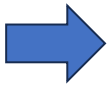
Bridging Planning and Reasoning in Natural Language with Foundational Models

Brief introduction to AI Planning

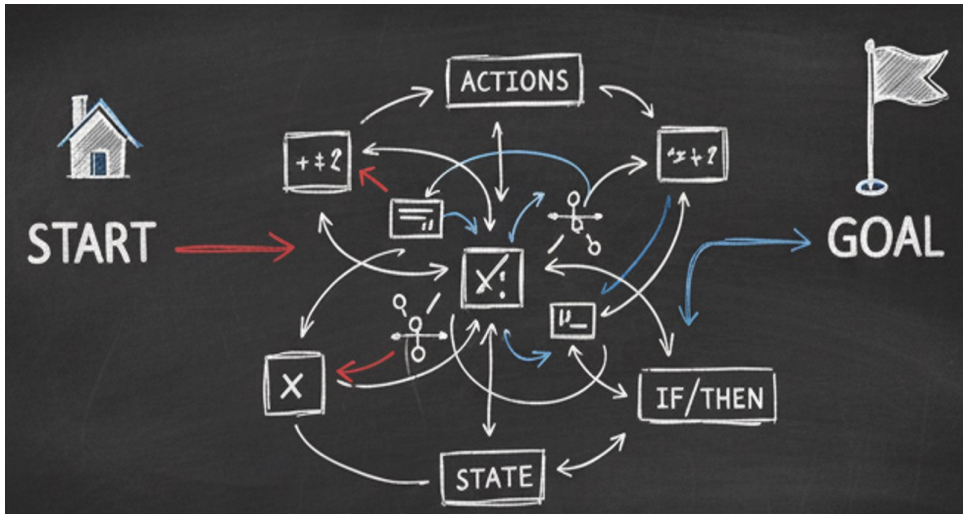
Shirin Sohrabi

Planning in the Era of Language Models



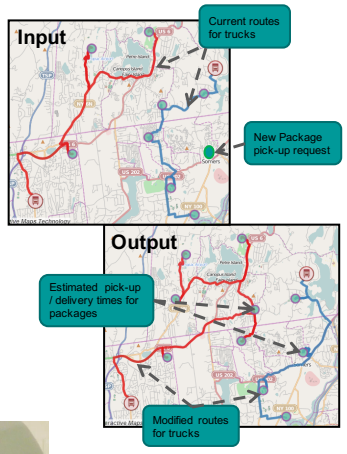
- **Introduction: What is Planning**
  - What kind of planning problem do I have aka Planning Formalisms
  - What kind of solutions am I looking for aka Computational Problems
  - How can I describe my planning problem aka Planning Languages
- **Solving Planning Problems**
  - Pre-LLM era
  - Early LLM era
  - Modern LLM era

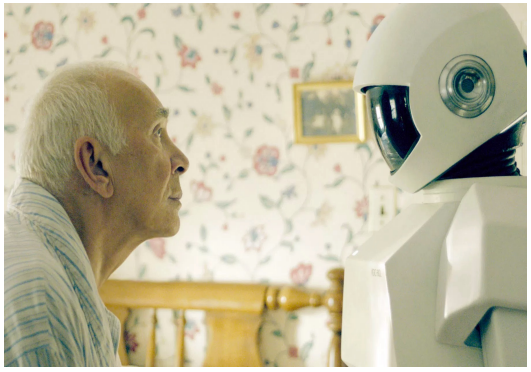
What is AI Planning



https://www.odtap.com/2018/10

AI Planning is everywhere

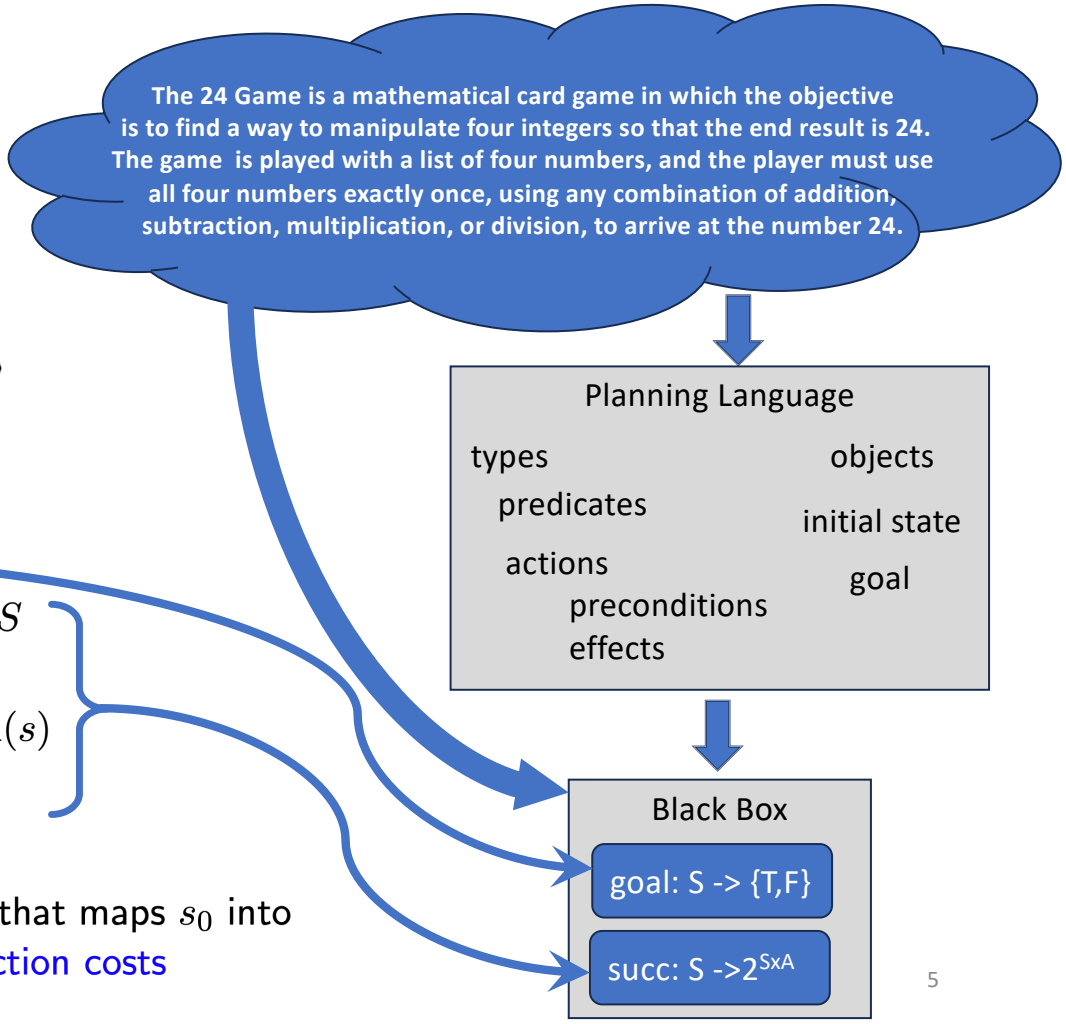




Source: https://www.livemint.com/Opinion/tijzm8flw2RY98Jvdm8LzJ/Opinion--Cyber-security-a-complex-behaviour-problem.html
Source: https://www.rigzone.com/news/schlumberger_in_400mm_deal_to_sell_drilling_assets-15-may-2019-158842-article/

4

Planning Problem



Mathematical Model: $\mathcal{S} = \langle S, s_0, S_G, A, f, c \rangle$

- finite and discrete state space S
- a known initial state $s_0 \in S$
- a set $S_G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each $s \in S$
- a deterministic transition function
  $s' = f(a, s)$ for $a \in A(s)$
- non-negative action costs $c(a, s)$

A solution is a sequence of applicable actions that maps s_0 into S_G, and it is optimal if it minimizes sum of action costs

Planning Formalisms

- Classical planning: initial state is **known**, action dynamics is **deterministic** and **instantaneous**, **discrete and finite** state space, **flat** hierarchies, **hard** goals that must be achieved
- Numeric Planning: numeric state variables, **infinite** state spaces
- Net-benefit/oversubscription planning: **utility of states, possibly no hard goals**
- Conformant planning: initial state is **uncertain**
- Probabilistic planning: action dynamics is **probabilistic**, and state spaces **do not have to be discrete or finite**
- Non-deterministic planning (ND): action dynamics is **non-deterministic**
- Temporal Planning: actions have **durations** and **temporal constraints**
- Hierarchical Task Network (HTN): initial state/goal are task networks (a set of tasks and constraints), with **recursive decomposition of high-level tasks into lower-level sub-tasks**

Computational Problems

Classical/Numeric Planning

- **Agile planning**: find a plan, quicker is better
- **Satisficing planning**: find a plan, cheaper plans a better
- **Cost-optimal planning**: find a plan that minimizes summed operator cost
- **Top-k planning**: find k plans such that no cheaper plans exist
- **Top-quality planning**: find all plans up to a certain cost
- **Diverse planning**: variety of problems, aiming at obtaining diverse set of plans, considering plan quality as well

Beyond Classical Planning (examples)

- **Net-benefit planning**: find a plan that minimizes the difference between utility of end state and summed operator cost
- **Oversubscription planning**: find a plan that maximizes the utility of end state, bounding summed operator cost
- **FOND planning**: find a policy that guarantees eventual goal achievement under fairness of outcomes assumption
- **PO Probabilistic planning**: find a policy that maximizes expected utility

Representing the Knowledge (Example: PDDL)

http://editor.planning.domains/

```
(define (problem mixed)
   (:domain miconic)
   (:objects p0 p1 p2 p3 f0 f1 f2 f3 f4 f5 f6 f7)

(:init
(passenger p0)(passenger p1)(passenger p2)(passenger p3)
(floor f0)(floor f1)(floor f2)(floor f3)(floor f4)
(floor f5)(floor f6)(floor f7)

(above f0 f1)(above f0 f2)(above f0 f3)(above f0 f4)
(above f0 f5)(above f0 f6)(above f0 f7)(above f1 f2)
(above f1 f3)(above f1 f4)(above f1 f5)(above f1 f6)
(above f1 f7)(above f2 f3)(above f2 f4)(above f2 f5)
(above f2 f6)(above f2 f7)(above f3 f4)(above f3 f5)
(above f3 f6)(above f3 f7)(above f4 f5)(above f4 f6)
(above f4 f7)(above f5 f6)(above f5 f7)(above f6 f7)

(origin p0 f0)(destin p0 f5)(origin p1 f7)(destin p1 f4)
(origin p2 f0)(destin p2 f7)(origin p3 f1)(destin p3 f6)

(lift-at f0))

(:goal (and
(served p0)(served p1)(served p2)(served p3))))
```

```
(define (domain miconic)
  (:requirements :strips)

(:predicates
(origin ?person ?floor)
(floor ?floor)
(passenger ?passenger)
(destin ?person ?floor)
(above ?floor1 ?floor2)
(boarded ?person)
(served ?person)
(lift-at ?floor))

(:action board
  :parameters (?f ?p)
  :precondition (and (floor ?f) (passenger ?p)(lift-at ?f) (origin ?p ?f))
  :effect (boarded ?p))

(:action depart
  :parameters (?f  ?p)
  :precondition (and (floor ?f) (passenger ?p) (lift-at ?f) (destin ?p ?f)
             (boarded ?p))
  :effect (and (not (boarded ?p))
          (served ?p)))

(:action up
  :parameters (?f1 ?f2)
  :precondition (and (floor ?f1) (floor ?f2) (lift-at ?f1) (above ?f1 ?f2))
  :effect (and (lift-at ?f2) (not (lift-at ?f1))))

(:action down
  :parameters (?f1 ?f2)
  :precondition (and (floor ?f1) (floor ?f2) (lift-at ?f1) (above ?f2 ?f1))
  :effect (and (lift-at ?f2) (not (lift-at ?f1)))))
```

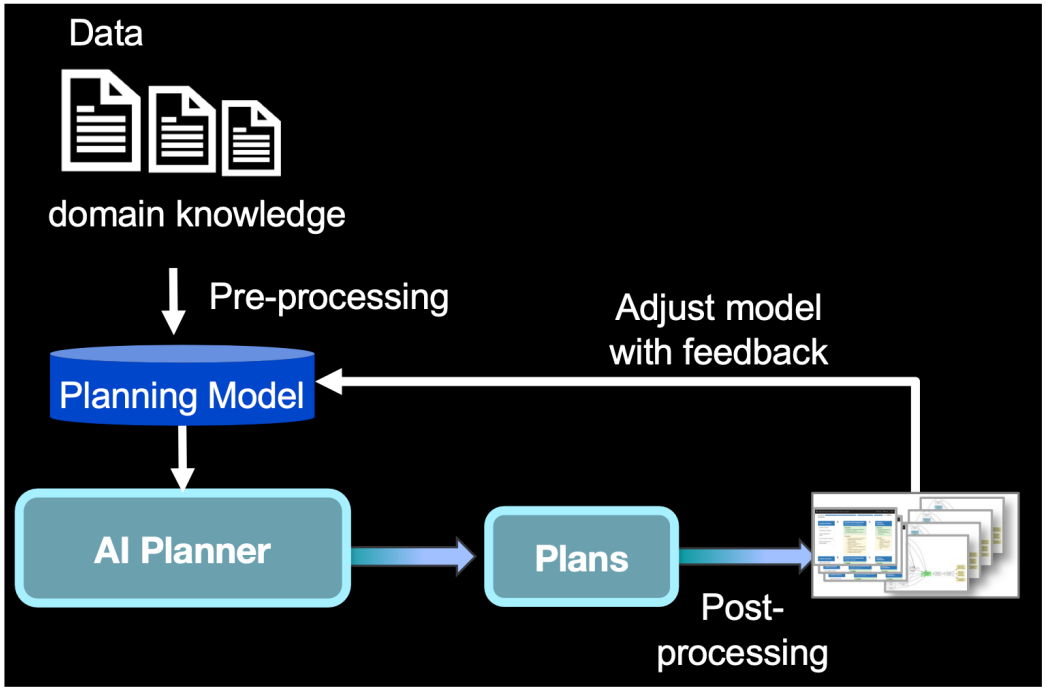
(up f0 f1)
(up f1 f7)
(board f7 p1)
(down f7 f4)
(depart f4 p1)
(down f4 f0)
(board f0 p0)
(up f0 f5)
(depart f5 p0)
(up f5 f6)
(down f6 f0)
(board f0 p2)
(up f0 f7)
(depart f7 p2)
(down f7 f1)
(board f1 p3)
(up f1 f6)
(depart f6 p3)

Planning in the Era of Language Models

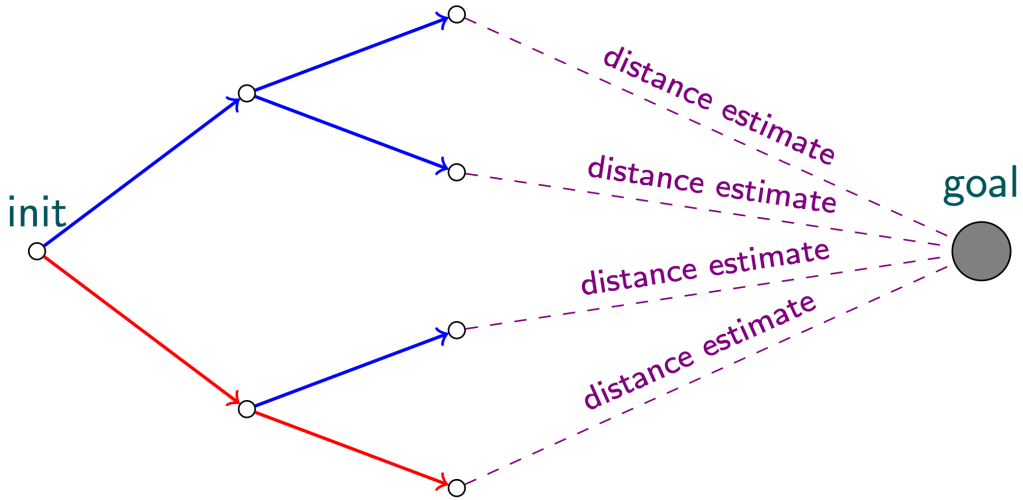
- **Introduction: What is Planning**
  - What kind of planning problem do I have aka Planning Formalisms
  - What kind of solutions am I looking for aka Computational Problems
  - How can I describe my planning problem aka Planning Languages
- **Solving Planning Problems**
  - Pre-LLM era
  - Early LLM era
  - Modern LLM era

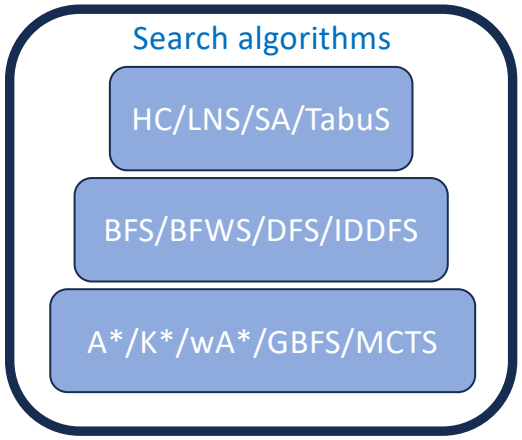
General Framework / Problems

- Model acquisition – how to derive / learn the planning model
- Planning – how to efficiently find solutions in the model
- Execution – how to efficiently execute model solutions in the environment



Solving Classical PDDL Planning





Major Planners/tools

- Fast-Forward (FF): classical satisficing, numeric, conformant, contingent (Hoffmann & Nebel, 2001)
- Fast Downward: classical, cost-optimal, satisficing, agile, cost-bouned, OSP, FOND, probabilistic, temporal (Helmert et al., 2006)
- SHOP2: HTN planning (Nau et al., 2003)
- LPG: classical, satisficing, numeric, temporal, diverse (Gerevini & Serina 2002)
- FOND planner PRP (Muise et al., 2012, 2014)
- OSP planners (Katz & Keyder 2019, Katz & Speck 2021)
- Top-k planners: K* (Katz et al., 2018), SymK (Speck et al., 2020)
- Forbid-iterative collection of planners for top-k, top-quality, diverse (Katz & Sohrabi 2020, Katz et al., 2020)

….

Planning in the Era of Language Models

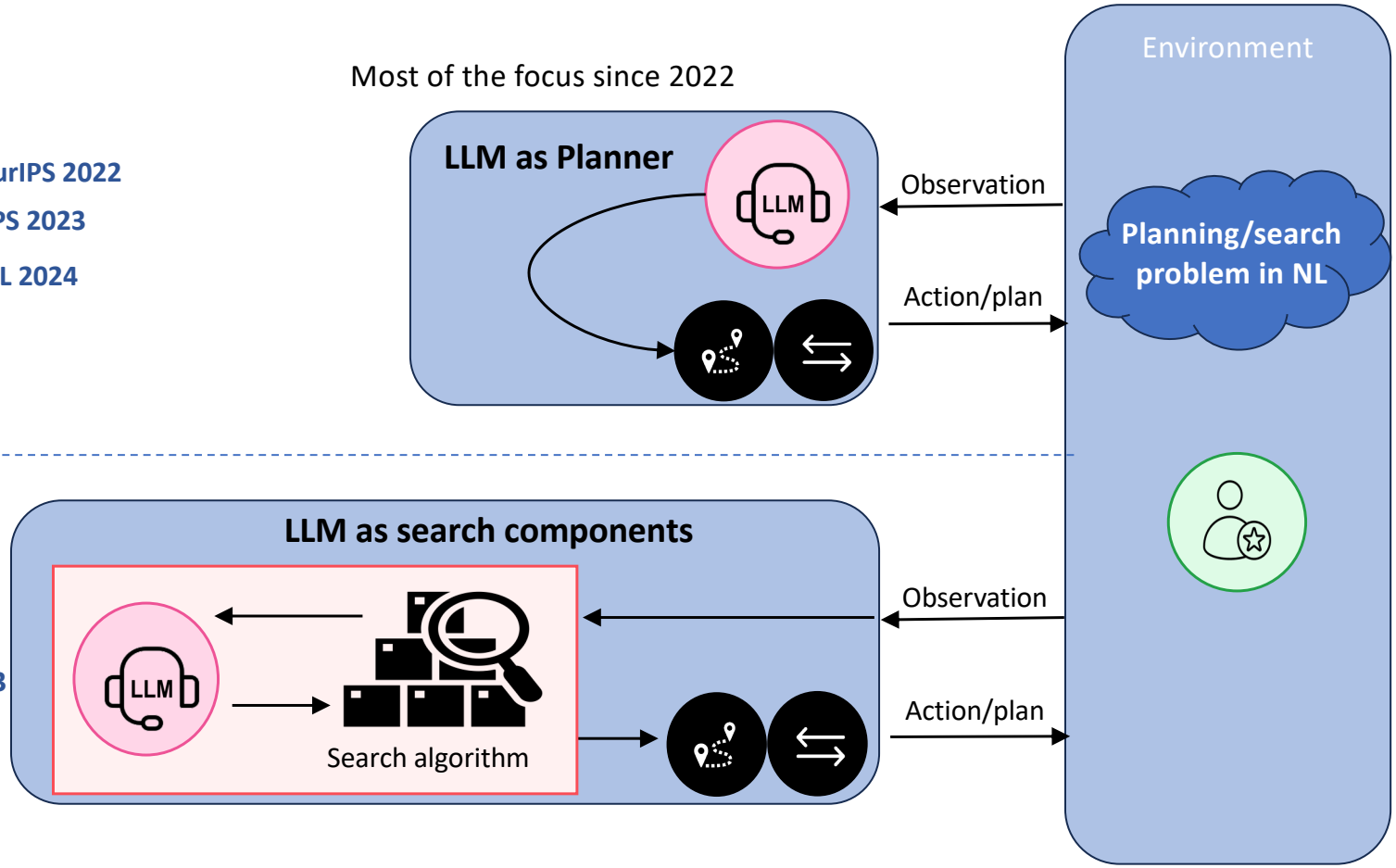
- **Introduction: What is Planning**
  - What kind of planning problem do I have aka Planning Formalisms
  - What kind of solutions am I looking for aka Computational Problems
  - How can I describe my planning problem aka Planning Languages
- **Solving Planning Problems**
  - Pre-LLM era
  - Early LLM era
  - Modern LLM era

Solving NL/PDDL Planning Problems

Most of the focus since 2022

Silver et al. FMDM@NeurIPS 2022
Valmeekam et al. NeurIPS 2023
Kambhampati et al. ICML 2024
Bohnet et al. Arxiv 2024
Zhao et al. ICLR 2025

Yao et al. ICLR 2023
Xu et al. Arxiv 2023
Hao et al. EMNLP 2023
Yao et al. NeurIPS 2023
Shinn et al. NeurIPS 2023
Zhou et al. ICML 2024
Sel et al. ICML 2024
Besta et al. AAAI 2024



What are the properties of these algorithms?

Sound? No! If validator exists, we can make it sound

Complete? No!

Optimal? No!

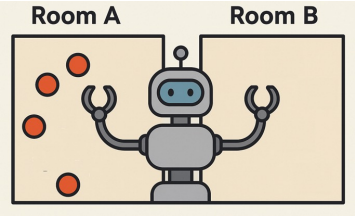
Computational complexity?

- Hard to measure. Most expensive operation is LLM call
- Computational effort vs. state space portion explored

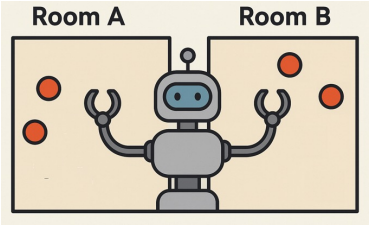
		24Game		Crossword		BlocksWorld		PrOntoQA	
Approach	Complexity	States	Calls	States	Calls	States	Calls	States	Calls
IO	O(D)	0.02%	1362	4e-9%	20	0.5%	502	4%	4000
CoT	O(D)	0.02%	1362	4e-9%	20	0.5%	502	4%	4000
ReAct	O(LD)	0.07%	4086	4e-8%	200	7.8%	8032	24.6%	24K
ReWOO	O(LD)	0.07%	4086	4e-8%	200	7.8%	8032	24.6%	24K
RAP	O(TbLD)	3.3%	245K	2e-6%	12K	388%	482K	1229%	1.44M
ToT	O(bmLD)	1.6%	102K	1e-6%	5K	194%	201K	615%	600K
GoT	O(bLD)	0.3%	20K	2e-7%	1K	39%	40K	122%	120K
Reflection	O(LTD)	0.7%	68K	4e-7%	2.4K	77.6%	90K	245%	320K
LATS	O(TbLD)	3.3%	286K	2e-6%	14K	388%	562K	1229%	1.68M

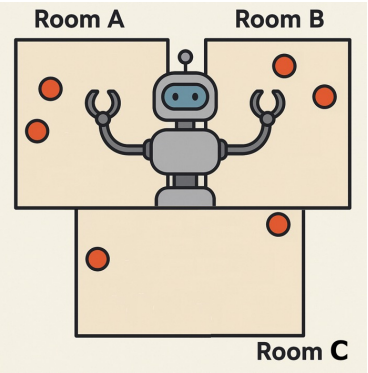
Katz et al, NeurIPS 2024, **Thought of Search: Planning with Language Models Through The Lens of Efficiency**

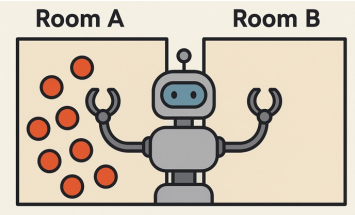
Why did it work in the first place?



(pick o1 A L)
(pick o2 A R)
(move A B)
(drop o1 B L)
(drop o2 B R)
(move B A)
(pick o3 A L)
(pick o4 A R)
(move A B)
(drop o3 B L)
(drop o4 B R)







(pick o1 A L)
(pick o2 A R)
(move A B)
(drop o1 B L)
(drop o2 B R)
(move B A)
…
(drop o7 B L)
(drop o8 B R)

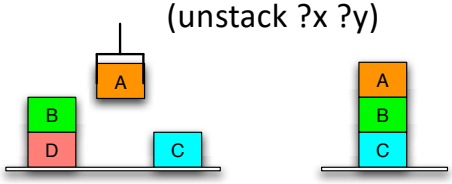
Conclusions (see [1]):

- Be aware of instance generator limitations
- Show generalization outside of training set
- Show performance on multiple domains

[1] Katz et al, Arxiv 2025, **Make Planning Research Rigorous Again!**

Why did it work in the first place?

(pickup ?x)
(putdown ?x)
(stack ?x ?y)
(unstack ?x ?y)



How do plans look like?

- pickup -> stack
- stack -> unstack | pickup
- unstack -> stack | putdown
- putdown -> unstack | pickup

What does it mean to restrict $|\pi| \leq 10$?

- At most 5 blocks are moved (even if the instance has 100s of blocks)
- Total of 1331 possible plan patterns
- When trained on a large collection of instances, most (all?) possible plan patterns appear in the training set

Conclusions (see [1]):

- There are many planning domains out there and BlocksWorld is among the simplest
- Show generalization outside of training set
- Show performance on multiple domains

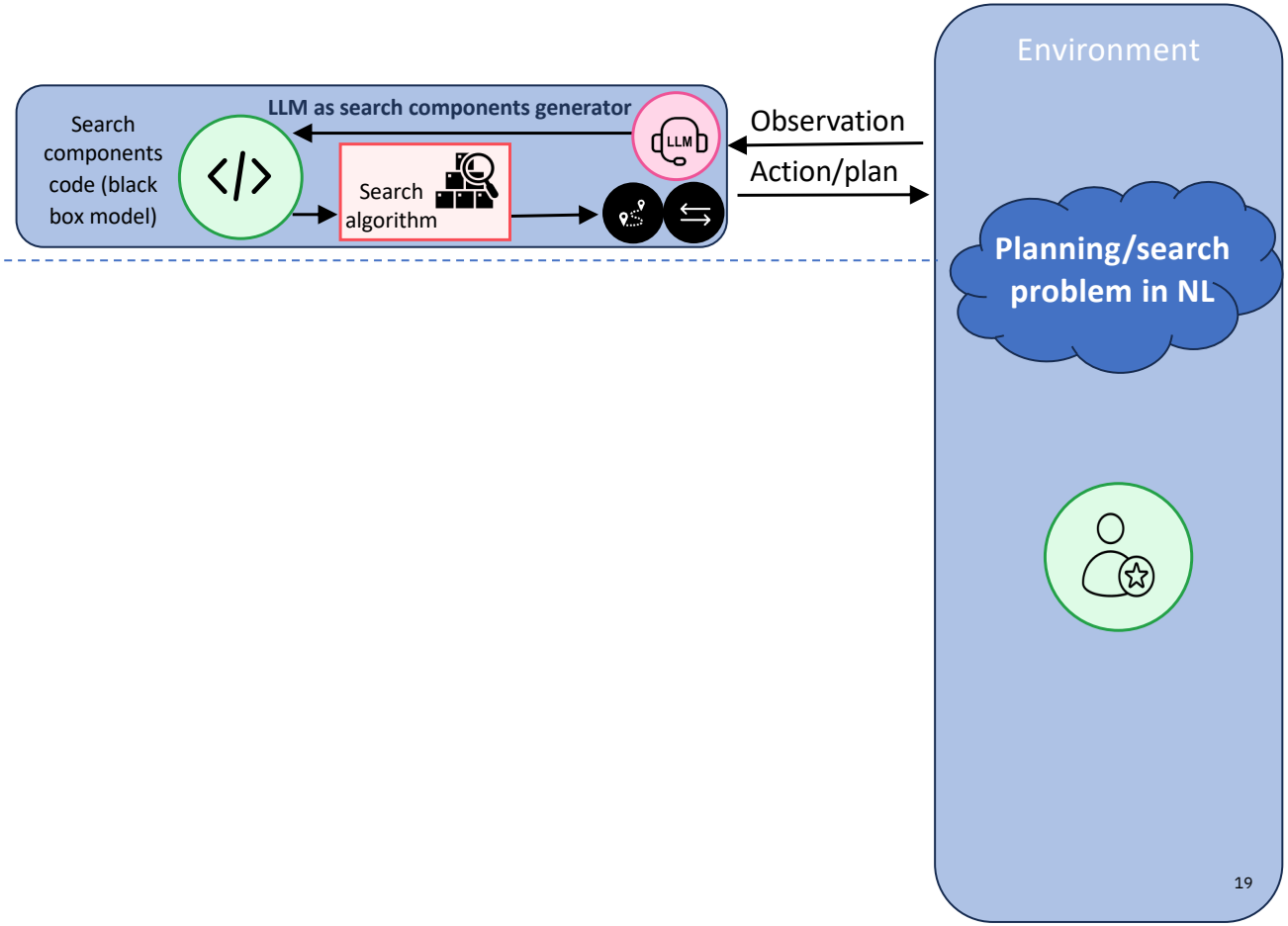
[1] Katz et al, Arxiv 2025, **Make Planning Research Rigorous Again!**

Planning in the Era of Language Models

- **Introduction: What is Planning**
  - What kind of planning problem do I have aka Planning Formalisms
  - What kind of solutions am I looking for aka Computational Problems
  - How can I describe my planning problem aka Planning Languages
- **Solving Planning Problems**
  - Pre-LLM era
  - Early LLM era
  - Modern LLM era

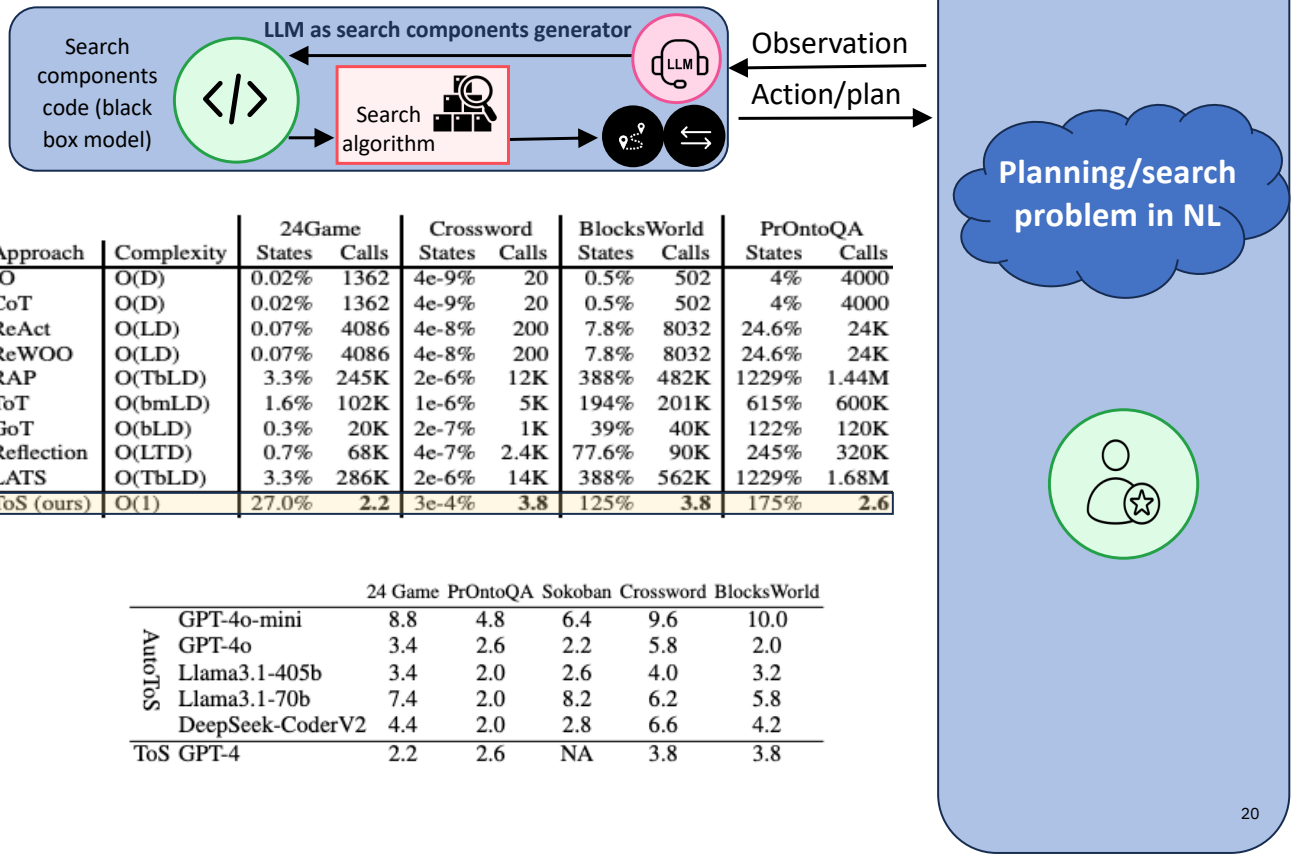
Solving NL/PDDL Planning Problems

Katz et al, NeurIPS 2024
Cao et al, OWA@NeurIPS 2024
Tuisov et al, Arxiv 2024
Correa et al, NeurIPS 2025



Solving NL/PDDL Planning Problems

Katz et al, NeurIPS 2024
Cao et al, OWA@NeurIPS 2024
Tuisov et al, Arxiv 2024
Correa et al, NeurIPS 2025



Approach	Complexity	24Game States	24Game Calls	Crossword States	Crossword Calls	BlocksWorld States	BlocksWorld Calls	PrOntoQA States	PrOntoQA Calls
IO	O(D)	0.02%	1362	4e-9%	20	0.5%	502	4%	4000
CoT	O(D)	0.02%	1362	4e-9%	20	0.5%	502	4%	4000
ReAct	O(LD)	0.07%	4086	4e-8%	200	7.8%	8032	24.6%	24K
ReWOO	O(LD)	0.07%	4086	4e-8%	200	7.8%	8032	24.6%	24K
RAP	O(TbLD)	3.3%	245K	2e-6%	12K	388%	482K	1229%	1.44M
ToT	O(bmLD)	1.6%	102K	1e-6%	5K	194%	201K	615%	600K
GoT	O(bLD)	0.3%	20K	2e-7%	1K	39%	40K	122%	120K
Reflection	O(LTD)	0.7%	68K	4e-7%	2.4K	77.6%	90K	245%	320K
LATS	O(TbLD)	3.3%	286K	2e-6%	14K	388%	562K	1229%	1.68M
ToS (ours)	O(1)	27.0%	**2.2**	3e-4%	**3.8**	125%	**3.8**	175%	**2.6**

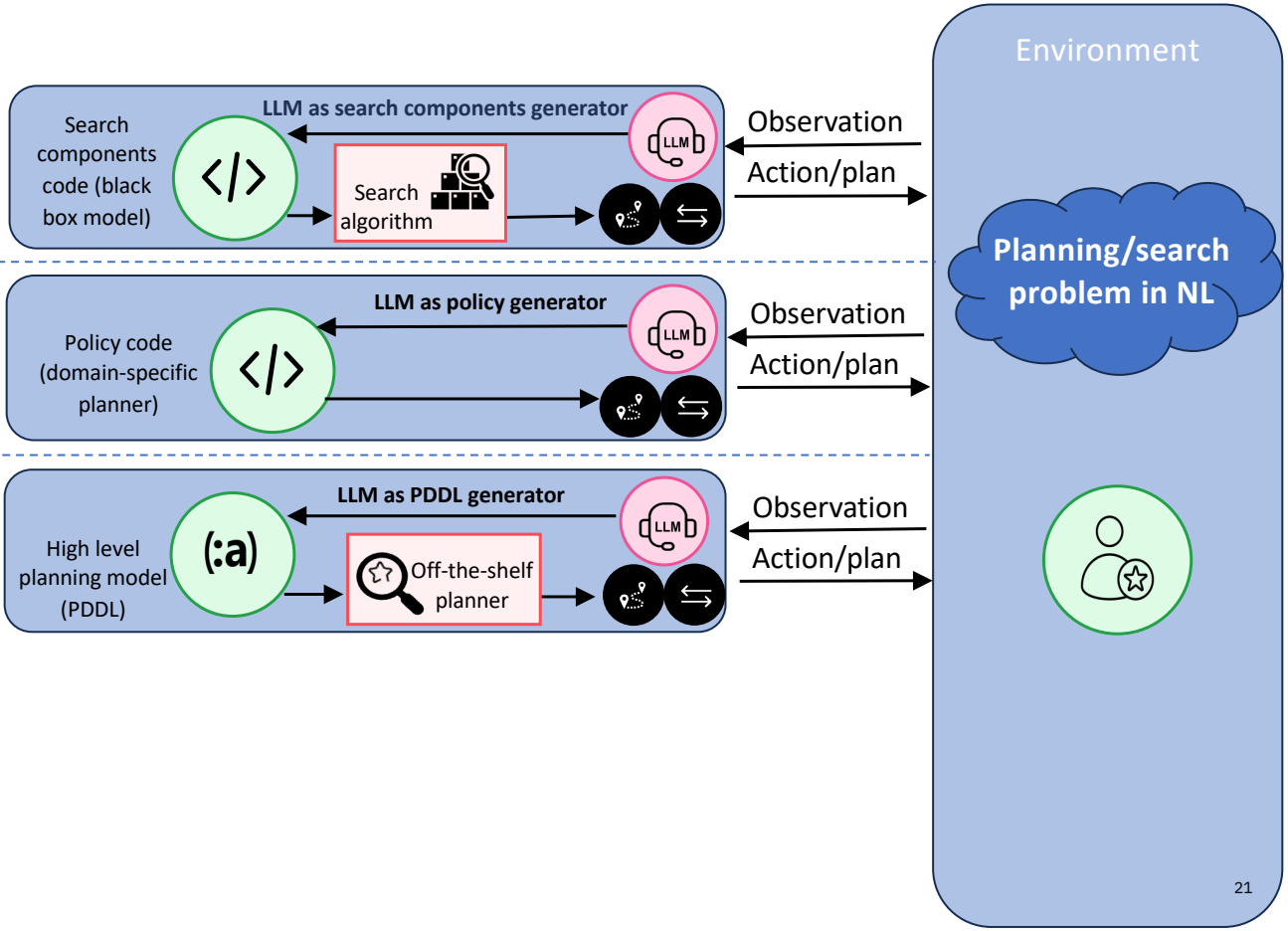
		24 Game	PrOntoQA	Sokoban	Crossword	BlocksWorld
AutoToS	GPT-4o-mini	8.8	4.8	6.4	9.6	10.0
	GPT-4o	3.4	2.6	2.2	5.8	2.0
	Llama3.1-405b	3.4	2.0	2.6	4.0	3.2
	Llama3.1-70b	7.4	2.0	8.2	6.2	5.8
	DeepSeek-CoderV2	4.4	2.0	2.8	6.6	4.2
ToS	GPT-4	2.2	2.6	NA	3.8	3.8

Solving NL/PDDL Planning Problems

Katz et al, NeurIPS 2024
Cao et al, OWA@NeurIPS 2024
Tuisov et al, Arxiv 2024
Correa et al, NeurIPS 2025

Silver et al, AAAI 2024
Hodel, BSc Thesis 2024
Stein et al, Arxiv 2025

Guan et al, NeurIPS 2023
Gestrin et al, Arxiv 2024
Oswald et al, ICAPS 2024
Huang et al, AAAI 2025
Tantakoun et al, ACL Findings 2025



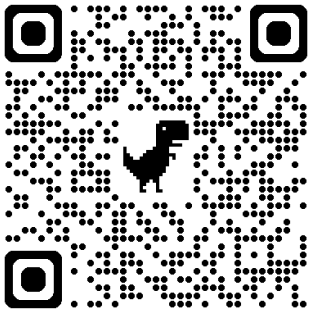
Links and references

Tutorial link:

- https://planning-llm-era.github.io/, **Planning in the Era of Language Models, NeurIPS 2025**
- https://aiplanning-tutorial.github.io/, AI Planning: Theory and Practice, AAAI 2022
- https://mp-tutorial.github.io/, Finding multiple plans for classical planning problems, ICAPS 2024

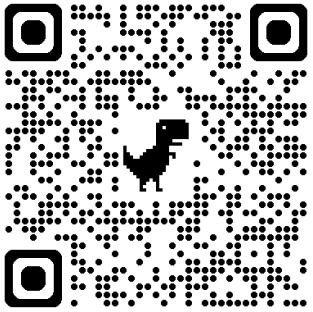
Planners available on github:

https://github.com/aibasel/downward
https://github.com/IBM/forbiditerative
https://github.com/IBM/kstar
https://github.com/speckdavid/symk

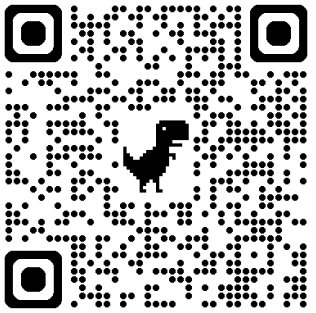


ICAPS conference

Git with references



ICAPS 2026 Summer School
June 22-25, 2026



AI Planning Community Git