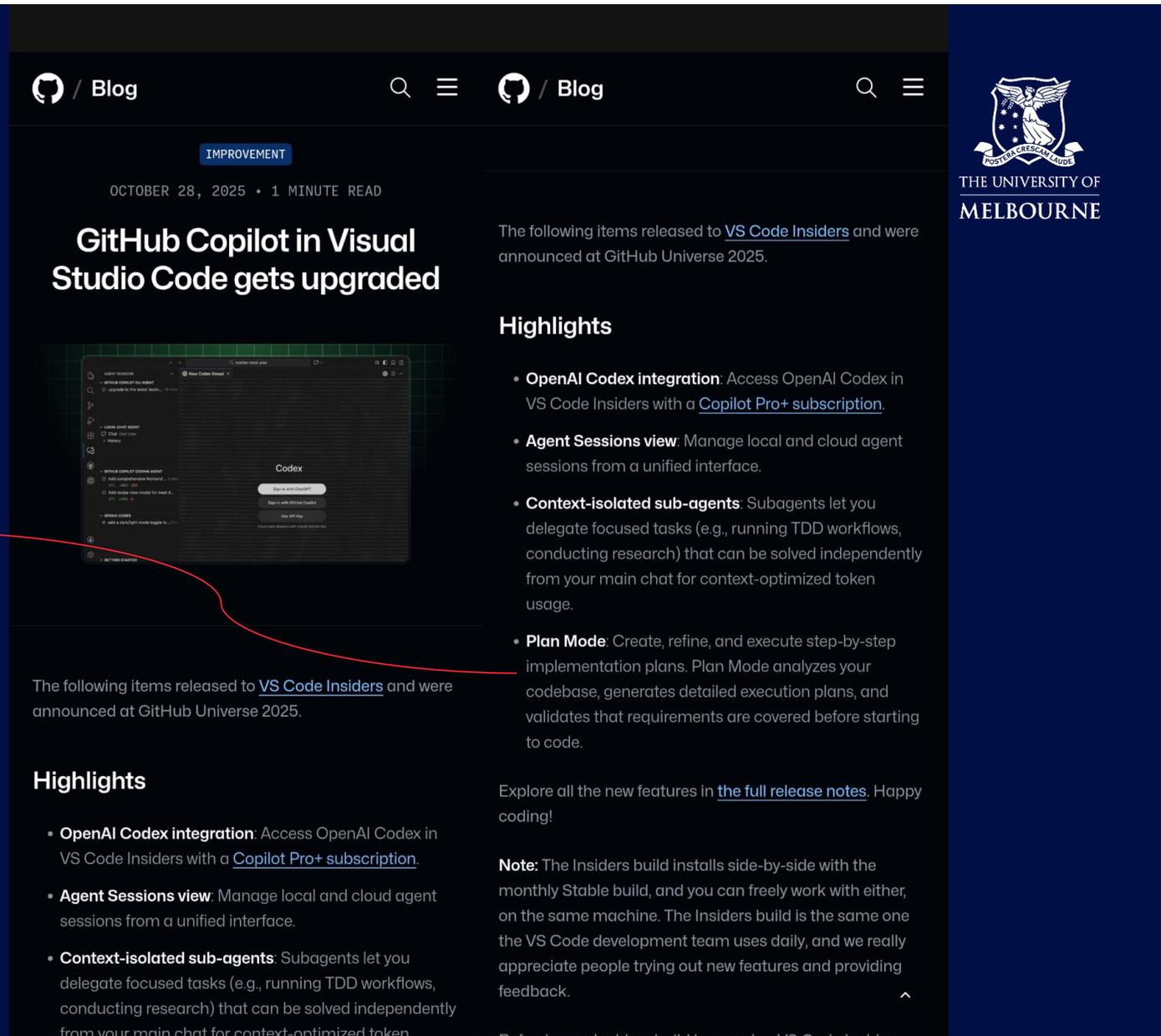
More results @ACL'25 paper

# LPW - Summary

**Plan Mode:** Create, refine, and execute step-by-step implementation plans.

**Plan Mode** analyzes your codebase, generates detailed execution plans, and validates that requirements are covered before starting to code.

Nir Lipovetzky | LM4Plan@ICAPS 2025



The screenshot shows a GitHub blog post from October 28, 2025, titled "GitHub Copilot in Visual Studio Code gets upgraded". The post is categorized as "IMPROVEMENT" and is 1 minute read. It features a "Codex" interface in the center, which is a chat window for OpenAI Codex. A red arrow points from the "Plan Mode" text on the left to the "Codex" interface in the screenshot. The post includes a "Highlights" section with the following items:

- **OpenAI Codex integration:** Access OpenAI Codex in VS Code Insiders with a [Copilot Pro+ subscription](#).
- **Agent Sessions view:** Manage local and cloud agent sessions from a unified interface.
- **Context-isolated sub-agents:** Subagents let you delegate focused tasks (e.g., running TDD workflows, conducting research) that can be solved independently from your main chat for context-optimized token usage.
- **Plan Mode:** Create, refine, and execute step-by-step implementation plans. Plan Mode analyzes your codebase, generates detailed execution plans, and validates that requirements are covered before starting to code.

The post also includes a "Note" section: "The Insiders build installs side-by-side with the monthly Stable build, and you can freely work with either, on the same machine. The Insiders build is the same one the VS Code development team uses daily, and we really appreciate people trying out new features and providing feedback."



# ARC - Summary

GPAR solves more training instances than other solvers and generalises better to test instances than Kaggle First place.

**Problem 1:** Identifying the most useful abstractions is still an open problem.

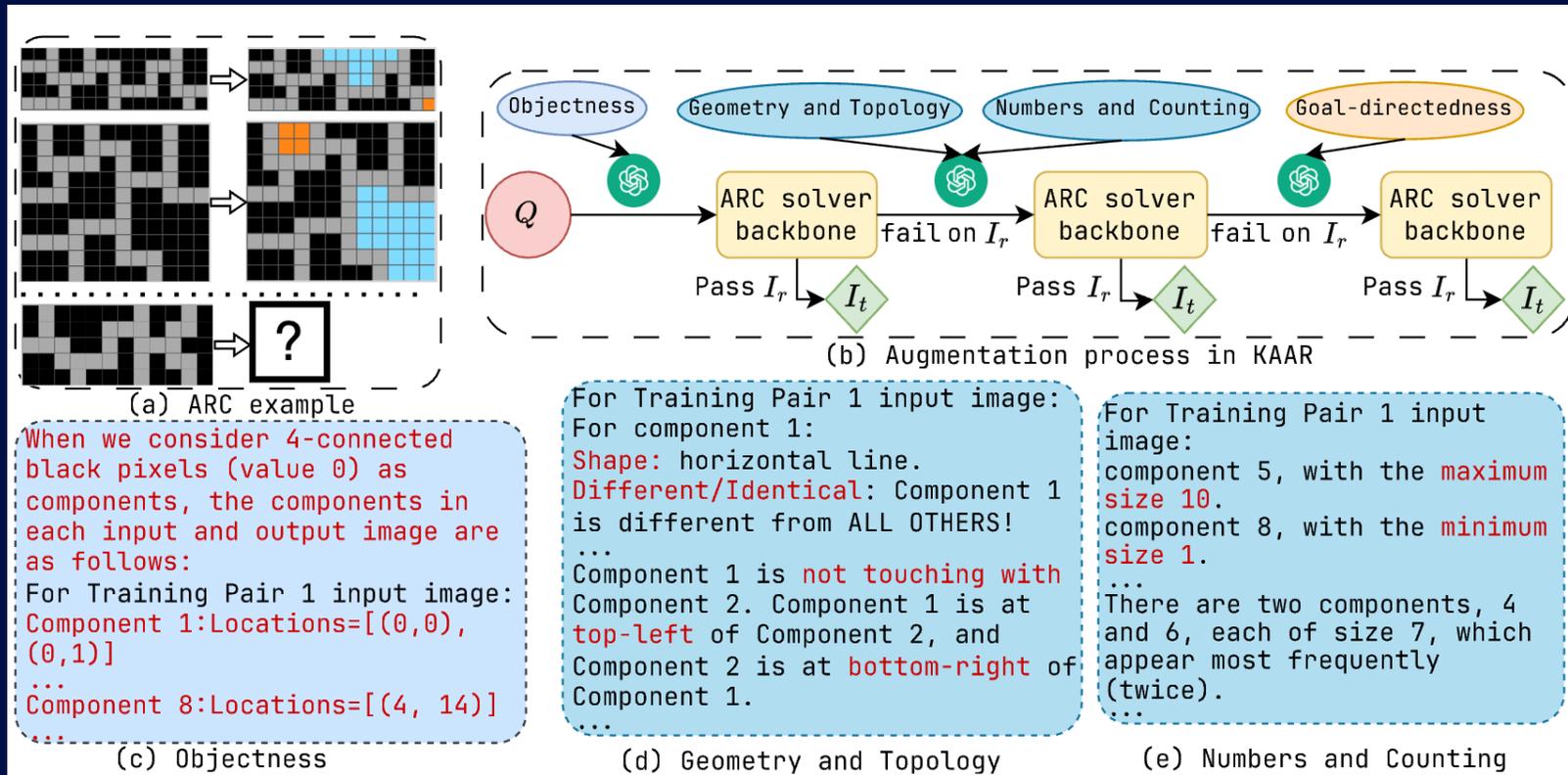
~~**Problem 2:** New heuristics can be defined to guide the search of programs, as finding a program is hard.~~

We can use LPW variants to address Problem 2. Let's address Problem 1!



# ARC – Knowledge augmented LLMs

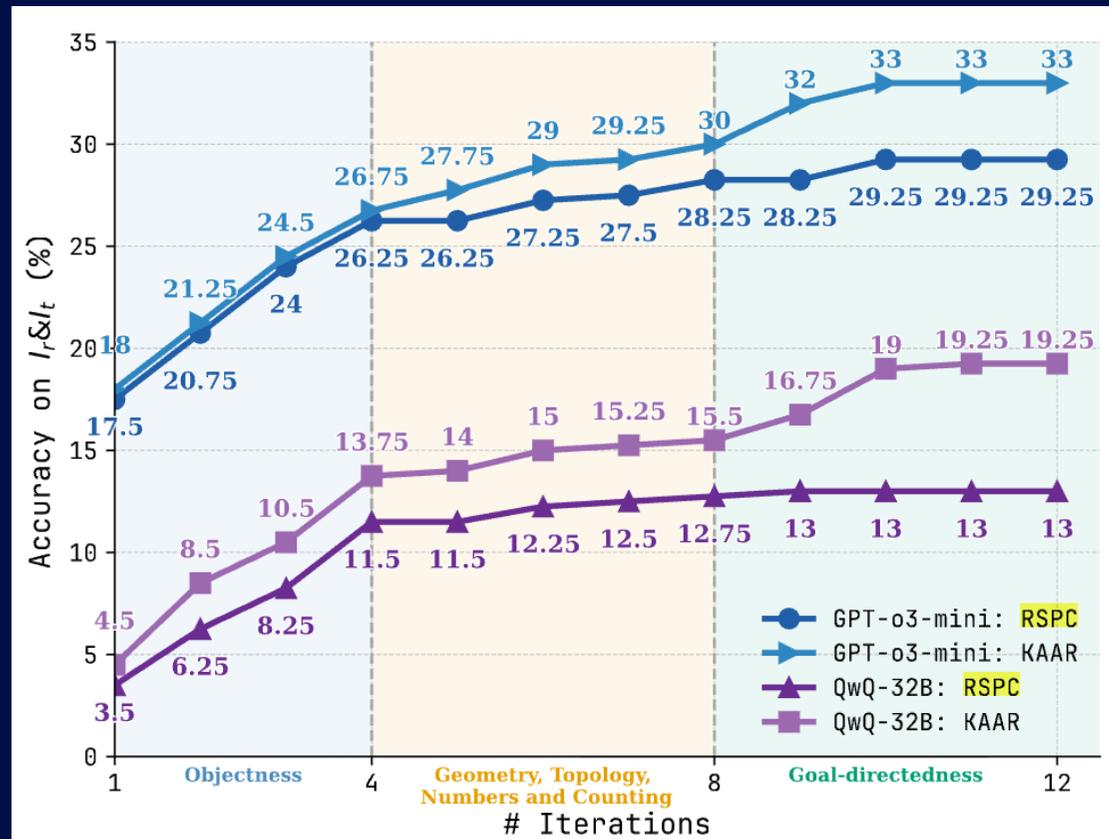
We use **GP** Knowledge Abstraction to augment **LLM Priors**.



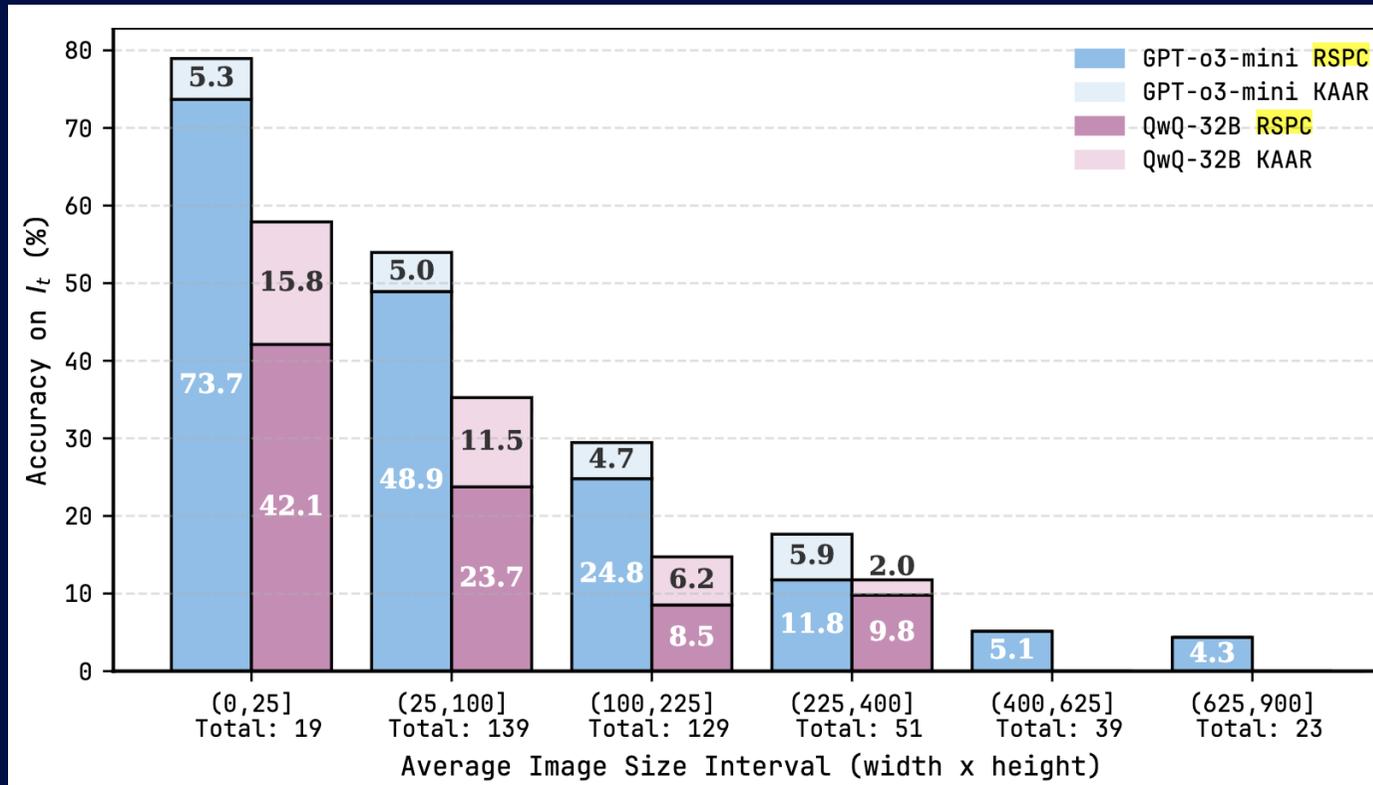
# Results – Prior integration



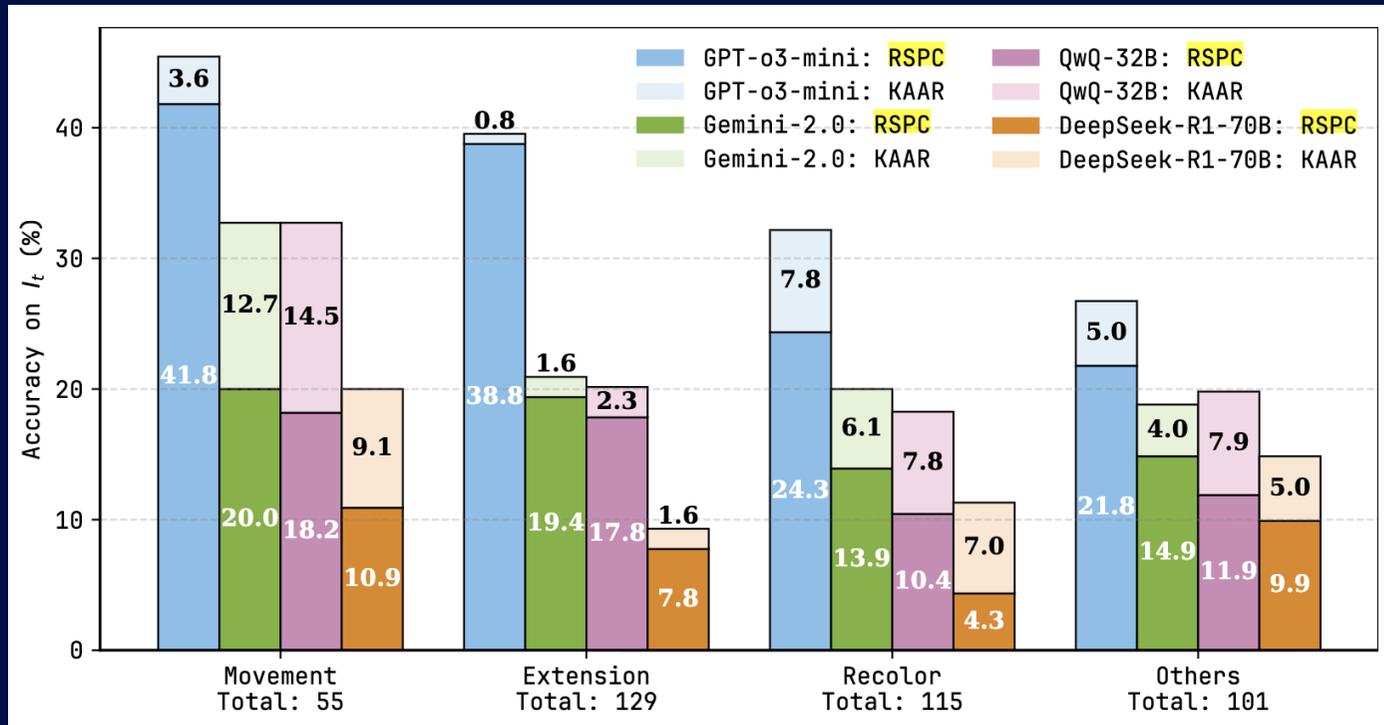
THE UNIVERSITY OF  
MELBOURNE



# Challenge – Image Size



# Challenge – Spatial Reasoning



# Knowledge augmented LLMs - Summary



**Generalisation** improves with Abstractions and Planning before coding, but LLM are far from solving the ARC challenge



From Reasoning to Generalization: Knowledge-Augmented LLMs for ARC Benchmark, Chao Lei et al. (Arxiv – work in progress)

# Take Aways



**Ambiguity in human language:** hard to capture correct semantics.

Hidden, **unexpressed knowledge** in descriptions.

Use coherence semantic filters + delete relaxed planning validation

Plenty of opportunities to integrate **LLM, Planning methods and modelling** in ARC and coding!

## Open problems

**Integrations of tractable** planning fragments for **LLM modelling and solving.**

Description languages to express and learn abstractions?

# Check out ICAPS 25



THE UNIVERSITY OF  
MELBOURNE

**Wednesday, 12 Nov (16:50 - 18:00) - PDDL-GenAI -- A Plugin for LLM-Based Support in Planning Domain Modeling**

*Jinbiao Wang, Rui Peng, Tong Chen, Chaowen Zeng, Weilin Wang, Sukai Huang, Nir Lipovetzky. ICAPS demo*

**Thursday, 13 Nov (11:30 - 11:50) - Chasing Progress, Not Perfection: Revisiting Strategies for End-to-End LLM Plan Generation**

*Sukai Huang, Nir Lipovetzky*

# Thanks!



THE UNIVERSITY OF  
MELBOURNE

**Contact** [nir.lipovetzky@unimelb.edu.au](mailto:nir.lipovetzky@unimelb.edu.au)

**More Info** <https://nirlipo.github.io/>