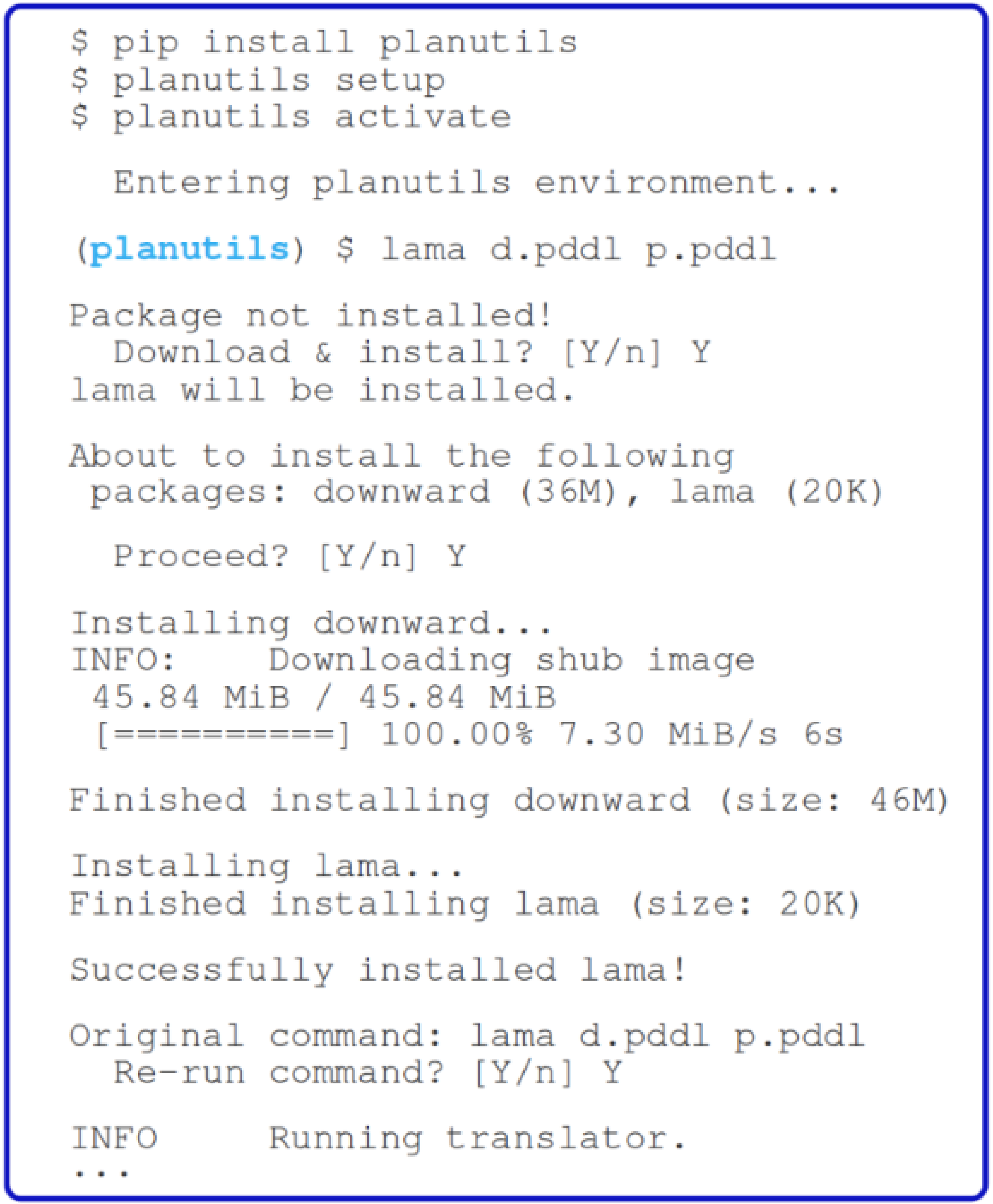
Tools to learn and work with planning models

1. Misc: modern toolbox
2. MACQ: time series > models
3. L2P: natural language > models

Misc Planning Utilities

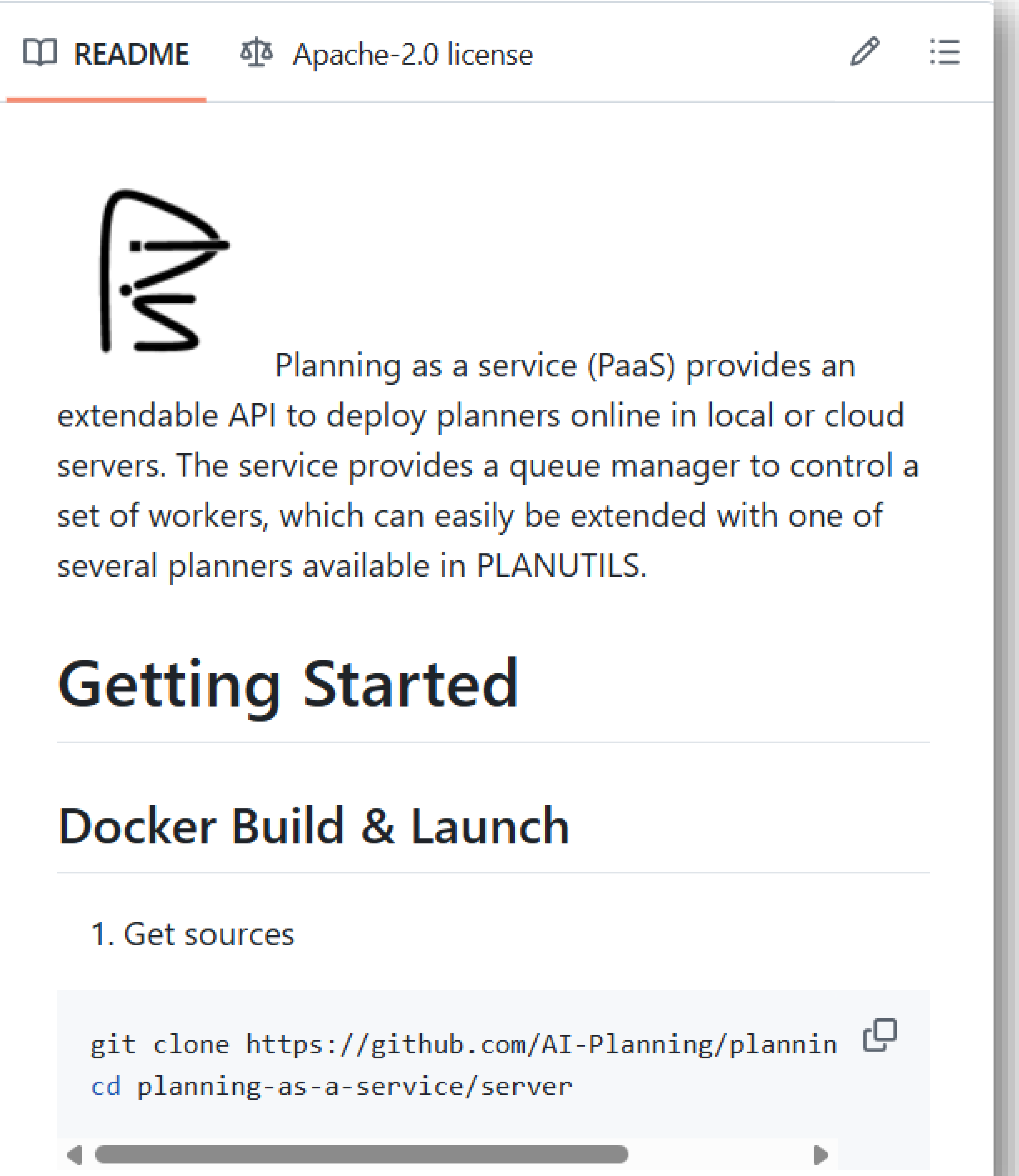
planutils

- Package manager and docker for planning tech
- Dozens of planners, all pre-compiled
- Ability to spin up lightweight local server
- Available via Dockerhub
    - The most common entry point



```
$ pip install planutils
$ planutils setup
$ planutils activate

  Entering planutils environment...

(planutils) $ lama d.pddl p.pddl

Package not installed!
  Download & install? [Y/n] Y
lama will be installed.

About to install the following
 packages: downward (36M), lama (20K)

  Proceed? [Y/n] Y

Installing downward...
INFO:    Downloading shub image
 45.84 MiB / 45.84 MiB
 [==========] 100.00% 7.30 MiB/s 6s

Finished installing downward (size: 46M)

Installing lama...
Finished installing lama (size: 20K)

Successfully installed lama!

Original command: lama d.pddl p.pddl
  Re-run command? [Y/n] Y

INFO     Running translator.
...
```

solver.planning.domains



Planning as a service (PaaS) provides an extendable API to deploy planners online in local or cloud servers. The service provides a queue manager to control a set of workers, which can easily be extended with one of several planners available in PLANUTILS.

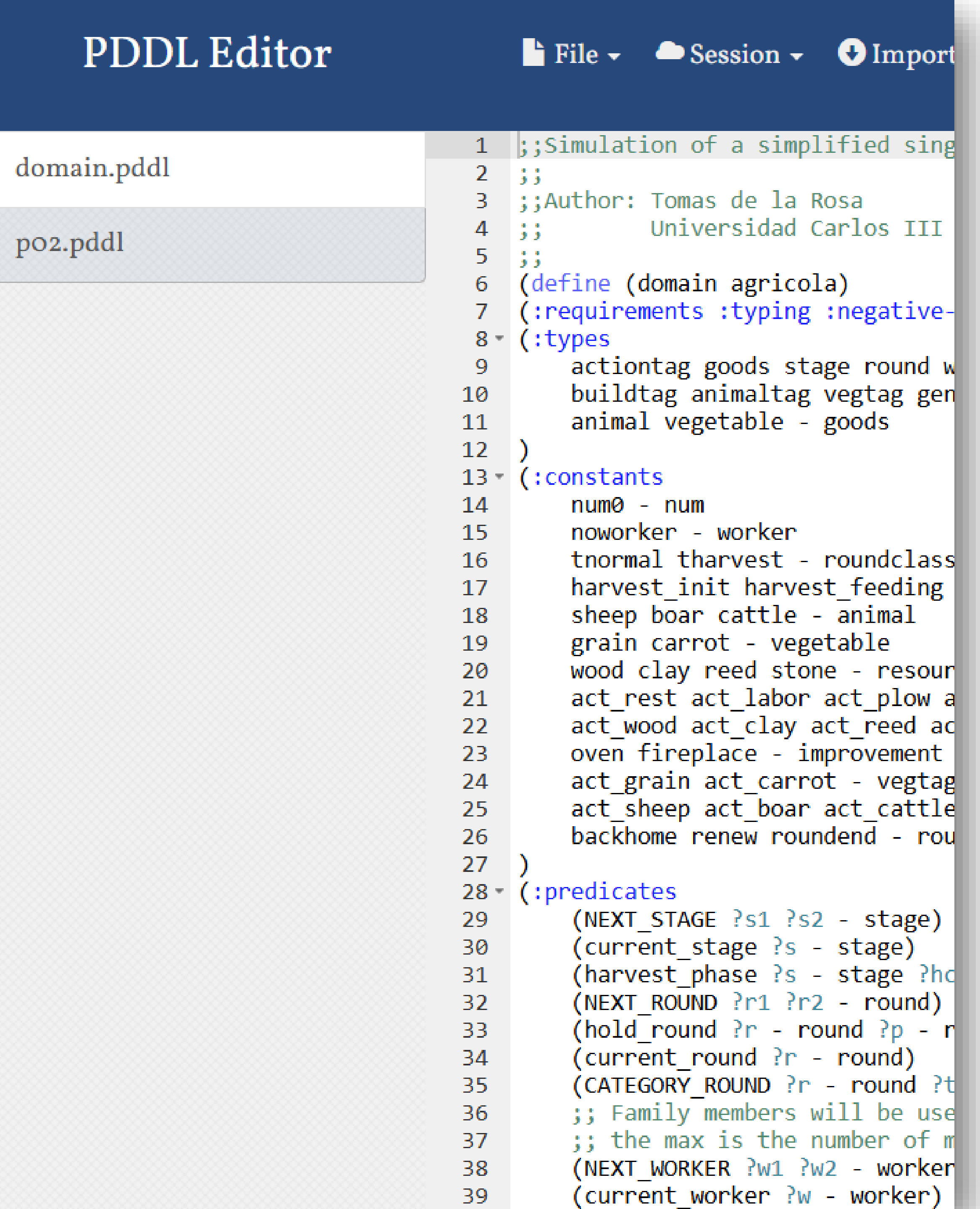
Getting Started

Docker Build & Launch

1. Get sources

```
git clone https://github.com/AI-Planning/plannin
cd planning-as-a-service/server
```

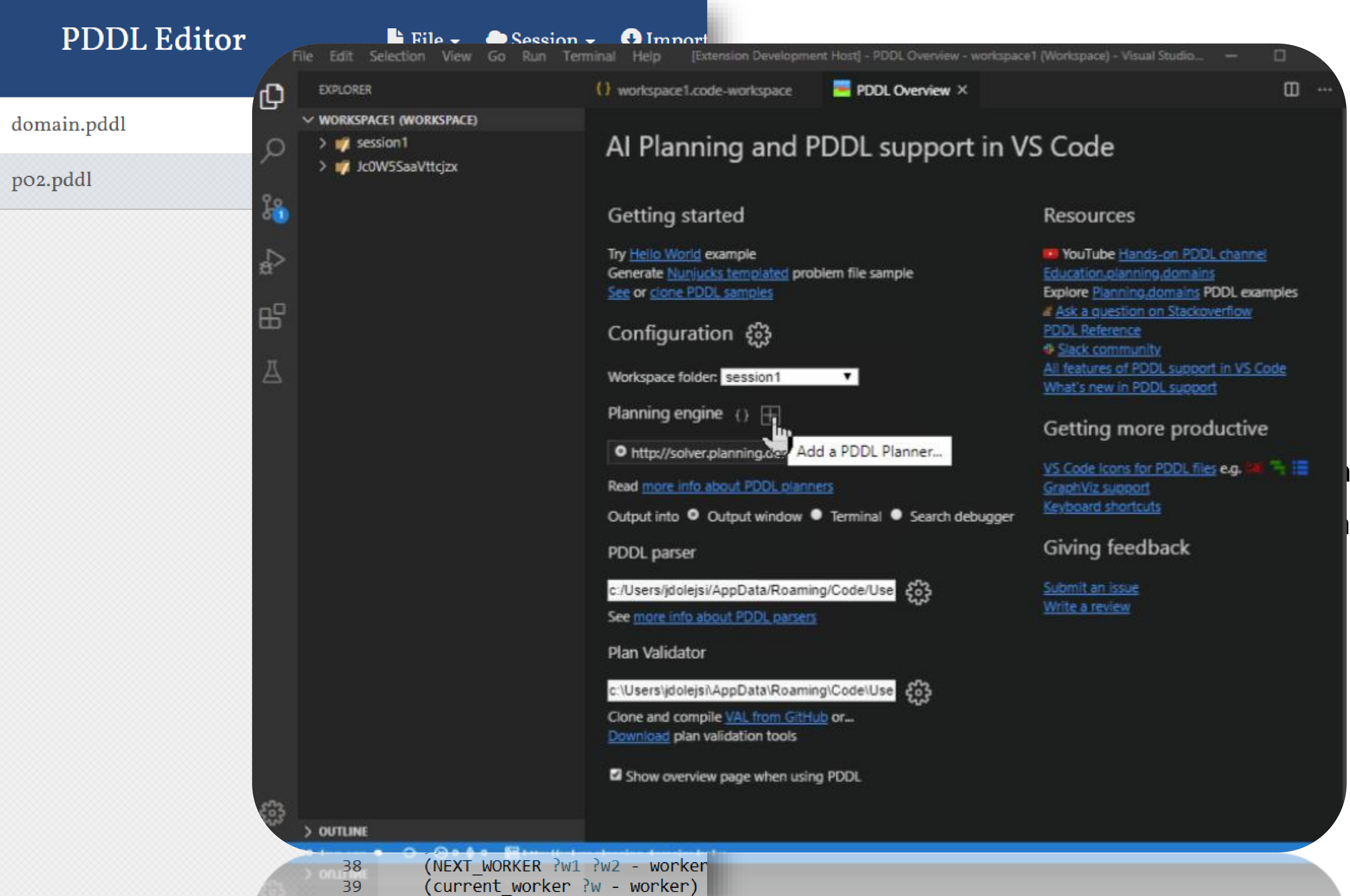
- Hosted on servers in Melbourne University
- Limited resource access to several planners
  - 500Mb & 30s
- Configured to host field’s most commonly used planning systems
- Built on the planutils package
- Also an open source project
  - Used by organizations for internal planner hosting



```
;;Simulation of a simplified sing
;;
;;Author: Tomas de la Rosa
;;        Universidad Carlos III
;;
(define (domain agricola)
(:requirements :typing :negative-
(:types
    actiontag goods stage round w
    buildtag animaltag vegtag gen
    animal vegetable - goods
)
(:constants
    num0 - num
    noworker - worker
    tnormal tharvest - roundclass
    harvest_init harvest_feeding
    sheep boar cattle - animal
    grain carrot - vegetable
    wood clay reed stone - resour
    act_rest act_labor act_plow a
    act_wood act_clay act_reed ac
    oven fireplace - improvement
    act_grain act_carrot - vegtag
    act_sheep act_boar act_cattle
    backhome renew roundend - rou
)
(:predicates
    (NEXT_STAGE ?s1 ?s2 - stage)
    (current_stage ?s - stage)
    (harvest_phase ?s - stage ?hc
    (NEXT_ROUND ?r1 ?r2 - round)
    (hold_round ?r - round ?p - r
    (current_round ?r - round)
    (CATEGORY_ROUND ?r - round ?t
    ;; Family members will be use
    ;; the max is the number of m
    (NEXT_WORKER ?w1 ?w2 - worker
    (current_worker ?w - worker)
```

editor.planning.domains

- Online editor for PDDL
- Import access to hundreds of benchmarks
- Solver access to solver.planning.domains
- Cloud session functionality (complete with read-only links)
- Plugin functionality with catalogue of custom built plugins from the planning community



AI Planning and PDDL support in VS Code

Getting started

Try Hello World example
Generate Nunjucks templated problem file sample
See or clone PDDL samples

Configuration

Workspace folder: session1

Planning engine

○ http://solver.planning.do Add a PDDL Planner...

Read more info about PDDL planners

Output into ● Output window ● Terminal ● Search debugger

PDDL parser

c:/Users/jdolejsi/AppData/Roaming/Code/Use

See more info about PDDL parsers

Plan Validator

c:\Users\jdolejsi\AppData\Roaming\Code\Use

Clone and compile VAL from GitHub or...
Download plan validation tools

☑ Show overview page when using PDDL

Resources

YouTube Hands-on PDDL channel
Education.planning.domains
Explore Planning.domains PDDL examples
Ask a question on Stackoverflow
PDDL Reference
Slack community
All features of PDDL support in VS Code
What's new in PDDL support

Getting more productive

VS Code Icons for PDDL files e.g.
GraphViz support
Keyboard shortcuts

Giving feedback

Submit an issue
Write a review

```
(NEXT_WORKER ?w1 ?w2 - worker
(current_worker ?w - worker)
```

ains

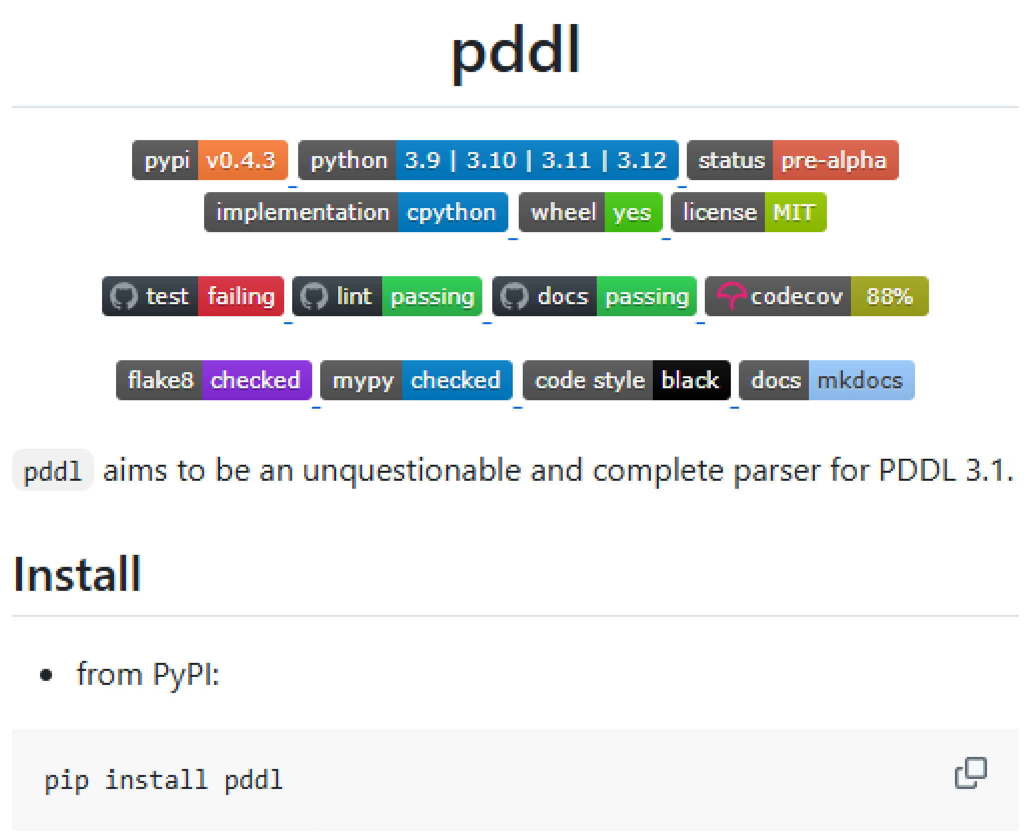
arks

ins

with

custom
unity

pddl



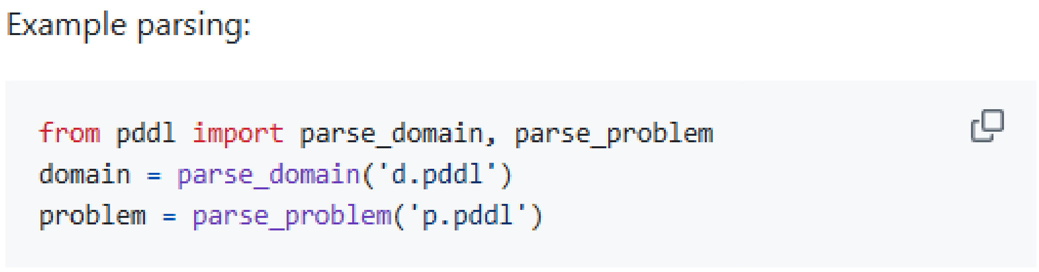
pddl

`pddl` aims to be an unquestionable and complete parser for PDDL 3.1.

Install

- from PyPI:

```
pip install pddl
```



Example parsing:

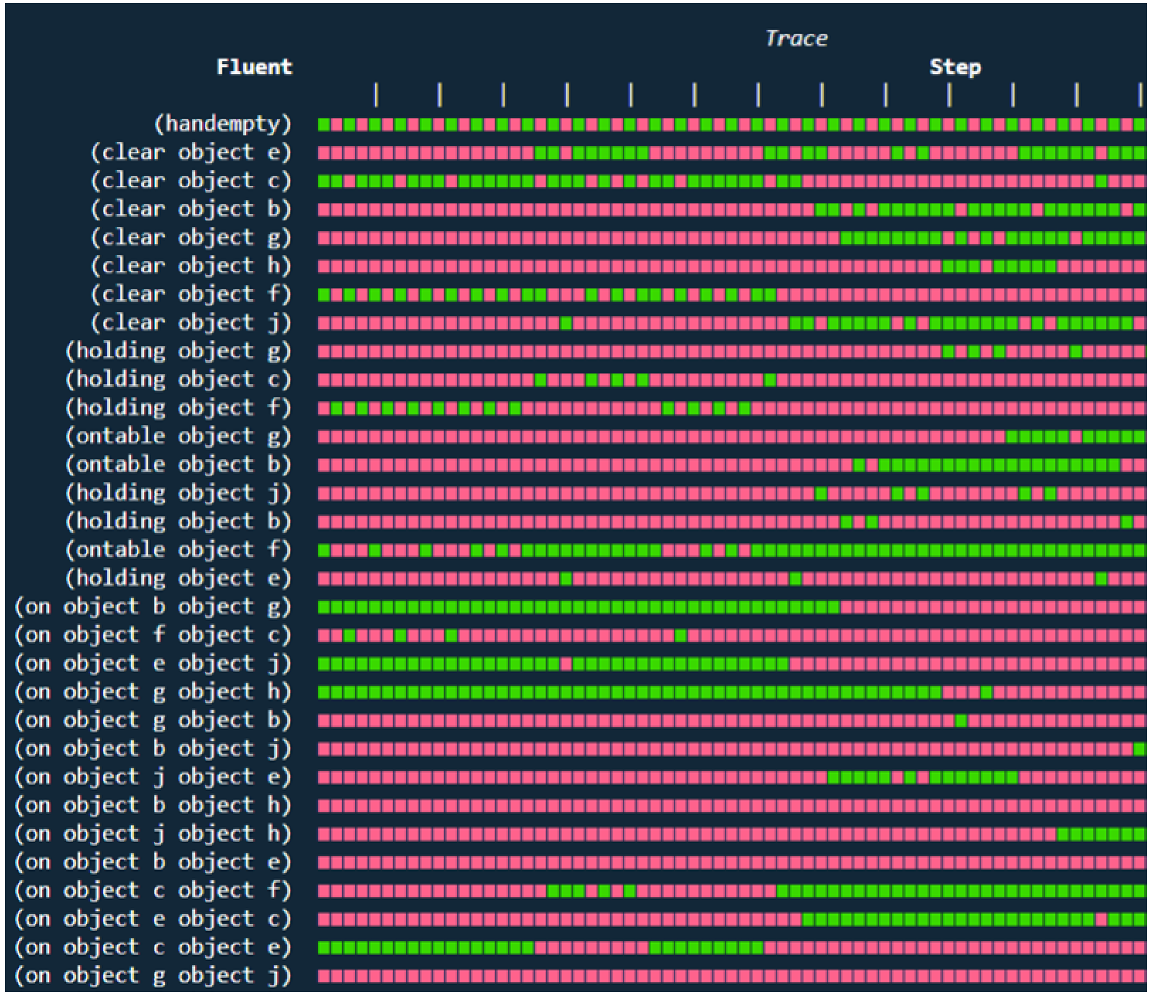
```
from pddl import parse_domain, parse_problem
domain = parse_domain('d.pddl')
problem = parse_problem('p.pddl')
```

- Python library for parsing and processing PDDL
- Rich flexibility in PDDL features
- Growing list of auxiliary libraries that leverage it (state maintenance, compilations, etc).

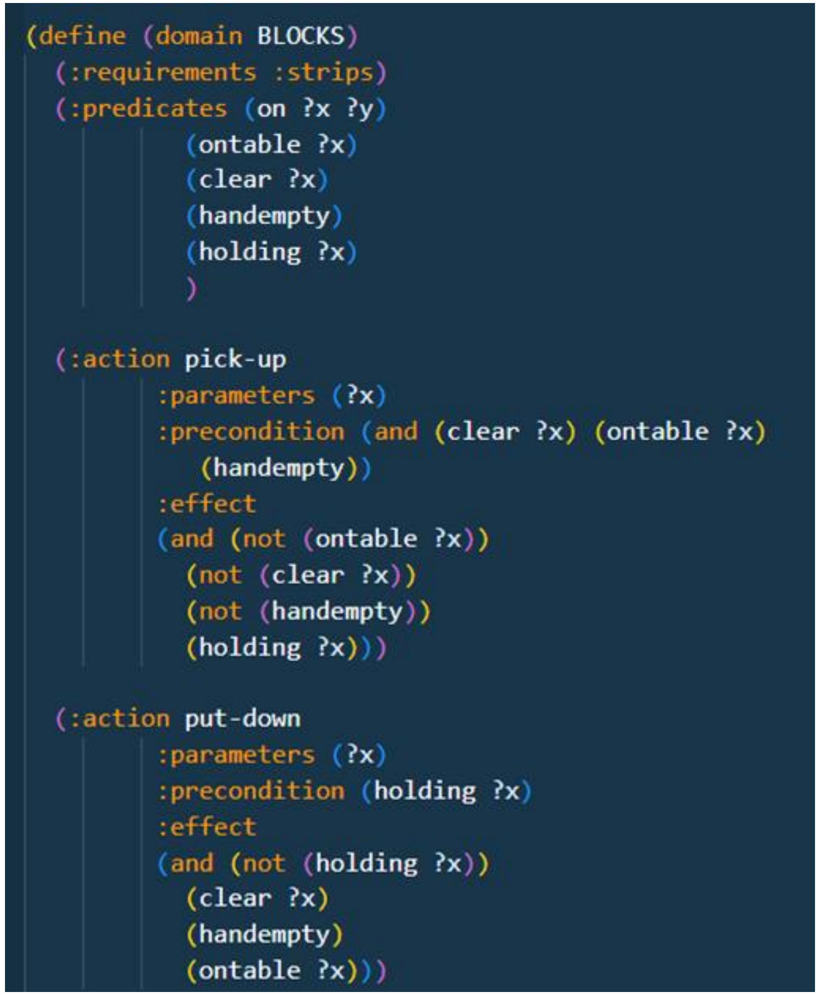
MACQ: The Model Acquisition Toolkit

What is Model Acquisition?

Input: State Trace Data

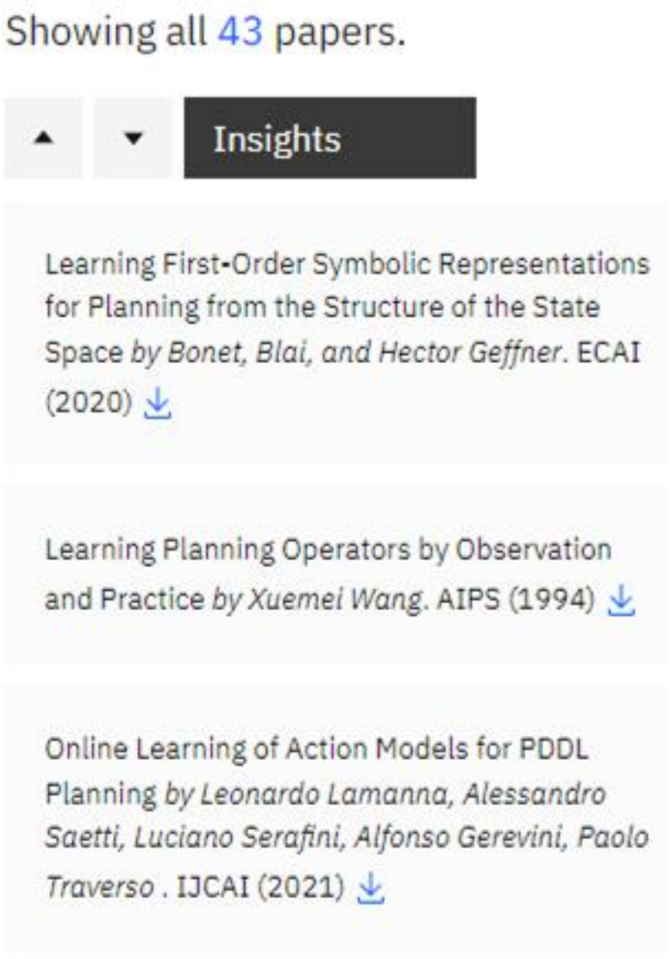


Output: PDDL Action Model



```
(define (domain BLOCKS)
  (:requirements :strips)
  (:predicates (on ?x ?y)
               (ontable ?x)
               (clear ?x)
               (handempty)
               (holding ?x)
               )

  (:action pick-up
             :parameters (?x)
             :precondition (and (clear ?x) (ontable ?x)
               (handempty))
             :effect
             (and (not (ontable ?x))
               (not (clear ?x))
               (not (handempty))
               (holding ?x)))

  (:action put-down
             :parameters (?x)
             :precondition (holding ?x)
             :effect
             (and (not (holding ?x))
               (clear ?x)
               (handempty)
               (ontable ?x)))
```

Research in Model Acquisition + Motivations



Showing all 43 papers.

Learning First-Order Symbolic Representations for Planning from the Structure of the State Space *by Bonet, Blai, and Hector Geffner*. ECAI (2020)

Learning Planning Operators by Observation and Practice *by Xuemei Wang*. AIPS (1994)

Online Learning of Action Models for PDDL Planning *by Leonardo Lamanna, Alessandro Saetti, Luciano Serafini, Alfonso Gerevini, Paolo Traverso* . IJCAI (2021)

etc...

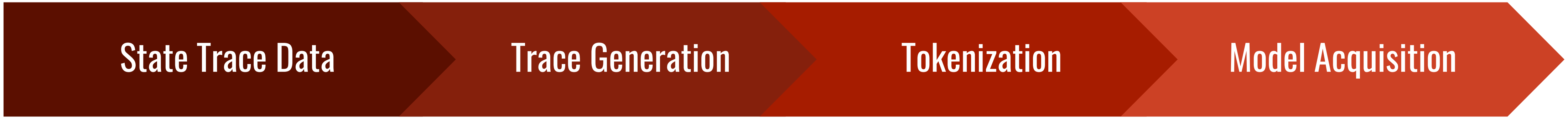
Still need a centralized API for:

Trace generation and visualization.

Converting traces into a format that varying model acquisition techniques will recognize.

Testing different model acquisition techniques on the fly.

Introducing MACQ



State Trace Data

- From PDDL files
- From a problem ID
- Raw data, i.e. from a CSV file

Trace Generation

- Generate trace data
- Dynamically change the initial state and/or goal
- Examples:
  - Vanilla Sampling
  - Goal-Oriented Sampling

Tokenization

- Convert traces into observation tokens
- Examples:
  - Partial obs.
  - Noisiness
  - Parallel actions
  - Disordered actions

Model Acquisition

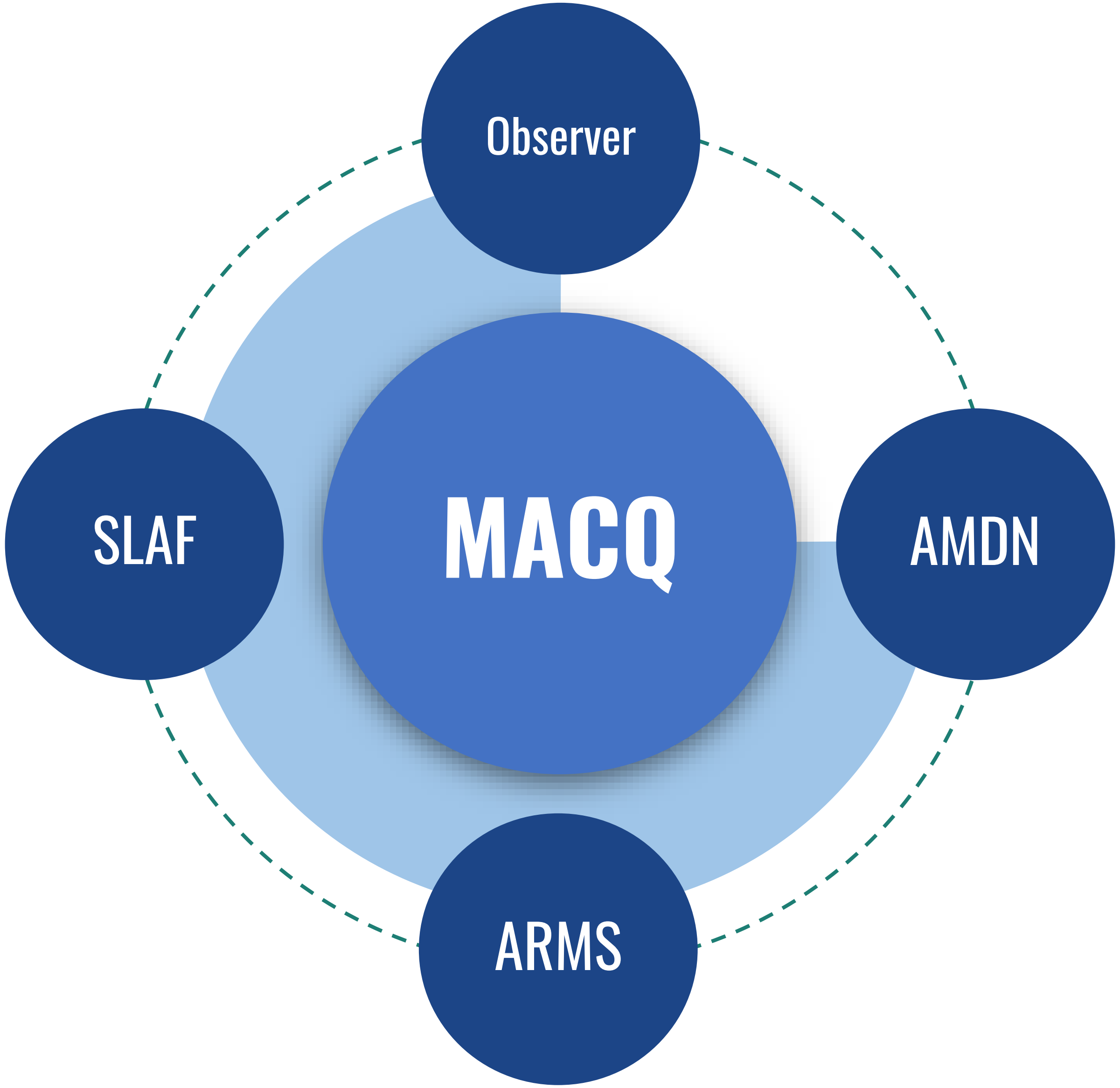
- Extract actions from observations
- Convert to PDDL
- Examples:
  - Observer
  - SLAF
  - ARMS
  - AMDN
  - LOCM
  - ...

Initial Choice of MACQ Techniques

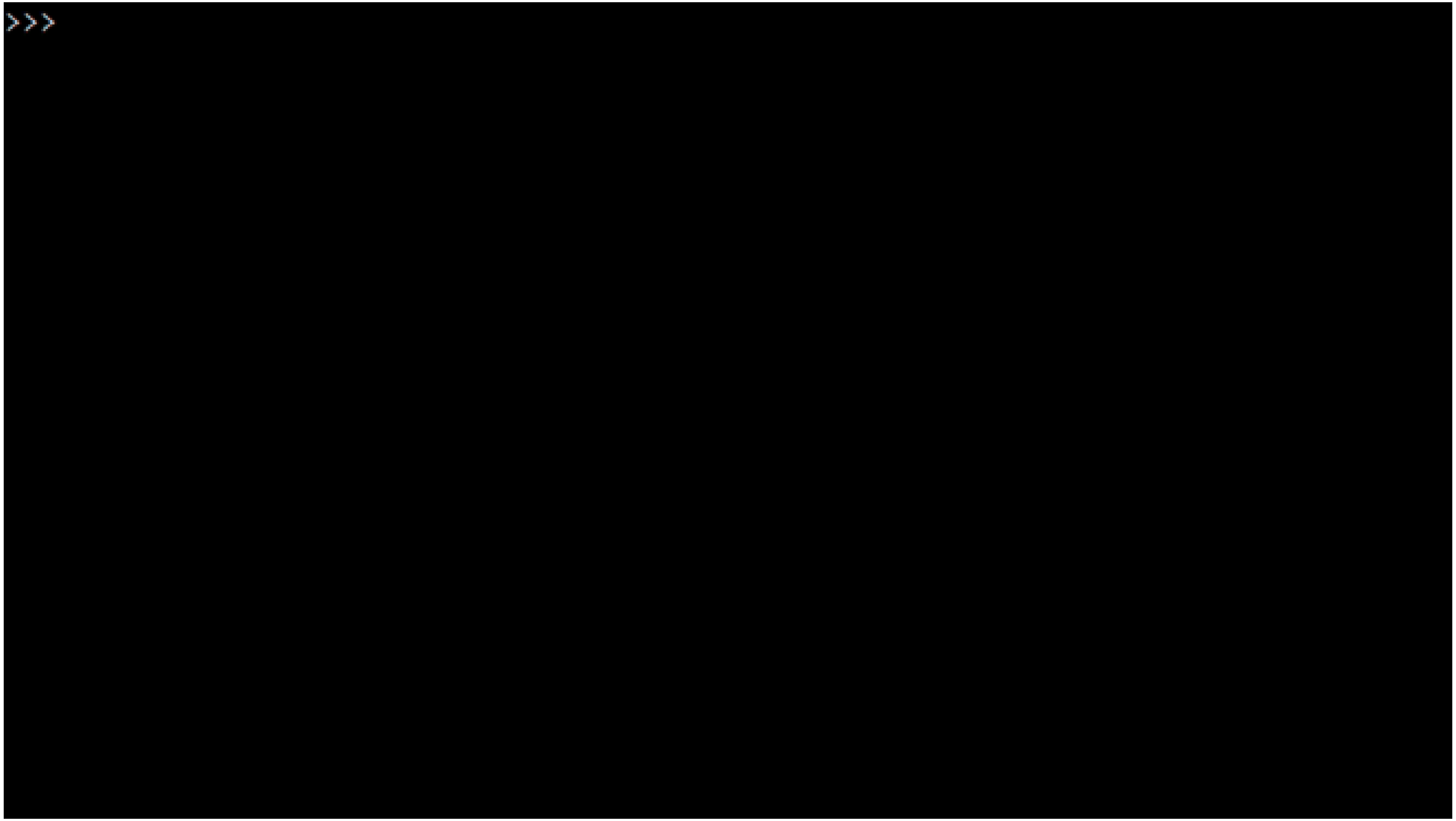
- Covers a large set of trace features, i.e.
  - Unmodified data
  - Partially observable data
  - Noisy data
  - Disordered and parallel actions

- Makes the library simple and easy to use upon release

- Open source, extensible API encourages the addition of more techniques

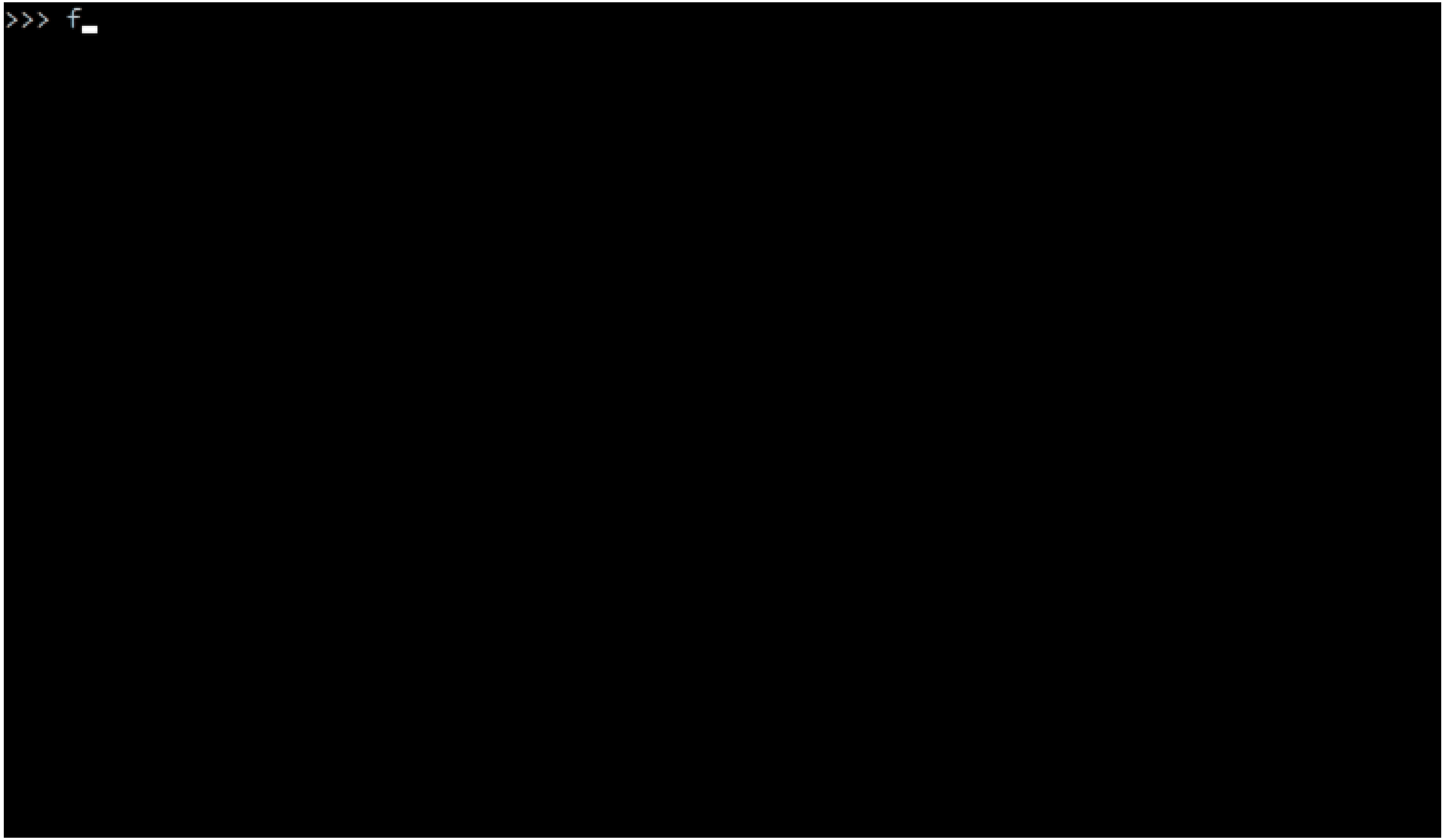


Trace Generation



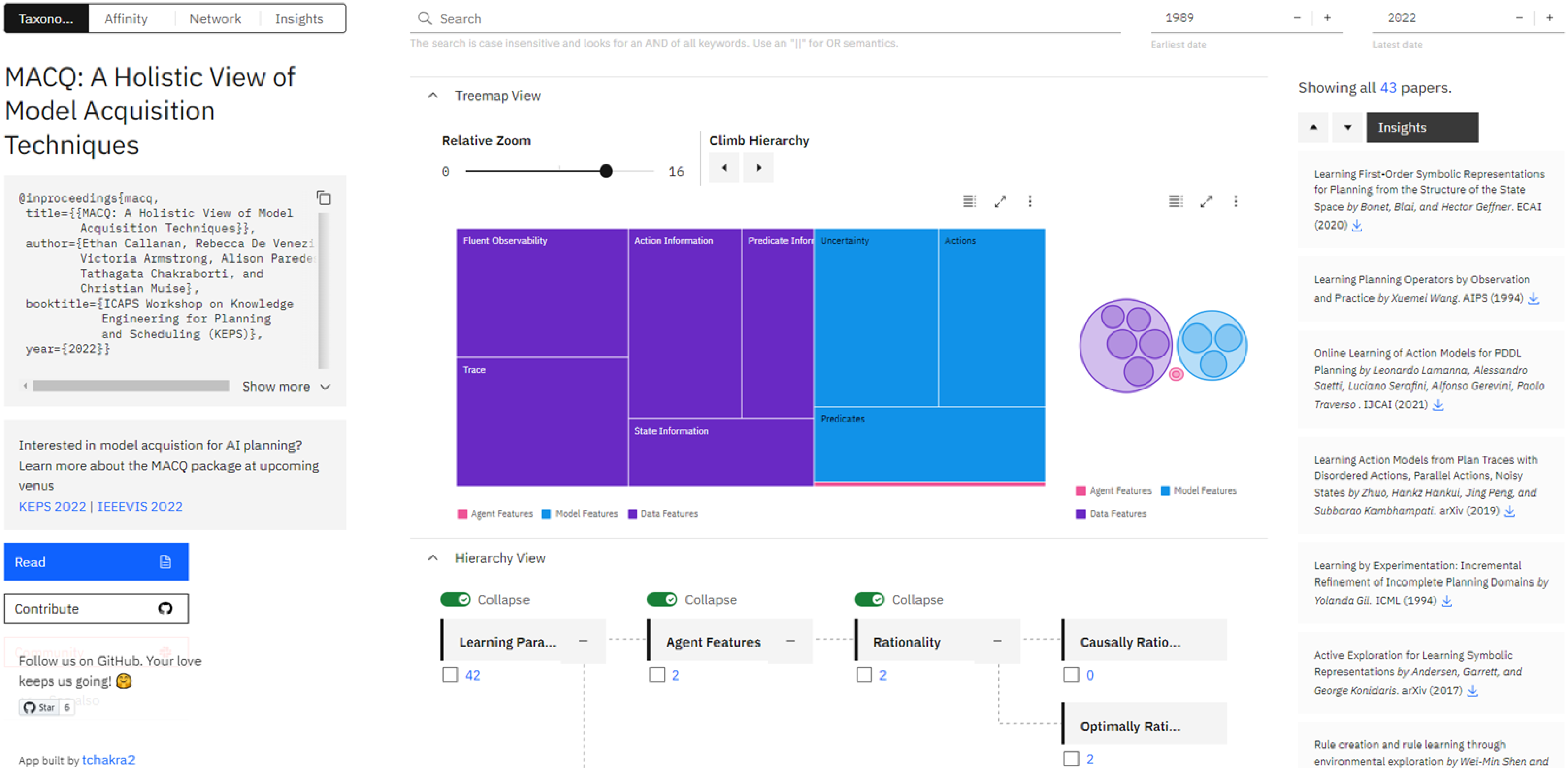
```
>>>
```

Tokenization + Model Acquisition



```
>>> f
```

MACQ Visualizer: A Holistic View of Model Acquisition Techniques



MACQ: A Holistic View of Model Acquisition Techniques

```
@inproceedings{macq,
 title={{MACQ: A Holistic View of Model
        Acquisition Techniques}},
 author={Ethan Callanan, Rebecca De Venezi
        Victoria Armstrong, Alison Parede
        Tathagata Chakraborti, and
        Christian Muise},
 booktitle={ICAPS Workshop on Knowledge
           Engineering for Planning
           and Scheduling (KEPS)},
 year={2022}}
```

Interested in model acquistion for AI planning? Learn more about the MACQ package at upcoming venus

KEPS 2022 | IEEEVIS 2022

Follow us on GitHub. Your love keeps us going! 🤗

App built by tchakra2

The search is case insensitive and looks for an AND of all keywords. Use an "||" for OR semantics.

Showing all 43 papers.

Learning First-Order Symbolic Representations for Planning from the Structure of the State Space *by Bonet, Blai, and Hector Geffner*. ECAI (2020)

Learning Planning Operators by Observation and Practice *by Xuemei Wang*. AIPS (1994)

Online Learning of Action Models for PDDL Planning *by Leonardo Lamanna, Alessandro Saetti, Luciano Serafini, Alfonso Gerevini, Paolo Traverso* . IJCAI (2021)

Learning Action Models from Plan Traces with Disordered Actions, Parallel Actions, Noisy States *by Zhuo, Hankz Hankui, Jing Peng, and Subbarao Kambhampati*. arXiv (2019)

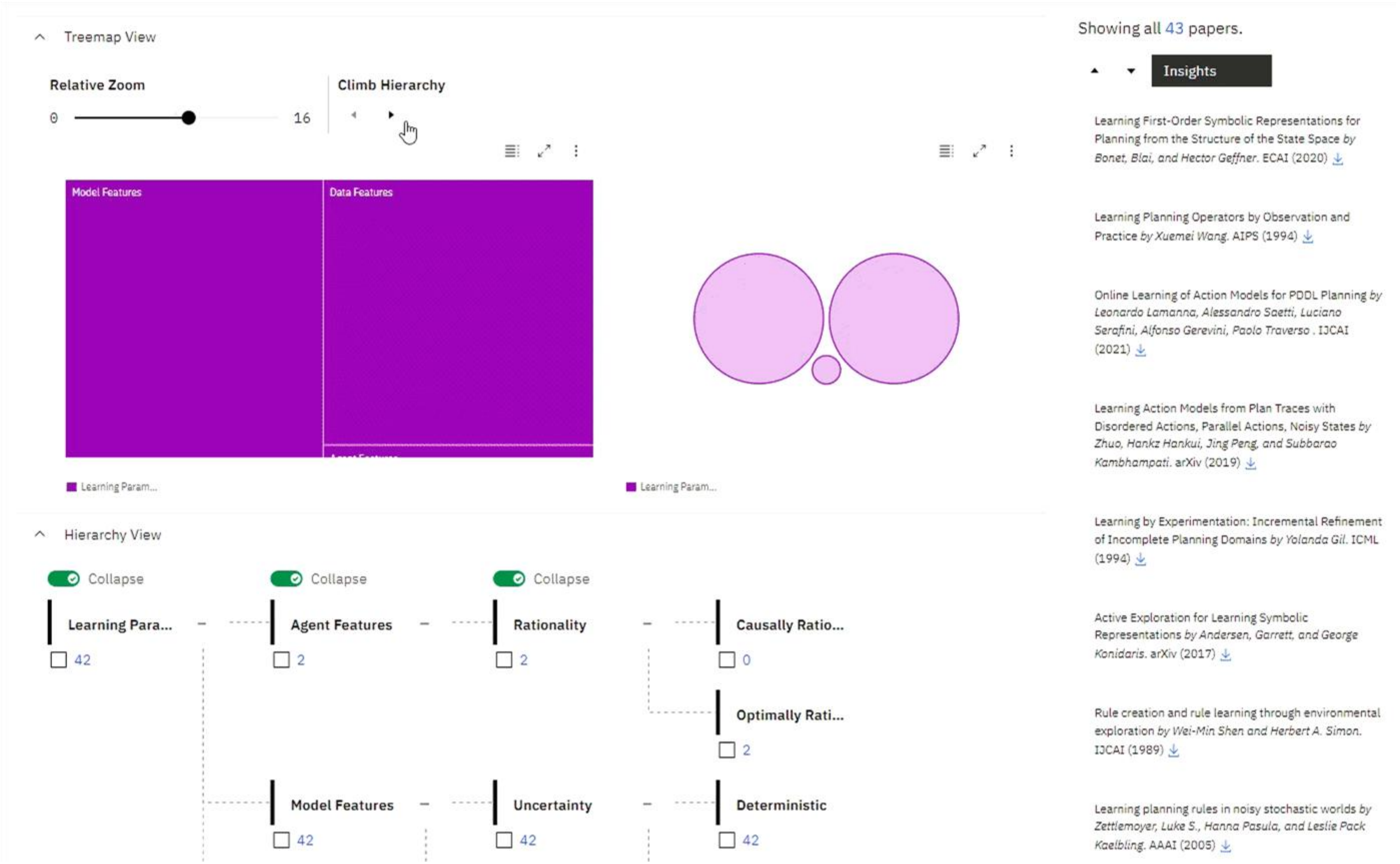
Learning by Experimentation: Incremental Refinement of Incomplete Planning Domains *by Yolanda Gil*. ICML (1994)

Active Exploration for Learning Symbolic Representations *by Andersen, Garrett, and George Konidaris*. arXiv (2017)

Rule creation and rule learning through environmental exploration *by Wei-Min Shen and*

macq.planning.domains

MACQ Visualizer: Taxonomy



Showing all 43 papers.

Learning First-Order Symbolic Representations for Planning from the Structure of the State Space *by Bonet, Blai, and Hector Geffner*. ECAI (2020)

Learning Planning Operators by Observation and Practice *by Xuemei Wang*. AIPS (1994)

Online Learning of Action Models for PDDL Planning *by Leonardo Lamanna, Alessandro Saetti, Luciano Serafini, Alfonso Gerevini, Paolo Traverso* . IJCAI (2021)

Learning Action Models from Plan Traces with Disordered Actions, Parallel Actions, Noisy States *by Zhuo, Hankz Hankui, Jing Peng, and Subbarao Kambhampati*. arXiv (2019)

Learning by Experimentation: Incremental Refinement of Incomplete Planning Domains *by Yolanda Gil*. ICML (1994)

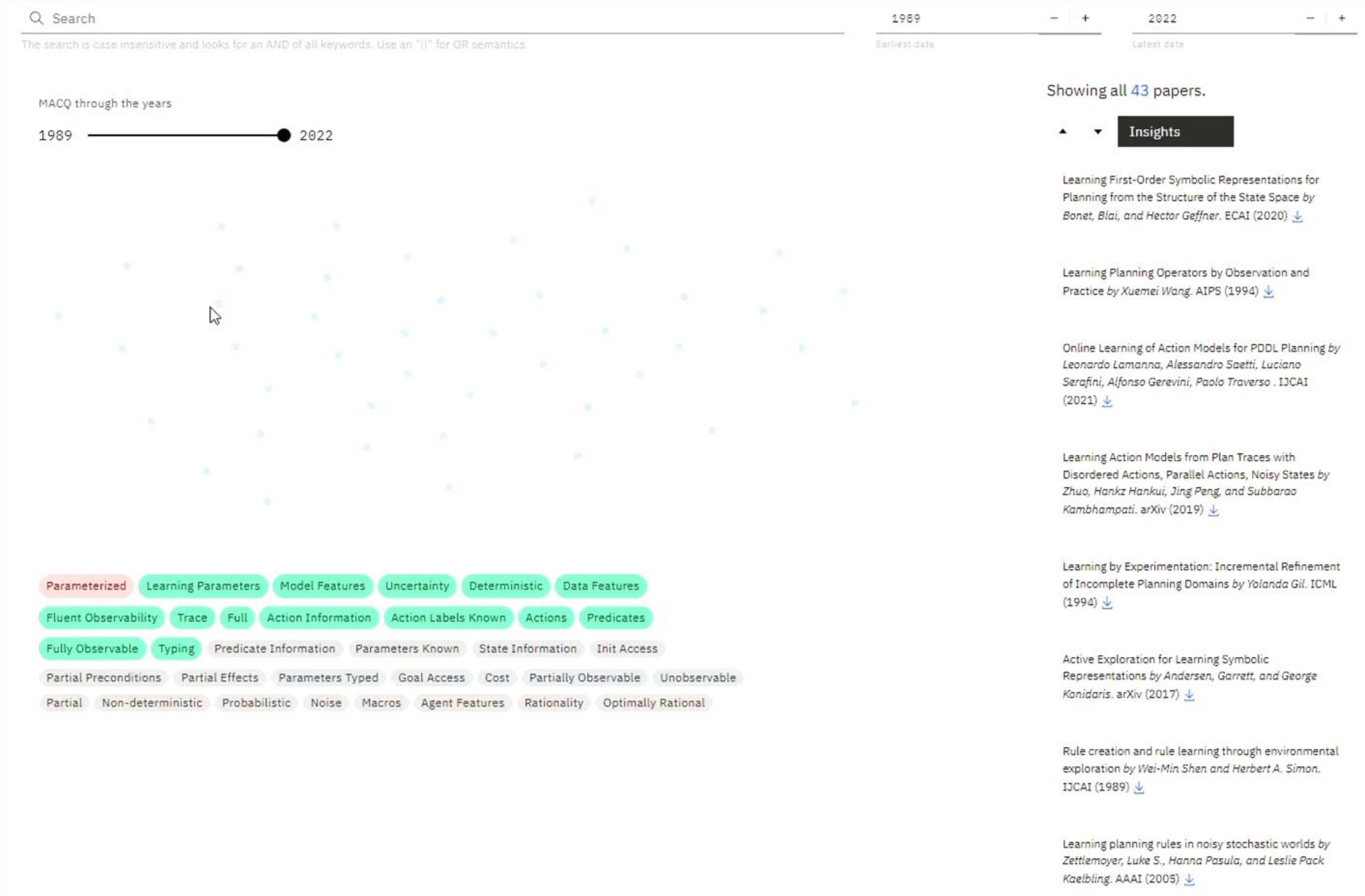
Active Exploration for Learning Symbolic Representations *by Andersen, Garrett, and George Konidaris*. arXiv (2017)

Rule creation and rule learning through environmental exploration *by Wei-Min Shen and Herbert A. Simon*. IJCAI (1989)

Learning planning rules in noisy stochastic worlds *by Zettlemoyer, Luke S., Hanna Pasula, and Leslie Pack Kaelbling*. AAAI (2005)

macq.planning.domains

MACQ Visualizer: Affinity



Search

The search is case insensitive and looks for an AND of all keywords. Use an "||" for OR semantics.

Earliest date: 1989

Latest date: 2022

MACQ through the years

1989 2022

Showing all 43 papers.

Learning First-Order Symbolic Representations for Planning from the Structure of the State Space *by Bonet, Blai, and Hector Geffner*. ECAI (2020)

Learning Planning Operators by Observation and Practice *by Xuemei Wang*. AIPS (1994)

Online Learning of Action Models for PDDL Planning *by Leonardo Lamanna, Alessandro Saetti, Luciano Serafini, Alfonso Gerevini, Paolo Traverso* . IJCAI (2021)

Learning Action Models from Plan Traces with Disordered Actions, Parallel Actions, Noisy States *by Zhuo, Hankz Hankui, Jing Peng, and Subbarao Kambhampati*. arXiv (2019)

Learning by Experimentation: Incremental Refinement of Incomplete Planning Domains *by Yolanda Gil*. ICML (1994)

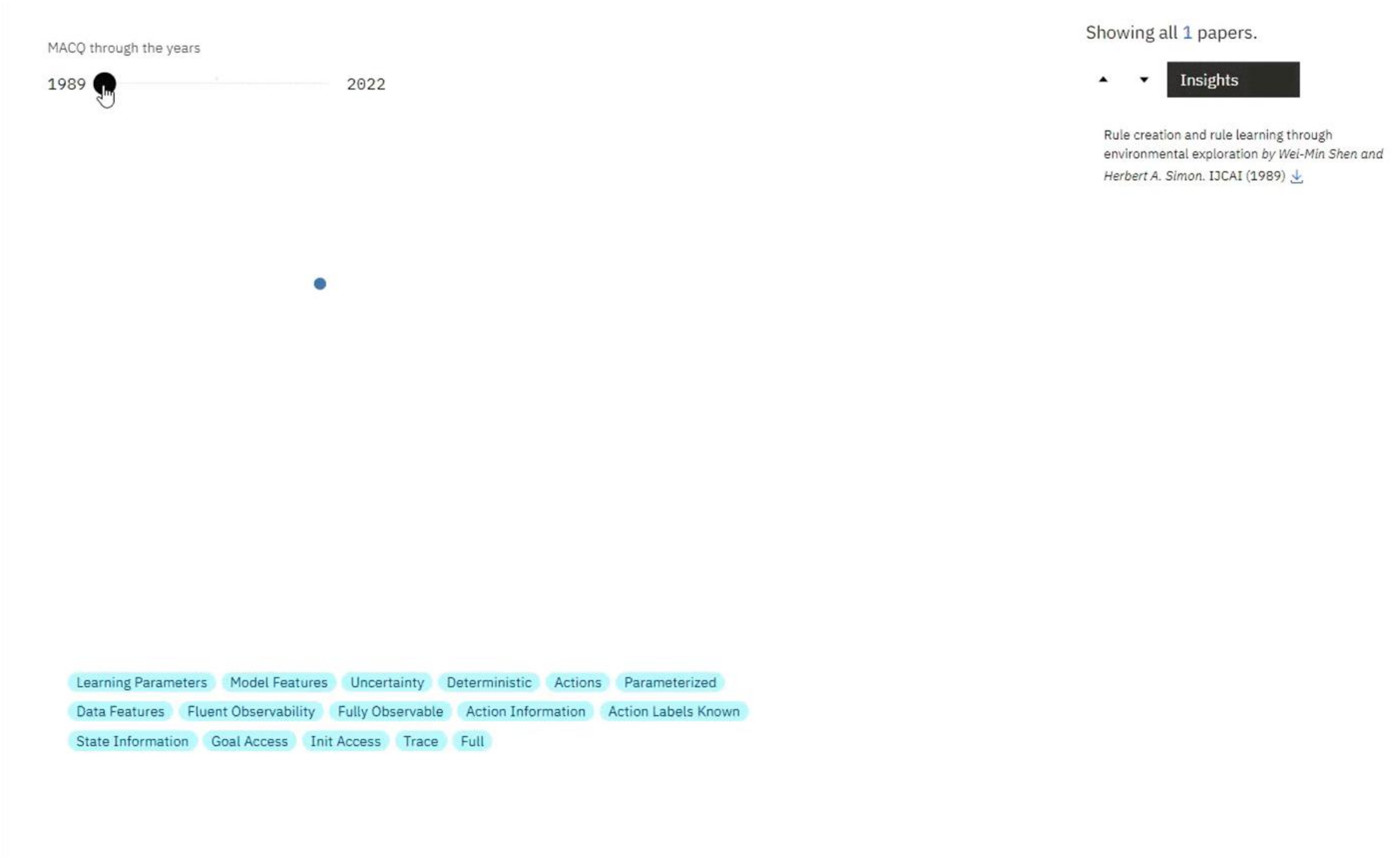
Active Exploration for Learning Symbolic Representations *by Andersen, Garrett, and George Konidaris*. arXiv (2017)

Rule creation and rule learning through environmental exploration *by Wei-Min Shen and Herbert A. Simon*. IJCAI (1989)

Learning planning rules in noisy stochastic worlds *by Zettlemoyer, Luke S., Hanna Pasula, and Leslie Pack Kaelbling*. AAAI (2005)

macq.planning.domains

MACQ Visualizer: Network

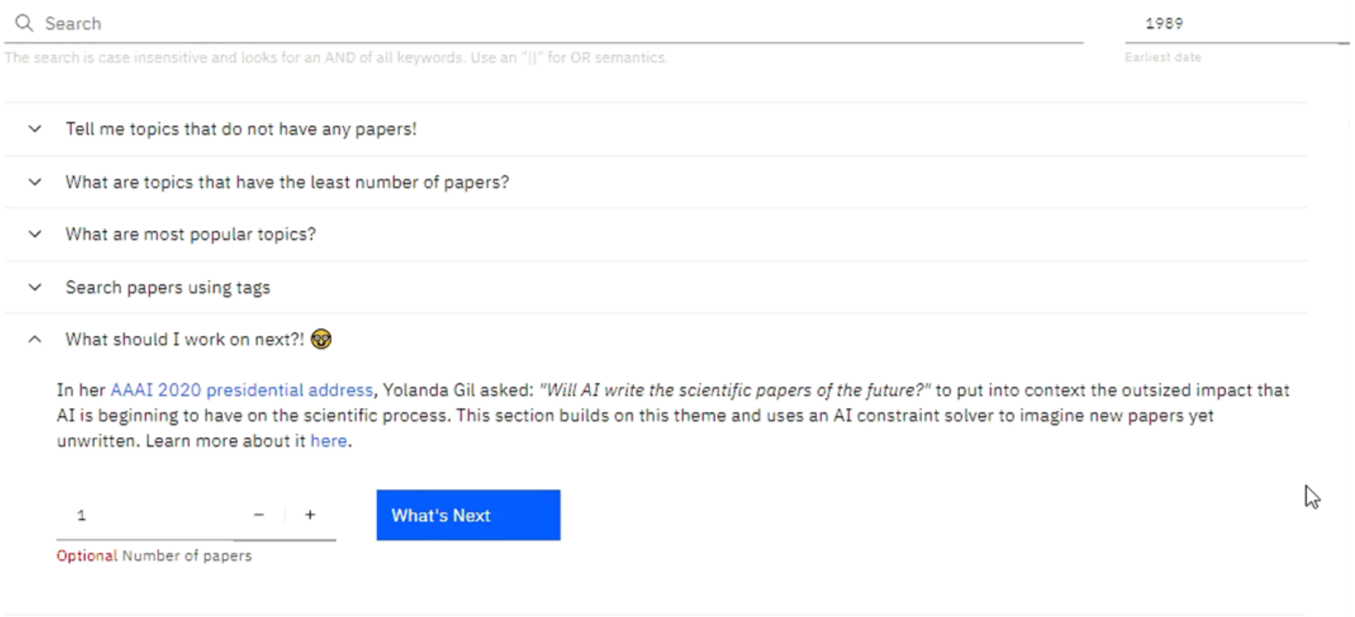


Showing all 1 papers.

Rule creation and rule learning through environmental exploration *by Wei-Min Shen and Herbert A. Simon.* IJCAI (1989)

macq.planning.domains

MACQ Visualizer: Insights



The search is case insensitive and looks for an AND of all keywords. Use an "||" for OR semantics.

Earliest date

Tell me topics that do not have any papers!
What are topics that have the least number of papers?
What are most popular topics?
Search papers using tags
What should I work on next?! 🤓

In her AAAI 2020 presidential address, Yolanda Gil asked: *"Will AI write the scientific papers of the future?"* to put into context the outsized impact that AI is beginning to have on the scientific process. This section builds on this theme and uses an AI constraint solver to imagine new papers yet unwritten. Learn more about it here.

Number of papers: 1 – +

Optional Number of papers

macq.planning.domains

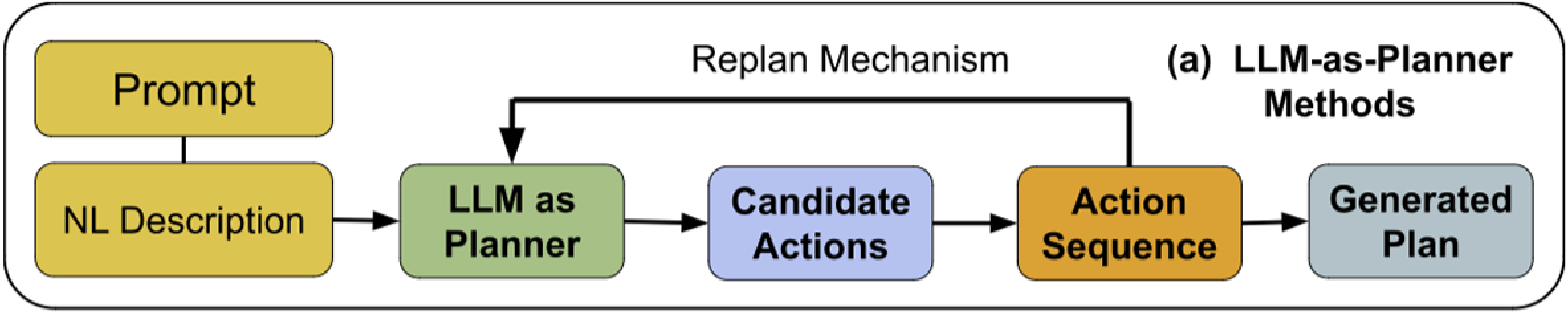
Summary

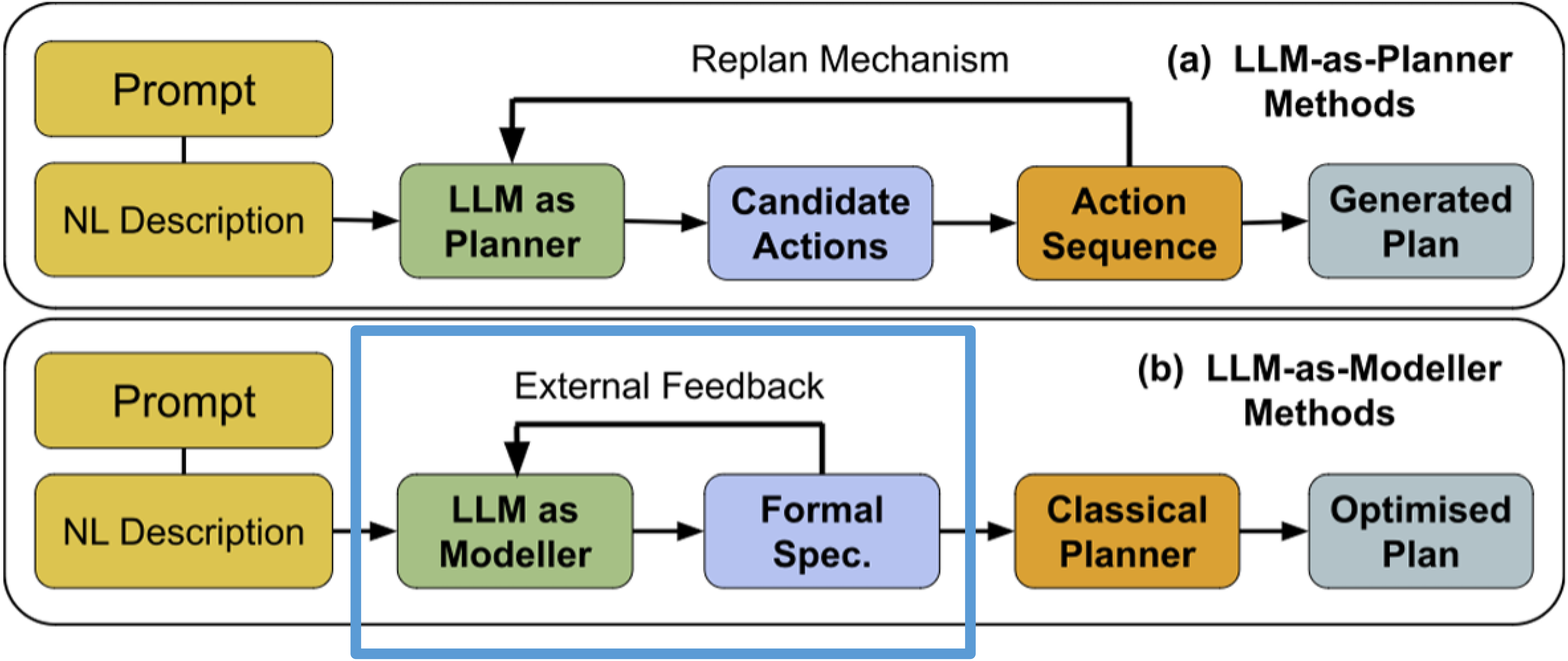
- MACQ is an open-source Python library that offers:
  - Trace generation
  - Trace visualization
  - Trace tokenization
  - Model acquisition

- The MACQ Visualizer is a website that offers:
  - A holistic view of model acquisition techniques
  - AI-Generated suggestions for future papers

- Contributions are welcome!

L2P:
Language-to-Plan


Replan Mechanism
(a) LLM-as-Planner Methods
Prompt
NL Description
LLM as Planner
Candidate Actions
Action Sequence
Generated Plan


Replan Mechanism
(a) LLM-as-Planner Methods
Prompt
NL Description
LLM as Planner
Candidate Actions
Action Sequence
Generated Plan
(b) LLM-as-Modeller Methods
External Feedback
Prompt
NL Description
LLM as Modeller
Formal Spec.
Classical Planner
Optimised Plan

PROBLEM

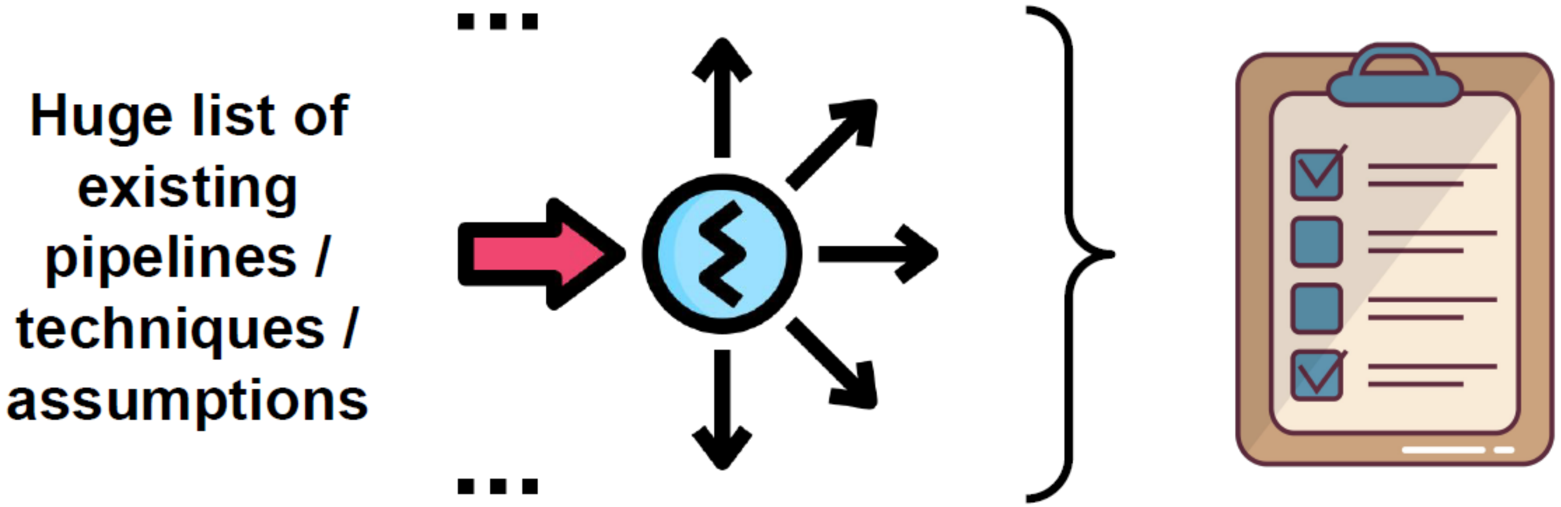
LLMs can be a bit … unpredictable

There is a fragmented landscape of NL-PDDL methods with each work possessing **different levels of assumptions:**

1. **Granularity** of natural language descriptions?	Explicit vs. Minimal Descriptions
2. What kind of **assumptions** are given?	Given info (levels of grounding)
3. What kind of **prompting styles** are used?	In-context, CoT, other styles…
4. What **generative techniques** are used?	Direct vs. Incremental Generation

MOTIVATION

Narrow down these techniques to provide actual insights to the limitations and advantages each of these works possess.

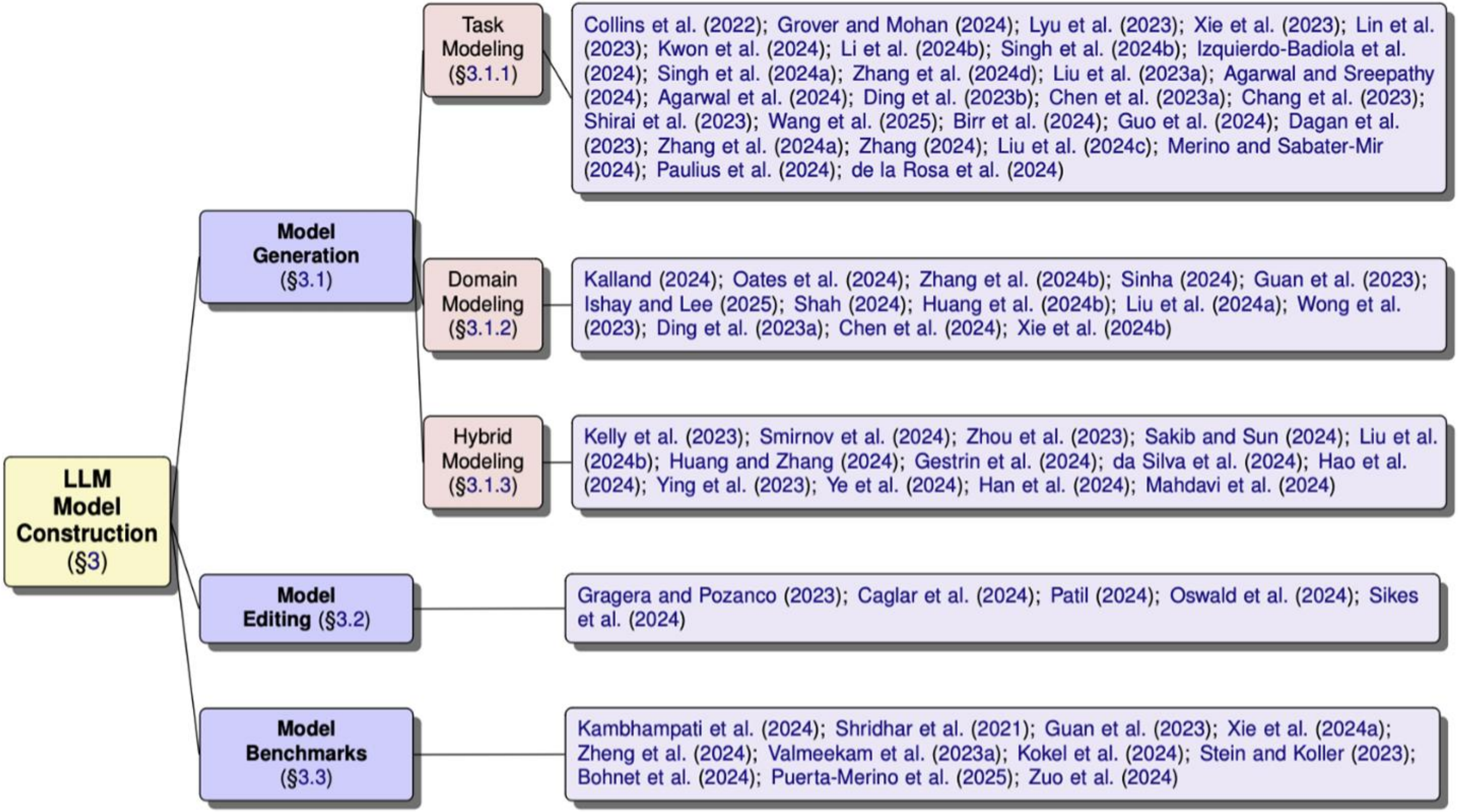


Huge list of existing pipelines / techniques / assumptions

...

...

The next steps towards applying LLMs in **real-world applications** is to establish a **standard, fair-comparison** of these frameworks – **What works? What doesn’t work?**


LLM Model Construction (§3)
Model Generation (§3.1)
Task Modeling (§3.1.1)
Collins et al. (2022); Grover and Mohan (2024); Lyu et al. (2023); Xie et al. (2023); Lin et al. (2023); Kwon et al. (2024); Li et al. (2024b); Singh et al. (2024b); Izquierdo-Badiola et al. (2024); Singh et al. (2024a); Zhang et al. (2024d); Liu et al. (2023a); Agarwal and Sreepathy (2024); Agarwal et al. (2024); Ding et al. (2023b); Chen et al. (2023a); Chang et al. (2023); Shirai et al. (2023); Wang et al. (2025); Birr et al. (2024); Guo et al. (2024); Dagan et al. (2023); Zhang et al. (2024a); Zhang (2024); Liu et al. (2024c); Merino and Sabater-Mir (2024); Paulius et al. (2024); de la Rosa et al. (2024)
Domain Modeling (§3.1.2)
Kalland (2024); Oates et al. (2024); Zhang et al. (2024b); Sinha (2024); Guan et al. (2023); Ishay and Lee (2025); Shah (2024); Huang et al. (2024b); Liu et al. (2024a); Wong et al. (2023); Ding et al. (2023a); Chen et al. (2024); Xie et al. (2024b)
Hybrid Modeling (§3.1.3)
Kelly et al. (2023); Smirnov et al. (2024); Zhou et al. (2023); Sakib and Sun (2024); Liu et al. (2024b); Huang and Zhang (2024); Gestrin et al. (2024); da Silva et al. (2024); Hao et al. (2024); Ying et al. (2023); Ye et al. (2024); Han et al. (2024); Mahdavi et al. (2024)
Model Editing (§3.2)
Gragera and Pozanco (2023); Caglar et al. (2024); Patil (2024); Oswald et al. (2024); Sikes et al. (2024)
Model Benchmarks (§3.3)
Kambhampati et al. (2024); Shridhar et al. (2021); Guan et al. (2023); Xie et al. (2024a); Zheng et al. (2024); Valmeekam et al. (2023a); Kokel et al. (2024); Stein and Koller (2023); Bohnet et al. (2024); Puerta-Merino et al. (2025); Zuo et al. (2024)

Language-to-Plan (L2P)

With the proliferations of emerging NL-PDDL extraction techniques, we introduce **Language-to-Plan (L2P)**, an open-source Python library that **unifies NL-PDDL frameworks into to a single umbrella** which can then be tested on rigorous benchmarks.

L2P possesses capabilities of constructing core PDDL components that enables researchers to create their own NL-PDDL pipelines

Comprehensive Tool Suite: easily plug in various LLMs for streamlined extraction experiments with our extensive collection of PDDL extraction and refining tools.

Modular Design: facilitates flexible PDDL generation, allowing users to explore prompting and create customized pipelines.

Autonomous Capability: building block support for fully autonomous pipeline, reducing manual efforts of producing LLM-AP pipelines from scratch.

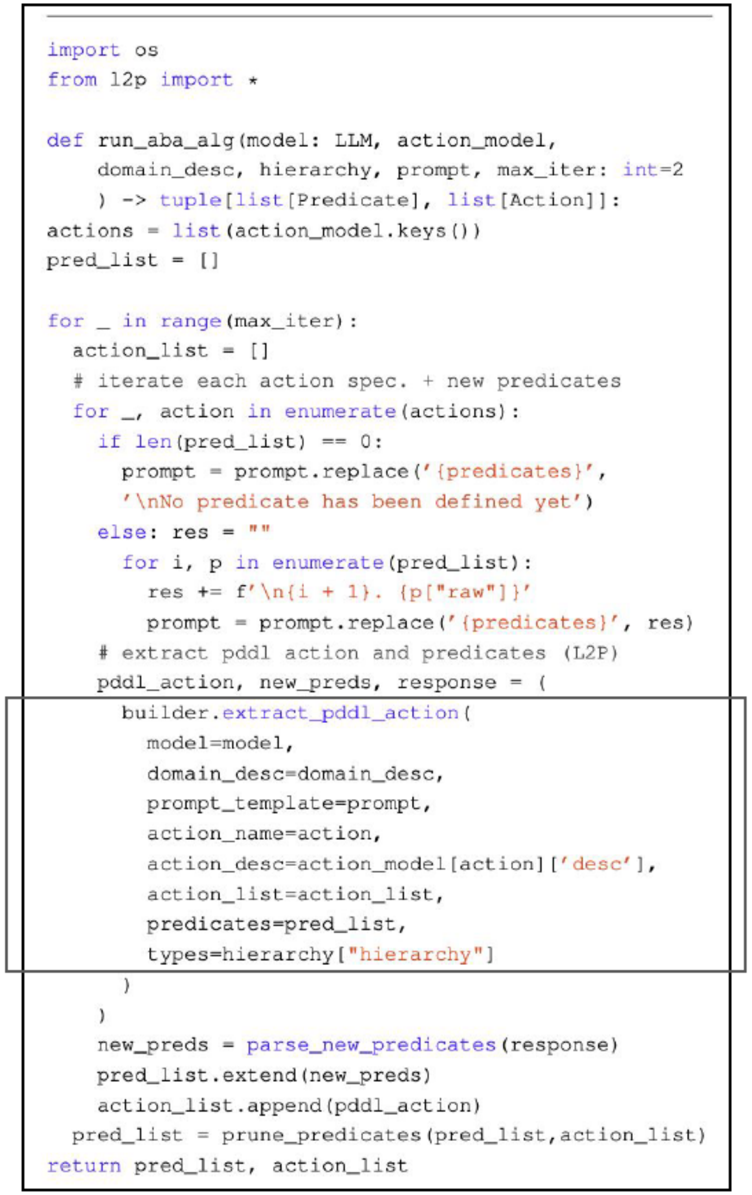
Language-to-Plan (L2P) – Examples

L2P can **recreate and encompass previous frameworks** for converting NL-PDDL, serving as a comprehensive foundation that integrates past approaches.

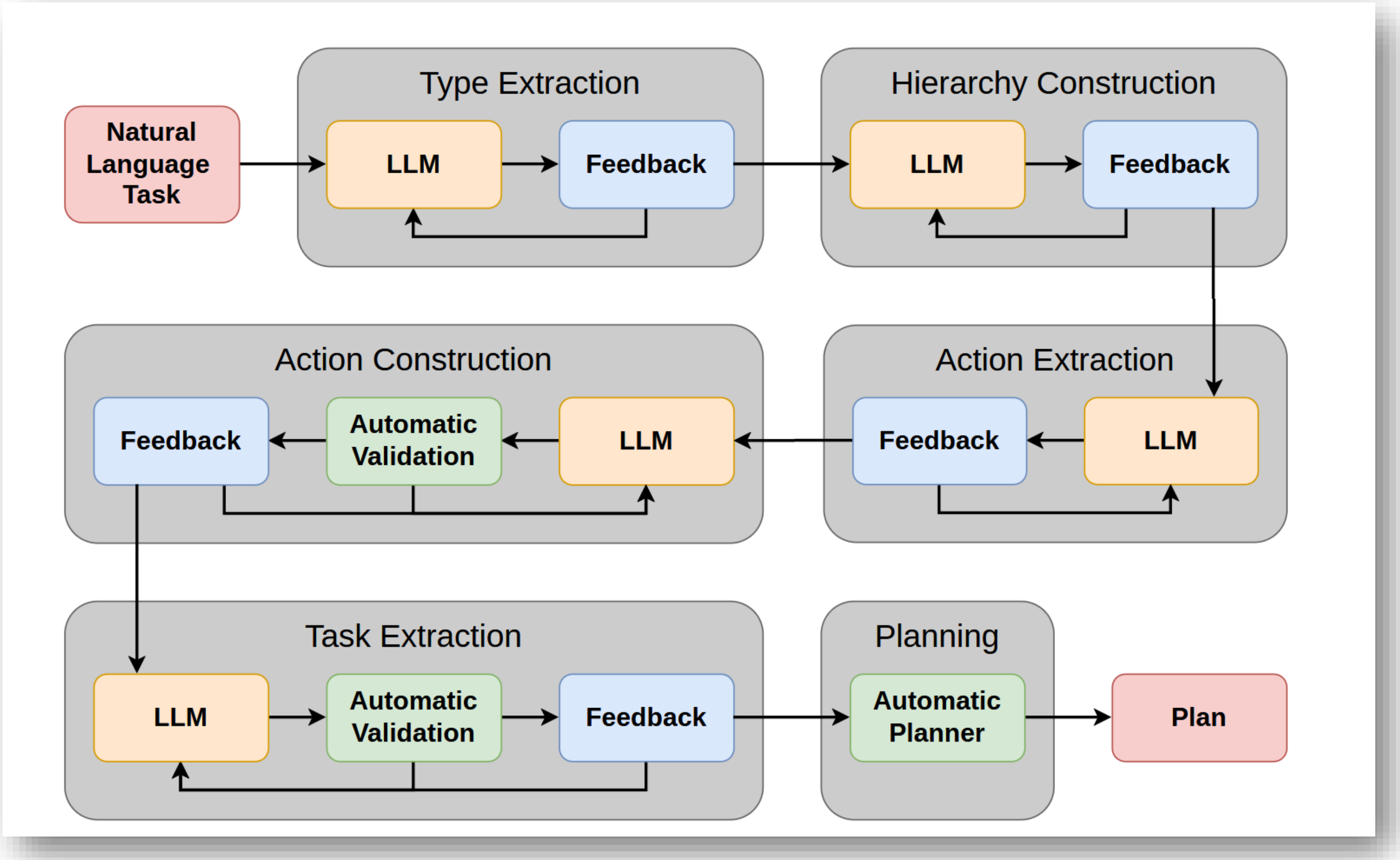
Some examples:

❑ LLM+P
❑ LLM-DM (example to the right)
❑ NL2Plan
❑ P+S
❑ PROC2PDDL

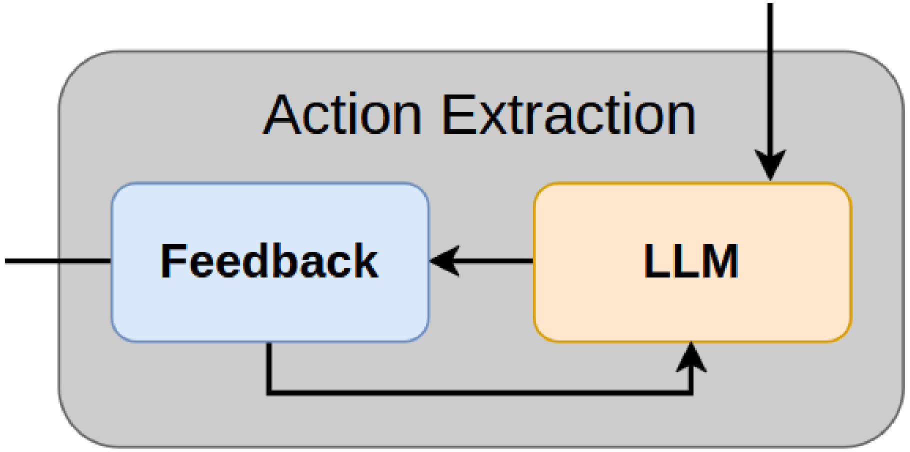
Shortened example of “Action-by-action” algorithm from (Guan et al. 2023)

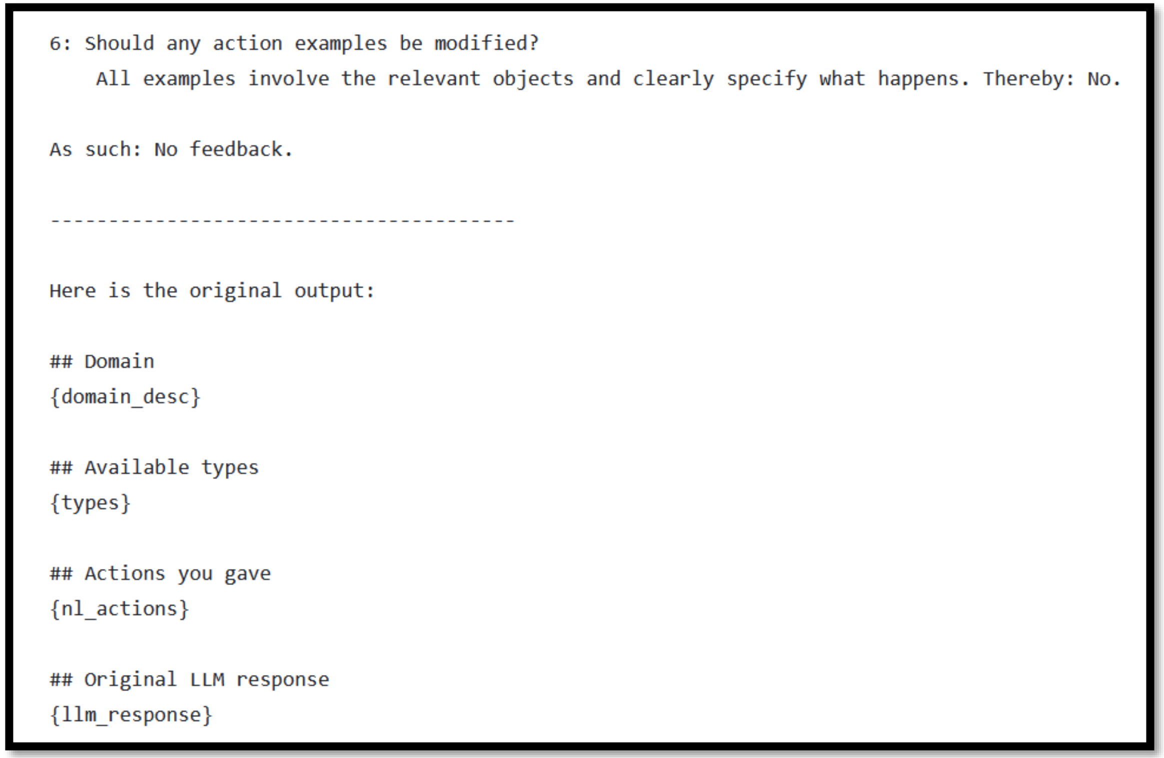


```
import os
from l2p import *

def run_aba_alg(model: LLM, action_model,
    domain_desc, hierarchy, prompt, max_iter: int=2
  ) -> tuple[list[Predicate], list[Action]]:
actions = list(action_model.keys())
pred_list = []

for _ in range(max_iter):
  action_list = []
  # iterate each action spec. + new predicates
  for _, action in enumerate(actions):
    if len(pred_list) == 0:
      prompt = prompt.replace(’{predicates}’,
      ’\nNo predicate has been defined yet’)
    else: res = ""
      for i, p in enumerate(pred_list):
        res += f’\n{i + 1}. {p["raw"]}’
        prompt = prompt.replace(’{predicates}’, res)
    # extract pddl action and predicates (L2P)
    pddl_action, new_preds, response = (
      builder.extract_pddl_action(
        model=model,
        domain_desc=domain_desc,
        prompt_template=prompt,
        action_name=action,
        action_desc=action_model[action][’desc’],
        action_list=action_list,
        predicates=pred_list,
        types=hierarchy["hierarchy"]
      )
    )
    new_preds = parse_new_predicates(response)
    pred_list.extend(new_preds)
    action_list.append(pddl_action)
  pred_list = prune_predicates(pred_list,action_list)
return pred_list, action_list
```

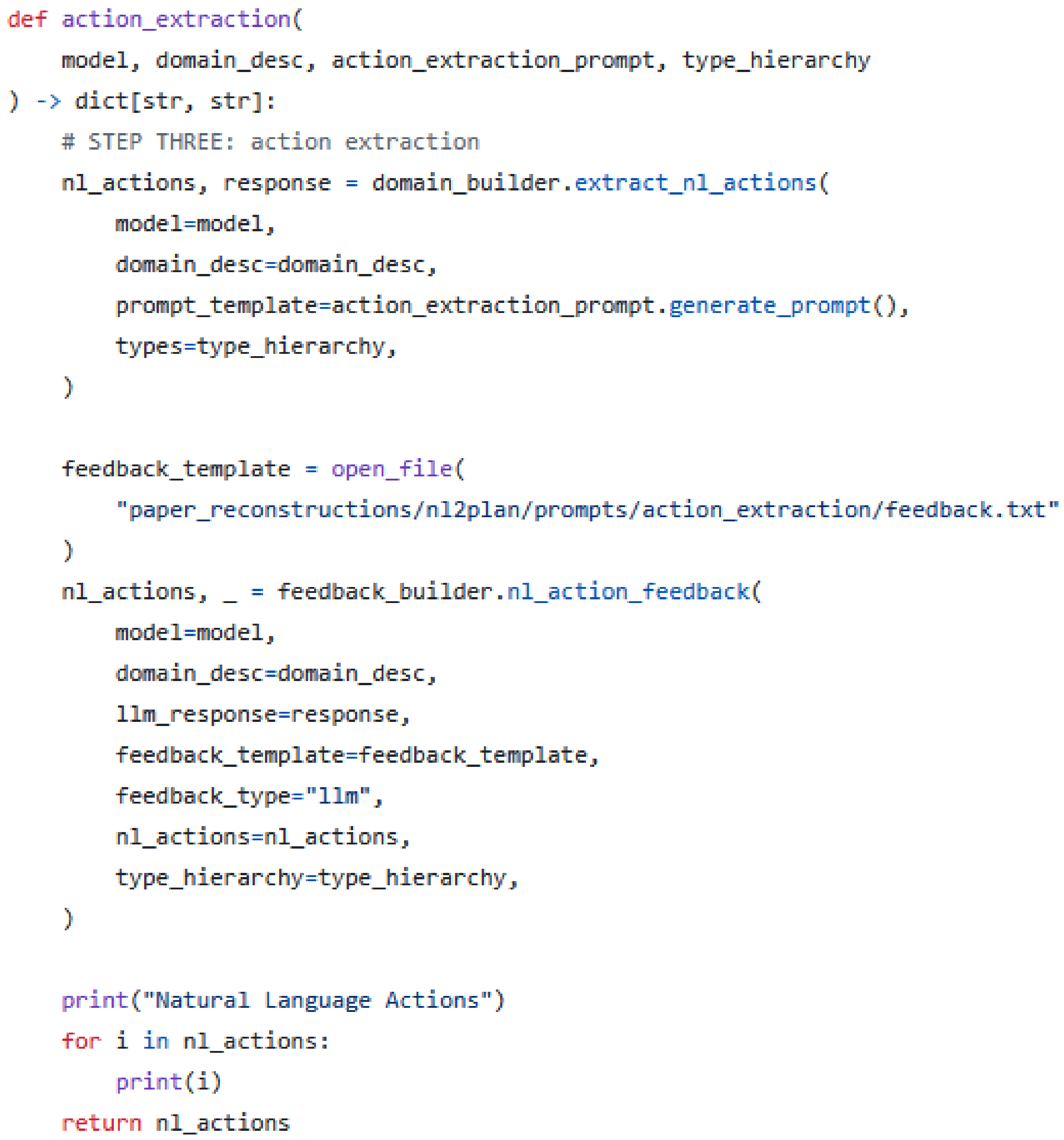


"**NL2Plan: Robust LLM-Driven Planning from Minimal Text Descriptions**" *Gestrin et al.* (2024)

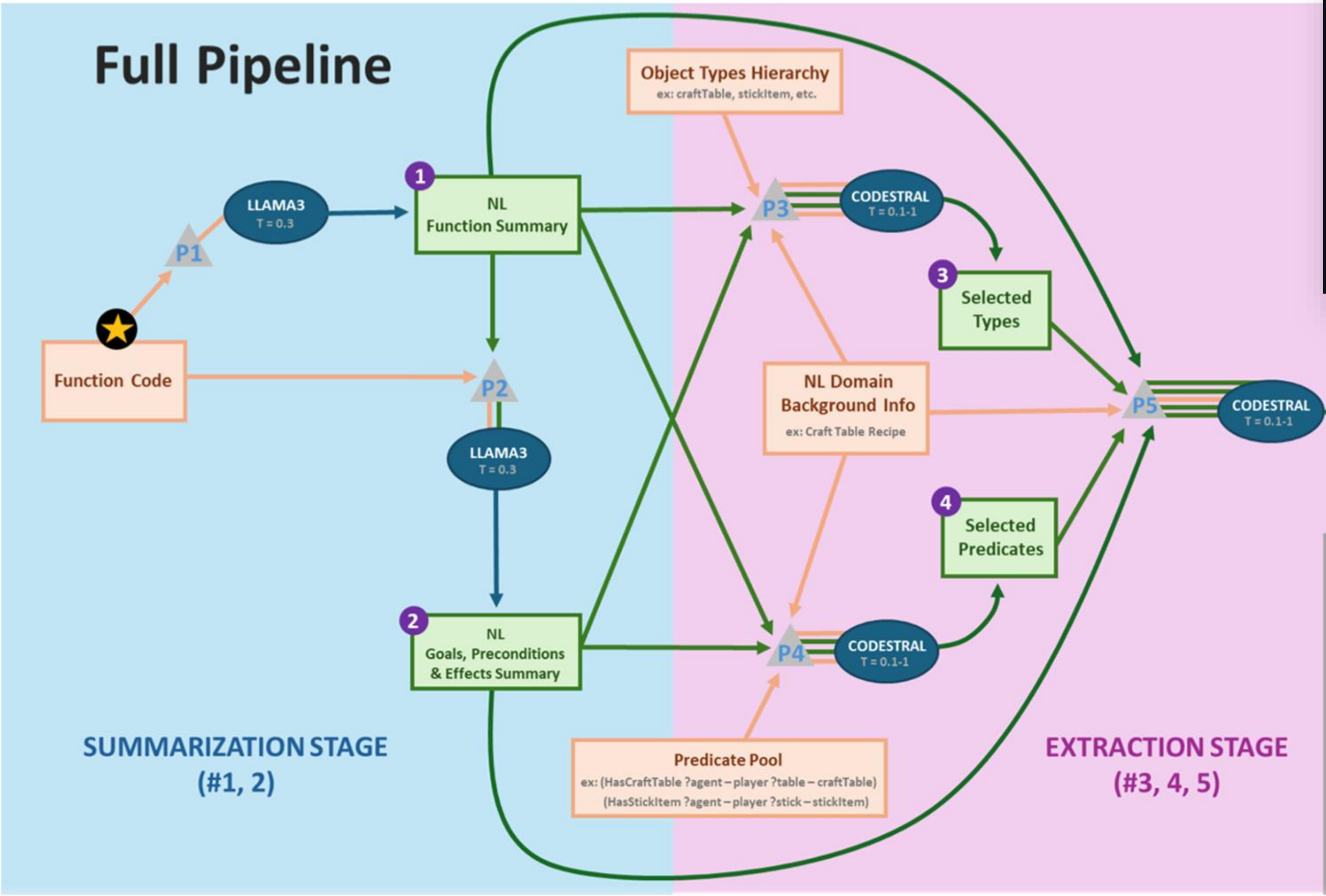


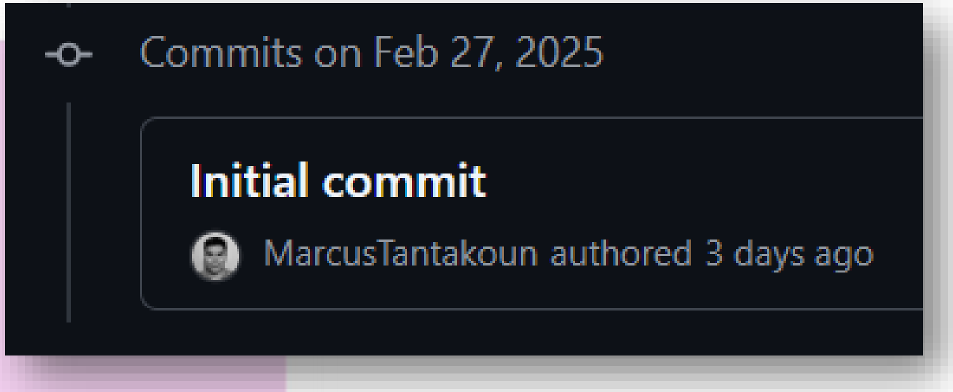


```
6: Should any action examples be modified?
    All examples involve the relevant objects and clearly specify what happens. Thereby: No.

As such: No feedback.

----------------------------------------

Here is the original output:

## Domain
{domain_desc}

## Available types
{types}

## Actions you gave
{nl_actions}

## Original LLM response
{llm_response}
```



```python
def action_extraction(
    model, domain_desc, action_extraction_prompt, type_hierarchy
) -> dict[str, str]:
    # STEP THREE: action extraction
    nl_actions, response = domain_builder.extract_nl_actions(
        model=model,
        domain_desc=domain_desc,
        prompt_template=action_extraction_prompt.generate_prompt(),
        types=type_hierarchy,
    )

    feedback_template = open_file(
        "paper_reconstructions/nl2plan/prompts/action_extraction/feedback.txt"
    )
    nl_actions, _ = feedback_builder.nl_action_feedback(
        model=model,
        domain_desc=domain_desc,
        llm_response=response,
        feedback_template=feedback_template,
        feedback_type="llm",
        nl_actions=nl_actions,
        type_hierarchy=type_hierarchy,
    )

    print("Natural Language Actions")
    for i in nl_actions:
        print(i)
    return nl_actions
```





Commits on Feb 27, 2025

Initial commit

MarcusTantakoun authored 3 days ago

github.com/MarcusTantakoun/JS-PDDL

"**Creating PDDL Models from Javascript using LLMs: Preliminary Results**" (Sikes et al. 2025)

Try it out!

```
pip install l2p
```

https://github.com/AI-Planning/l2p