
# User-Driven Capability Assessment of Taskable AI Systems


Pulkit Verma
MIT CSAIL


Siddharth Srivastava
AAIR Autonomous Agents and Intelligent Robots
ASU Arizona State University

bit.ly/aia25-tutorial

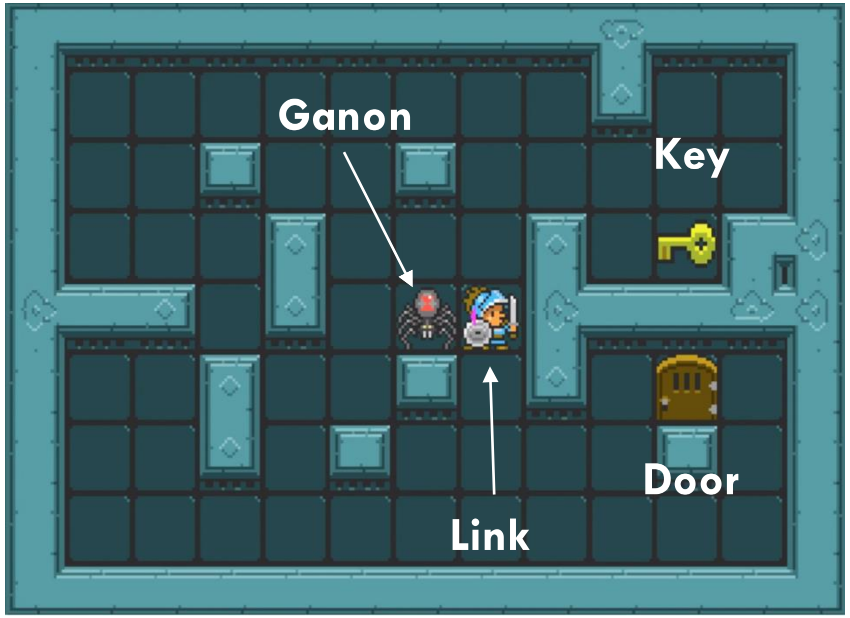
# Schedule

| | |
|---|---|
| 08:30 AM | Meet and Greet over Coffee |
| 09:00 AM | Session 1<br>• Introduction and Motivation<br>• Assessment through Model Learning<br>• Assessment of Black-Box AI Systems in Stationary Settings |
| 10:30 AM | Coffee Break |
| 11:00 AM | Session 2<br>• Discovering Capabilities for Black-Box AI Assessment<br>• AI Assessment in Adaptive Settings<br>• Future Directions and Conclusion |
| 12:30 PM | Lunch |


# Discovering Capabilities for Black-Box AI Assessment

# Capability v/s Functionality

- **Functionality:** Set of possible low-level actions of the agent.
- **Capability:** What agent's planning and learning algorithms can do.




| Agent Actions (Keystrokes) | Learned Capabilities |
|---|---|
| W | (defeat ganon) |
| A | (go to door) |
| S | (go to key) |
| D | (go to ganon) |
| E | (pick key) |
| | (open door) |

**Knowledge of primitive actions might be insufficient to understand the agent's capabilities**

# Discovering User-Interpretable Capabilities of Black-Box Planning Agents

*Pulkit Verma, Shashank Rao Marpally, and Siddharth Srivastava*

KR 2022

# Vocabulary Acquisition

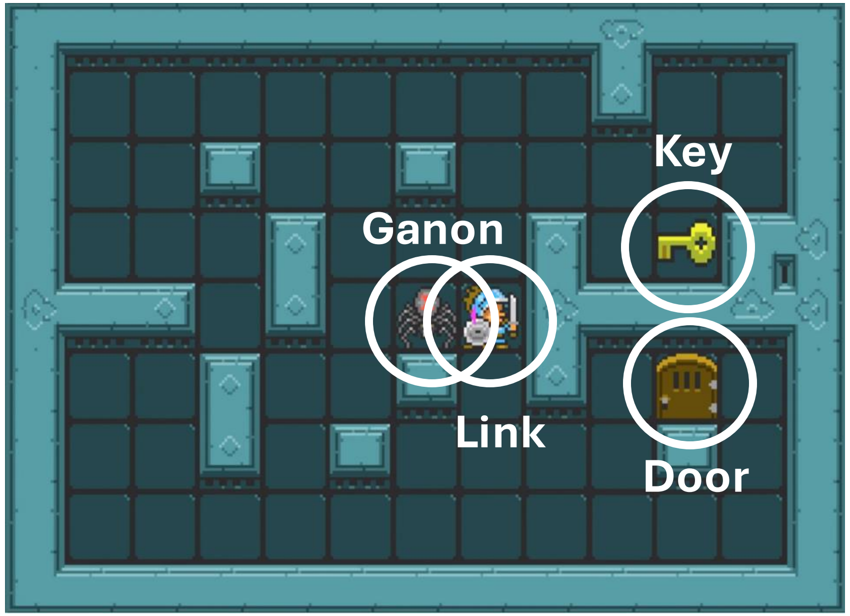
- Users share same vocabulary in same workspaces.
  - E.g., factory workers, coworkers, etc.

- Training the users on some predefined vocabulary.

- Using vocabulary acquisition techniques like TCAV†, etc.

†Kim et al. Interpretability beyond feature attribution: Testing with Concept Activation Vectors. In Proc. ICML 2018.

# User Vocabulary can be Less Expressive




| Agent's State Representation | State Representation in User's Vocabulary |
|---|---|
| pixel_1_1(#42A8B3) | (at ganon 5,3) |
| pixel_1_2(#42A8B3) | (at link 6,3) |
| . | (at key 9,4) |
| . | (at door 9,2) |
| . | |
| pixel_n_m(#203A3D) | |

# Discovering Capabilities

## Input

- Predicates (User vocabulary)
  - With their evaluation functions
- Samplers: high-level state to low-level state.
- Low-level state transitions.


## Output

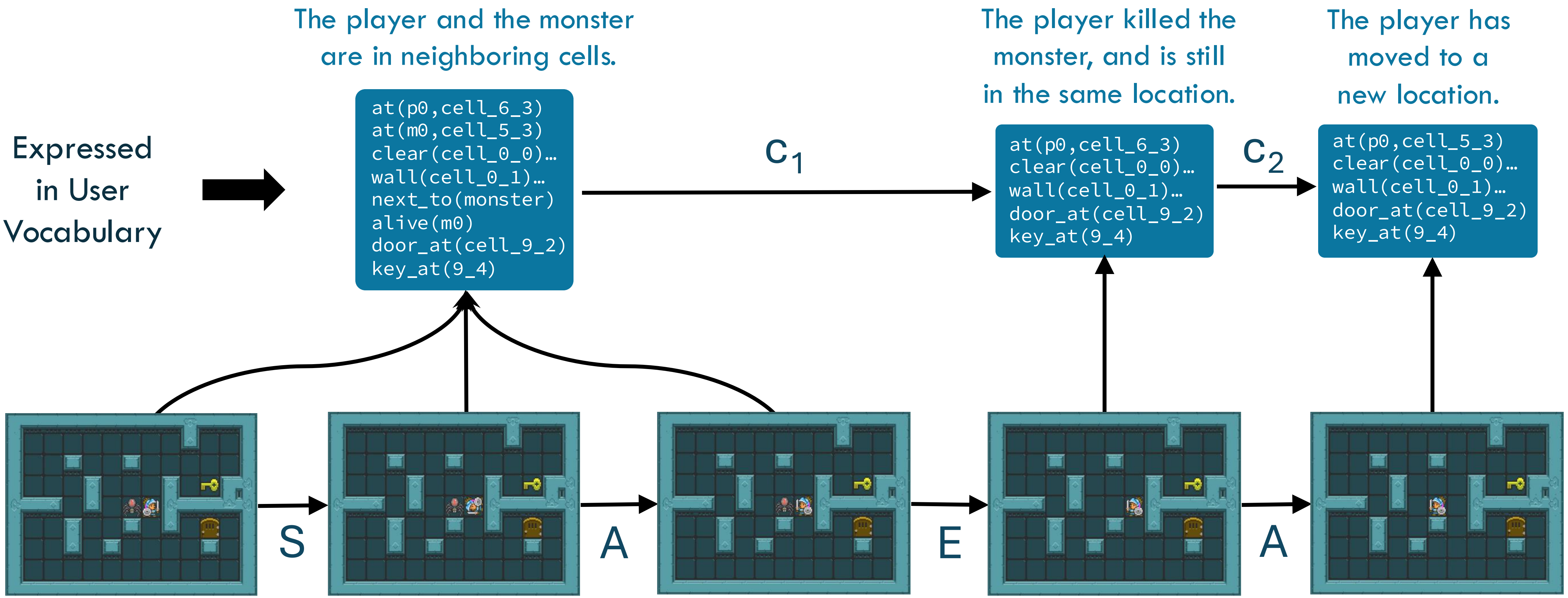
- List of capabilities.
- PDDL-like description of each capability.

[**Verma**, Marpally, Srivastava; KR '22]

## Assumptions

- ~~User's vocabulary matches simulator's vocabulary.~~
- Black-Box AI provides a list of ~~capabilities.~~ transitions.
- Stationary agent model.
- Deterministic environment.
- Fully observable setting.

# Discovering Capabilities using Input Predicates as Abstractions



# Parameterizing a Capability

```
at(p0,cell_6_3)
at(m0,cell_5_3)
clear(cell_0_0)…
wall(cell_0_1)…
next_to(monster)
alive(m0)
door_at(cell_9_2)
key_at(9_4)
```

→

```
at(p0,cell_6_3)
clear(cell_0_0)…
wall(cell_0_1)…
door_at(cell_9_2)
key_at(9_4)
```

[Sample pre and post states of a capability]

```
(:capability c4
 :parameters (?player1 ?cell1
   ?monster1 ?cell2)
 :precondition
  (and (alive ?monster1)
    (at ?player1 ?cell1)
    (at ?monster1 ?cell2)
    (next_to ?monster1))
 :effect
  (and (clear ?cell2)
   (not(alive ?monster1))
   (not(at ?monster1 ?cell2))
   (not(next_to ?monster1))))
```

How to learn these?

[Learned capability description]

For each capability:

- Extract what predicates were different in the pre and post-states of the capability.
- Extract the parameters from those predicates to create a candidate parameter set.
- Complete the parameter set along with capability description as precondition and effect of a capability by active querying.

# Query Refinement

$s_0, \langle c_1, c_2, \dots, c_n \rangle$

What will happen if you execute the plan $\langle c_1, c_2, \dots, c_n \rangle$ starting in a state $s_0$?

**Plan Outcome Query**

```
(:capability c1
  :parameters (…)
  :precondition (…)
  :effect (…))
```

```
(:capability c2
  :parameters (…)
  :precondition (…)
  :effect (…))
```

. . .

$s_0, c_1, s_1, c_2, s_2, \dots, c_n, s_n$

$\langle s_0, s_1 \rangle, \langle s_1, s_2 \rangle, \dots, \langle s_{n-1}, s_n \rangle$

**State Refinement**

$\langle \bar{s}_0, \bar{s}_1 \rangle, \langle \bar{s}_1, \bar{s}_2 \rangle, \dots, \langle \bar{s}_{n-1}, \bar{s}_n \rangle$

Can you reach state $\bar{s}_1$ from state $\bar{s}_0$?

Can you reach state $\bar{s}_2$ from state $\bar{s}_1$?

. . .

Can you reach state $\bar{s}_n$ from state $\bar{s}_{n-1}$?

**State Reachability Queries**

# Response Interpretation

High-level Query Generator

$\langle n, s_n \rangle$

$\langle \bar{s}_0, \bar{s}_1 \rangle$

$\langle \bar{s}_1, \bar{s}_2 \rangle$

•
•
•

$\langle \bar{s}_{n-1}, \bar{s}_n \rangle$

Iterative Capability Model Learning [iCaML]

Can you reach state $\bar{s}_1$ from state $\bar{s}_0$?

Can you reach state $\bar{s}_2$ from state $\bar{s}_1$?

Can you reach state $\bar{s}_n$ from state $\bar{s}_{n-1}$?

YES

# Response Interpretation

High-level Query Generator

$\langle 1, s_1 \rangle$

$\langle \bar{s}_0, \bar{s}_1 \rangle$

$\langle \bar{s}_1, \bar{s}_2 \rangle$

$\vdots$

$\langle \bar{s}_{n-1}, \bar{s}_n \rangle$

Iterative Capability Model Learning [iCaML]

Can you reach state $\bar{s}_1$ from state $\bar{s}_0$?

Can you reach state $\bar{s}_2$ from state $\bar{s}_1$?

No

# Example of a Learned Capability Description

```
(:capability c4
 :parameters (?player1 ?cell1
   ?monster1 ?cell2)
 :precondition
  (and (alive ?monster1)
    (at ?player1 ?cell1)
    (at ?monster1 ?cell2)
    (next_to ?monster1))
 :effect
  (and (clear ?cell2)
   (not(alive ?monster1))
   (not(at ?monster1 ?cell2))
   (not(next_to ?monster1))))
```

Position of Link has not changed

Ganon is not at its previous location

Ganon is not alive anymore

Link is not next to Ganon

This capability is: "Defeat Ganon"

# User Study Setup to Verify Interpretability

Preconditions

4. **Capability C4**:
The *player* can execute this capability when:

- The *monster* is not defeated.
- The *player* is in *cell1*.
- The monster is in *cell2*.
- The player is in a cell adjacent to the *monster*.

Effects

After the *player* executes this capability:

- *Cell2* is empty.
- The *monster* is defeated.
- The *monster* is not in *cell2*.
- The player is not in a cell adjacent to the *monster*.

**Question 4 of 12:**
Select the phrase that best summarizes the capability **C4**? We will use your response while referring to this capability **C4** later in the survey.

Go next to Door
Go next to Ganon
Go next to Key
Go next to Wall
Defeat Ganon
Break Key
Pick Key
Open Door

Possible options to choose from

[Capability Description Example]

Keystroke Description

**W**: Pressing this key does the following:

- If Link is facing up and there is no wall, door, or key in the cell above, then Link moves to the cell above.
- If there is a wall, door, or key in the cell above Link, then Link stays in the same cell.
- If Link is facing Left, Right, or Down before pressing W, then Link faces up but stays in the same cell.

**Question 1 of 11:**
Select the phrase that best summarizes pressing **W**? We will use your response while referring to this key **W** later in the survey.

Up
Down
Left
Right
Interact

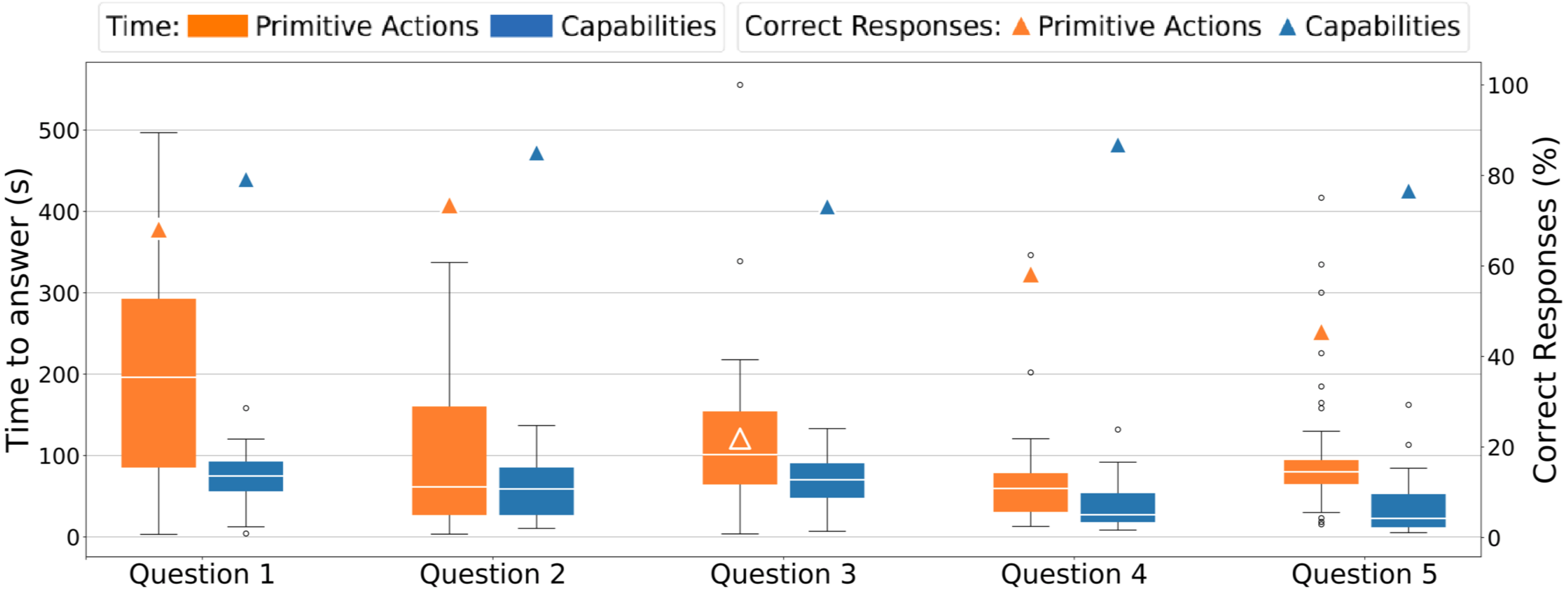
Possible options to choose from

[Functionality Description Example]

# Utility of Discovered Capability Descriptions

If Link starts in the state shown below:

Which sequence of actions can Link take to reach the state shown below:



# Results: Capability Summarization Study

1. **Action A1**

*Link* can execute this action when:

- Link's cell is not empty.
- *Link*'s cell is connected via a path to a destination cell adjacent to *Ganon*.
- The destination cell is empty.
- *Link* is not in the destination cell.

After *Link* executes this action:

- *Link* is in the destination cell.
- *Link* is no longer in its previous cell.
- Link's previous cell is empty.
- *Link* and *Ganon* are in adjacent cells.

Go next to Door
Go next to Ganon
Go next to Key
Go next to Wall ✗
Defeat Ganon
Break Key ✗
Pick Key
Open Door

**Question 1 of 12:**

Select the phrase that best summarizes action **A1**? We will use your response while referring to this action **A1** later in the survey.

| | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 |
|---|---|---|---|---|---|---|---|---|
| C1 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C2 | 0 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C3 | 0 | 0 | 0.94 | 0 | 0 | 0 | 0.06 | 0 |
| C4 | 0 | 0 | 0 | 1.0 | 0 | 0 | 0 | 0 |
| C5 | 0 | 0 | 0 | 0 | 0.94 | 0 | 0 | 0.06 |
| C6 | 0 | 0 | 0 | 0 | 0 | 1.00 | 0 | 0 |

# Learned Capability Descriptions are Maximally Consistent

- Theorem (*consistency*):
  The learned descriptions are consistent with the observations and the queries.

- Theorem (*maximal consistency*):
  This approach is maximally consistent, i.e., we cannot add any more literals to the preconditions or effects without ruling out some truly possible models.

- Theorem (*probabilistic completeness*):
  In the limit of infinite execution traces, the probability of discovering all capabilities expressible in the user vocabulary is 1.

# Learning Neuro-Symbolic Skills for Bilevel Planning

*Tom Silver, Ashay Athalye, Josh Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling*

CoRL 2022

# Learn High-Level Skills for Robots

Concepts/Symbols

```
cupfilled (?c)
potOnPlate (?p ?p1)
aboveCup(?r ?c)
Holding (?r ?p)
…
```

$\langle s_0, a_1, s_1 \rangle$

$\langle s_1, a_2, s_2 \rangle$

$\vdots$

$\langle s_{n-1}, a_n, s_n \rangle$

Low-Level Transitions

**Learning**

Learned Skills

Pick up Pot

Place on Plate

Turn Plate On

…

Learning such skills can lead to efficient long horizon planning in continuous state and action spaces.

# Key Properties of a Skill

Abstractions are lossy, hence:

1. A skill should be able to reach many different environment states (“subgoals”) that correspond to the same abstract state.

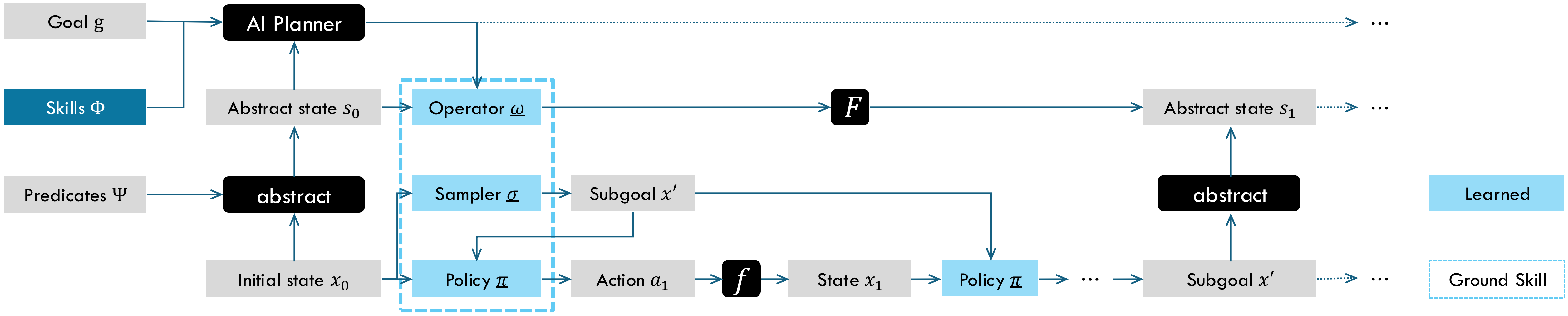
2. An agent should be able to consider multiple skill sequences that reach the same goal from the same initial abstract state.

# Components of a Skill

- Each Skill has 3 components:

- A Symbolic Operator (like action in PDDL)
- Neural subgoal-conditioned policy (like an option in RL)
- Neural subgoal sampler

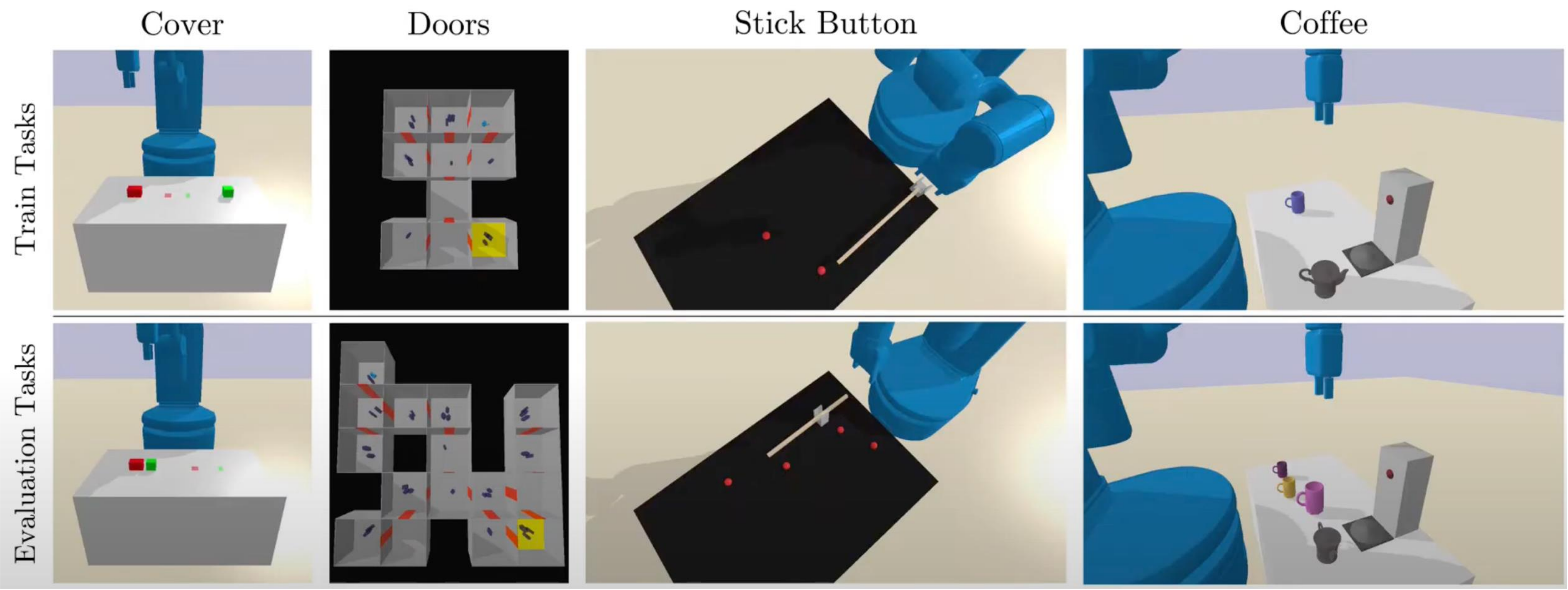
# Architecture



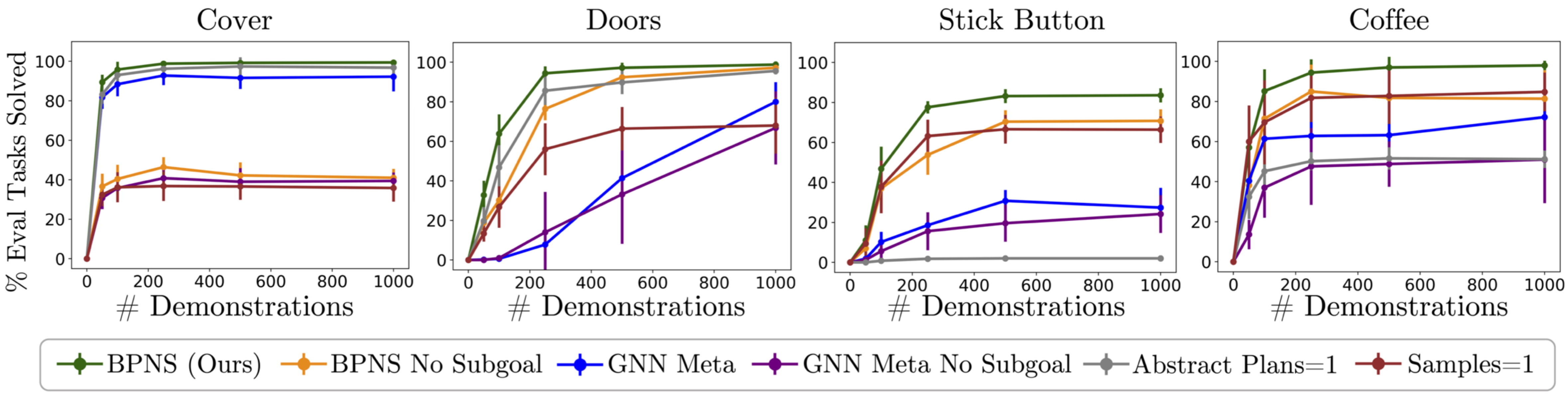
# Learning Neuro-Symbolic Skills

1. Preprocess demonstrations into skill datasets
2. Learn operators: symbolic techniques
3. Learn policies: supervised learning
4. Learn samplers: distribution learning

# Empirical Evaluation



# Efficient and Better Generalization across all domains

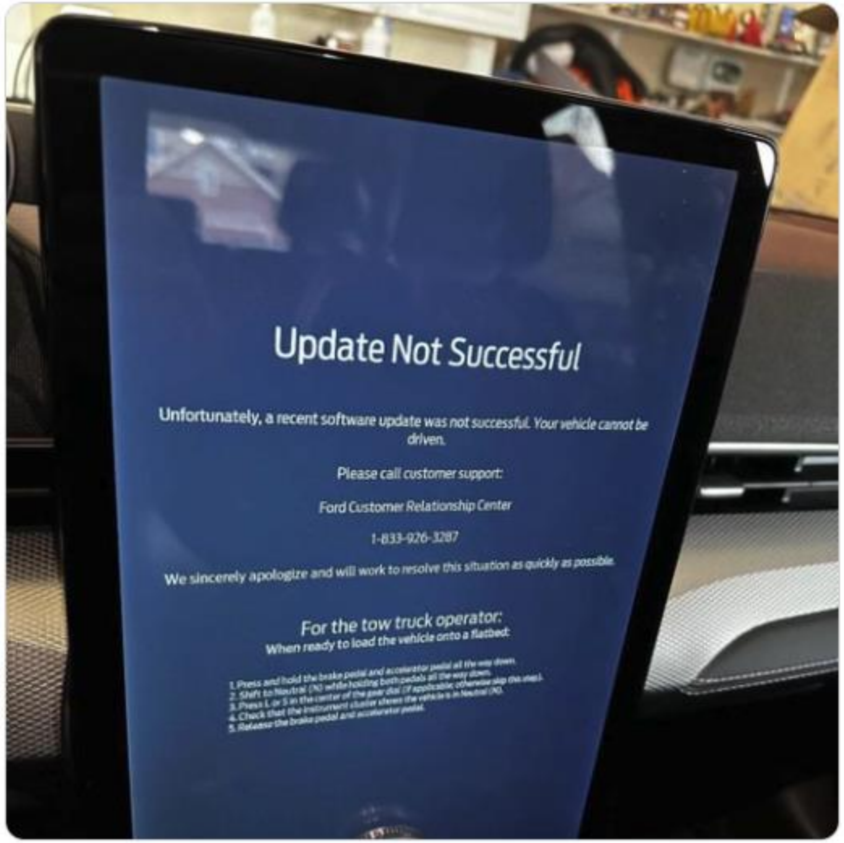



# AI Assessment in Adaptive Settings

# Adaptive Taskable AI Systems

**Dan Luu**
@danluu · **Follow**

"Unfortunately, a recent software update was not successful. Your vehicle cannot be driven. Please call customer support:"




1:46 PM · Dec 25, 2023

USA TODAY

## Tesla self-driving software update begins rollout though company says to use with caution

Charisse Jones, USA TODAY
July 12, 2021 · 2 min read

## Lucid Owners Facing Software Glitches That Brick EVs Or Drive the Wrong Direction

1

Owen Bellwood
November 10, 2022 · 2 min read

## AEG combi microwave thinks it is a steam oven and no longer works after an incorrect update

By **Julian Huijbregts** 04-03-2022 • 10:48
News editor

NEWS

## Nest thermostat software bug chills users once again

A faulty software update for the smart thermostat made batteries drain and home temperatures drop.

By Jared Newman
TechHive | JAN 14, 2016 8:38 AM PST

# Maintaining Evolving Domain Models

*Dan Bryce, J. Benton, and Michael W. Boldt*

IJCAI 2016

# Model Maintenance Problem

- Real-world domains evolve (e.g., changes in effectors or conditions).
- **Model drift :** Ground-truth and the model diverge
- Model Maintenance: A user's understanding (mental model) of a domain evolves, drifting away from the formal computational model of the domain

# Model Maintenance Problem

- Real-world domains evolve (e.g., changes in effectors or conditions).
- **Model drift :** Ground-truth and the model diverge

Domain Expert
knows model $M^u$

$M^u$ can evolve over time

**Marshal**

Model Maintenance Tool
that must keep $M$ updated
according to $M^u$

Automated Planner
using model $M$

# Model Representation

- User query is $u_t$, and observation received is $z_t$
- Formulate the prior distribution $P(\vec{X}_0)$ over models by assuming starting model where every feature is $\perp$
- Each possible model is a particle that can be sampled from a proposal distribution that considers both model drift and observations

$$q(\vec{x}_t^{(i)} | \vec{x}_{t-1}^{(i)}, z_t)$$

Model $\vec{x}$

```
(:action open-door
  :parameters (?l1)
  :precondition (and
x1  (⊤/⊥) (has_key)
x2  (⊤/⊥) (door_open)
x3  (⊤/⊥) (door_adjacent ?l1)
x4  (⊤/⊥) (player_at ?l1))
  :effect (and
x5  (⊤/⊥) (has_key)
x6  (⊤/⊥) (door_open)
x7  (⊤/⊥) (door_adjacent ?l1)
x8  (⊤/⊥) (player_at ?l1)))
```

# Marshal’s Learning Process

1. Query the User: Query the user with $u_t$ and receive $z_t$.

2. Generate N Samples based from proposal distribution:

$$\vec{x}_t^{(i)} \sim q(\vec{x}_t^{(i)} | \vec{x}_{t-1}^{(i)}, z_t)$$

3. Weight the Particles with their likelihoods

$$w_t^{(i)} = \frac{P\left(z_t \middle| \vec{x}_t^{(i)}\right) P\left(\vec{x}_t^{(i)} \middle| \vec{x}_{t-1}^{(i)}\right)}{q\left(\vec{x}_t^{(i)} \middle| \vec{x}_{t-1}^{(i)}, z_t\right)}$$

4. Resample Particles from the set of normalized-weighted particles to create the next belief state $\{\vec{x}_t^{(i)}\}$.

# Updating the Particles

- Verbatim (V)
- Uniform Drift (U)
- Uniform Drift Generalization (UG)
- Well-Formed Drift (W)
- Well-Formed Generalization (WG)

$$w_t^{(i)} = \frac{P\left(z_t \middle| \vec{x}_t^{(i)}\right) P\left(\vec{x}_t^{(i)} \middle| \vec{x}_{t-1}^{(i)}\right)}{q\left(\vec{x}_t^{(i)} \middle| \vec{x}_{t-1}^{(i)}, z_t\right)}$$

# Updating the Particles

- Verbatim (V): Update particles to be complicit with user domain update and query response observations. Ignore plan observations.

- Uniform Drift (U)

- Uniform Drift Generalization (UG)

- Well-Formed Drift (W)

- Well-Formed Generalization (WG)

$$w_t^{(i)} = \frac{P\left(z_t \middle| \vec{x}_t^{(i)}\right) P\left(\vec{x}_t^{(i)} \middle| \vec{x}_{t-1}^{(i)}\right)}{q\left(\vec{x}_t^{(i)} \middle| \vec{x}_{t-1}^{(i)}, z_t\right)} = \begin{cases} 1, \text{ if } \vec{x}_t^{(i)} \text{ respects } \vec{x}_{t-1}^{(i)} \text{ aside from updates specified by } z_t \\ 0, \text{ otherwise} \end{cases}$$

# Updating the Particles

- Verbatim (V)

- Uniform Drift (U): Similar to verbatim, but for plan observations, uniformly sample a single domain model feature to add (remove).

- Uniform Drift Generalization (UG)

- Well-Formed Drift (W)

- Well-Formed Generalization (WG)

$$w_t^{(i)} = \frac{P\left(Z_t \middle| \vec{x}_t^{(i)}\right) P\left(\vec{x}_t^{(i)} \middle| \vec{x}_{t-1}^{(i)}\right)}{q\left(\vec{x}_t^{(i)} \middle| \vec{x}_{t-1}^{(i)}, Z_t\right)} = \begin{cases} \frac{1}{|\mathcal{X}|}, & \text{if } \vec{x}_t^{(i)} \text{ differs from } \vec{x}_{t-1}^{(i)} \text{ by exactly one assignment} \\ 0, & \text{otherwise} \end{cases}$$

# Updating the Particles

- Verbatim (V)

$$w_t^{(i)} = \frac{P\left(z_t \middle| \vec{x}_t^{(i)}\right) P\left(\vec{x}_t^{(i)} \middle| \vec{x}_{t-1}^{(i)}\right)}{q\left(\vec{x}_t^{(i)} \middle| \vec{x}_{t-1}^{(i)}, z_t\right)} = \begin{cases} \frac{1}{|\mathcal{X}_G|}, & \text{if } \vec{x}_t^{(i)} \text{ differs from } \vec{x}_{t-1}^{(i)} \text{ by exactly one group of assignments} \\ 0, & \text{otherwise} \end{cases}$$

- Uniform Drift (U)
- Uniform Drift Generalization (UG): In addition to uniform drift, also add (remove) related domain model features.
- Well-Formed Drift (W)
- Well-Formed Generalization (WG)

# Updating the Particles

- $\alpha$ is a normalization constant
- a comes from a heuristic (see paper)

- Verbatim (V)
- Uniform Drift (U)
- Uniform Drift Generalization (UG)
- Well-Formed Drift (W): Similar to uniform drift, but treat plans differently.
- Well-Formed Generalization (WG)

$$w_t^{(i)} = \frac{P\left(Z_t \middle| \vec{x}_t^{(i)}\right) P\left(\vec{x}_t^{(i)} \middle| \vec{x}_{t-1}^{(i)}\right)}{q\left(\vec{x}_t^{(i)} \middle| \vec{x}_{t-1}^{(i)}, Z_t\right)} = \begin{cases} \alpha a, & \text{if } \vec{x}_t^{(i)} \text{ differs from } \vec{x}_{t-1}^{(i)} \text{ by exactly one group of assignments} \\ 0, & \text{otherwise} \end{cases}$$

# Weighting Particles

$$w_t^{(i)} = \frac{P\left(z_t \middle| \vec{x}_t^{(i)}\right) P\left(\vec{x}_t^{(i)} \middle| \vec{x}_{t-1}^{(i)}\right)}{q\left(\vec{x}_t^{(i)} \middle| \vec{x}_{t-1}^{(i)}, z_t\right)}$$

$$P\left(z_t \middle| \vec{x}_t^{(i)}\right) = \frac{1}{2^{|X|}}$$

$P\left(\vec{x}_t^{(i)} \middle| \vec{x}_{t-1}^{(i)}\right)$ set such that observations agreeing with a domain model have high probability (0.99) and those disagreeing have low probability (0.01)

# Empirical Evaluation

**Q1**. Must Marshal assume an evolving model, or can static change be assumed?

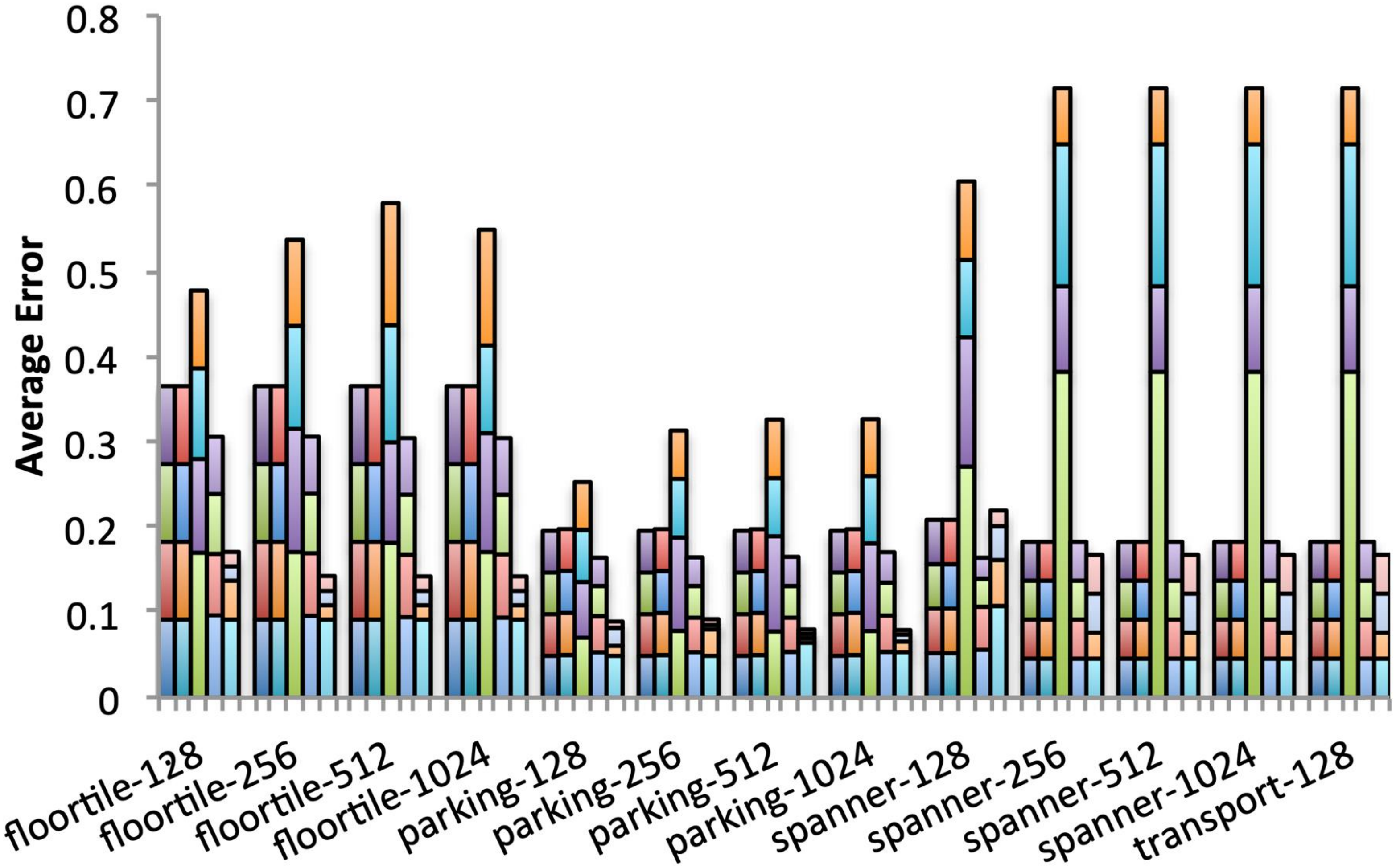
**Q2**. Do answers to queries help with the learning process, or are plan observations enough?

**Q3**. Does a uniform transition function operate as effectively as a more well-formed transition function that captures common traits of domains?

# Empirical Evaluation

- The user updates their mental model six times. Each change is over a precondition, add or delete effect in an action schema.
- After each change, the user provides a series of 108 plans that they believe are valid.
- After each plan, the user answers a series of Marshal’s questions in the order that Marshal determines.
- After each series of plans, and just prior to the next drift in the user’s model, we ask Marshal to calculate the probability (given its distribution over models) that each plan within a testing set of 28 plans is valid.
- Marshal uses 128, 256, 512, or 1024 particles in its particle filter

# Observations beyond just plans are useful to learn drifted models




- Each stacked column lists results for a method, from left to right (V, U, UG, W, WG).
- Within each column the results from the bottom to the top of the stack are for each number of queries per plan (0, 1, 2, 3).

# Differential Assessment of Black-Box AI Agents

*Rashmeet Kaur Nayyar*, Pulkit Verma*, and Siddharth Srivastava*

AAAI 2022

# Differential Assessment

## Input

- Initial model of the AI system.
  - Predicates (User vocabulary)
    - With their evaluation functions
  - List of capabilities.
- Observations of AI system working in the environment.

## Output

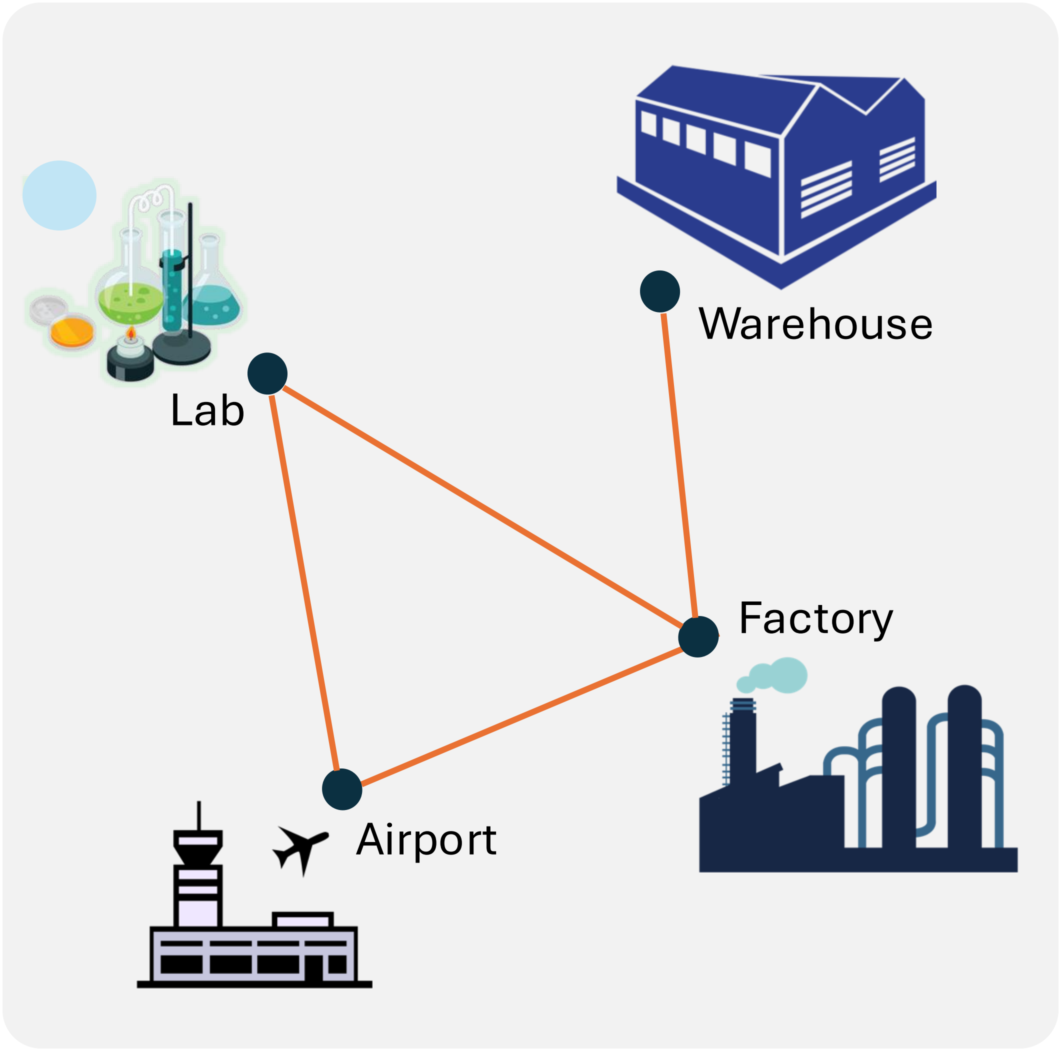
- Updated PDDL-like description of each capability.

Can we learn an updated model without doing a complete assessment?

## Assumptions

- User's vocabulary matches simulator's vocabulary.
- Black-Box AI provides a list of capabilities.
- Adaptive ~~Stationary~~ agent model.
- Deterministic environment.
- Fully observable setting.


Will it be able to safely navigate from the lab to the warehouse?
It is not how I knew it was supposed to navigate from the lab to warehouse. What has changed?


Warehouse
Lab
Factory
Airport

## Agent updates

E.g., software update,

new deployment,
adapted for user needs, etc.

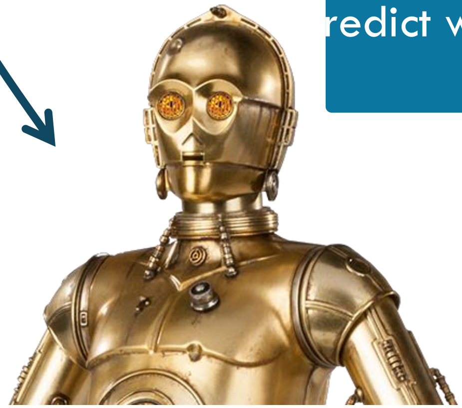
simulator

## Challenge 1

How to identify what has changed from sparse observations of agent's behavior?

Sparse Observations
(collected once)

Jse observations and $M_{init}$ t
redict what might've change

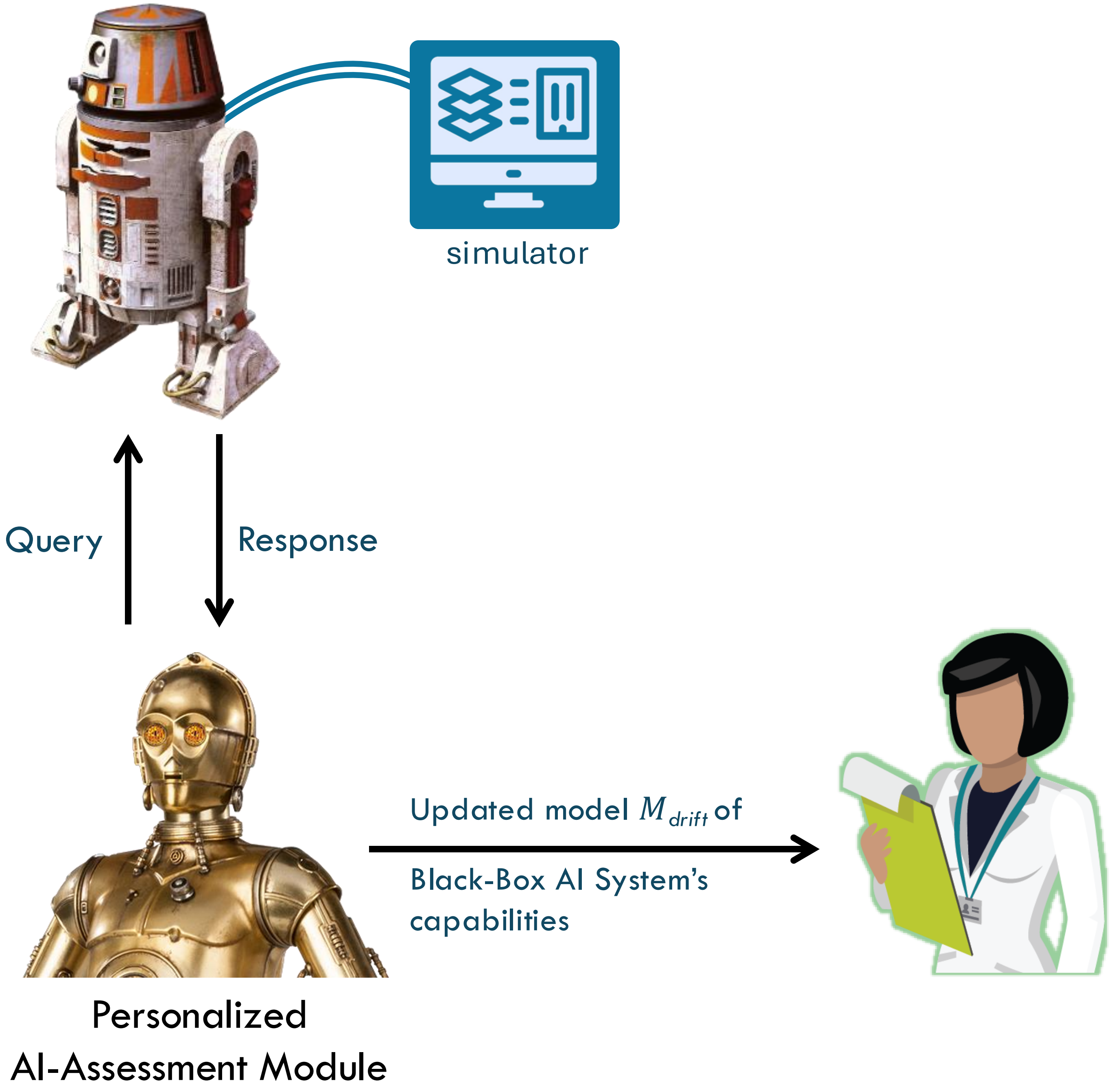


Initial Model
known to the user
$M_{init}$

Personalized
AI-Assessment Module

## Challenge 2

How to identify how the model has changed given what has changed?


simulator
Query
Response
Updated model $M_{drift}$ of
Black-Box AI System's
capabilities
Personalized
AI-Assessment Module

# What can change?

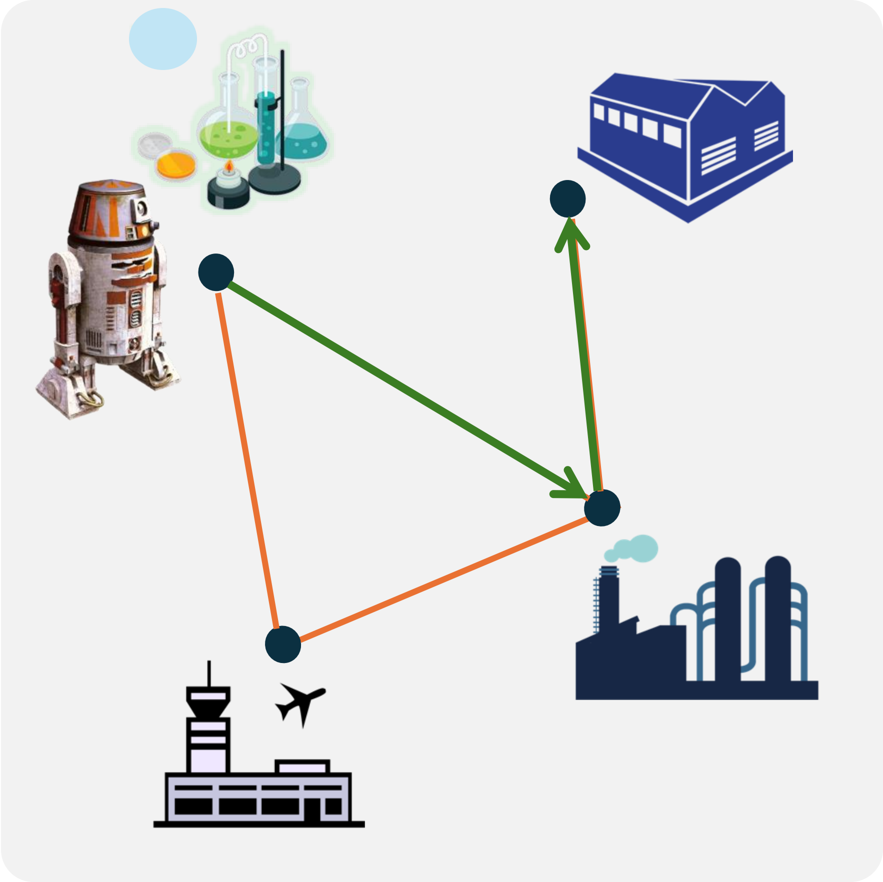
## Two broad categories

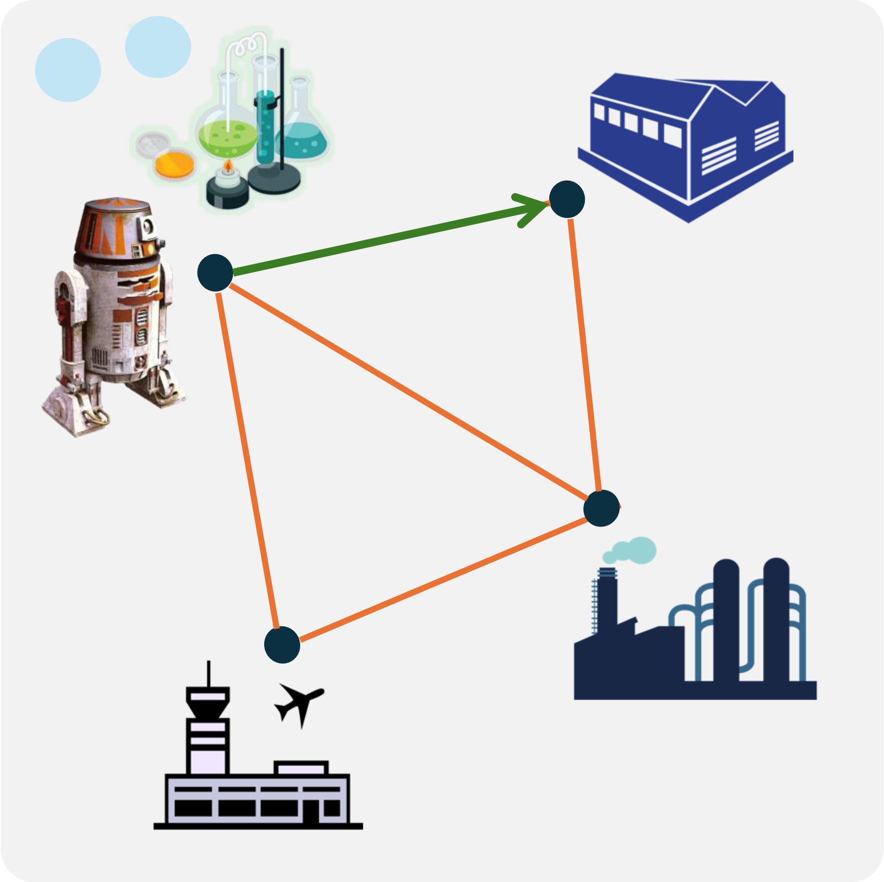
- Any of these could change their form:
  - From + to –, e.g., (handempty) to (not(handempty))
  - From – to +, e.g., (not(handempty)) to (handempty)
- Can get dropped from precondition or effect.

- Another predicate can get added as a precondition or effect.

```
(:action pick-samples
  :parameters (?s)
  :precondition (and
    (handempty)
    (onshelf ?s))
  :effect (and
    (not (handempty))
    (not (onshelf ?s))
    (holding ?s))
)
```

# Increased Functionality



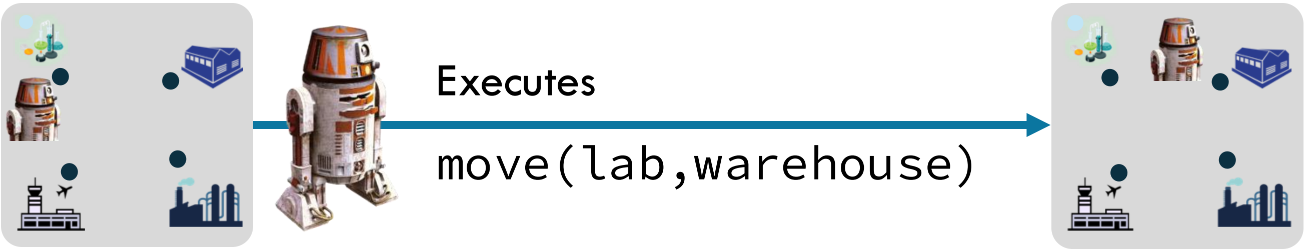


Observed that the agent executed
`move(lab,warehouse)`

# How to identify increased functionality ?

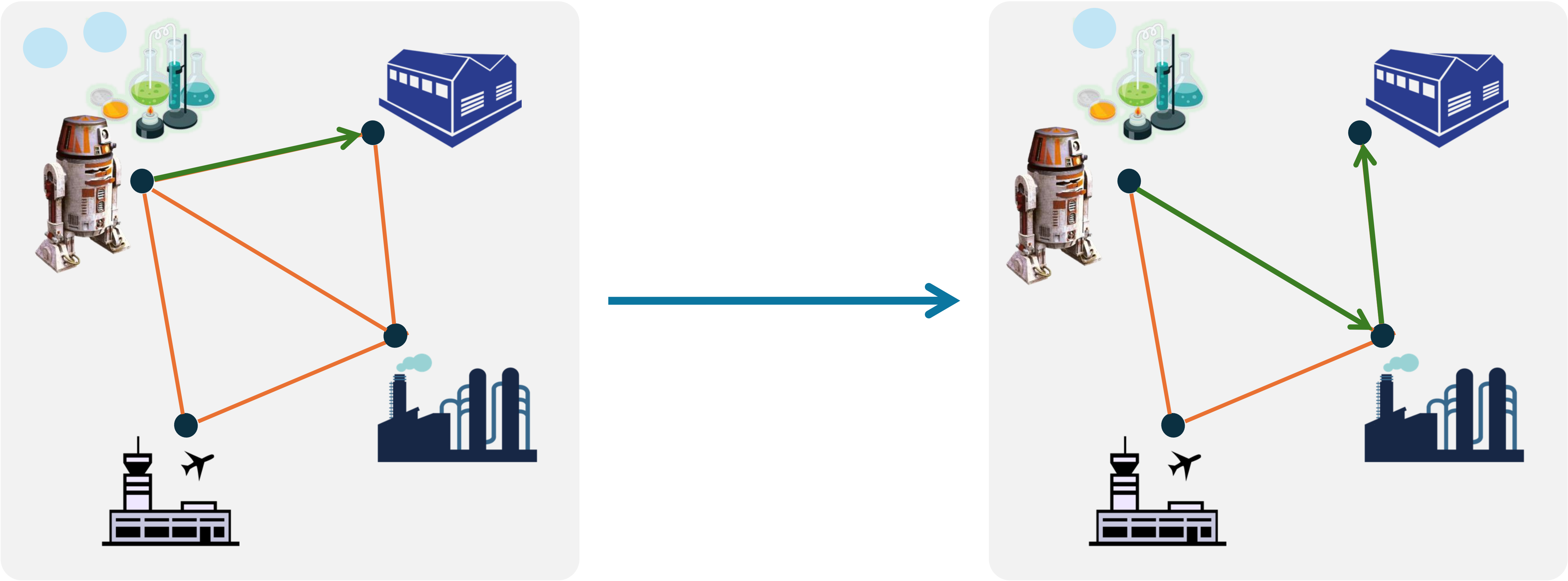

## Observation Traces

- “State -> Action -> State” tuples.




Many approaches learn models based on such observations but…

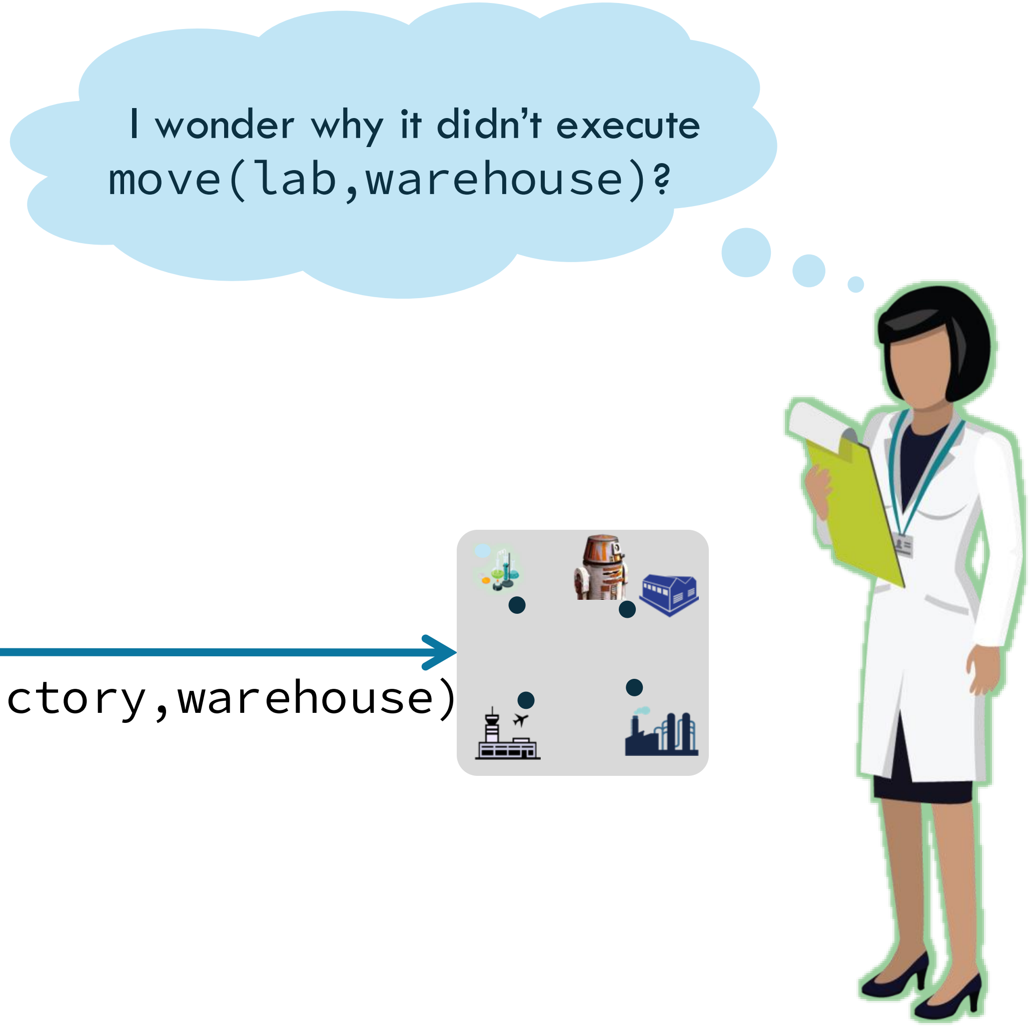
# Reduced Functionality

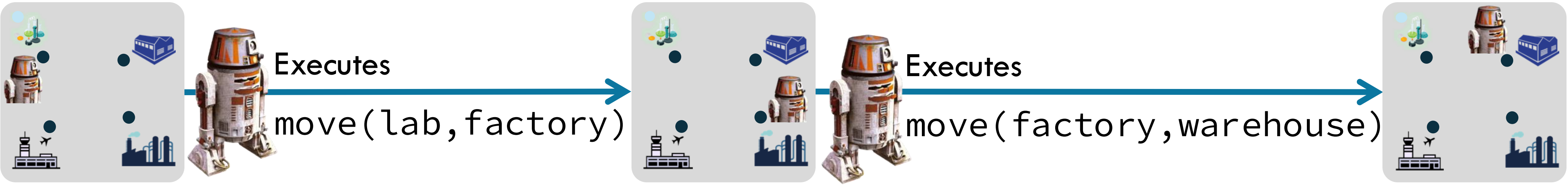


Observed that the agent executed
⟨move(lab,factory),
move(factory,warehouse)⟩

# How to identify reduced functionality?

- Not a trivial problem to solve.
- Observations not directly available.




- Can we use similar intuitions of optimality?

Agent updates

E.g., software update,

new deployment,
adapted for user needs, etc.

Agent placed in an
optimal planning mode

Sparse Observations
(collected once)

$\langle s_0, a_1, s_1 \rangle \langle s_1, a_2, s_2 \rangle \; ... \; \langle s_{n-1}, a_n, s_n \rangle$

$\langle s_0, s_k \rangle$ (length of plan = k)

planning
problem

Initial Model
known to the user
$M_{init}$

optimal plan

If length of plan < k, then subset of
actions in the plan has changed!

Solves Challenge 1

How to identify how the model
has changed given what has
changed?

# Marking the changes

- Combine knowledge of increased and reduced functionality to identify parts of model that may have changed.

```
(:action pick-samples
  :parameters (?s)
  :precondition (and
     (handempty)
     (+/-/Ø)(onshelf ?s))
  :effect (and
     (+/-/Ø)(handempty)
     (not(onshelf ?s))))
```

Only some parts of action changed

```
(:action pick-samples
  :parameters (?s)
  :precondition (and
     (+/-/Ø)(handempty)
     (+/-/Ø)(onshelf ?s))
  :effect (and
     (+/-/Ø)(handempty)
     (+/-/Ø)(onshelf ?s)))
```

Complete action changed

- How do we identify their correct form?

# Experimental Setup

- Randomly generate initially known agents using IPC benchmark suite.
- Generate observations for unknown drifted IPC agent using IPC problems.
- Using previous model and available observations, predict what may have changed.
- Learn the updated model by querying for changed portions of the model.
- Evaluate performance of the assessment module and compare it with the vanilla active querying approach of assessing model from scratch.

# Fewer Queries Needed Compared to Learning from Scratch

| Domain | #Tuples | AIA | DAAISy |
|---|---|---|---|
| Gripper | 20 | 15.0 | 6.5 |
| Miconic | 36 | 32.0 | 7.7 |
| Satellite | 50 | 34.0 | 9.0 |
| Blocksworld | 52 | 40.0 | 11.4 |
| Termes | 134 | 115.0 | 27.0 |
| Rovers | 402 | 316.0 | 61.0 |

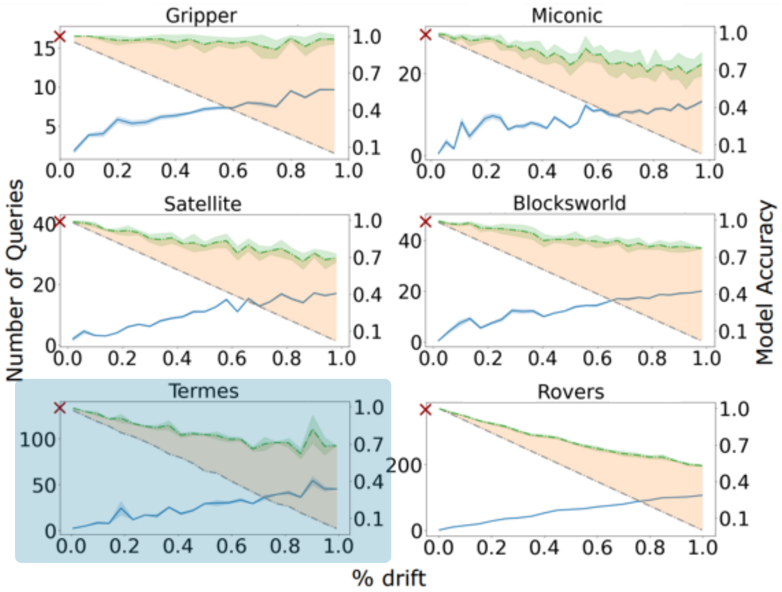
The average number of queries to achieve same level of accuracy for 50% drifted models

- Results with FD planner with LM-Cut.
- AIA takes up to 5 times more number of queries than our approach, DAAISy.

# Fewer Queries Needed Compared to Learning from Scratch

Random deterministic planning agent from IPC

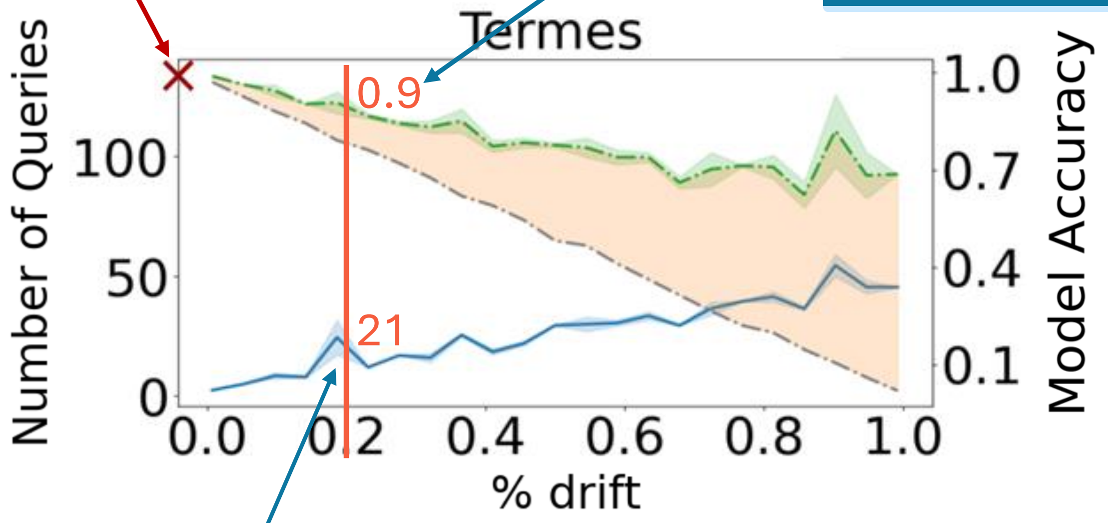
- Accuracy of initial model
- Accuracy gained by AAM
- Accuracy of model computed by AAM
- Number of queries by AAM
- Number of queries when learning from scratch




134 queries needed if starting from scratch

Accuracy

Used 10 observations per domain




Number of queries are much lower than 134

# Learned Updated Capability Descriptions are Consistent

- Theorem (*consistency*): The learned descriptions are consistent with the observations and the query responses.


# Future Directions and Conclusion

# Interpretability Analysis of Symbolic Representations for Sequential Decision-Making Systems

*Pulkit Verma and Julie A. Shah*

HRI 2025 Workshop on Explainability for Human-Robot Collaboration

# Interpretable Representations

Representations beyond PDDL.

- Temporal Logic (LTL/STL)
- Bayesian Networks
- RDDL

How interpretable each representation is?

Which representation fits the requirements of end user well?

# Interpretable Representations

| Representation | Interpretability Level | Formalism Type | Temporal Expressiveness | Abstraction Level | Explanation Type | Domain Speci-ficity | Human Interaction |
|---|---|---|---|---|---|---|---|
| Markov Decision Processes (MDPs) | Medium | Symbolic | Discrete Time | Low-Level | Global | General | Indirect |
| Finite State Machines (FSMs) | Medium | Symbolic | Discrete Time | Low-Level | Global | General | Direct |
| Decision Trees | High | Symbolic | N/A | Low-Level | Global | General | Direct |
| Rule-Based Systems | High | Symbolic | N/A | Low-Level | Global | General | Direct |
| Temporal Logic (LTL, STL, etc.) | Medium | Symbolic | Continuous Time | Low-Level | Global | Domain | Indirect |
| Program Synthesis | Low | Symbolic | N/A | High-Level | Global | Domain | Indirect |
| Planning Domain Definition Language (PDDL) | Medium | Symbolic | Discrete Time | High-Level | Global | Domain | Indirect |
| Hierarchical Task Networks (HTNs) | Medium | Symbolic | Hierarchical Time | High-Level | Global | Domain | Indirect |
| Relational Dynamic Influence Diagram Language (RDDL) | Medium | Symbolic | Discrete Time | High-Level | Global | Domain | Indirect |
| Causal Models | Medium | Hybrid | N/A | Multi-Level | Global | General | Indirect |
| Neuro-Symbolic Integration | Low | Hybrid | N/A | Multi-Level | Global | General | Indirect |

Classification of Interpretable Representations for Sequential Decision-Making Systems along different dimensions

# ∀uto∃∨∧l: Autonomous Evaluation of LLMs for Truth Maintenance and Reasoning Tasks

*Rushang Karia*, Daniel Bramblett*, Daksh Dobhal, and Siddharth Srivastava*

ICLR 2025

# Emerging Direction: Evaluation of LLM Based Agents

Can LLMs maintain factual accuracy when translating formal language?

Autoformalization: converting natural language into formal language

E.g., Code synthesis, synthesis of formal safety specifications in linear temporal logic

Informalization: converting formal language into natural language

E.g., code summarization, summarization of legal documents, interpretation of bug reports

Objectives:

1. Generating out-of-distribution datasets without human annotators
2. Accurately measure a LLM’s truth maintenance capabilities
3. Use our metric as a predicter of performance on other metrics

# Dataset Generation Using CFG Parse Trees

$$S \to S \land S$$
$$S \to (P \lor P \lor P)$$
$$P \to \neg v | v$$

$(a)$ 3–CNF

$$S \to (S \land S)|(S \lor S)$$
$$S \to (\neg S)$$
$$S \to \neg v | v$$

$(b)$ Propositional Logic

$$S \to F|(\forall f.\, S)|(\exists f.\, S)$$
$$F \to (F \land F)|(F \lor F)$$
$$F \to (\neg F)|\neg p|p$$

$(c)$ First-order Logic

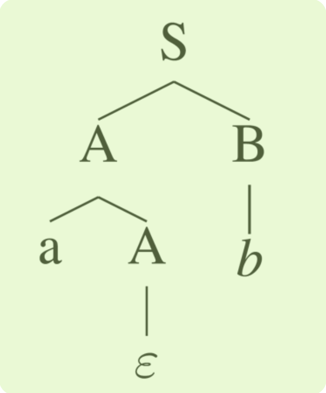
$$S \to (S)K|\Sigma K$$
$$S \to S\Sigma K$$
$$K \to *|\varepsilon$$

$(d)$ Regular Expression

CFGs used for synthesizing the datasets in ∀UTO∃V∧L

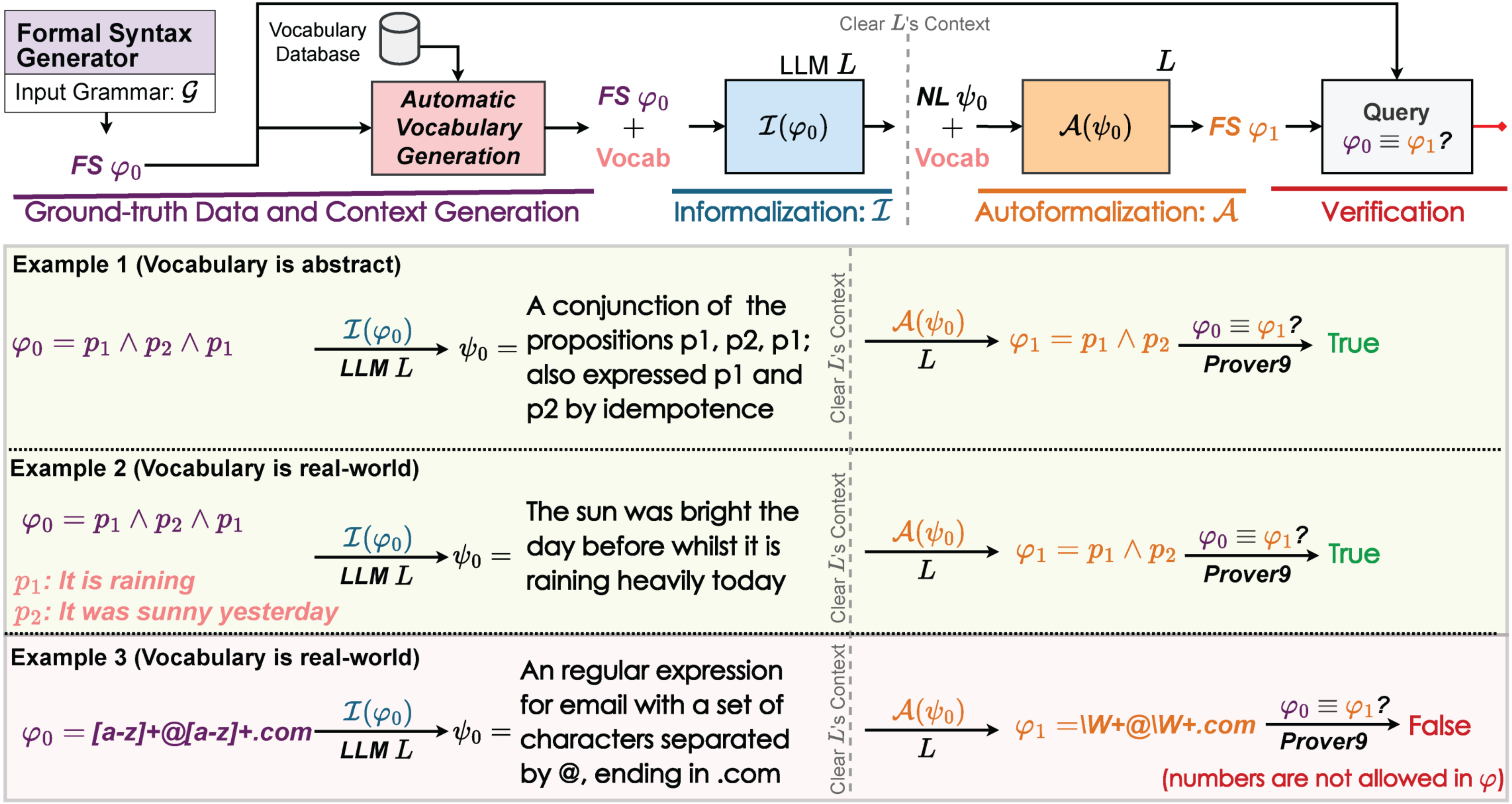
$$S \to AB$$
$$A \to aA|\varepsilon$$
$$B \to b$$

$(a)$ CFG $\mathcal{G}$ for the language $a^*b$




$(b)$ Parse tree when using $\mathcal{G}$ to obtain the string $ab$ $(d = 2)$

# AutoEval Process



# Example: Prompt for Informalization

Your task is to convert a ⟨Propositional Logic, First-order Logic⟩ formula, appearing after [FORMULA], to a natural description that represents the formula. Only natural language terms are allowed to be used and do not copy the formula in your description. Your description should allow one to reconstruct the formula without having access to it, so make sure to use the correct names in your description. Explicitly describe the predicates. You may use terms verbatim as specified in the vocabulary below.

[VOCABULARY]

| | |
|---|---|
| Operators: | List of operators followed by their NL interpretations |
| Objects: | The objects in the universe (if any) |
| Propositions: | The propositions in the universe and their NL interpretations (if any) |
| Predicates: | The predicates in the universe and their NL interpretations (if any) |
| Examples: | Few-shot examples of the task (if any) |

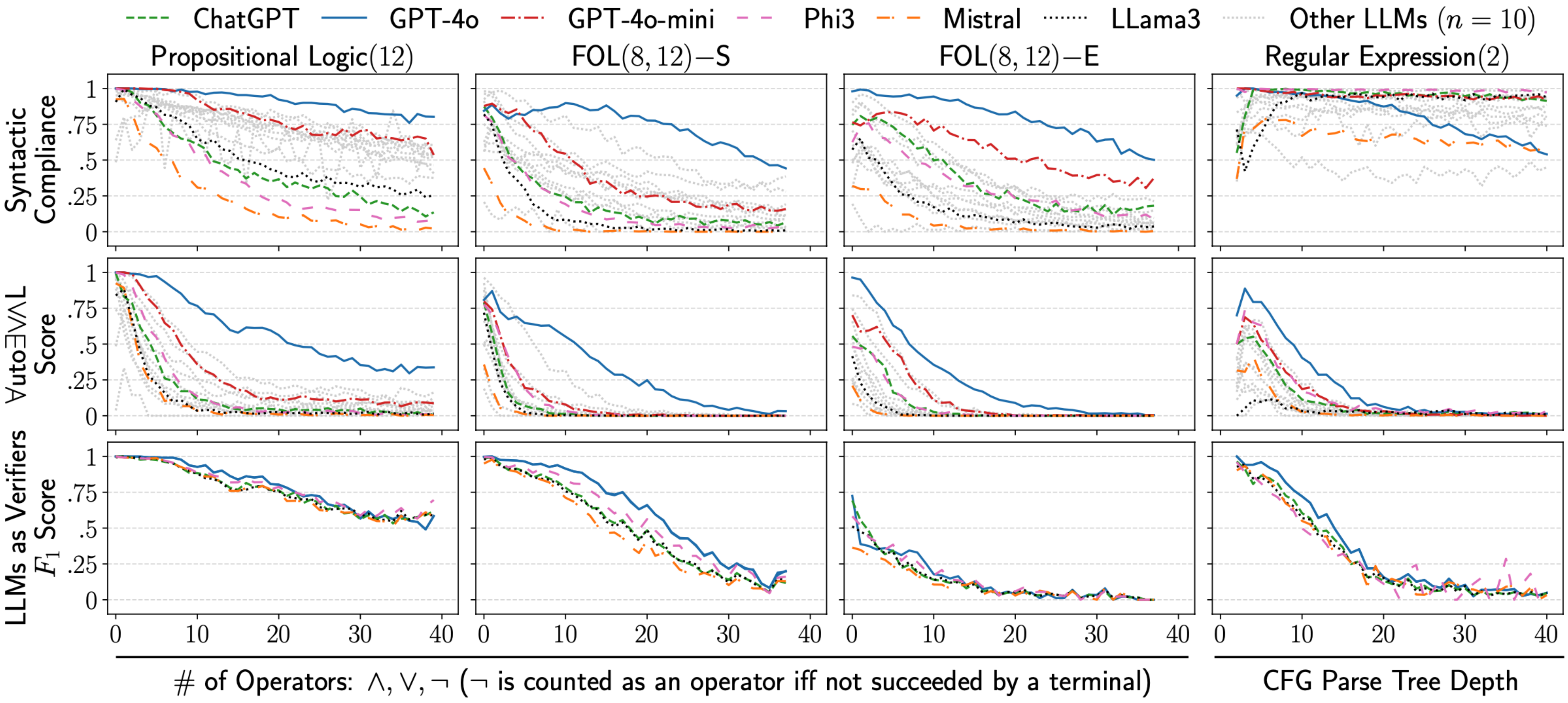
*Example Prompt*

Your task . . .

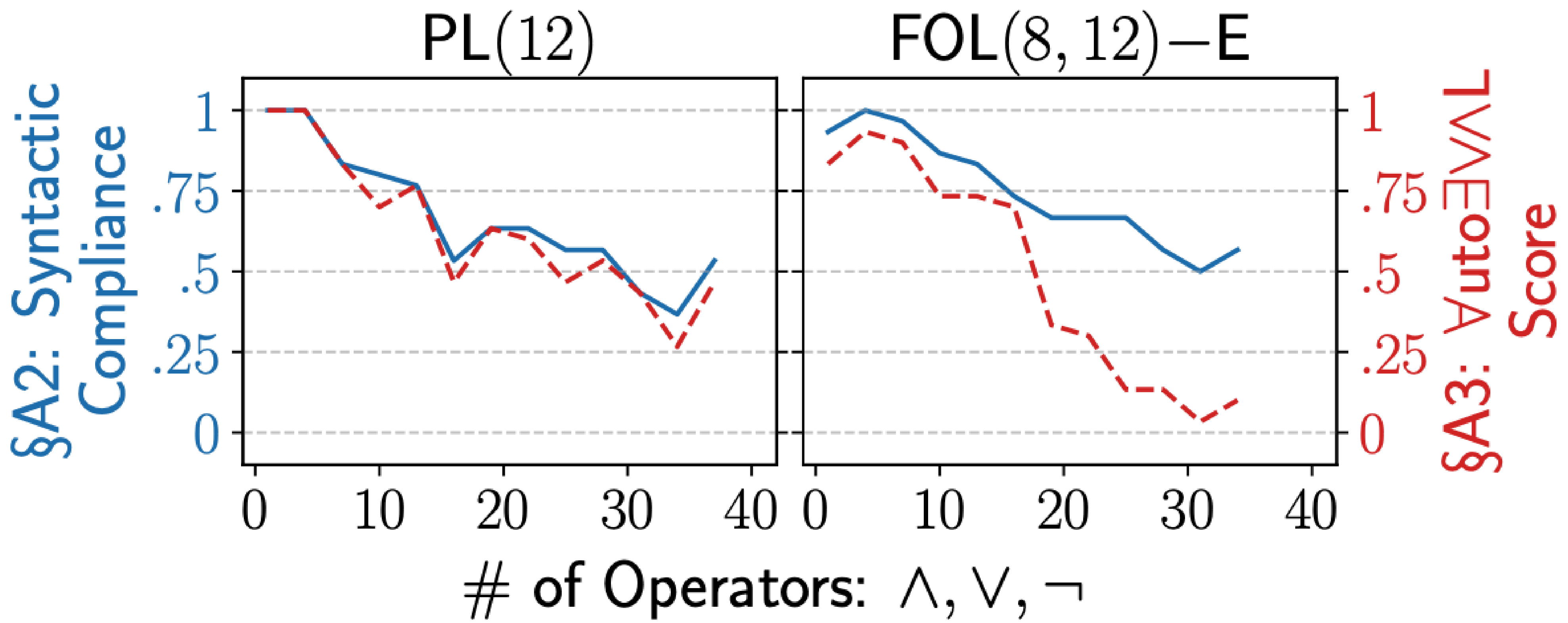
| | |
|---|---|
| Operators: | $\wedge$ represents conjunction, $\vee$ represents disjunction, . . . |
| Propositions: | $p_1$ : *It is raining*, $p_2$ : *It was sunny yesterday* |
| Formula: | $p_1 \wedge p_2 \wedge p_1$ |

*Example Response:* `The sun was bright the day before whilst it is raining today.`

# Evaluating LLMs Using AutoEval



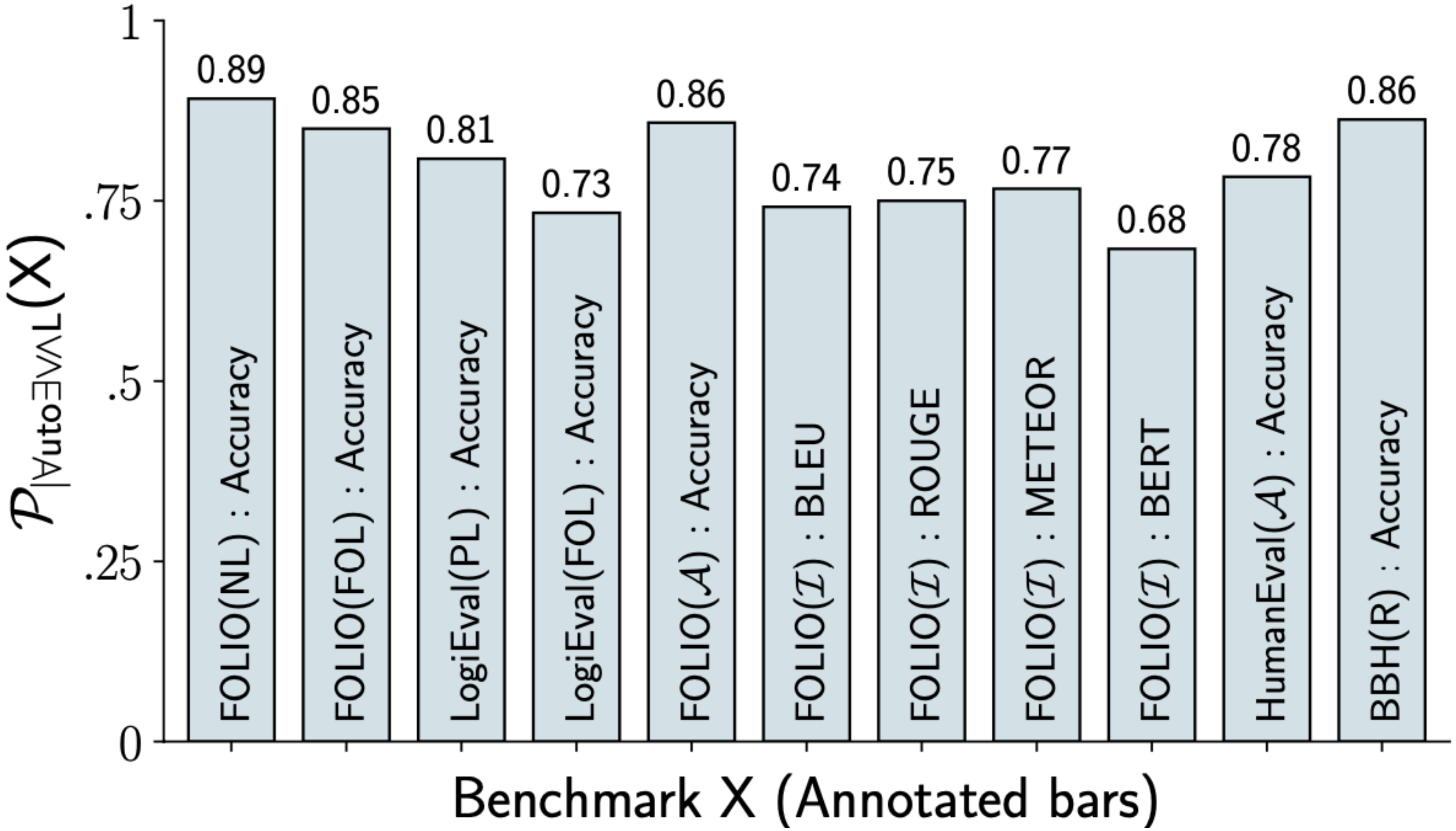
# Evaluating Large Reasoning Models: o1



# Predictive Power of AutoEval

Let $L_1$ and $L_2$ be two language models evaluated on two benchmarks $A$ and $B$ with ranks $\geq_A$ and $\geq_B$ . The **predictive power of $\boldsymbol{\geq_A}$ over $\boldsymbol{\geq_B}$** is defined as:

$$\mathcal{P}_{\geq_A}(\geq_B) = \Pr(L_1 \geq_B L_2 | L_1 \geq_A L_2)$$



# What Remains to Be Done

# Revisiting AI Assessment

User's Intent

User's Specification

Agent's Goal/Control Objective/Cost/Constraints

Agent's Behavior Synthesis

- Constraints (unknown to user)
- Adaptive code (unknown to designer)
- Mostly suboptimal

Only for stationary systems: known at design-stage

User-Driven AI Assessment

Executable Program/Controller

# Revisiting AI Assessment

Reward Hacking

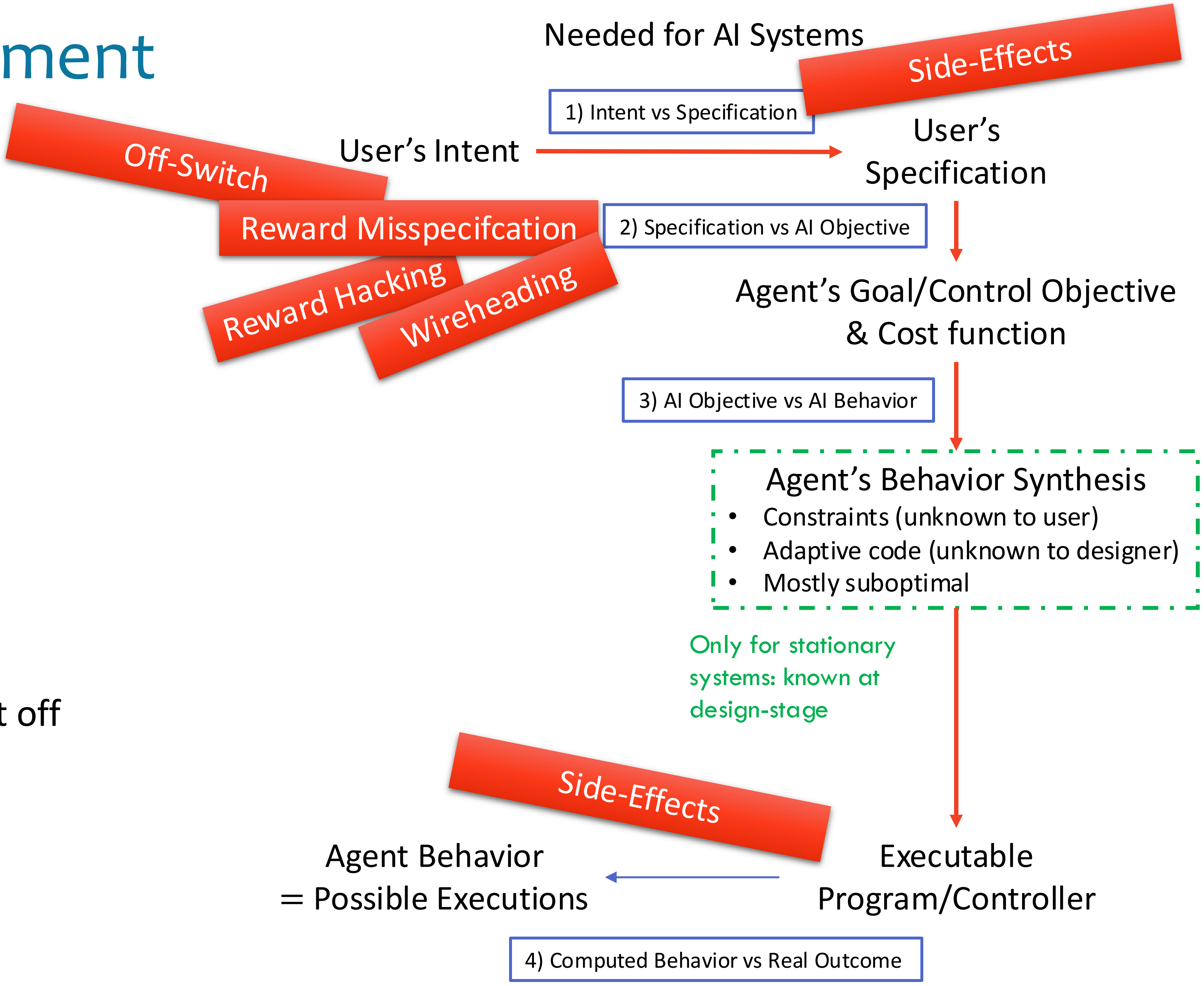
Wireheading

Reward Misspecification

Side Effects

Off-Switch

Agent doesn’t let the user turn it off



# Revisiting AI Assessment

Reward Hacking

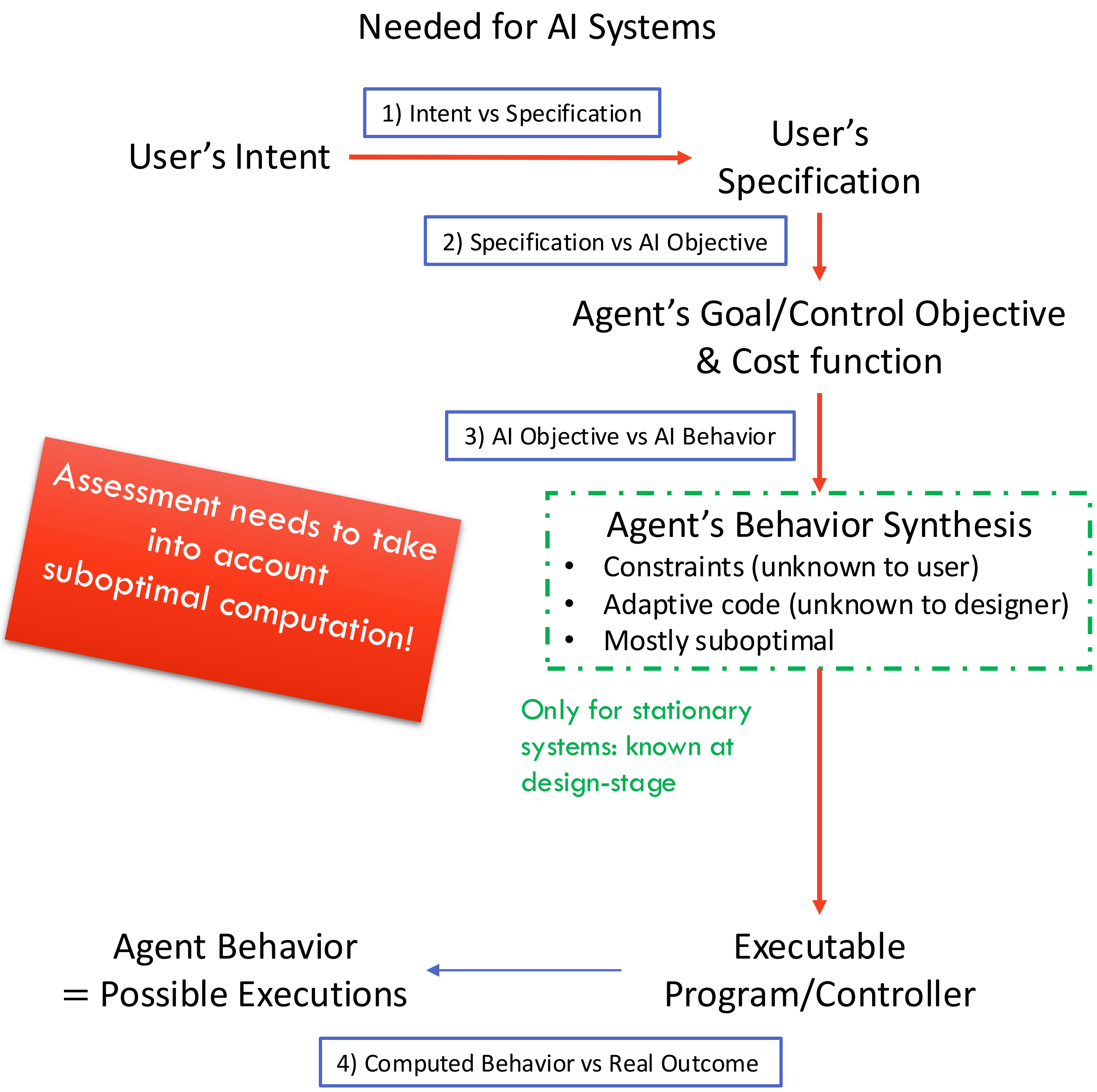
Wireheading

Reward Misspecification

Side Effects

Off-Switch

Several gaps in ongoing research



The 39th Annual AAAI Conference on Artificial Intelligence

FEBRUARY 25 – MARCH 4, 2025 | PHILADELPHIA, PENNSYLVANIA, USA

# User-Driven Capability Assessment of Taskable AI Systems

bit.ly/aia25-tutorial

Pulkit Verma

MIT CSAIL

Siddharth Srivastava

AAIR Autonomous Agents and Intelligent Robots

ASU Arizona State University