Language Models For Generalised PDDL Planning: Synthesising Sound and Programmatic Policies

VECTOR INSTITUTE | INSTITUT VECTEUR

Dillon Z. Chen

Johannes Zenn

Tristan Cinquin

Sheila A. McIlraith

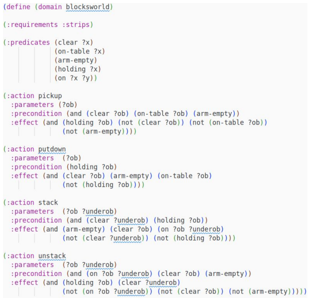
UNIVERSITY OF TORONTO

PDDL Planning

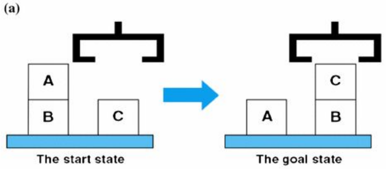
- sequential decision making on first-order, finite domain, symbolic models [1, 2]
- PDDL *de facto* formalism for fully-observable, deterministic transition models

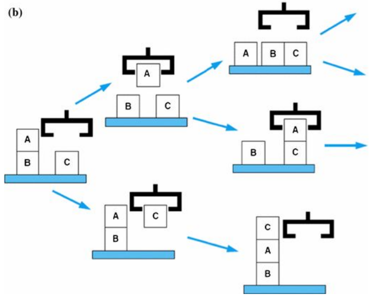
`INPUT:` *PDDL planning problem*

`OUTPUT:` *plan* – a sequence of actions that progresses the initial state to a goal



```
(define (domain blocksworld)

(: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)))))
```





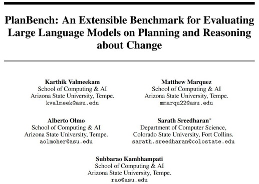
[1] McDermott et al., 1998
[2] Haslum et al., 2019

Q: How can we best use LLMs for PDDL planning?

A: By generating *sound* planning programs!

Language Models for PDDL Planning So Far: Approaches

(1) Generate Plans



PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change

Karthik Valmeekam
School of Computing & AI
Arizona State University, Tempe.
kvalmeek@asu.edu

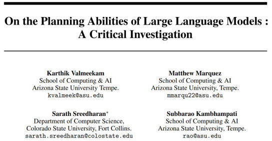
Matthew Marquez
School of Computing & AI
Arizona State University, Tempe.
mmarqu22@asu.edu

Alberto Olmo
School of Computing & AI
Arizona State University, Tempe.
aolmoher@asu.edu

Sarath Sreedharan*
Department of Computer Science,
Colorado State University, Fort Collins.
sarath.sreedharan@colostate.edu

Subbarao Kambhampati
School of Computing & AI
Arizona State University, Tempe.
rao@asu.edu

NeurIPS’23



On the Planning Abilities of Large Language Models : A Critical Investigation

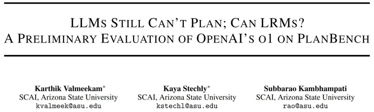
Karthik Valmeekam
School of Computing & AI
Arizona State University Tempe.
kvalmeek@asu.edu

Matthew Marquez
School of Computing & AI
Arizona State University, Tempe.
mmarqu22@asu.edu

Sarath Sreedharan*
Department of Computer Science,
Colorado State University, Fort Collins.
sarath.sreedharan@colostate.edu

Subbarao Kambhampati
School of Computing & AI
Arizona State University, Tempe.
rao@asu.edu

NeurIPS’23



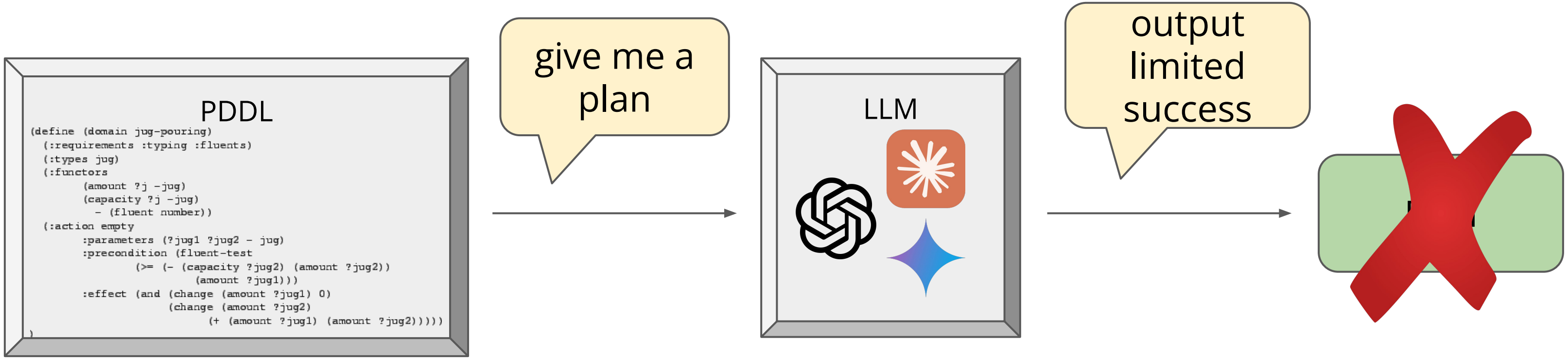
LLMs Still Can’t Plan; Can LRMs?
A Preliminary Evaluation of OpenAI’s o1 on PlanBench

Karthik Valmeekam*
SCAI, Arizona State University
kvalmeek@asu.edu

Kaya Stechly*
SCAI, Arizona State University
kstechl@asu.edu

Subbarao Kambhampati
SCAI, Arizona State University
rao@asu.edu

Arxiv’24



```
(define (domain jug-pouring)
  (:requirements :typing :fluents)
  (:types jug)
  (:functors
        (amount ?j -jug)
        (capacity ?j -jug)
          - (fluent number))
  (:action empty
        :parameters (?jug1 ?jug2 - jug)
        :precondition (fluent-test
                   (>= (- (capacity ?jug2) (amount ?jug2))
                              (amount ?jug1)))
        :effect (and (change (amount ?jug1) 0)
                     (change (amount ?jug2)
                            (+ (amount ?jug1) (amount ?jug2)))))
)
```

Language Models for PDDL Planning So Far: Approaches

(1) Generate Plans

PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change

On the Planning Abilities of Large Language Models : A Critical Investigation

LLMs Still Can't Plan; Can LRMs?
LIMINARY EVALUATION OF OPENAI'S O1 ON PLANBENCH

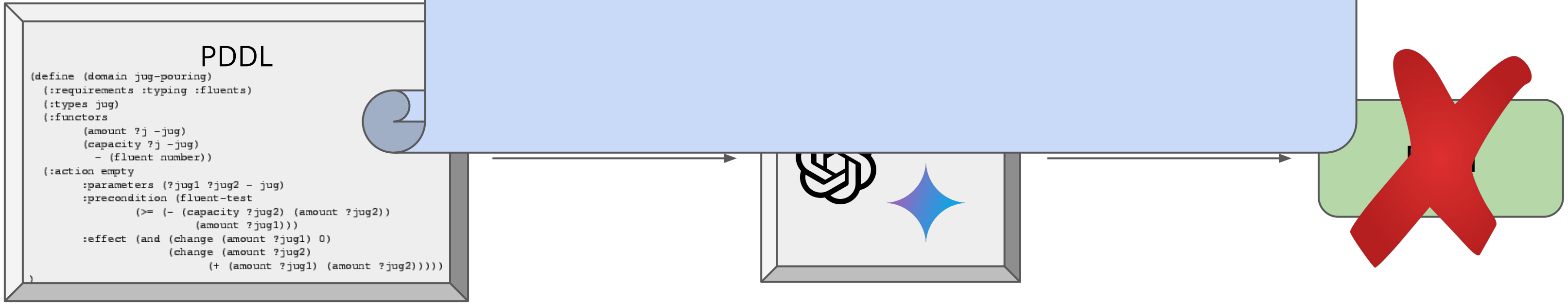
arthik Valmeekam*
SCAI, Arizona State University
kvalmeek@asu.edu

Kaya Stechly*
SCAI, Arizona State University
kstechl@asu.edu

Subbarao Kambhampati
SCAI, Arizona State University
rao@asu.edu

Arxiv'24

LLMs cannot plan reliably even when given a (PDDL) model



PDDL

```
(define (domain jug-pouring)
  (:requirements :typing :fluents)
  (:types jug)
  (:functors
        (amount ?j -jug)
        (capacity ?j -jug)
          - (fluent number))
  (:action empty
        :parameters (?jug1 ?jug2 - jug)
        :precondition (fluent-test
                (>= (- (capacity ?jug2) (amount ?jug2))
                      (amount ?jug1)))
        :effect (and (change (amount ?jug1) 0)
                  (change (amount ?jug2)
                        (+ (amount ?jug1) (amount ?jug2)))))
)
```

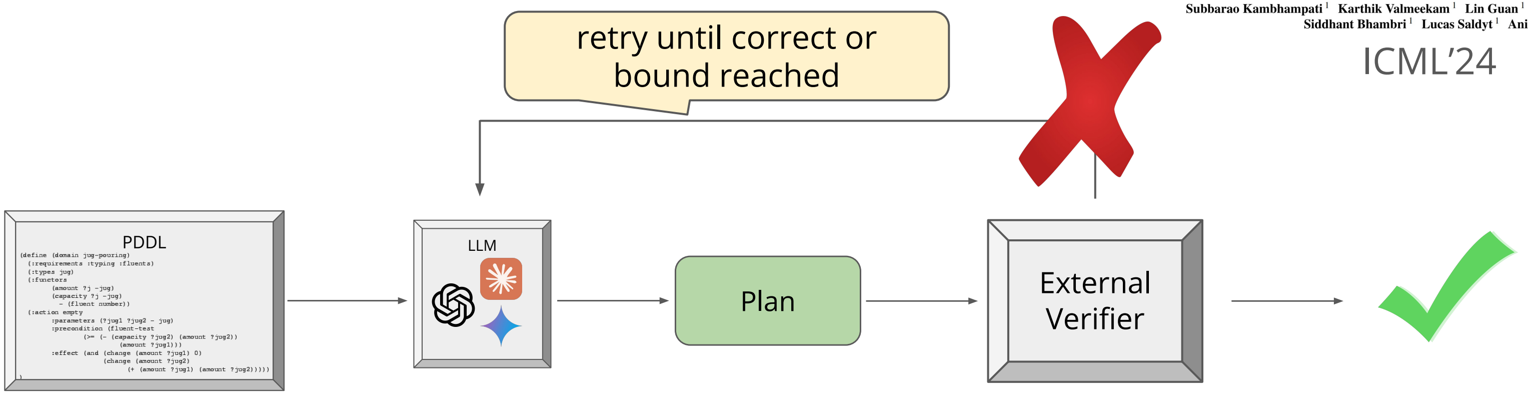
Language Models for PDDL Planning So Far: Approaches

(1a) “LLM-Modulo”: Generate Plans and Verify

**Position: LLMs Can’t Plan,
But Can Help Planning in LLM-Modulo Frameworks**

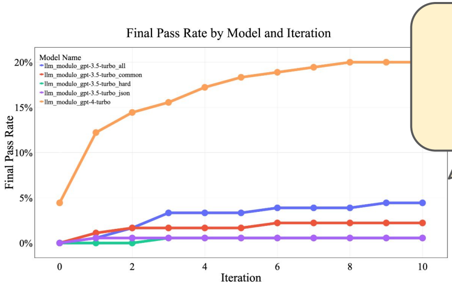
Subbarao Kambhampati[1] Karthik Valmeekam[1] Lin Guan[1] Mudit Verma[1] Kaya Stechly[1] Siddhant Bhambri[1] Lucas Saldyt[1] Anil Murthy[1]

ICML’24



retry until correct or bound reached

```
(define (domain jug-pouring)
  (:requirements :typing :fluents)
  (:types jug)
  (:functors
        (amount ?j -jug)
        (capacity ?j -jug)
          - (fluent number))
  (:action empty
        :parameters (?jug1 ?jug2 - jug)
        :precondition (fluent-test
                (>= (- (capacity ?jug2) (amount ?jug2))
                         (amount ?jug1)))
        :effect (and (change (amount ?jug1) 0)
                     (change (amount ?jug2)
                          (+ (amount ?jug1) (amount ?jug2))))
)
```



Still poor performance

Language Models for PDDL Planning So Far: Approaches

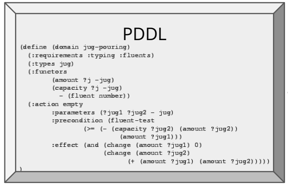
(1a) “LLM-Modulo”:

Position: LLMs Can’t Plan,
nning in LLM-Modulo Frameworks

i[1] Karthik Valmeekam[1] Lin Guan[1] Mudit Verma[1] Kaya Stechly[1]
ldhant Bhambri[1] Lucas Saldyt[1] Anil Murthy[1]

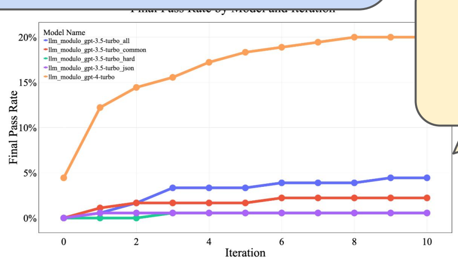
ICML’24

LLMs cannot plan reliably even when given a (PDDL) model and external verifiers



PDDL

```
(define (domain jug-pouring)
  (:requirements :typing :fluents)
  (:types jug)
  (:functors
        (amount ?j -jug)
        (capacity ?j -jug)
          - (fluent number))
  (:action empty
        :parameters (?jug1 ?jug2 - jug)
        :precondition (fluent-test
                (>= (- (capacity ?jug2) (amount ?jug2))
                        (amount ?jug1)))
        :effect (and (change (amount ?jug1) 0)
                     (change (amount ?jug2)
                        (+ (amount ?jug1) (amount ?jug2)))))
)
```



Still poor performance

Language Models for PDDL Planning So Far: Approaches

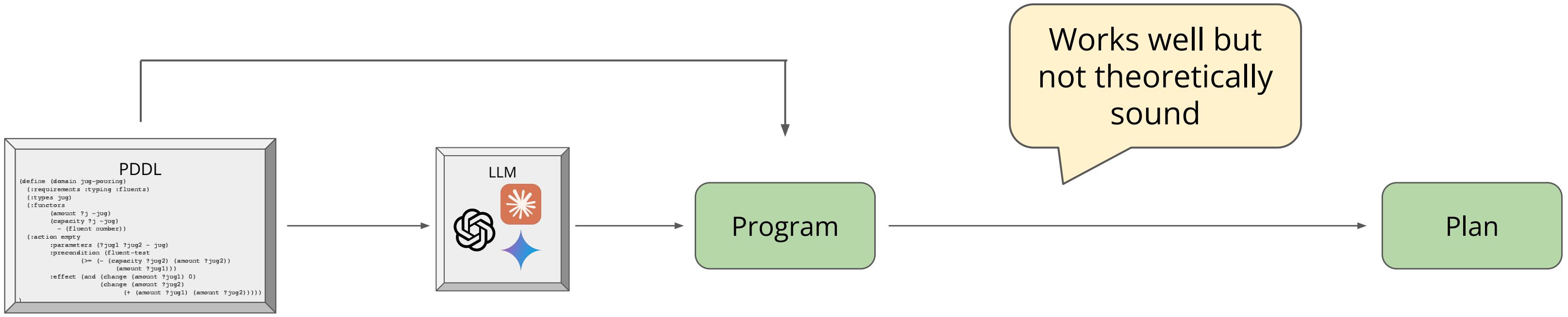
Generalized Planning in PDDL Domains with Pretrained Large Language Models

Tom Silver[1], Soham Dan[2], Kavitha Srinivas[2], Joshua Tenenbaum[1], Leslie Kaelbling[1], Michael Katz[2]

[1]MIT Computer Science and Artificial Intelligence Laboratory; [2]IBM Research

AAAI'24

(2a) Generate Planning Programs

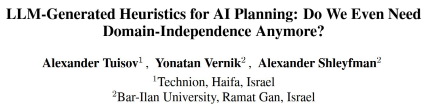


```
(define (domain jug-pouring)
  (:requirements :typing :fluents)
  (:types jug)
  (:functors
        (amount ?j -jug)
        (capacity ?j -jug)
          - (fluent number))
  (:action empty
        :parameters (?jug1 ?jug2 - jug)
        :precondition (fluent-test
                (>= (- (capacity ?jug2) (amount ?jug2))
                         (amount ?jug1)))
        :effect (and (change (amount ?jug1) 0)
                     (change (amount ?jug2)
                          (+ (amount ?jug1) (amount ?jug2)))))
)
```

Works well but not theoretically sound

Language Models for PDDL Planning So Far: Approaches

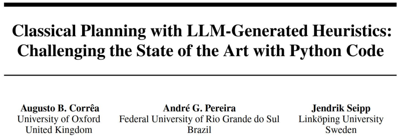
(2b) Generate Value Function Programs



LLM-Generated Heuristics for AI Planning: Do We Even Need Domain-Independence Anymore?

Alexander Tuisov[1], **Yonatan Vernik**[2], **Alexander Shleyfman**[2]
[1]Technion, Haifa, Israel
[2]Bar-Ilan University, Ramat Gan, Israel

Arxiv'25



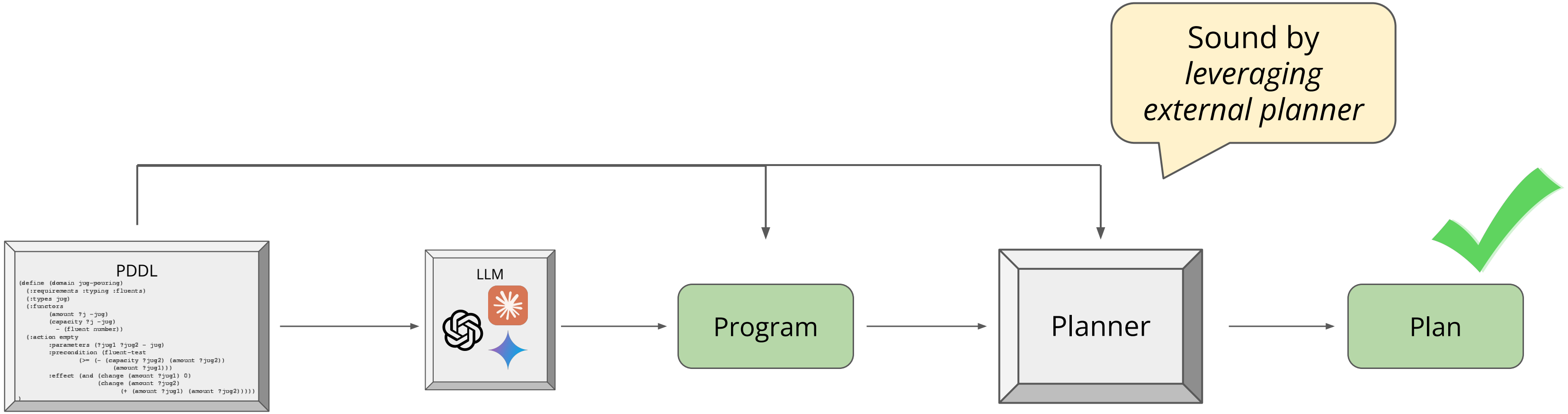
Classical Planning with LLM-Generated Heuristics: Challenging the State of the Art with Python Code

Augusto B. Corrêa
University of Oxford
United Kingdom

André G. Pereira
Federal University of Rio Grande do Sul
Brazil

Jendrik Seipp
Linköping University
Sweden

Arxiv'25

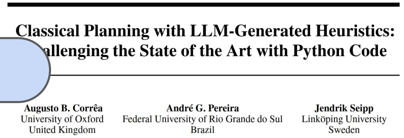


Sound by *leveraging external planner*

```
(define (domain jug-pouring)
  (:requirements :typing :fluents)
  (:types jug)
  (:functors
        (amount ?j -jug)
        (capacity ?j -jug)
          - (fluent number))
  (:action empty
        :parameters (?jug1 ?jug2 - jug)
        :precondition (fluent-test
                (>= (- (capacity ?jug2) (amount ?jug2))
                         (amount ?jug1)))
        :effect (and (change (amount ?jug1) 0)
                     (change (amount ?jug2)
                           (+ (amount ?jug1) (amount ?jug2)))))
)
```

Language Models for PDDL Planning So Far: Approaches

(2b) Generate Valu



Classical Planning with LLM-Generated Heuristics: allenging the State of the Art with Python Code

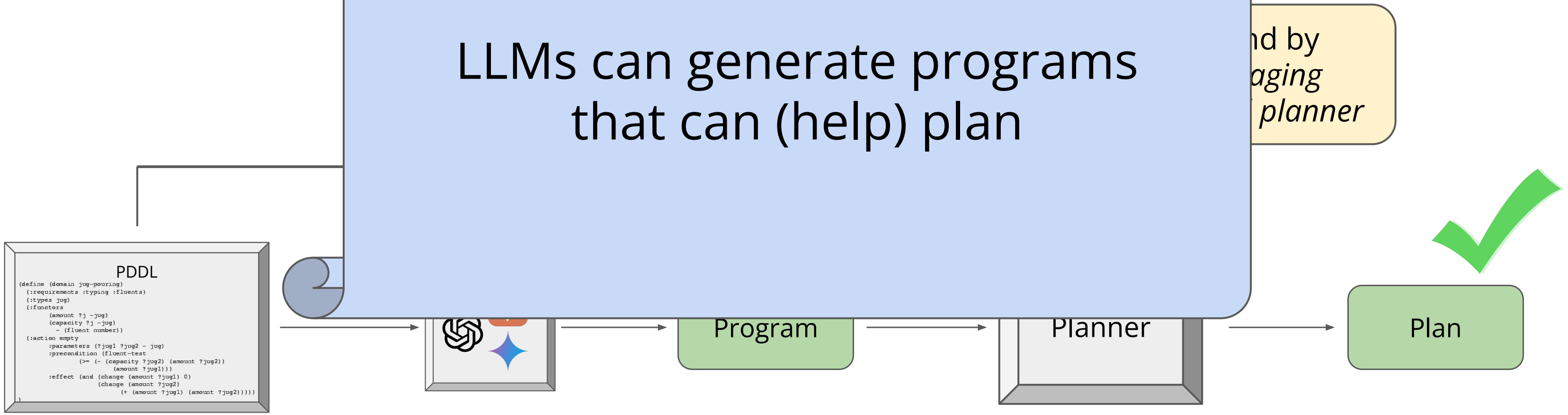
Augusto B. Corrêa
University of Oxford
United Kingdom

André G. Pereira
Federal University of Rio Grande do Sul
Brazil

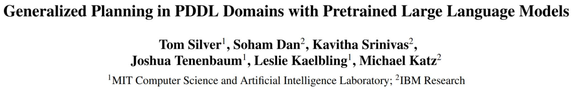
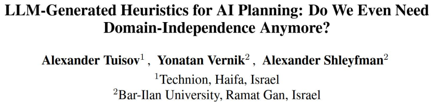
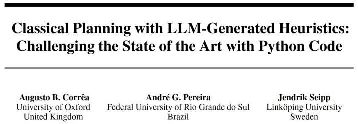
Jendrik Seipp
Linköping University
Sweden

Arxiv'25

LLMs can generate programs that can (help) plan

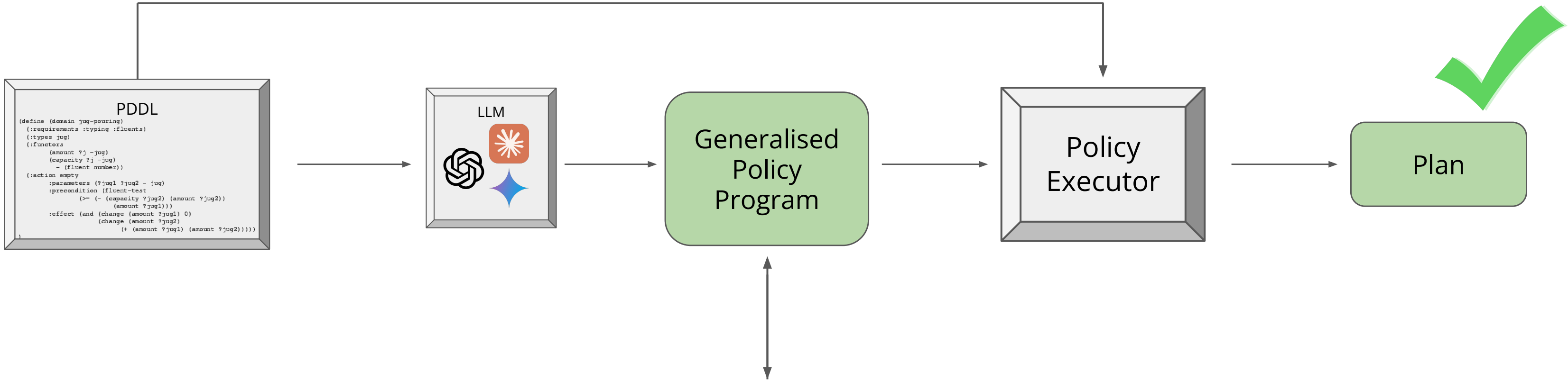


```
(define (domain jug-pouring)
  (:requirements :typing :fluents)
  (:types jug)
  (:functors
        (amount ?j -jug)
        (capacity ?j -jug)
          - (fluent number))
  (:action empty
        :parameters (?jug1 ?jug2 - jug)
        :precondition (fluent-test
                (>= (- (capacity ?jug2) (amount ?jug2))
                        (amount ?jug1)))
        :effect (and (change (amount ?jug1) 0)
                     (change (amount ?jug2)
                          (+ (amount ?jug1) (amount ?jug2)))))
)
```

Approach	Sound	No External Planner
 **Generalized Planning in PDDL Domains with Pretrained Large Language Models** **Tom Silver[1], Soham Dan[2], Kavitha Srinivas[2], Joshua Tenenbaum[1], Leslie Kaelbling[1], Michael Katz[2]** [1]MIT Computer Science and Artificial Intelligence Laboratory; [2]IBM Research AAAI’24		✓
 **LLM-Generated Heuristics for AI Planning: Do We Even Need Domain-Independence Anymore?** **Alexander Tuisov[1], Yonatan Vernik[2], Alexander Shleyfman[2]** [1]Technion, Haifa, Israel [2]Bar-Ilan University, Ramat Gan, Israel Arxiv’25	✓	
 **Classical Planning with LLM-Generated Heuristics: Challenging the State of the Art with Python Code** **Augusto B. Corrêa** University of Oxford United Kingdom **André G. Pereira** Federal University of Rio Grande do Sul Brazil **Jendrik Seipp** Linköping University Sweden Arxiv’25	✓	

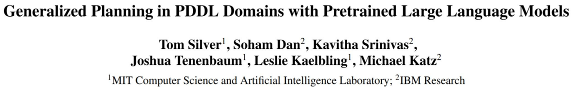
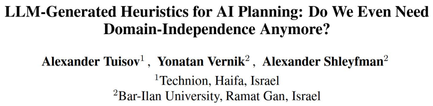
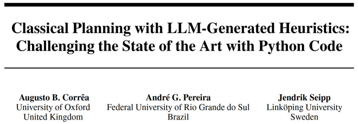
Contribution 1:
Sound LM Programs without External Planners

LLMs for Generating Sound Policy Programs



```
(define (domain jug-pouring)
  (:requirements :typing :fluents)
  (:types jug)
  (:functors
        (amount ?j -jug)
        (capacity ?j -jug)
          - (fluent number))
  (:action empty
        :parameters (?jug1 ?jug2 - jug)
        :precondition (fluent-test
                (>= (- (capacity ?jug2) (amount ?jug2))
                         (amount ?jug1)))
        :effect (and (change (amount ?jug1) 0)
                     (change (amount ?jug2)
                           (+ (amount ?jug1) (amount ?jug2)))))
)
```

$\pi(\mathit{Action} \mid \mathit{State}, \mathit{Problem}) \in$ set of applicable actions in *State*

Approach	Sound	No External Planner
 **Generalized Planning in PDDL Domains with Pretrained Large Language Models** **Tom Silver[1], Soham Dan[2], Kavitha Srinivas[2], Joshua Tenenbaum[1], Leslie Kaelbling[1], Michael Katz[2]** [1]MIT Computer Science and Artificial Intelligence Laboratory; [2]IBM Research AAAI'24		✓
 **LLM-Generated Heuristics for AI Planning: Do We Even Need Domain-Independence Anymore?** **Alexander Tuisov[1], Yonatan Vernik[2], Alexander Shleyfman[2]** [1]Technion, Haifa, Israel [2]Bar-Ilan University, Ramat Gan, Israel Arxiv'25	✓	
 **Classical Planning with LLM-Generated Heuristics: Challenging the State of the Art with Python Code** **Augusto B. Corrêa** University of Oxford United Kingdom **André G. Pereira** Federal University of Rio Grande do Sul Brazil **Jendrik Seipp** Linköping University Sweden Arxiv'25	✓	
Language Models For Generalised PDDL Planning: Synthesising Sound and Programmatic Policies PRL@RLC'25!! (ours)	✓	✓

Results: LMs can generate powerful, sound planning programs

- Baselines: state-of-the-art planners
- Metric (↑): numbe limit

Planner		ed Ratio
GBFS + h^{FF}		
WL-GOOSE		
LAMA		
[ours] LMPlan		
[ours] **LMPlan + Search**	**630**	**70%**

LLMs competitive with
state-of-the-art PDDL planners

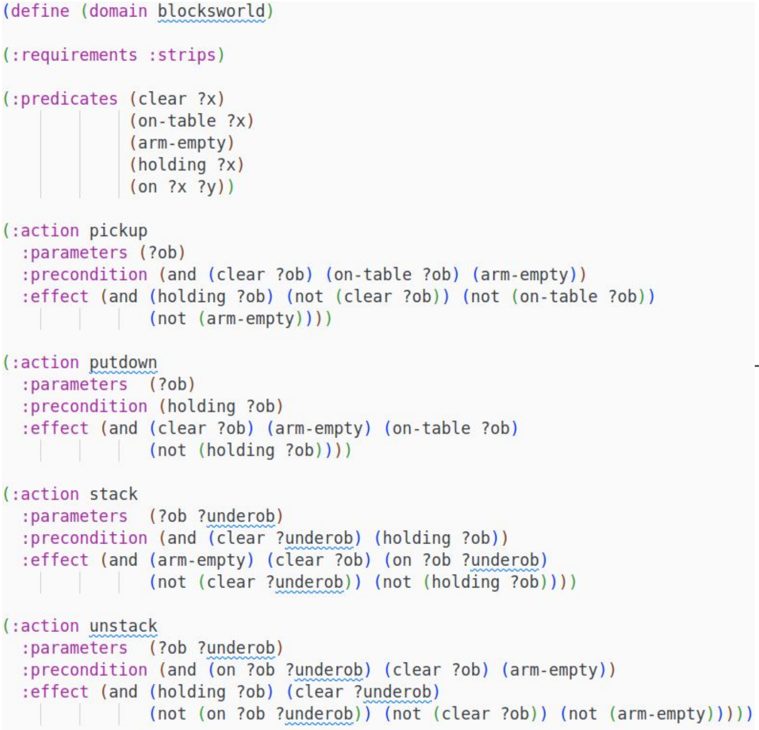
Results: LMs can generate powerful, sound planning programs

- Baselines: state-of-the-art planners
- Metric (↑): number of problems solved correctly in time and memory limit

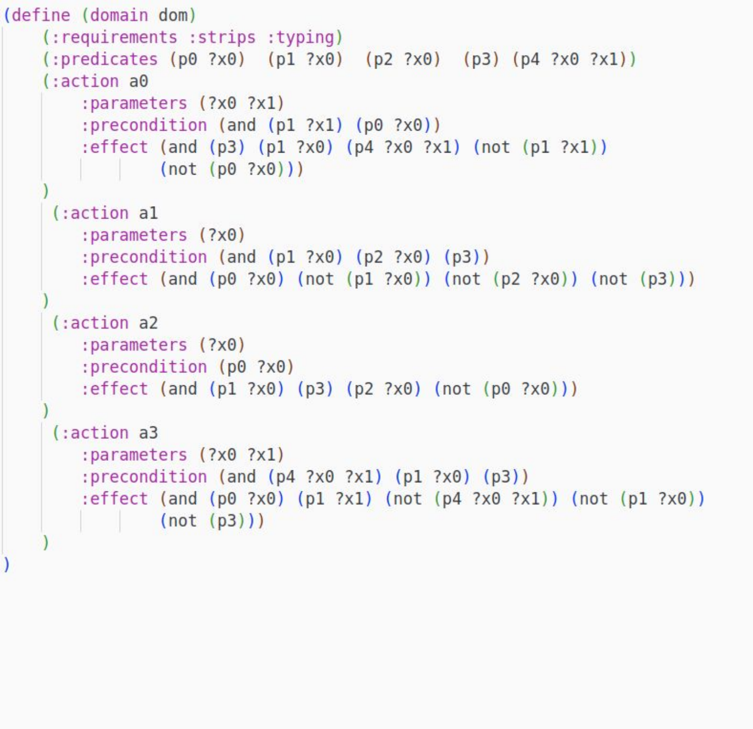
Planner	Problems Solved	Problems Solved Ratio
GBFS + h^{FF}	430	48%
WL-GOOSE	502	56%
LAMA	557	62%
[ours] LMPlan	563	63%
[ours] **LMPlan + Search**	**630**	**70%**

Contribution 2:
Sound Programs Can Plan with Meaningless Symbols

Setup: Replace Natural Language with Meaningless Symbols



```
(define (domain blocksworld)

(: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)))))
```



```
(define (domain dom)
    (:requirements :strips :typing)
    (:predicates (p0 ?x0)  (p1 ?x0)  (p2 ?x0)  (p3) (p4 ?x0 ?x1))
    (:action a0
        :parameters (?x0 ?x1)
        :precondition (and (p1 ?x1) (p0 ?x0))
        :effect (and (p3) (p1 ?x0) (p4 ?x0 ?x1) (not (p1 ?x1))
                (not (p0 ?x0)))
    )
     (:action a1
        :parameters (?x0)
        :precondition (and (p1 ?x0) (p2 ?x0) (p3))
        :effect (and (p0 ?x0) (not (p1 ?x0)) (not (p2 ?x0)) (not (p3)))
    )
     (:action a2
        :parameters (?x0)
        :precondition (p0 ?x0)
        :effect (and (p1 ?x0) (p3) (p2 ?x0) (not (p0 ?x0)))
    )
     (:action a3
        :parameters (?x0 ?x1)
        :precondition (and (p4 ?x0 ?x1) (p1 ?x0) (p3))
        :effect (and (p0 ?x0) (p1 ?x1) (not (p4 ?x0 ?x1)) (not (p1 ?x0))
                (not (p3)))
    )
)
```

Experimental Results

LLM performance (surprisingly!) does not degrade significantly when removing language from inputs

Language Models For Generalised PDDL Planning: Synthesising Sound and Programmatic Policies

VECTOR INSTITUTE | INSTITUT VECTEUR

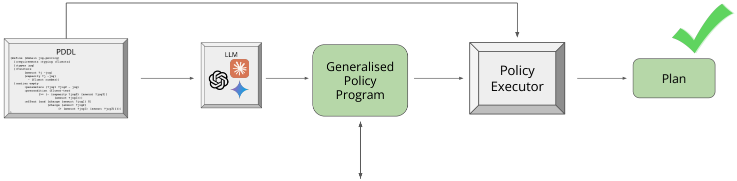
Dillon Z. Chen   Johannes Zenn   Tristan Cinquin   Sheila A. McIlraith

UNIVERSITY OF TORONTO

Q: How can we best use LLMs for PDDL planning?

A: By generating *sound* planning programs!

Contribution 1: Sound LM Programs without External Planners

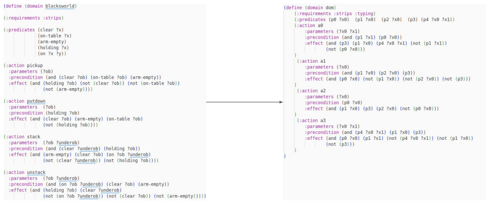


```
(define (domain jug-pouring)
  (:requirements :typing :fluents)
  (:types jug)
  (:functors
        (amount ?j -jug)
        (capacity ?j -jug)
          - (fluent number))
  (:action empty
        :parameters (?jug1 ?jug2 - jug)
        :precondition (fluent-test
                (>= (- (capacity ?jug2) (amount ?jug2))
                        (amount ?jug1)))
        :effect (and (change (amount ?jug1) 0)
                     (change (amount ?jug2)
                        (+ (amount ?jug1) (amount ?jug2)))))
)
```

$\pi(\mathit{Action} \mid \mathit{State}, \mathit{Problem}) \in$ set of applicable actions in *State*

Planner	Problems Solved	Problems Solved Ratio
GBFS + h^{FF}	430	48%
WL-GOOSE	502	56%
LAMA	557	62%
LMPlan	563	63%
LMPlan + Search	630	70%

Contribution 2: Sound Programs Can Plan with Meaningless Symbols



```
(define (domain blocksworld)

(: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)))))
```

```
(define (domain dom)
    (:requirements :strips :typing)
    (:predicates (p0 ?x0)  (p1 ?x0)  (p2 ?x0)  (p3) (p4 ?x0 ?x1))
    (:action a0
        :parameters (?x0 ?x1)
        :precondition (and (p1 ?x1) (p0 ?x0))
        :effect (and (p3) (p1 ?x0) (p4 ?x0 ?x1) (not (p1 ?x1))
                (not (p0 ?x0)))
    )
     (:action a1
        :parameters (?x0)
        :precondition (and (p1 ?x0) (p2 ?x0) (p3))
        :effect (and (p0 ?x0) (not (p1 ?x0)) (not (p2 ?x0)) (not (p3)))
    )
     (:action a2
        :parameters (?x0)
        :precondition (p0 ?x0)
        :effect (and (p1 ?x0) (p3) (p2 ?x0) (not (p0 ?x0)))
    )
     (:action a3
        :parameters (?x0 ?x1)
        :precondition (and (p4 ?x0 ?x1) (p1 ?x0) (p3))
        :effect (and (p0 ?x0) (p1 ?x1) (not (p4 ?x0 ?x1)) (not (p1 ?x0))
                (not (p3)))
    )
)
```

	Bl	Ch	Fe	Fl	Mi	Ro	Sa	So	Sp	Tr	Σ
V_{sem}	33	15	59	2	63	32	60	32	46	55	397
V_{sym}	33	24	61	1	70	34	48	30	63	42	406
π_{sem}	90	11	90	0	90	12	90	0	90	90	563
π_{sym}	1	12	90	0	90	46	6	0	90	89	424