July 16, 2026 | CeRAI Faculty Seminar Series Talk

Beyond the Lab: User-Aligned Assessment of Taskable AI Systems

Pulkit Verma

pulkitv@cse.iitm.ac.in

pulkitverma.net

01

Introduction

02

Foundational Approach

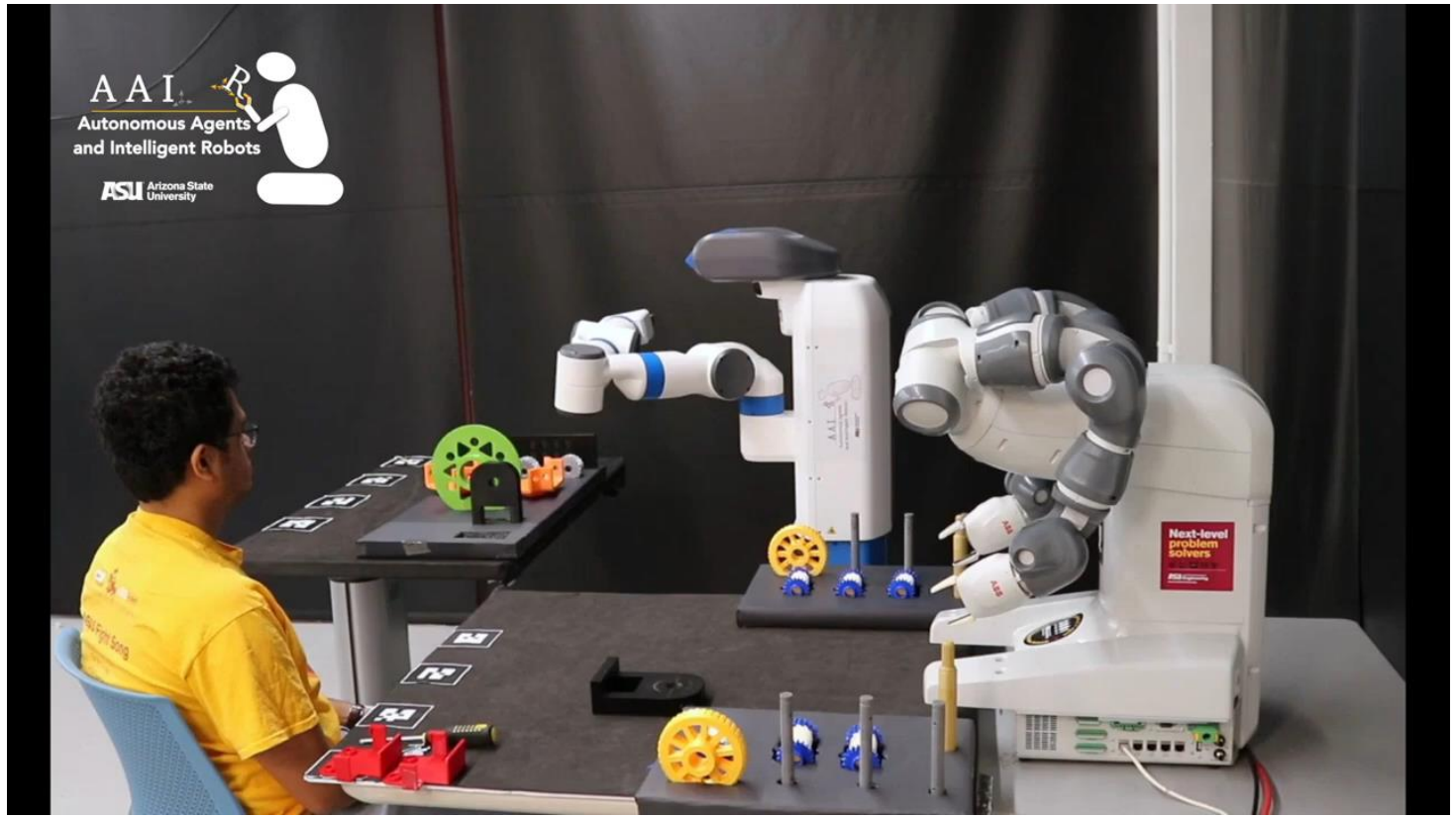
03

Generalizations

04

Conclusion and Future Work

Taskable AI Systems: Systems that can Learn and Plan



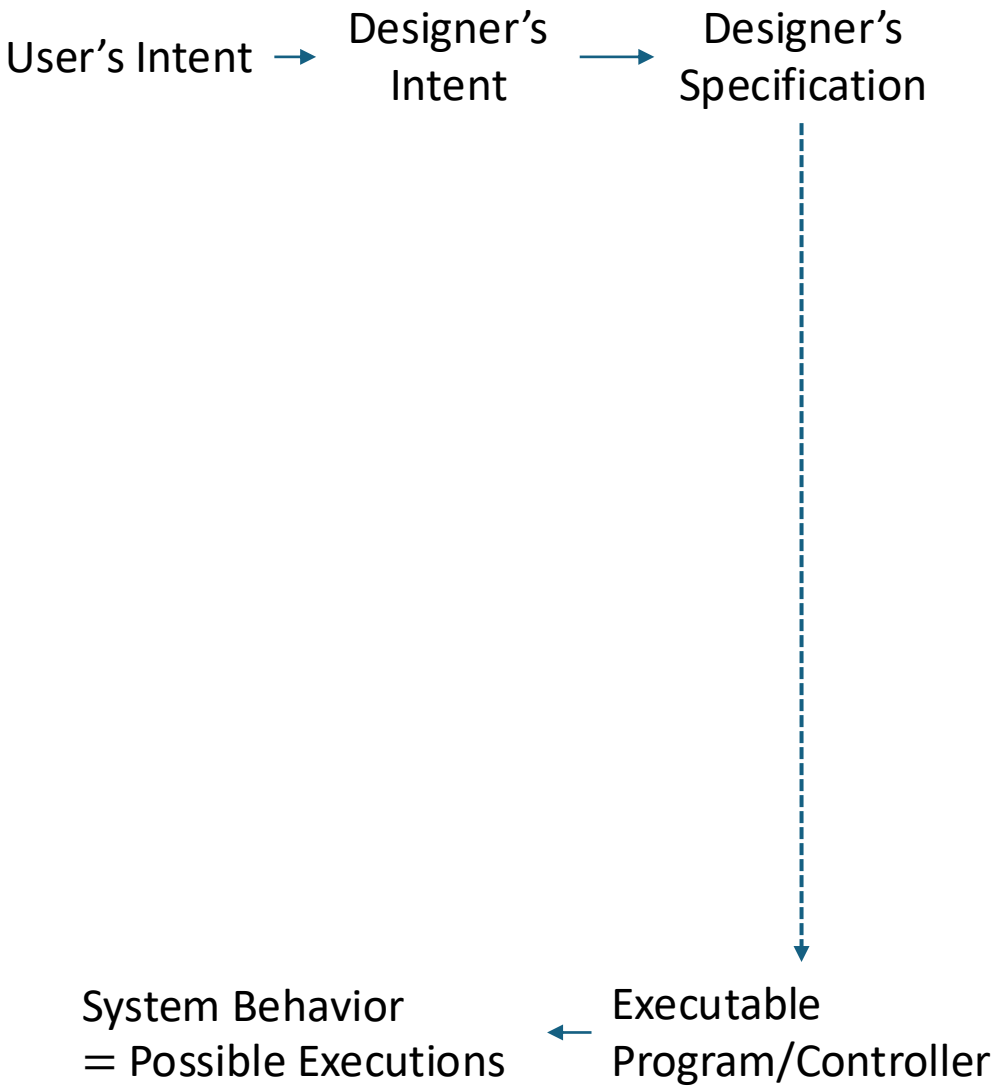
User gives a task and Robots have to complete it

- Sequential Decision-Making Systems.
- Systems designed to be able to help user.
- User has a task in mind and expects the AI system to help in that task.

Personalized Assessment of AI Systems that can Learn and Plan

- Users would like to give AI systems multiple tasks.
  - How would users know what the AI systems can do?

Classical Notion of Verification

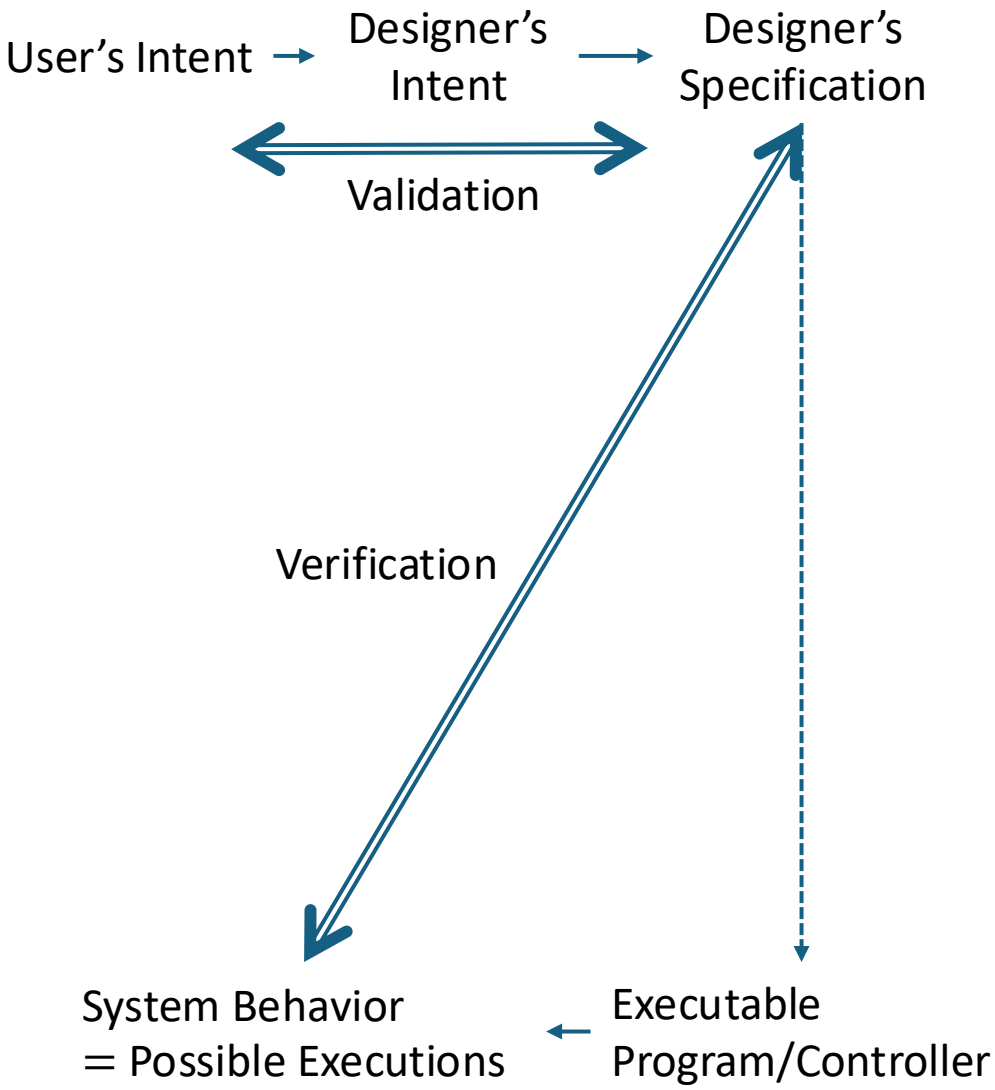


[Srivastava, Fainekos, **Verma**, Bramblett; AIA @ IJCAI '25]

Classical Notion of Verification

Input for Verification: Executable Program/Controller
(includes task spec)
+ Model/assumptions on env
+ Safety property

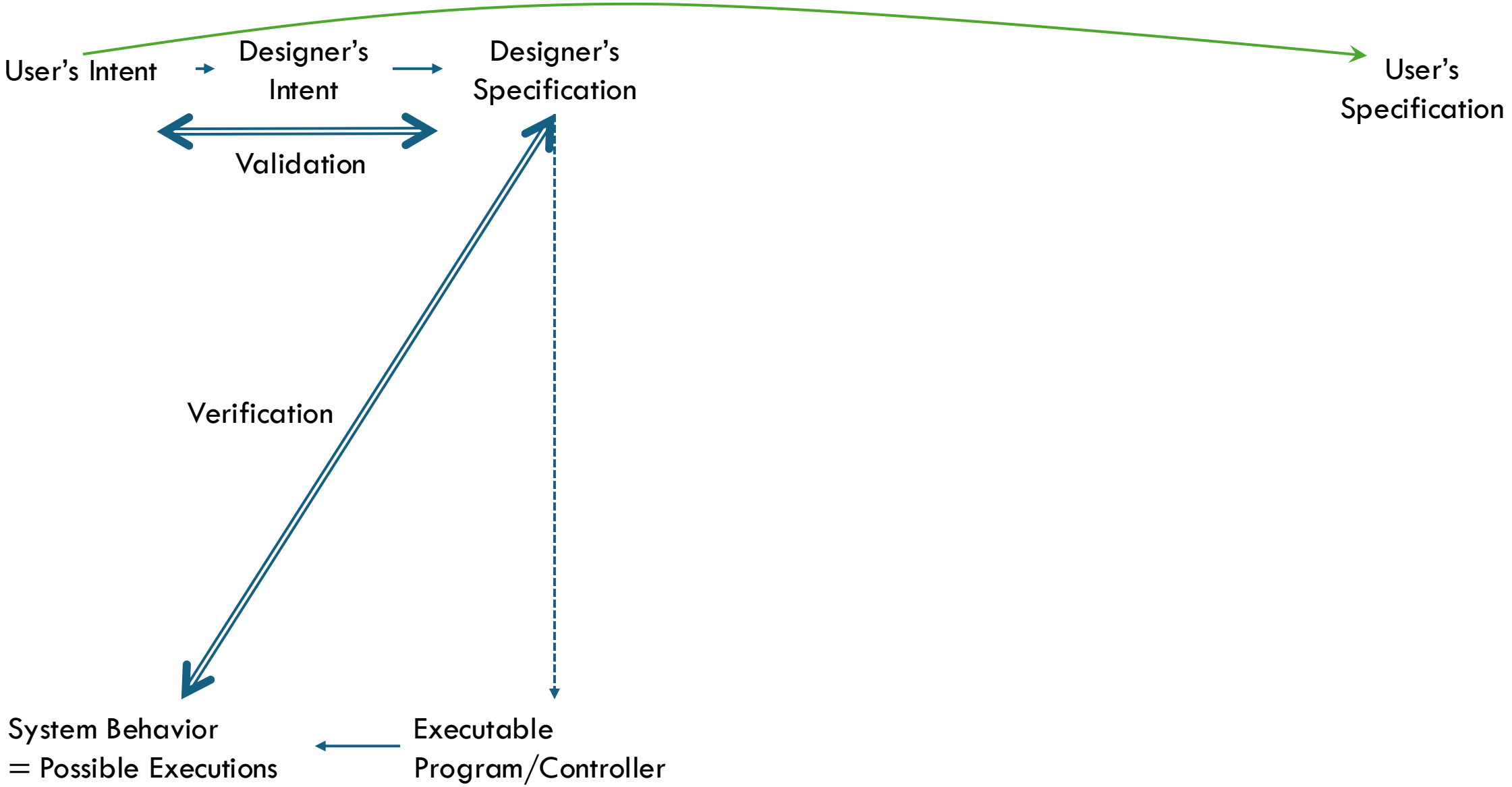
The designer plays a central role



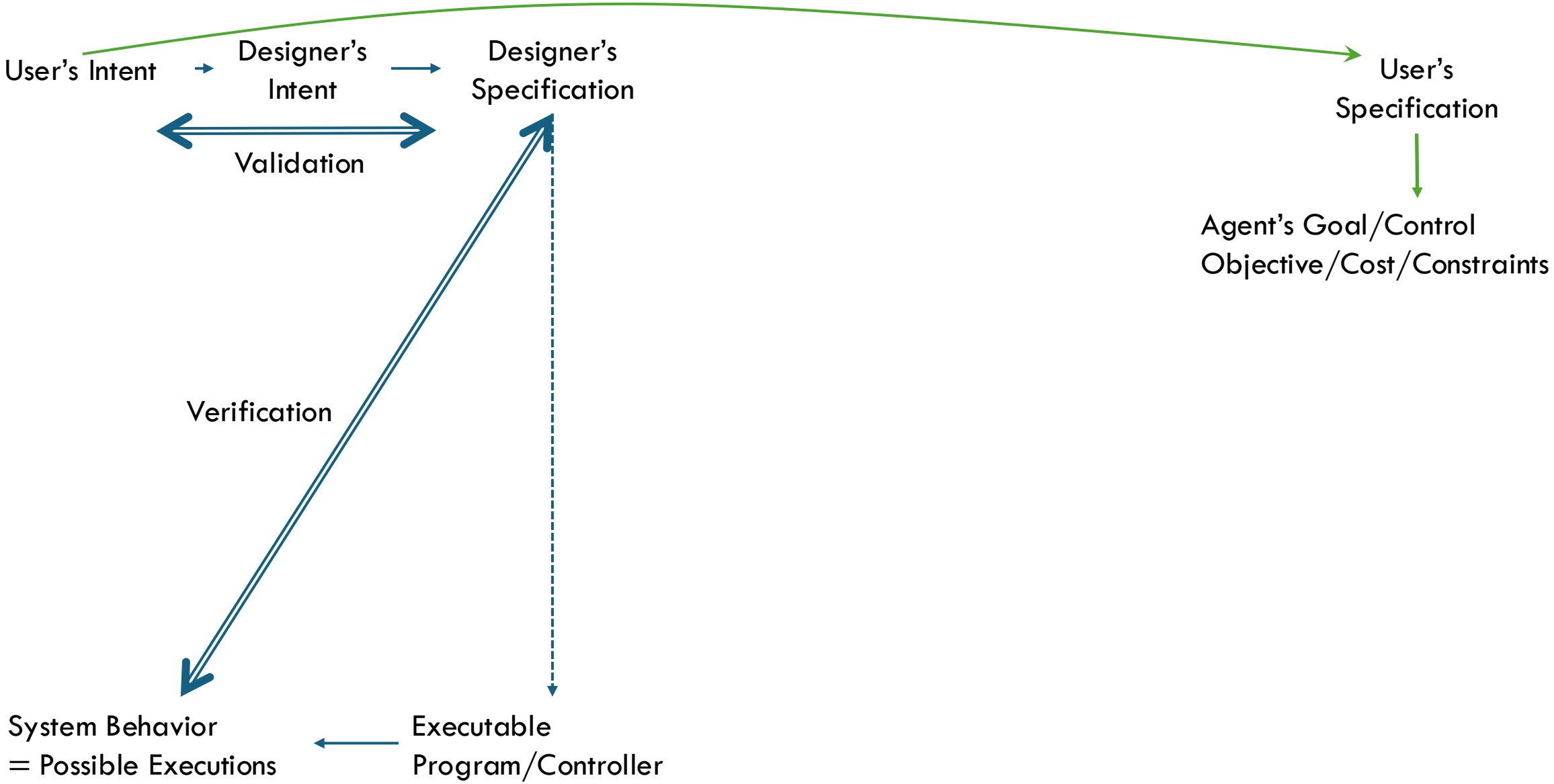
[Srivastava, Fainekos, **Verma**, Bramblett; AIA @ IJCAI '25]

Personalized Assessment of AI Systems that can Learn and Plan

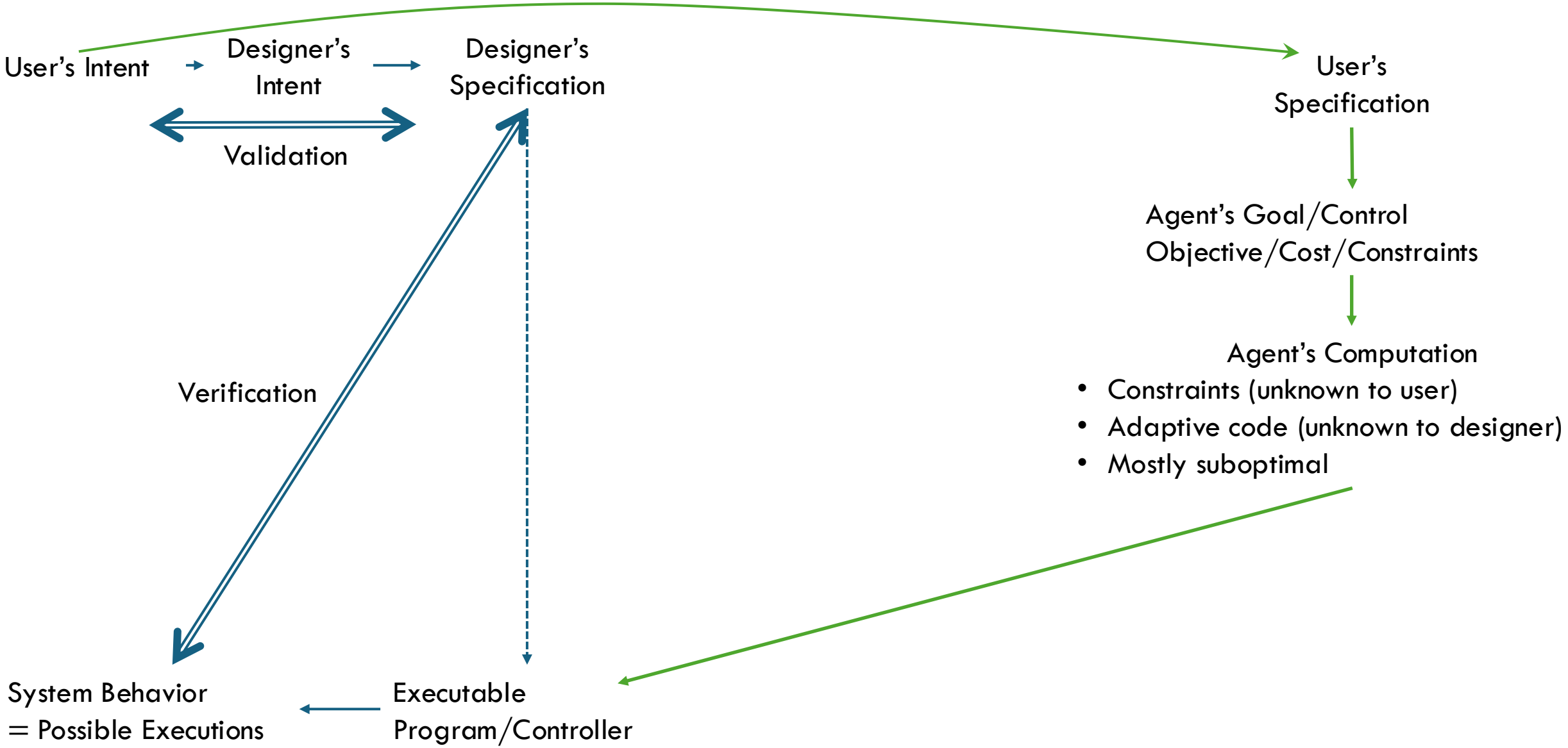
- Users would like to give AI systems multiple tasks.
  - How would users know what the AI systems can do?
- AI systems should support third-party assessment.
- The assessment should work with:
  - Adaptive AI systems.
  - Black-Box AI systems.



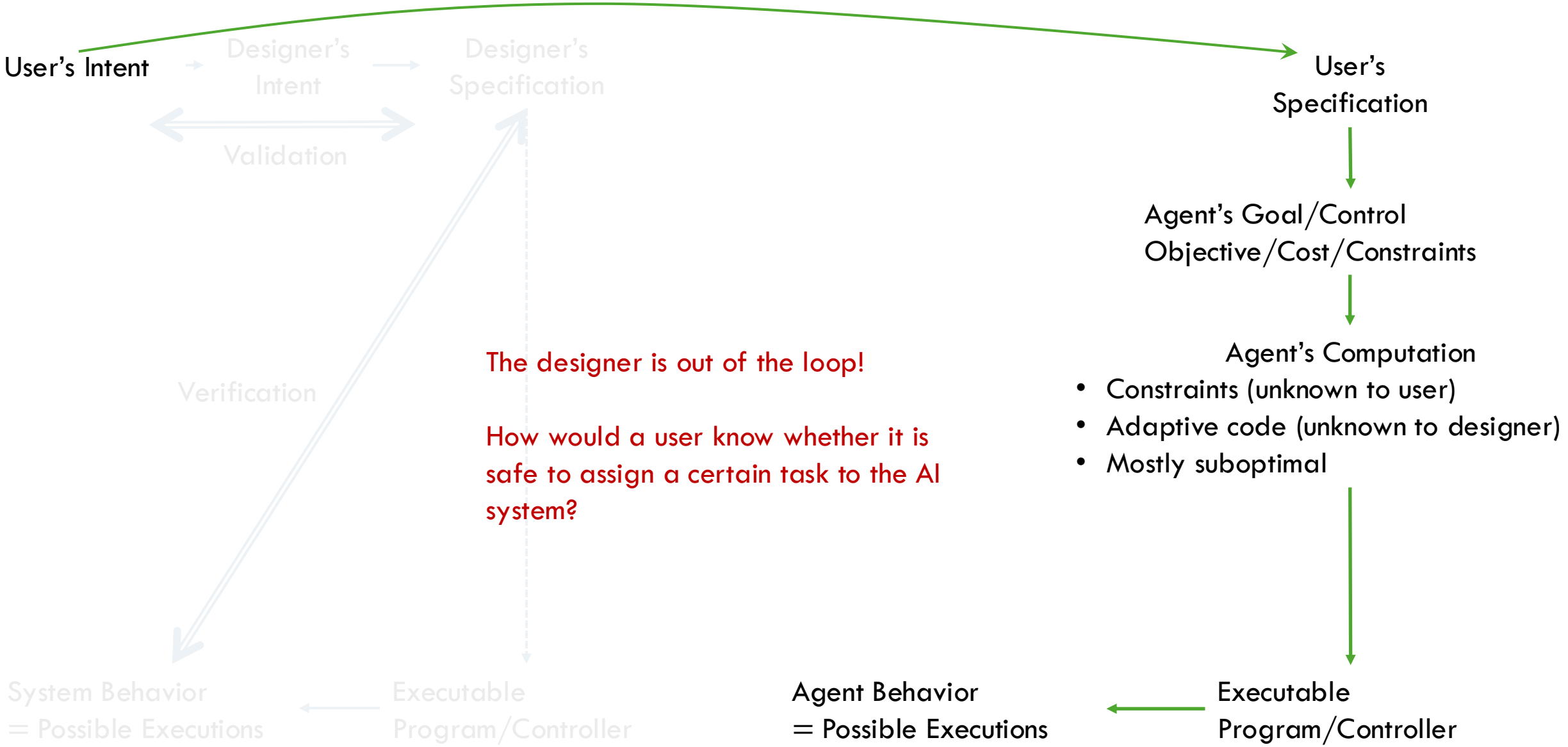
[Srivastava, Fainekos, **Verma**, Bramblett; AIA @ IJCAI '25]



[Srivastava, Fainekos, **Verma**, Bramblett; AIA @ IJCAI '25]



[Srivastava, Fainekos, **Verma**, Bramblett; AIA @ IJCAI '25]

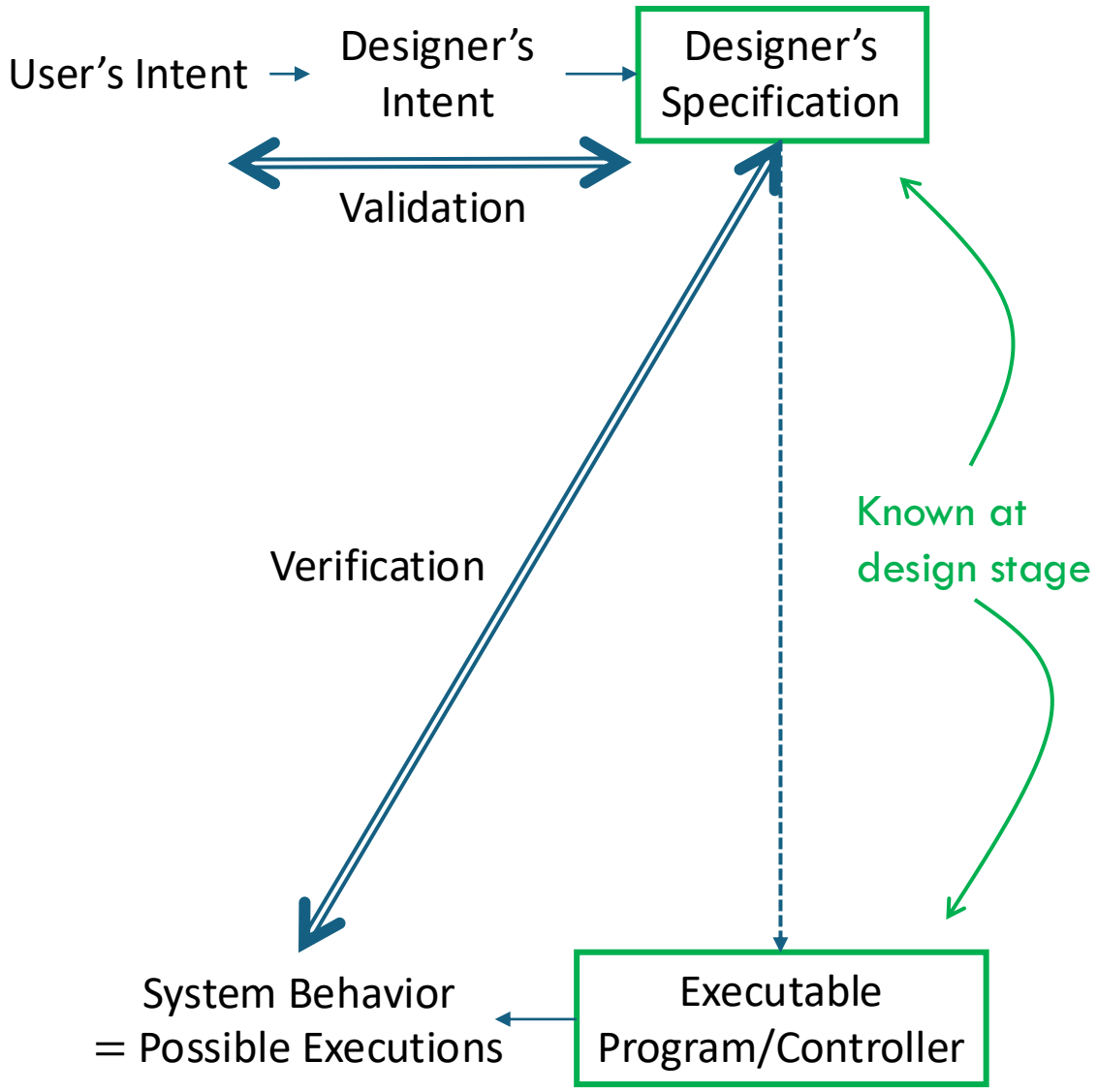


The designer is out of the loop!

How would a user know whether it is safe to assign a certain task to the AI system?

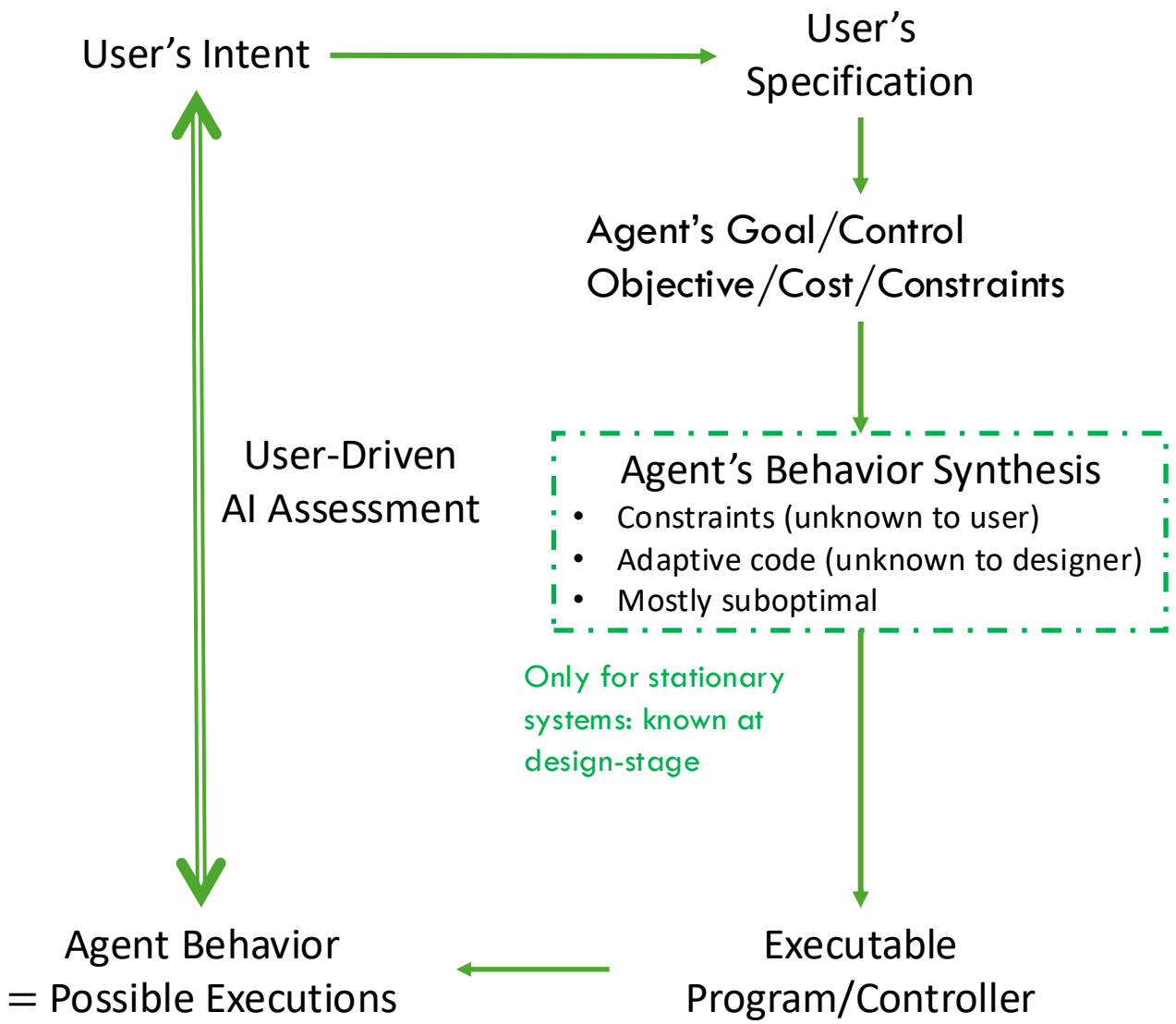
[Srivastava, Fainekos, **Verma**, Bramblett; AIA @ IJCAI '25]

Conventional Verification

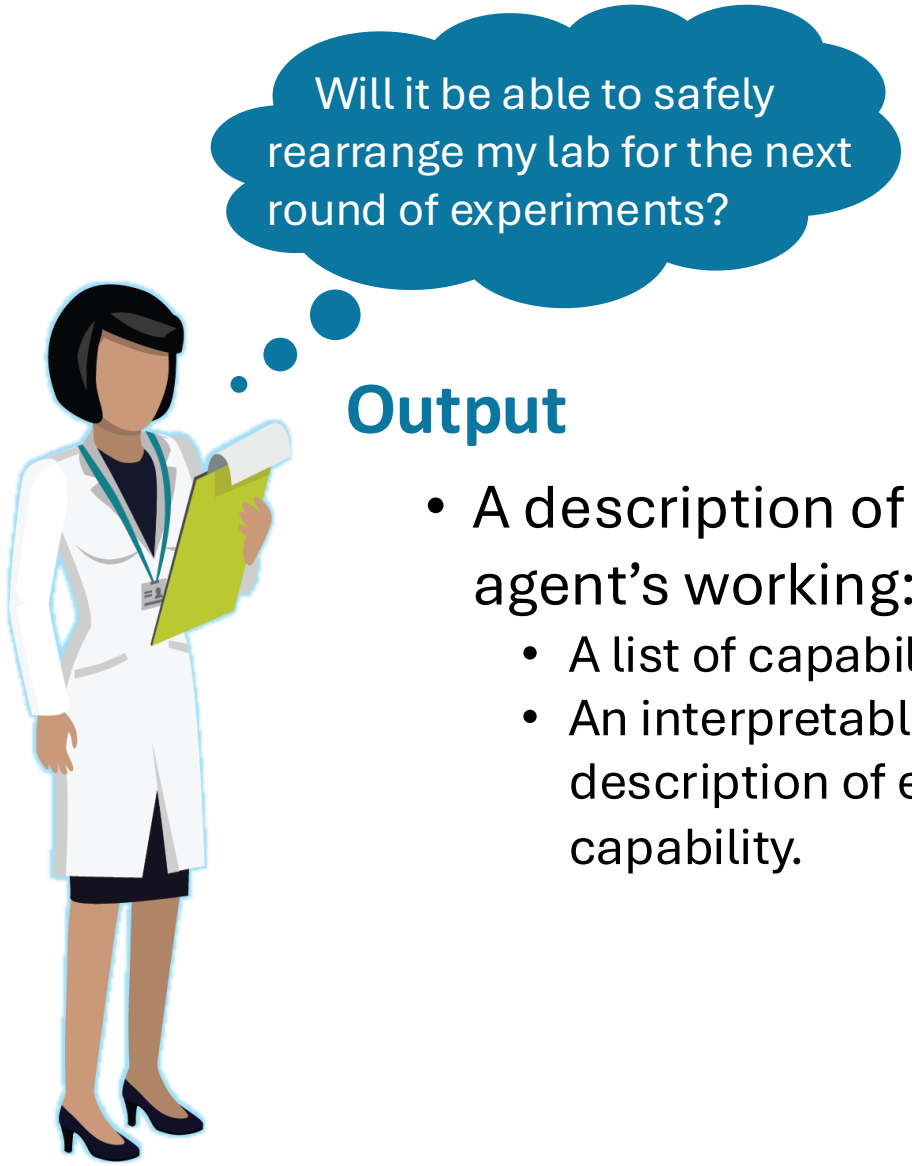


[Srivastava, Fainekos, **Verma**, Bramblett; AIA @ IJCAI '25]

Needed for AI Systems

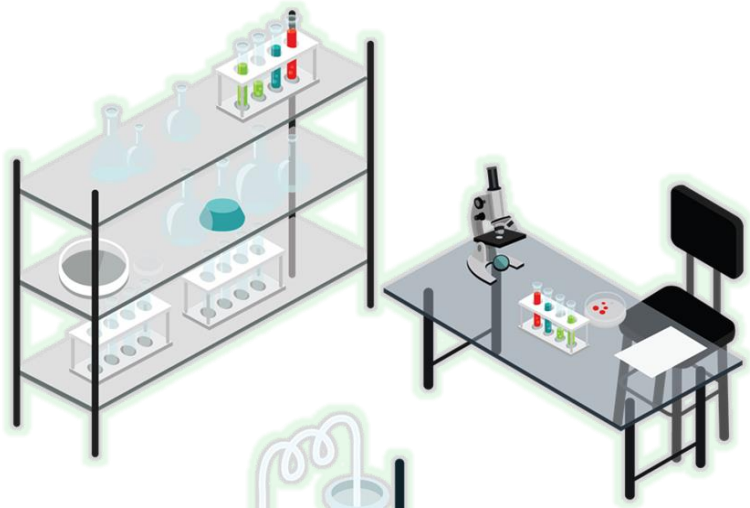


The Assessment Problem



Output

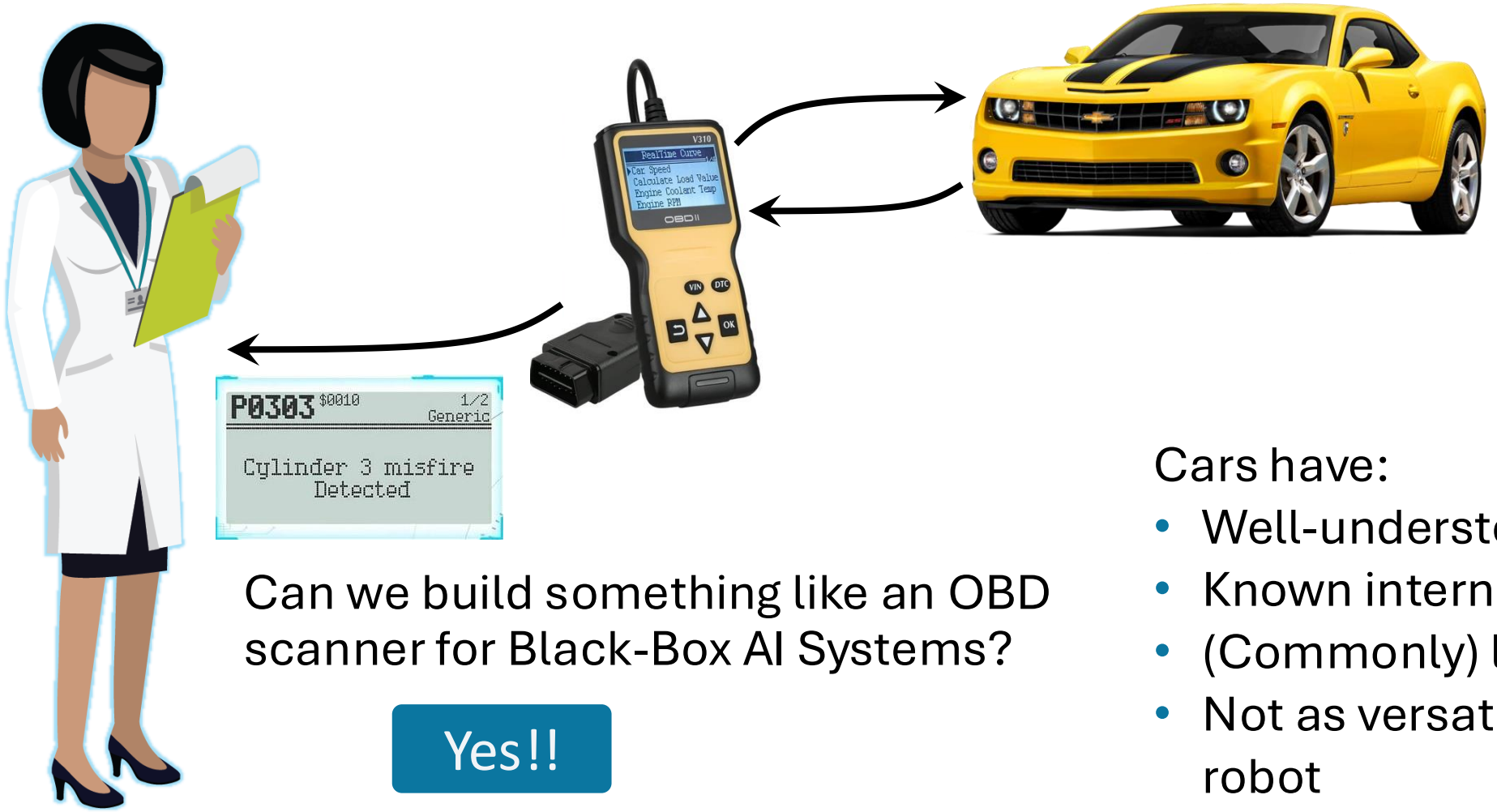
- A description of agent's working:
  - A list of capabilities.
  - An interpretable description of each capability.



Black-Box AI

- Arbitrary internal implementation
- Comes with a Trained Policy

OBD Scanner for Black-Box AI?



Can we build something like an OBD scanner for Black-Box AI Systems?

Yes!!

But we'll need something more general and powerful.

Cars have:

- Well-understood components
- Known internal design
- (Commonly) limited functionality
- Not as versatile as a household robot

```
at(p0,cell_6_3)
clear(cell_0_0)
door_at(cell_9_2)
next_to(m0)
alive(m0)
key_at(9_4)
```

[Input]
Concepts that the user understands

Personalized AI-Assessment Module (AAM)

Black-Box AI

Arbitrary internal implementation

Doesn't know user's vocabulary

```
at(p0,cell_6_3)
clear(cell_0_0)
door_at(cell_9_2)
next_to(m0)
alive(m0)
key_at(9_4)
```

[Input]
Concepts that the user understands

Personalized AI-Assessment Module (AAM)

Query

Response

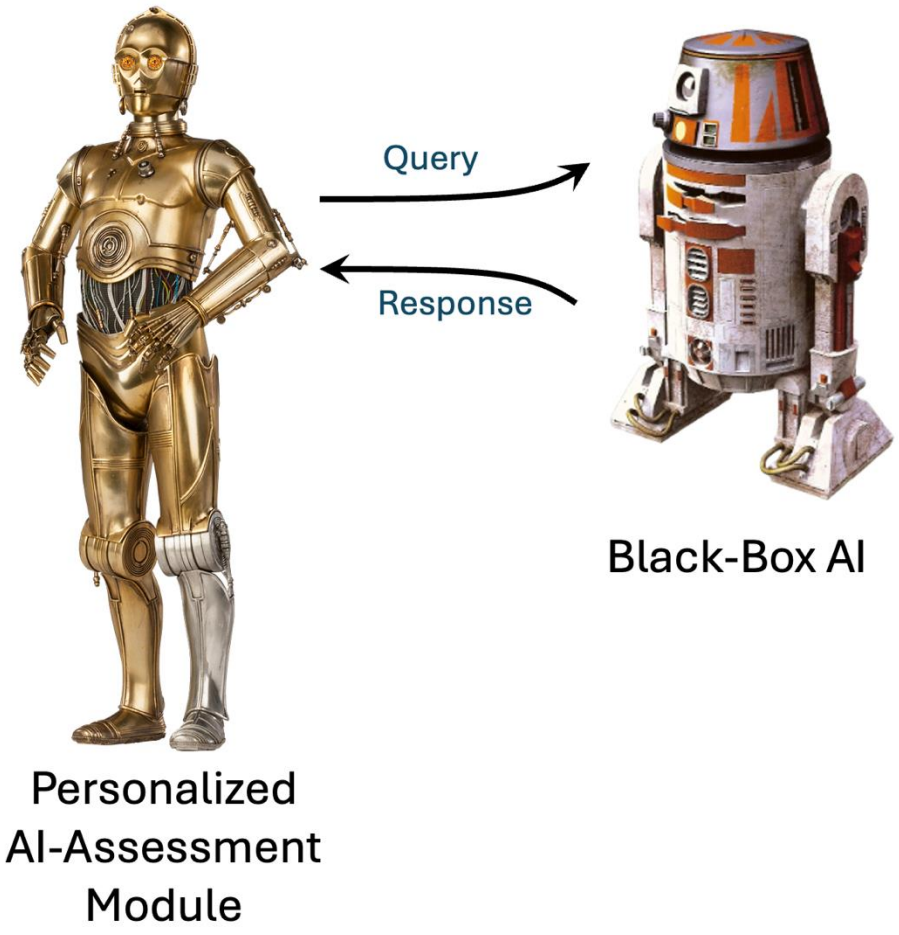
simulator

Black-Box AI

Arbitrary internal implementation

Doesn't know user's vocabulary

Black-Box AI System Interface



- Simple Query-Response Interface
- Should work for a variety of taskable AI systems.
  - Independent of their internals.
- Requirement: ⟨QueryType, ResponseType, ρ⟩.

[Input]
Concepts that the user understands

[Output]
User-Interpretable description of Black-Box AI's capabilities

Personalized AI-Assessment Module (AAM)

This Work

Query

Response

simulator

Black-Box AI

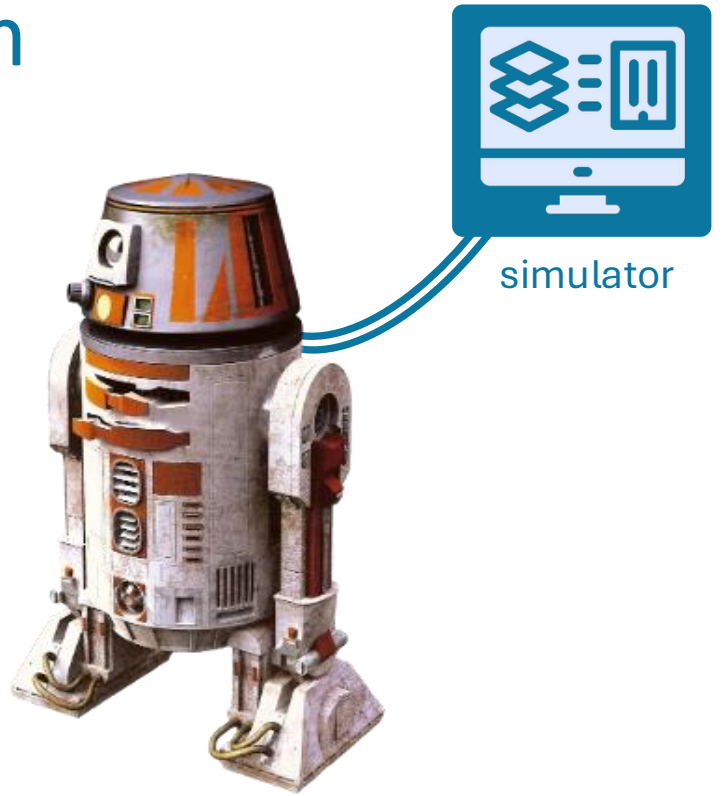
Arbitrary internal implementation

Doesn't know user's vocabulary

The Assessment Problem

Black-Box (Taskable) AI

- Can be connected to a simulator.
- Can have arbitrary internal implementation.
- Does not know input vocabulary.



Black-Box AI

Input

- Predicates (User vocabulary)
  - With their evaluation functions

Output

- A description of agent's working:
  - A list of capabilities.
  - An interpretable description of each capability.

Interpretable Description: PDDL/PPDDL

```
(:action open-door
   :parameters (?l1)
   :precondition (and
      (has_key)
      (player_at ?l1)
      (door_adjacent ?l1))
   :effect (probabilistic
      0.95 (and (door_open))
      0.05 (and (not (has_key))
                (game-over))
)
```

Precondition: This condition must be true for this action to execute

Effect: This is a set of conditions, one of which becomes true when this action is executed

Probabilities: Each set of effect has an associated probability with which that effect set is executed

Interpretable: Easily Convertible to Natural Language

```
(:action open-door
  :parameters (?l1)
  :precondition (and
     (has_key)
     (player_at ?l1)
     (door_adjacent ?l1))
  :effect (probabilistic
     0.95 (and (door_open))
     0.05 (and (not(has_key))
               (game-over))
)
```

The player can open the door when in location ?l1 if:

- It has the key
- The player is at location ?l1
- The door is adjacent to location ?l1

After executing that capability:

- With 95% probability, the door will open
- With 5% probability, the player will not have the key and the game will be over

01

Introduction

02

Foundational Approach

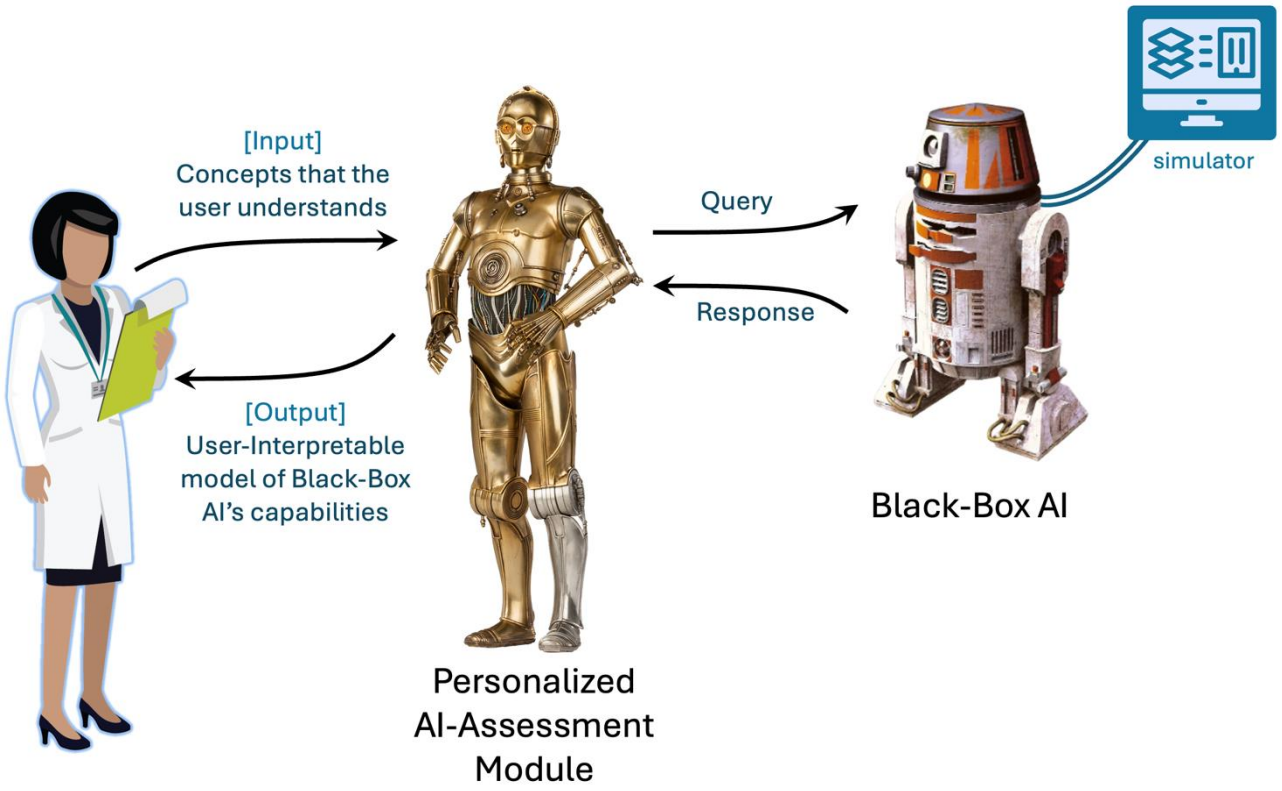
03

Generalizations

04

Conclusion and Future Work

Deterministic and Stationary Setting



Assumptions

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

Exponential Search for Learning Correct Description

- Consider the following 4 predicates/concepts:
  - `(has_key)`
  - `(door_open)`
  - `(door_adjacent ?x)`
  - `(player_at ?x)`
- Consider just one capability: `(open-door ?x)`
- $9^{|C|\times|P|}$ = $9^{1\times4}$=6561 possible models (Assuming deterministic models/ descriptions, i.e., no probabilities).

```
(:action open-door
  :parameters (?l1)
  :precondition (and
    (+/-/∅)(has_key)
    (+/-/∅)(door_open)
    (+/-/∅)(door_adjacent ?l1)
    (+/-/∅)(player_at ?l1))
  :effect (and
    (+/-/∅)(has_key)
    (+/-/∅)(door_open)
    (+/-/∅)(door_adjacent ?l1)
    (+/-/∅)(player_at ?l1))
```

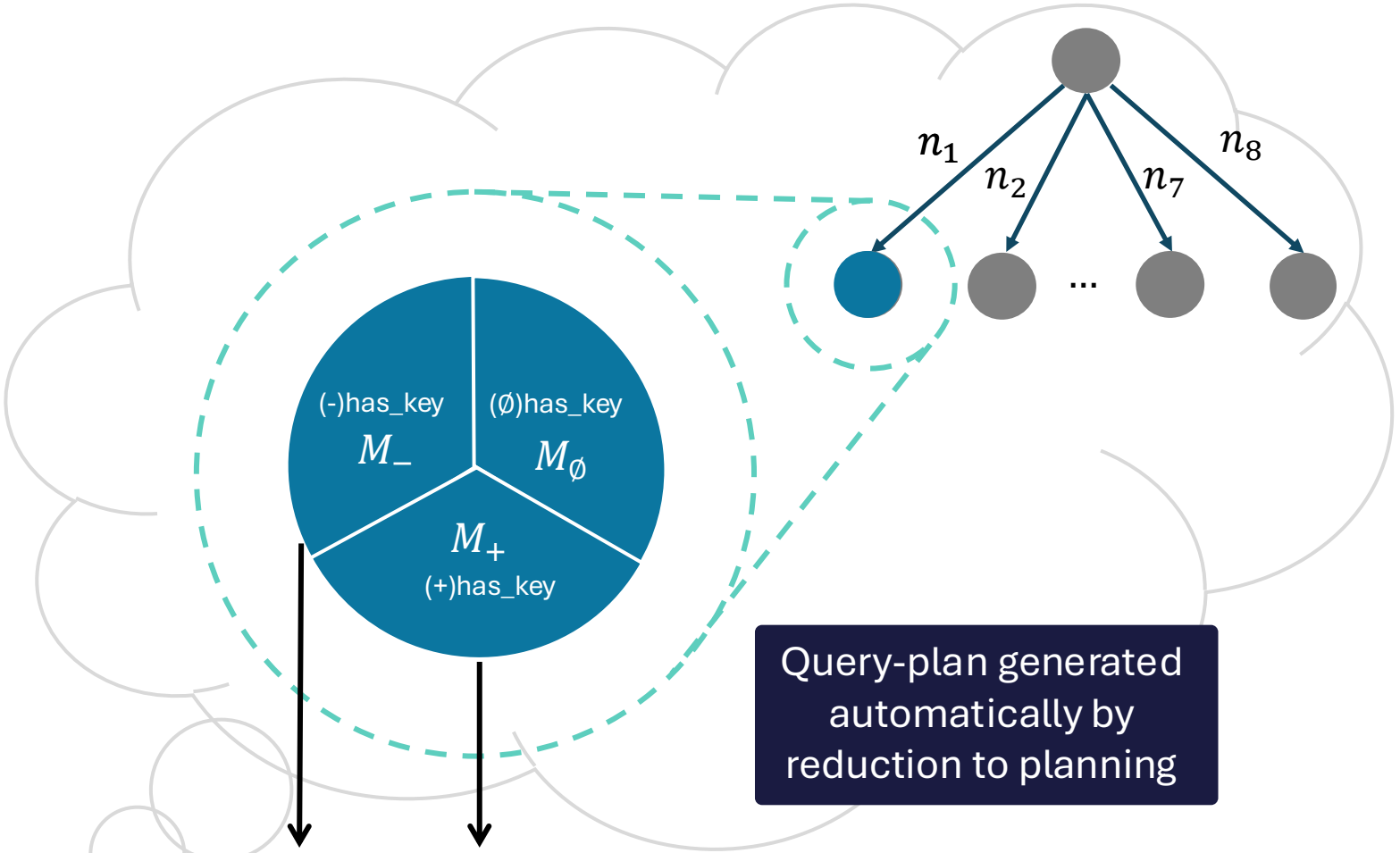
Simple Queries

	Plan Outcome Queries	State Reachability Query
Query	In state s_I, what will happen if you execute the plan $\pi = \langle c_1, \ldots, c_n \rangle$?	Can you go from state s_I to state s_F?
Response	I can execute first ℓ steps of the plan, ending up in state s_F.	Yes / No.

- How to generate the queries?
- How to use the responses to generate models?

We have a reduction that converts this to a planning problem, so it automatically generates queries.

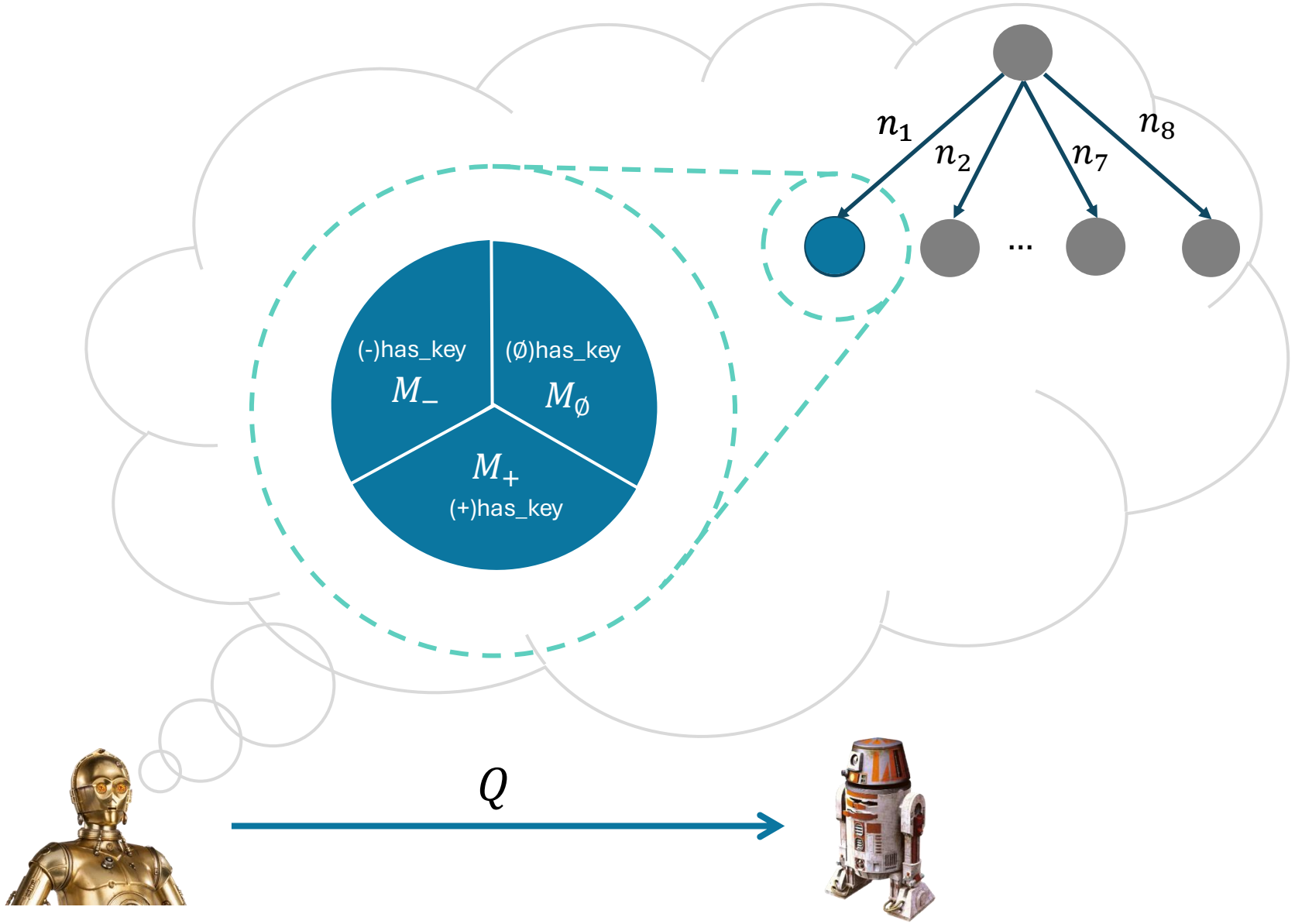
Hierarchical Query Synthesis



```
(:action open-door
  :parameters (?l1)
  :precondition (and
n1 (+/-/∅)(has_key)
n2 (+/-/∅)(door_open)
n3 (+/-/∅)(door_adjacent ?l1)
n4 (+/-/∅)(player_at ?l1))
  :effect (and
n5 (+/-/∅)(has_key)
n6 (+/-/∅)(door_open)
n7 (+/-/∅)(door_adjacent ?l1)
n8 (+/-/∅)(player_at ?l1))
```

[**Verma**, Marpally, Srivastava; AAAI '21]

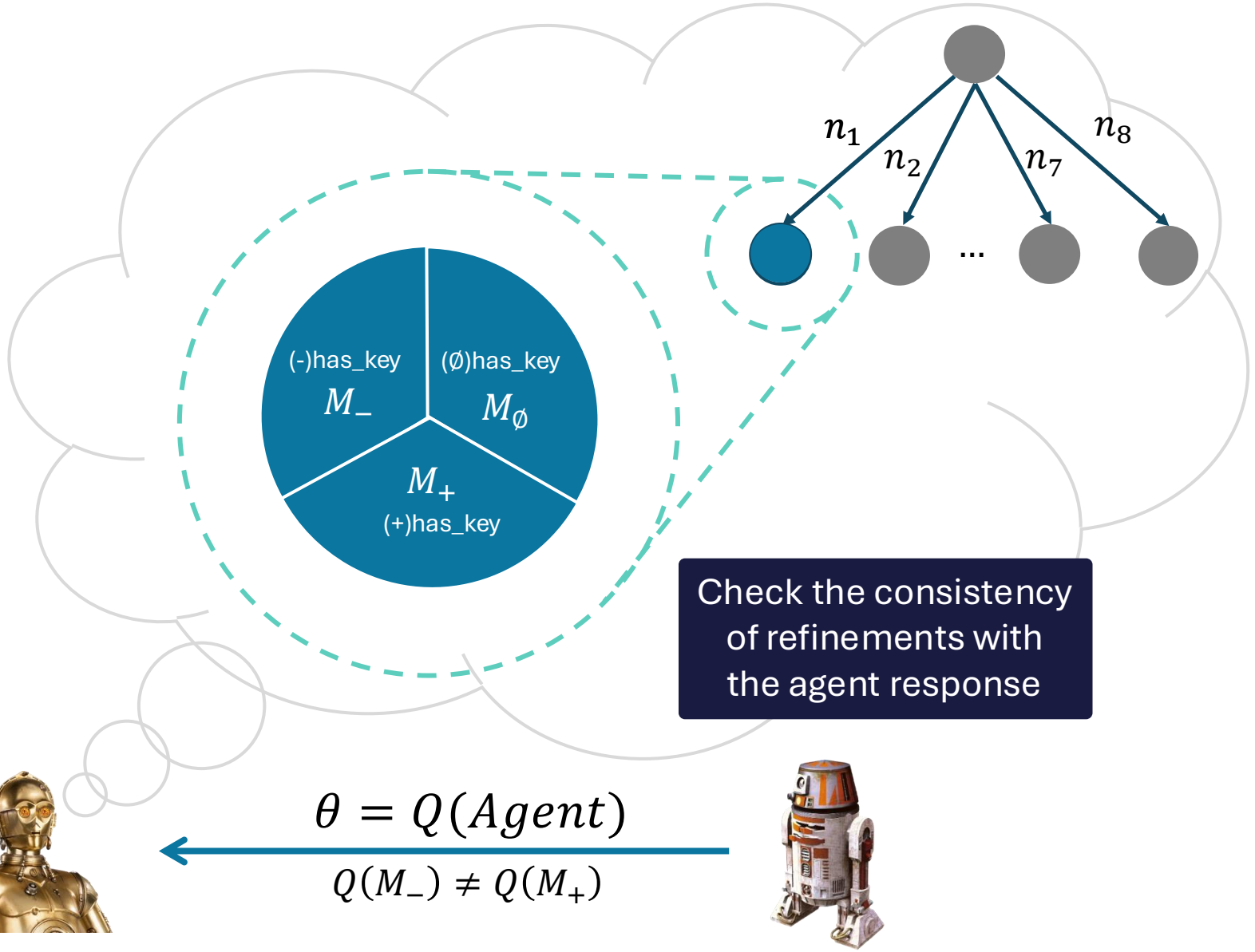
Hierarchical Query Synthesis




```
(:action open-door
  :parameters (?l1)
  :precondition (and
n1 (+/-/∅)(has_key)
n2 (+/-/∅)(door_open)
n3 (+/-/∅)(door_adjacent ?l1)
n4 (+/-/∅)(player_at ?l1))
  :effect (and
n5 (+/-/∅)(has_key)
n6 (+/-/∅)(door_open)
n7 (+/-/∅)(door_adjacent ?l1)
n8 (+/-/∅)(player_at ?l1))
```

[**Verma**, Marpally, Srivastava; AAAI '21]

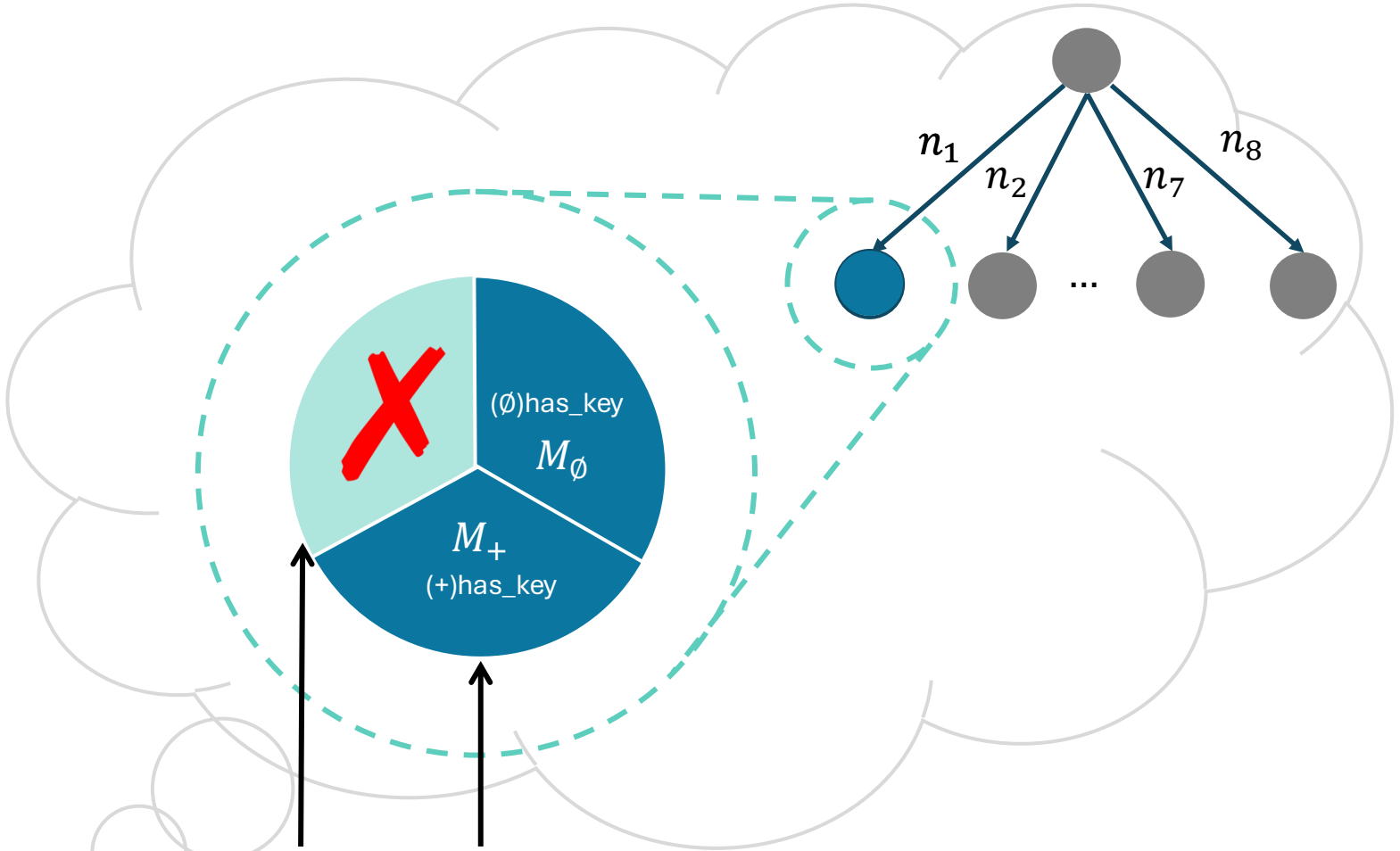
Hierarchical Query Synthesis




```
(:action open-door
  :parameters (?l1)
  :precondition (and
n1  (+/-/Ø)(has_key)
n2  (+/-/Ø)(door_open)
n3  (+/-/Ø)(door_adjacent ?l1)
n4  (+/-/Ø)(player_at ?l1))
  :effect (and
n5  (+/-/Ø)(has_key)
n6  (+/-/Ø)(door_open)
n7  (+/-/Ø)(door_adjacent ?l1)
n8  (+/-/Ø)(player_at ?l1))
```

[**Verma**, Marpally, Srivastava; AAAI '21]

Hierarchical Query Synthesis

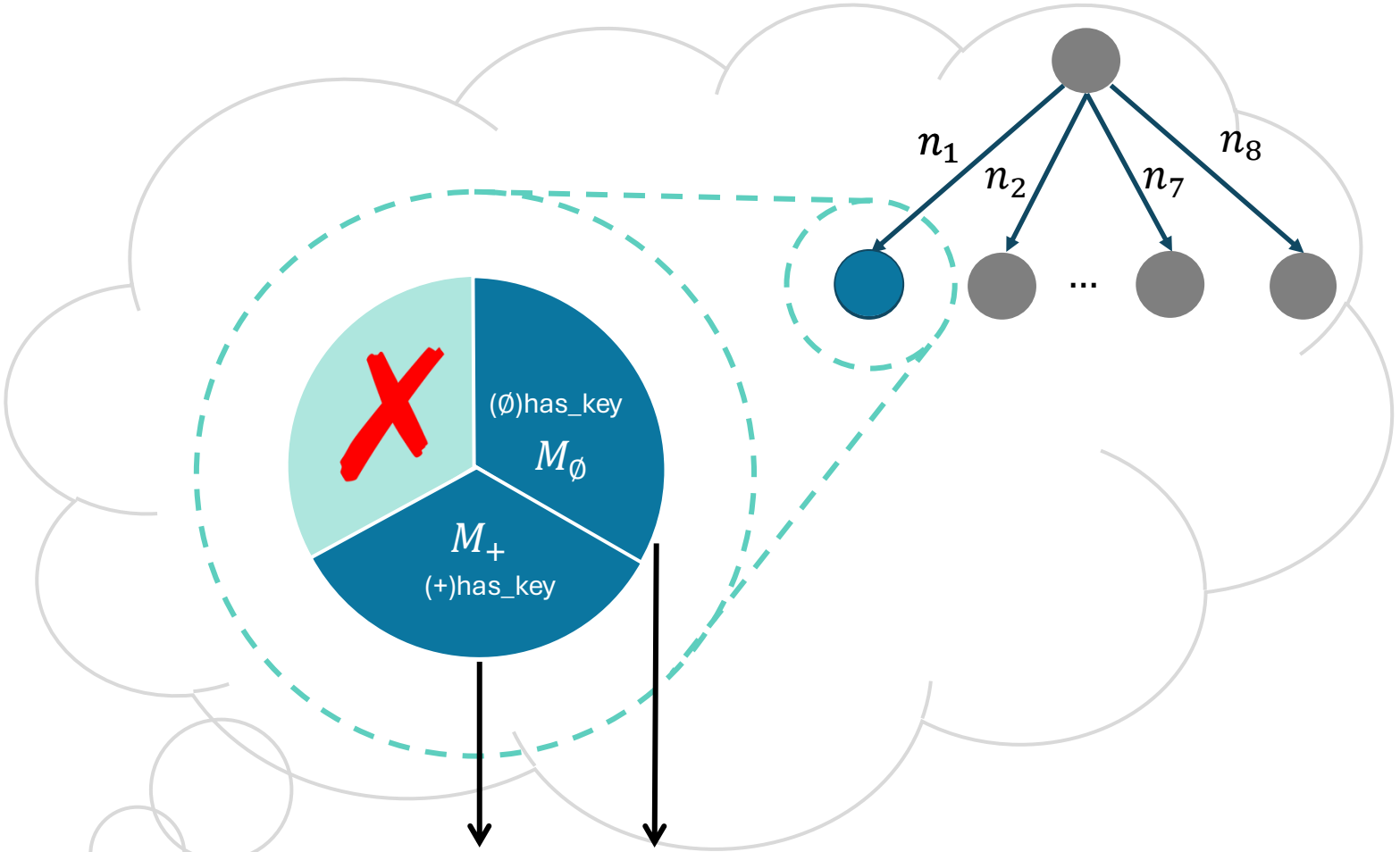



```
(:action open-door
  :parameters (?l1)
  :precondition (and
n1   (+/Ø)(has_key)
n2   (+/-/Ø)(door_open)
n3   (+/-/Ø)(door_adjacent ?l1)
n4   (+/-/Ø)(player_at ?l1))
  :effect (and
n5   (+/-/Ø)(has_key)
n6   (+/-/Ø)(door_open)
n7   (+/-/Ø)(door_adjacent ?l1)
n8   (+/-/Ø)(player_at ?l1))
```

Reject abstract model(s) that are not consistent with the agent

[**Verma**, Marpally, Srivastava; AAAI '21]

Hierarchical Query Synthesis

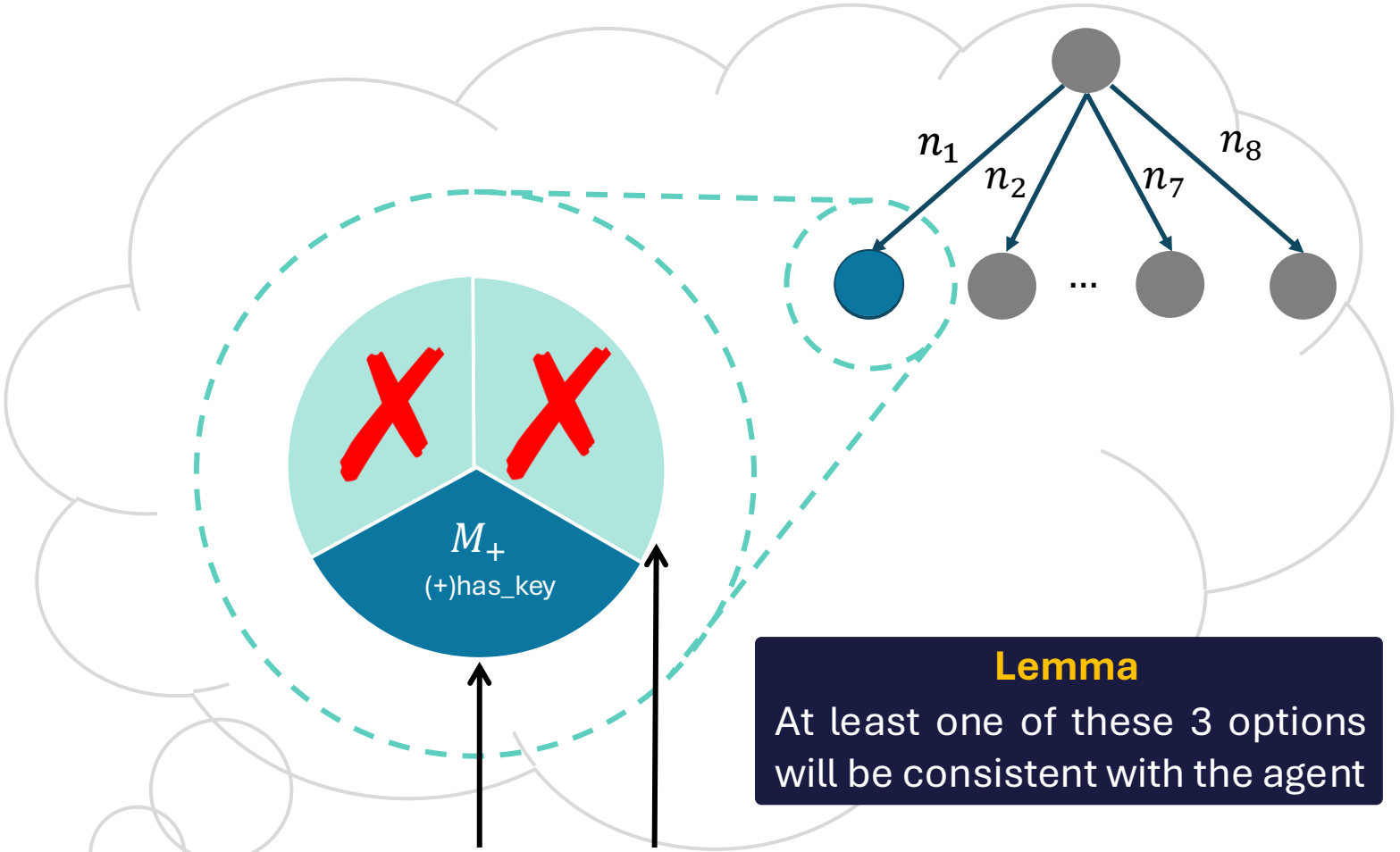


Generate a distinguishing query
for these two abstract models

```
(:action open-door
  :parameters (?l1)
  :precondition (and
n1   (+/Ø)(has_key)
n2   (+/-/Ø)(door_open)
n3   (+/-/Ø)(door_adjacent ?l1)
n4   (+/-/Ø)(player_at ?l1))
  :effect (and
n5   (+/-/Ø)(has_key)
n6   (+/-/Ø)(door_open)
n7   (+/-/Ø)(door_adjacent ?l1)
n8   (+/-/Ø)(player_at ?l1))
```

[**Verma**, Marpally, Srivastava; AAAI '21]

Hierarchical Query Synthesis



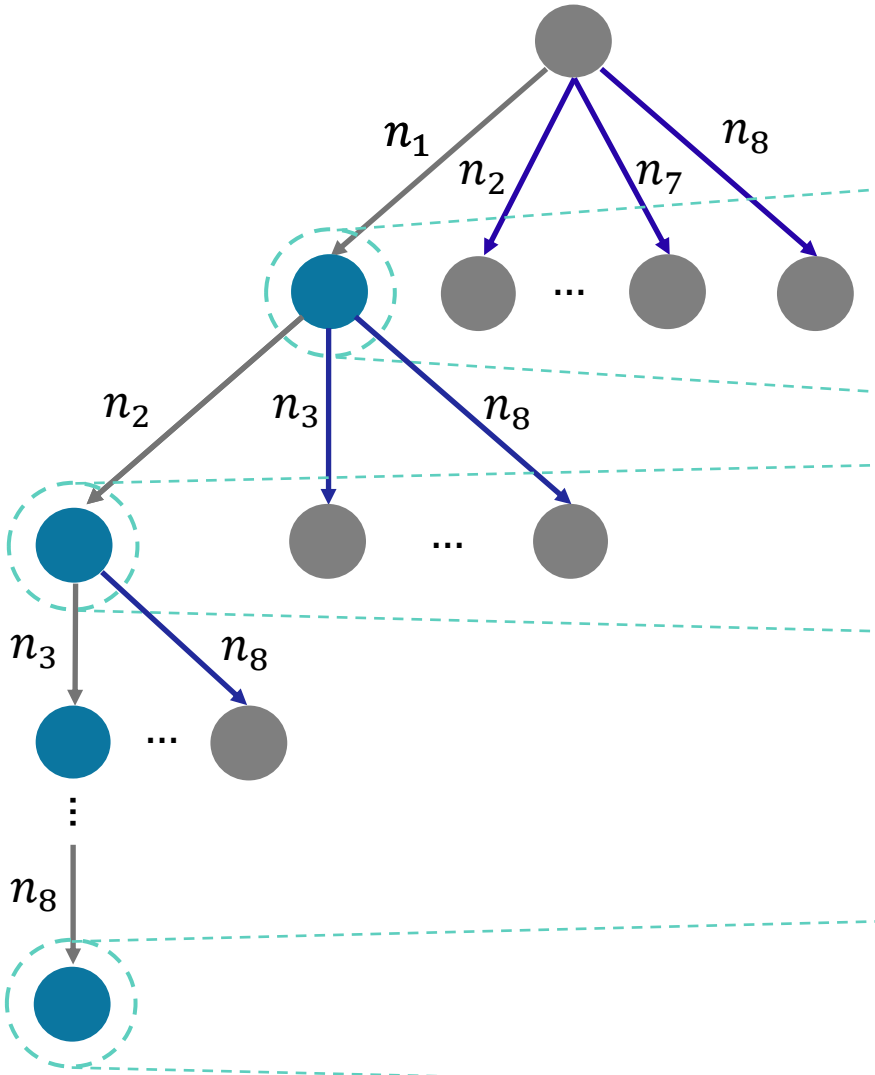
Lemma
At least one of these 3 options will be consistent with the agent

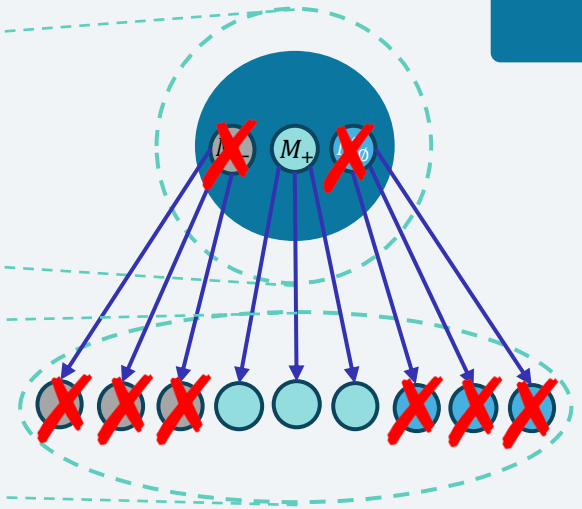
```
(:action open-door
  :parameters (?l1)
  :precondition (and
n1      (+)(has_key)
n2    (+/-/∅)(door_open)
n3    (+/-/∅)(door_adjacent ?l1)
n4    (+/-/∅)(player_at ?l1))
  :effect (and
n5    (+/-/∅)(has_key)
n6    (+/-/∅)(door_open)
n7    (+/-/∅)(door_adjacent ?l1)
n8    (+/-/∅)(player_at ?l1))
```

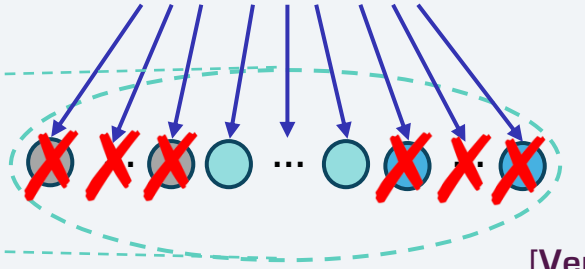
Reject the abstract model
that is not consistent with the agent

[**Verma**, Marpally, Srivastava; AAAI '21]

Hierarchical Query Synthesis







Key feature of the algorithm

Whenever we prune an abstract model, we prune a large number of concrete models.

Active Learning

[**Verma**, Marpally, Srivastava; AAAI '21]

Deterministic and Stationary Setting

Input

- Predicates (User vocabulary)
  - With their evaluation functions
- List of capabilities.

Output

- PDDL-like description of each capability.

Assumptions

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

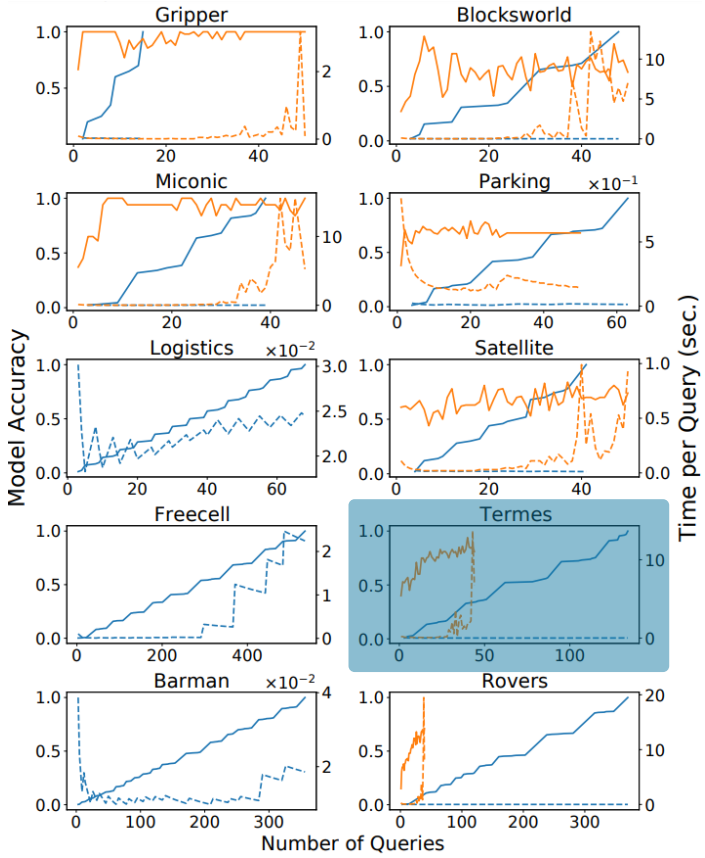
AAM learns Accurate Model with fewer Queries

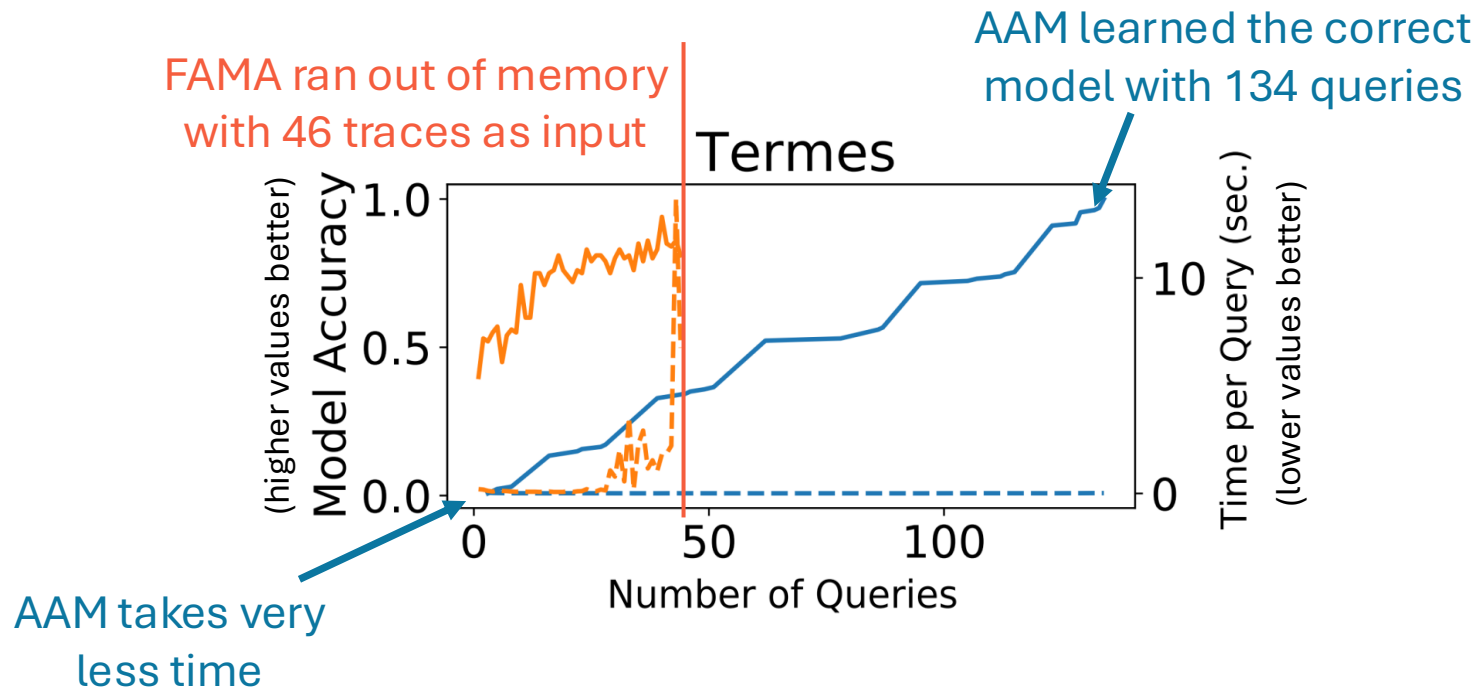
- Asses by learning the model and compare with ground truth.
- Baseline†: A passive learner (FAMA) that observes agent behavior

Random deterministic planning agent from IPC

Accuracy: AAM FAMA

Time: AAM FAMA





[**Verma**, Marpally, Srivastava; AAAI '21]

†Aineto, D.; Celorrio, S. J.; and Onaindia, E. 2019. *Learning Action Models With Minimal Observability*. Artificial Intelligence 275: 104–137.

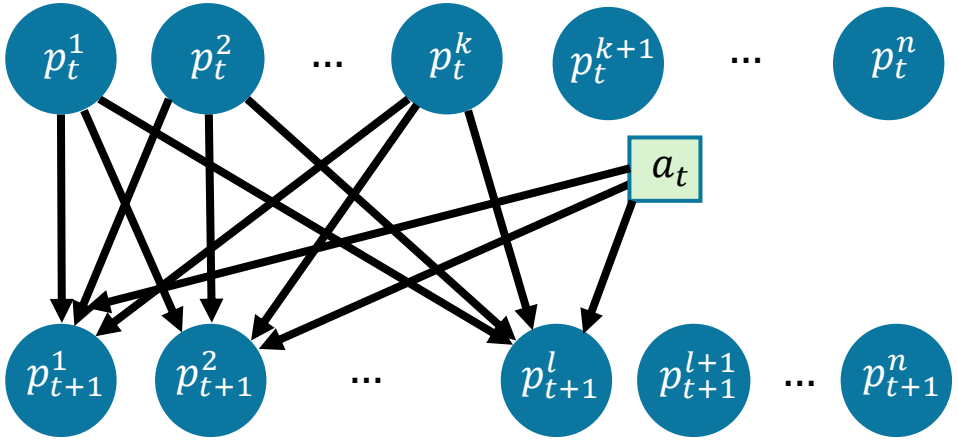
AAM learns Accurate Deterministic Models

- Theorem (*termination*) : The algorithm terminates after a finite number of iterations.

- Theorem (*soundness*): The resulting (set of) model(s) is(are) functionally equivalent to the ground truth model.

Causal Accuracy Analysis

- Use the framework for *Actual Causality*† to define the causal accuracy of the models that we learn.
- Explain theoretically why models learned using passive learners may not be causally accurate.
- Show that the models AAM learns are causally accurate†.



[**Verma**, Srivastava; 2024]

†According to AC2 definition of Actual Cause. Joseph Halpern. *Actual Causality*. MIT Press. 2016.

01
Introduction

02
Foundational Approach

03
Generalizations

04
Conclusion and Future Work

Stochastic and Stationary Setting

Input

- Predicates (User vocabulary)
  - With their evaluation functions
- List of capabilities.

Output

- PPDDL-like description of each capability.

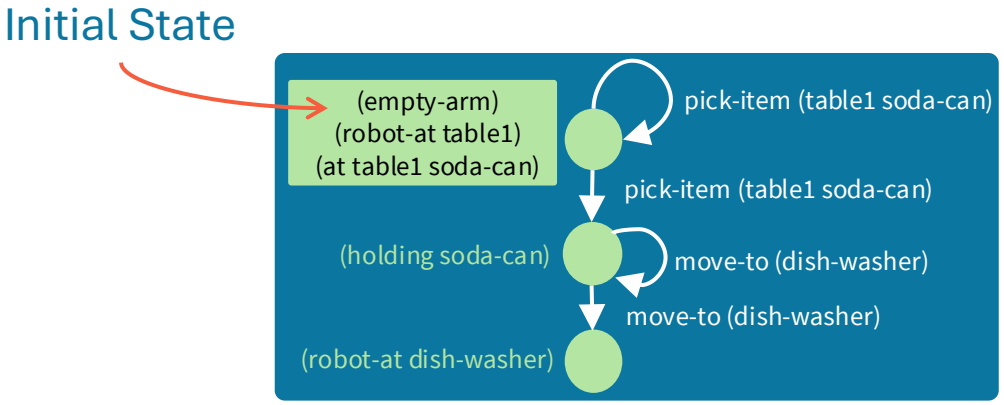
[**Verma**, Karia, Srivastava; NeurIPS '23]

Assumptions

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

Changes for Stochastic Settings

New Queries



Policy: Generated Autonomously by Reduction to Non-Deterministic Planning

What happens if you start in the given initial state and follow this partial policy?

[**Verma**, Karia, Srivastava; NeurIPS '23]

Assumptions

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

Changes for Stochastic Settings

Step 1: Learn a Non-Deterministic Model

```
(:action open-door
  :parameters (?l1)
  :precondition (and
    (+/-/∅)(has_key)
    (+/-/∅)(door_open)
    (+/-/∅)(door_adjacent ?l1)
    (+/-/∅)(player_at ?l1))
  :effect (oneof
    (and
      (+/-/∅)(has_key)
      (+/-/∅)(door_open)
      (+/-/∅)(door_adjacent ?l1)
      (+/-/∅)(player_at ?l1))
    (and
      (+/-/∅)(has_key)
      (+/-/∅)(door_open)
      (+/-/∅)(door_adjacent ?l1)
      (+/-/∅)(player_at ?l1))))
```

Apply Maximum Likelihood Estimation on the observed data (query responses)

Step 2: Convert to Probabilistic Model

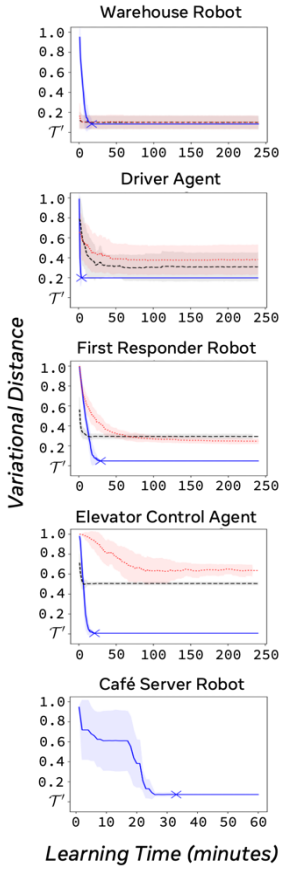
```
(:action open-door
  :parameters (?l1)
  :precondition (and
    (+/-/∅)(has_key)
    (+/-/∅)(door_open)
    (+/-/∅)(door_adjacent ?l1)
    (+/-/∅)(player_at ?l1))
  :effect (probabilistic
    0.xx (and
      (+/-/∅)(has_key)
      (+/-/∅)(door_open)
      (+/-/∅)(door_adjacent ?l1)
      (+/-/∅)(player_at ?l1))
    0.yy (and
      (+/-/∅)(has_key)
      (+/-/∅)(door_open)
      (+/-/∅)(door_adjacent ?l1)
      (+/-/∅)(player_at ?l1))))
```

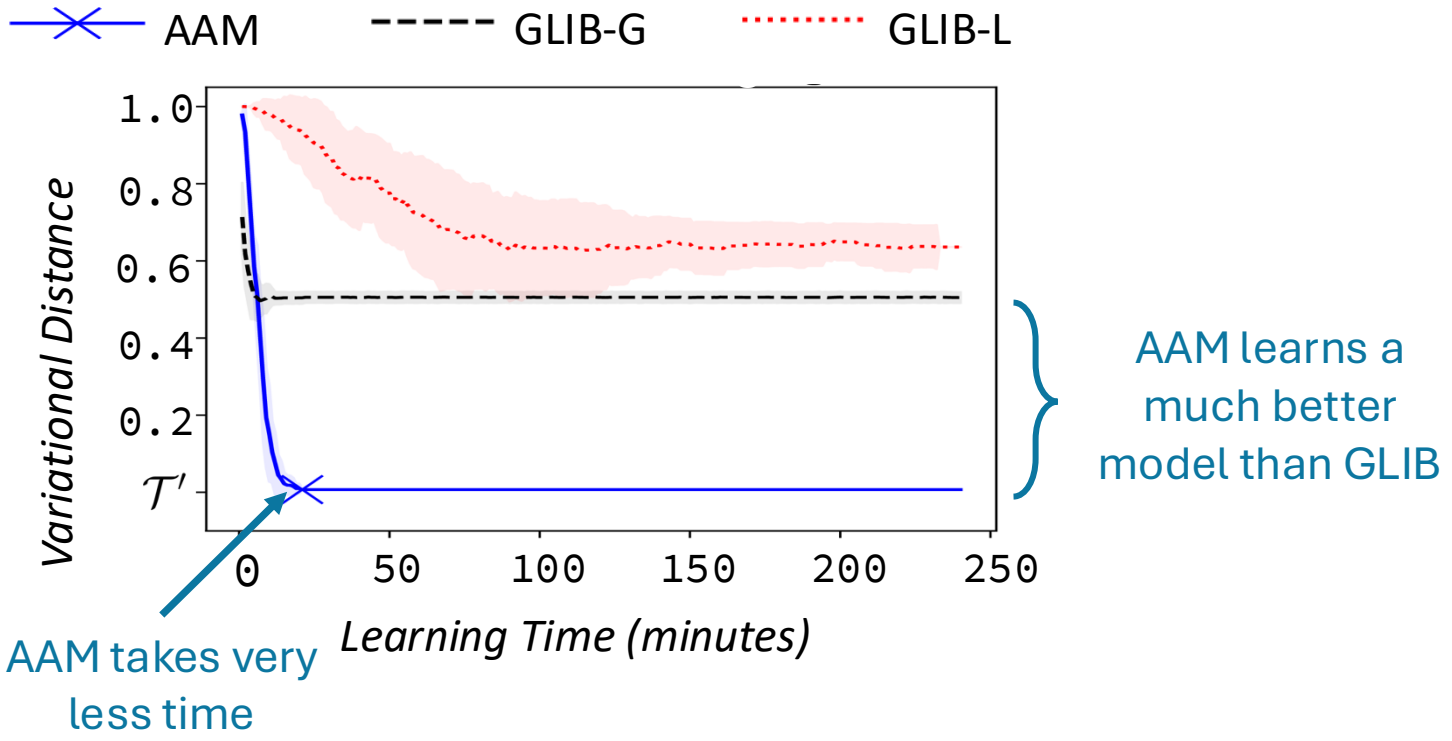
[**Verma**, Karia, Srivastava; NeurIPS '23]

AAM learns accurate probabilistic models faster

- Baseline: directed exploration approach (GLIB)
- Increase the time taken to learn the model.

Random probabilistic planning agent from IPC



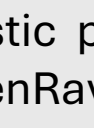


[**Verma**, Karia, Srivastava; NeurIPS '23]

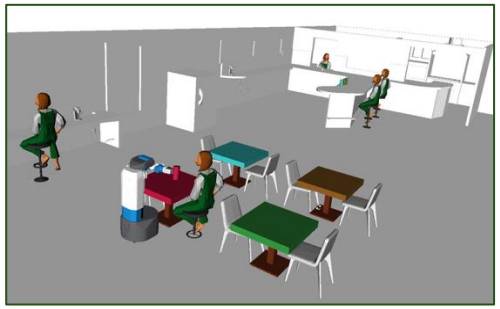
†Chitnis, R.; Silver, T.; Tenenbaum, J.; Kaelbling, L. P.; Lozano-Perez, T. GLIB: Efficient Exploration for Relational MBRL via Goal-Literal Babbling. AAAI 2021.

AAM learns accurate models for Continuous Domains

- Use Task and Motion Planning (TMP) to convert actions into motion plans.
- Increase the time taken to learn the model.

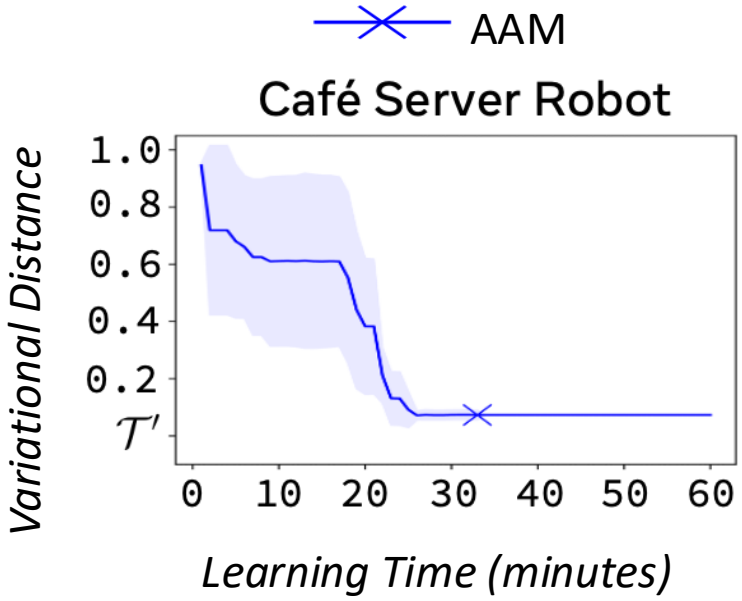


Probabilistic planning agent using OpenRave and TMP



	x	y	z	θ	φ	ψ
robot-base	1.0	-3.2	4.7	0.9	1.3	3.1
soda-can1	6.0	-2.8	3.5	8.3	6.7	9.2
⋮					⋮	
table4	-2.1	4.1	1.9	3.7	9.5	4.8

(empty-arm)
(robot-at table1)
(at table1 soda-can)



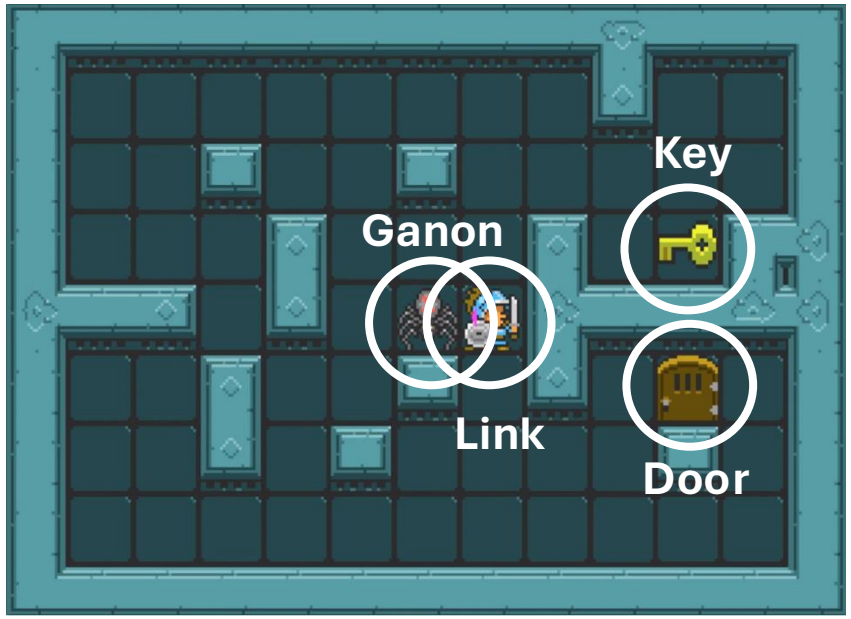
[**Verma**, Karia, Srivastava; NeurIPS '23]

AAM learns Accurate Probabilistic Models

- Theorem (*soundness and completeness*):
  The intermediate non-deterministic model (after step 1) is sound and complete w.r.t. the ground truth model.

- Theorem (*probabilistic correctness*):
  The resulting probabilistic model is correct w.r.t. the ground truth model.

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)	

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

Discovering Capabilities

Input

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

Output

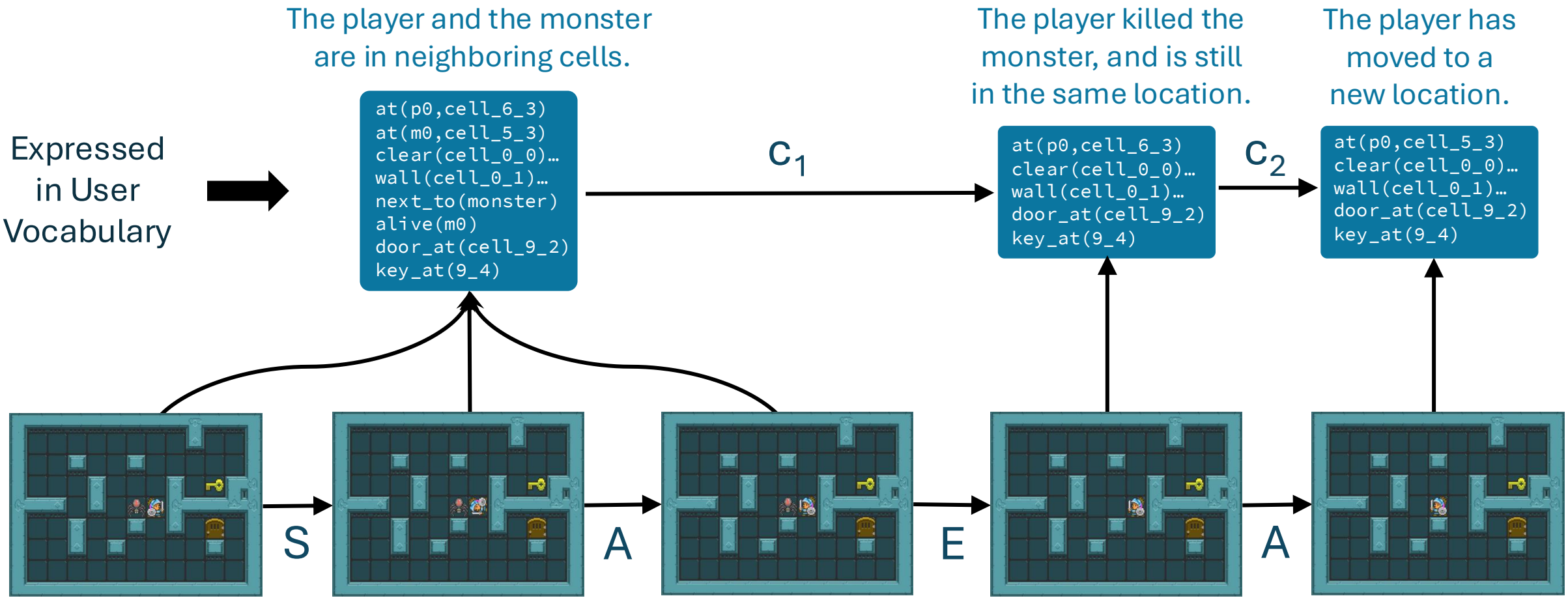
- List of capabilities.
- PDDL-like description of each capability.

[**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



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

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”

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

User Study Setup to Verify Interpretability

Preconditions

Effects

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*.

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

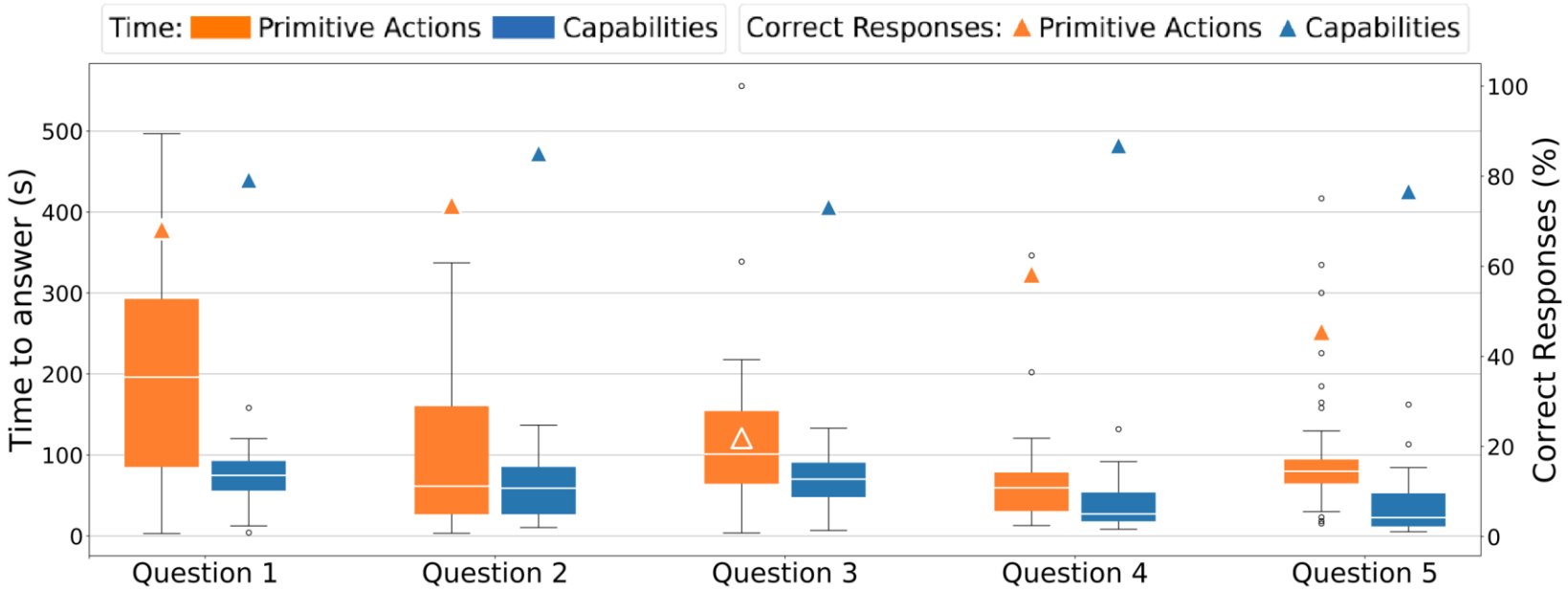
[Functionality Description Example]

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

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:



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

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.

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

Discovering Capabilities in Stochastic Settings

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.

[Bramblett, Karia, Ciotinga, **Verma**, Choi, Srivastava; GenPlan '26]

Assumptions

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

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.

Agent updates

E.g., software update,

new deployment,
adapted for user needs, etc.

simulator

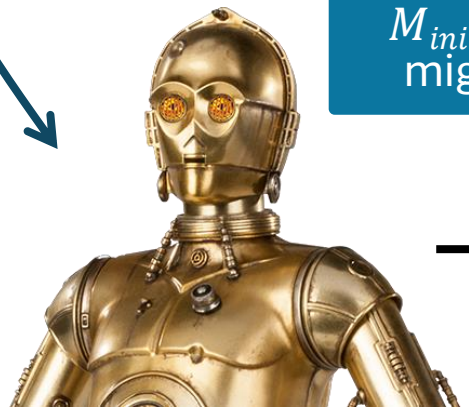
Observations
(collected once)

Query

Response

Use observations and M_{init} to predict what might've changed.

Initial Model
known to the user
M_{init}



Personalized
AI-Assessment Module

Updated model M_{drift} of

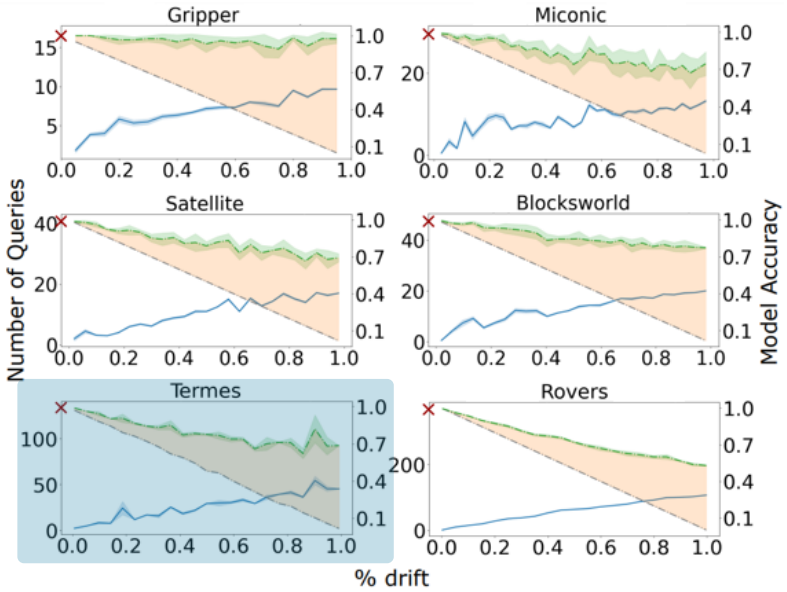
Black-Box AI System's
capabilities

[Nayyar*, **Verma***, Srivastava; AAAI '22]

Fewer Queries Needed Compared to Learning from Scratch

Random deterministic planning agent from IPC

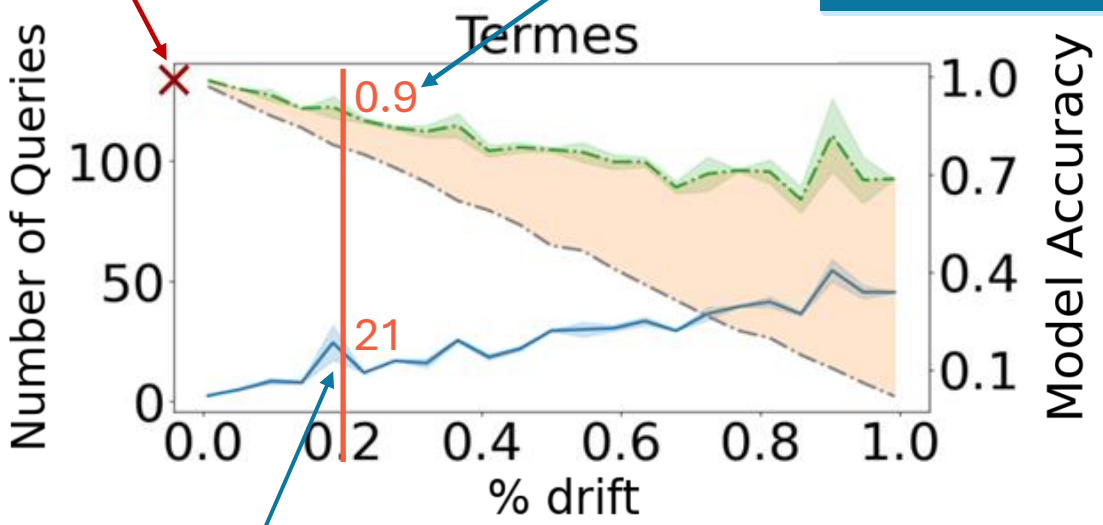
Accuracy of initial model
Accuracy gained by AAM
Accuracy of model computed by AAM
Number of queries by AAM
× Number of queries when learning from scratch



134 queries needed if starting from scratch

Accuracy

Used 10 observations per domain



Number of queries are much lower than 134

[Nayyar*, **Verma***, Srivastava; AAAI '22]

Learned Updated Capability Descriptions are Consistent

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

[Nayyar*, **Verma***, Srivastava; AAAI '22]

01

Introduction

02

Foundational Approach

03

Generalizations

04

Conclusion and Future Work

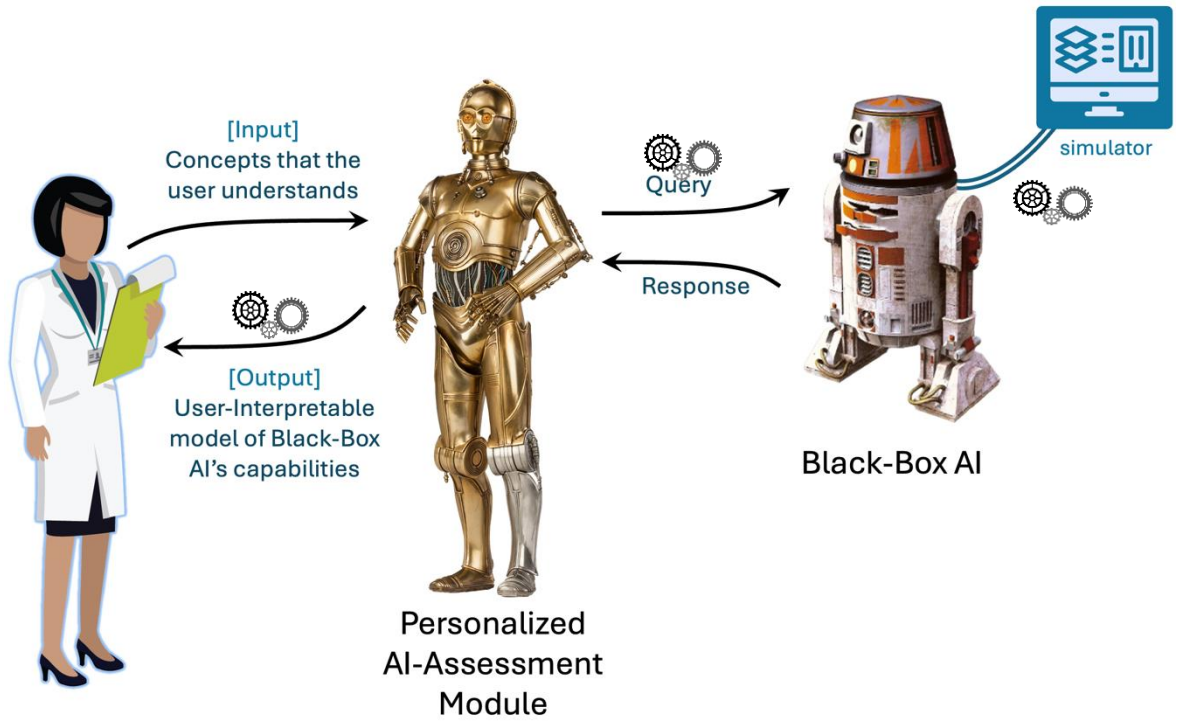
Contributions

- Formally defined the Third-Party AI Assessment Problem for the Taskable AI Systems.
- The first work that shows we can make some assumptions about the interface and assess black-box AI systems on the fly.
- We explain how to assess an adaptive agent after it is deployed.
- Explain theoretically why models learned using passive learners may not be causally accurate.

Future Work: Complexity Analysis

- Perform extensive analysis of queries in terms of:
  - Complexity of generating them.
  - Complexity of answering them.
  - Complexity of inferring models from Black-Box AI's responses.

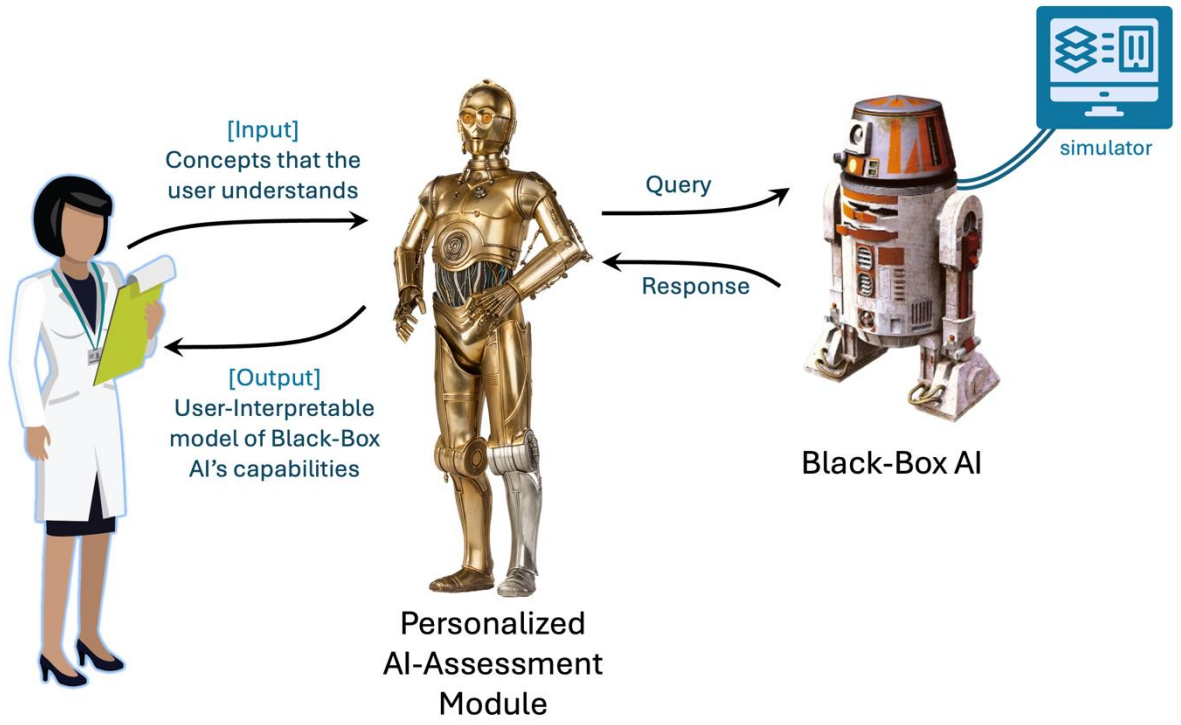
- Extend the work for partially observable settings.



Future Work: Interpretable Representations

- Representations beyond PDDL.
  - LTL – Linear Temporal Logic
  - STL- Signal Temporal Logic

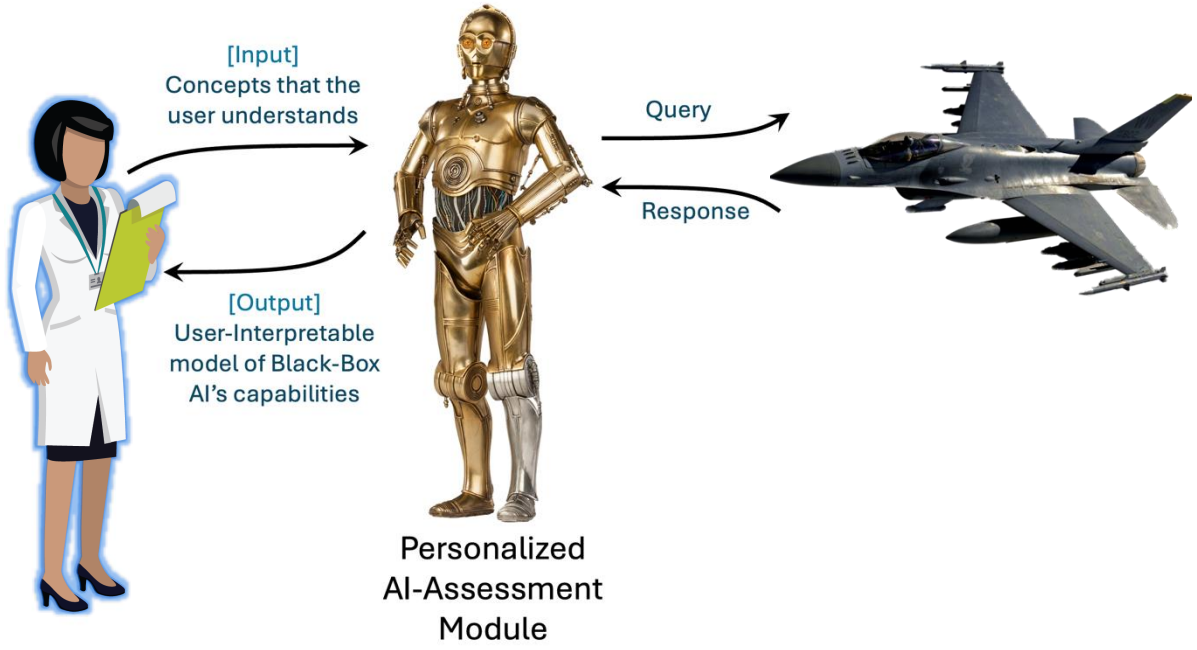
- How interpretable each representation is?



Future Work: Real World Setting

- Flying Extended Trail
- Human must collaborate with the AI-powered aircraft.

- How can we explain the aircraft's policy to the human in an interpretable way?



[Input]
Concepts that the user understands

[Output]
User-Interpretable description of Black-Box AI's capabilities

Future Work

Personalized AI-Assessment Module (AAM)

Query

Response

simulator

Black-Box AI

Arbitrary internal implementation

Doesn't know user's vocabulary

[Input]
Concepts that the user understands

[Output]
User-Interpretable description of Black-Box AI's capabilities

Future Work

Future Work

simulator

Personalized AI-Assessment Module (AAM)

Query

Response

Black-Box AI

Arbitrary internal implementation

Doesn't know user's vocabulary

Rushang Karia

Shashank Marpally

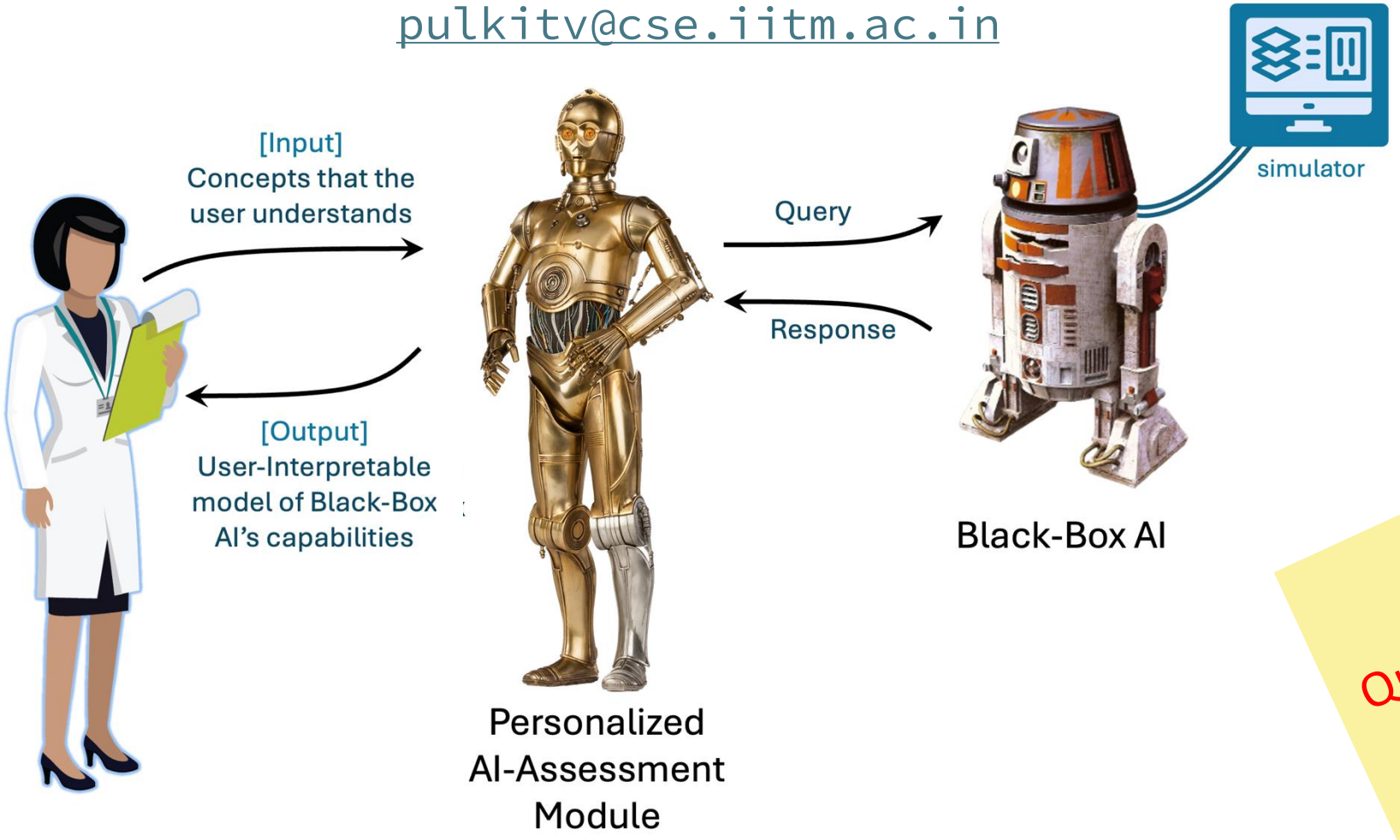
Rashmeet Nayyar

Siddharth Srivastava

Beyond the Lab: User-Aligned Assessment of Taskable AI Systems

Pulkit Verma

pulkitv@cse.iitm.ac.in



Questions?