

October 11, 2024

Learning Interpretable Models for Autonomous Assessment of Taskable AI Systems

Pulkit Verma

✉ pulkitv@mit.edu

🌐 pulkitverma.net



Interactive
ROBOTICS

01

Introduction

02

Foundational
Approach

03

Generalizations

04

Conclusion and
Future Work

Taskable AI Systems: Systems that can Learn and Plan



- 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.

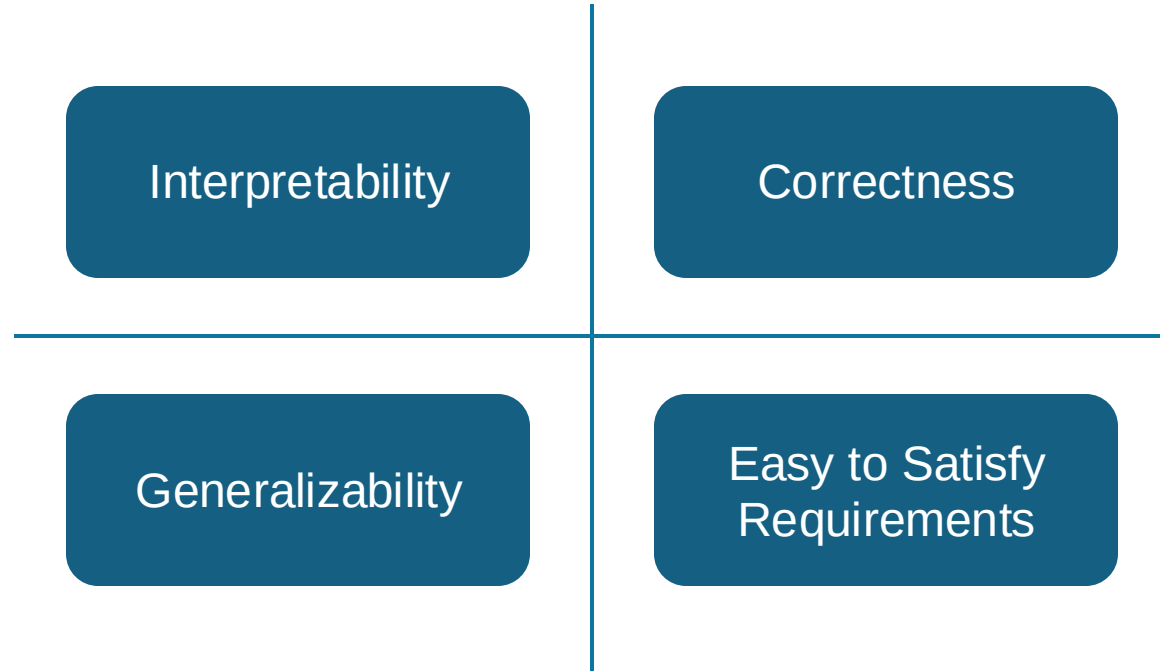
User gives a task and Robots have to complete it

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.



Desiderata of Assessment System



The Assessment Problem

Will it be able to safely rearrange my lab for the next round of experiments?

Output

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

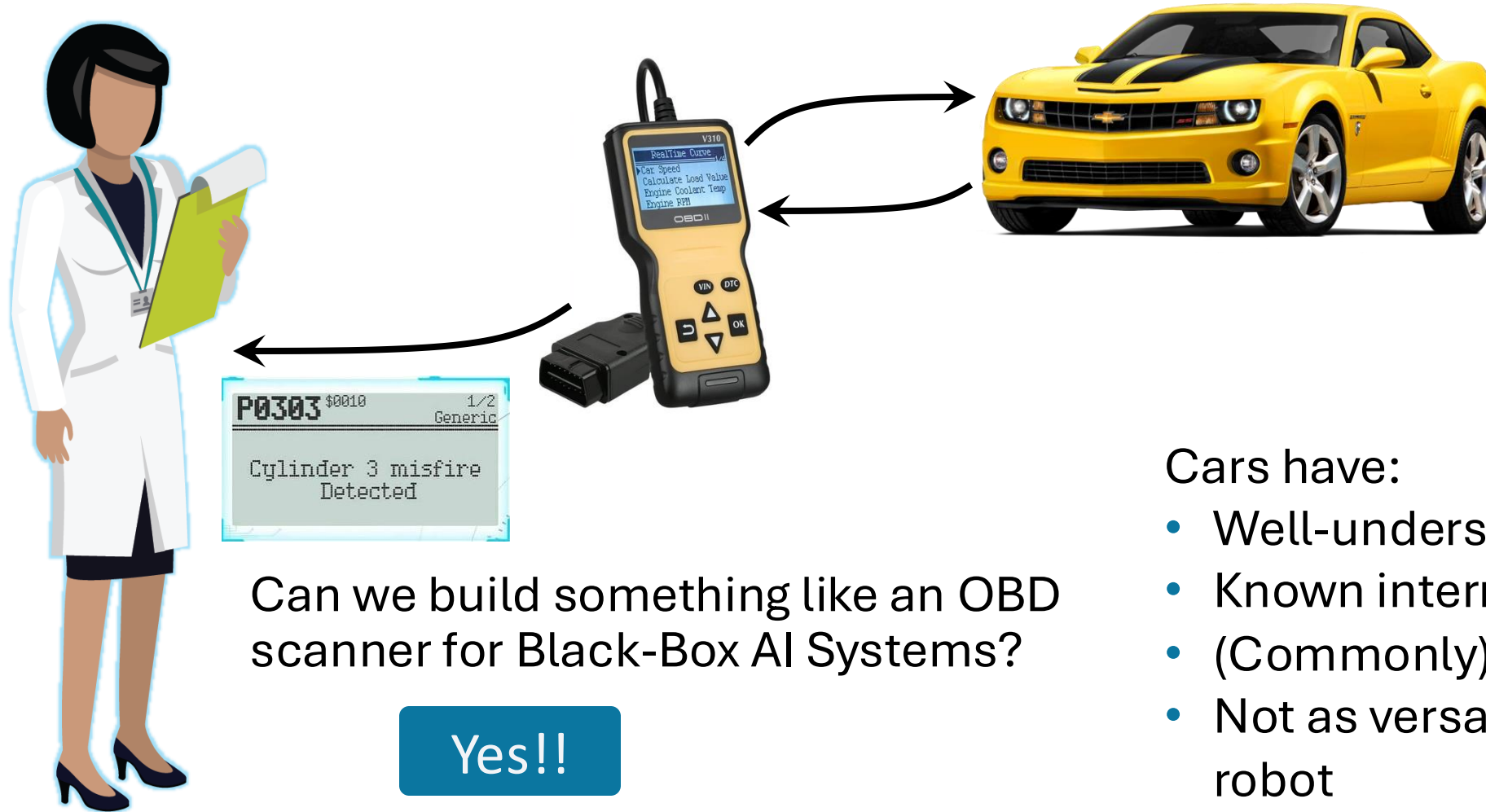


Black-Box AI

- 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)



Arbitrary internal
implementation

Doesn't know
user's vocabulary

Black-Box AI

```
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



Black-Box AI

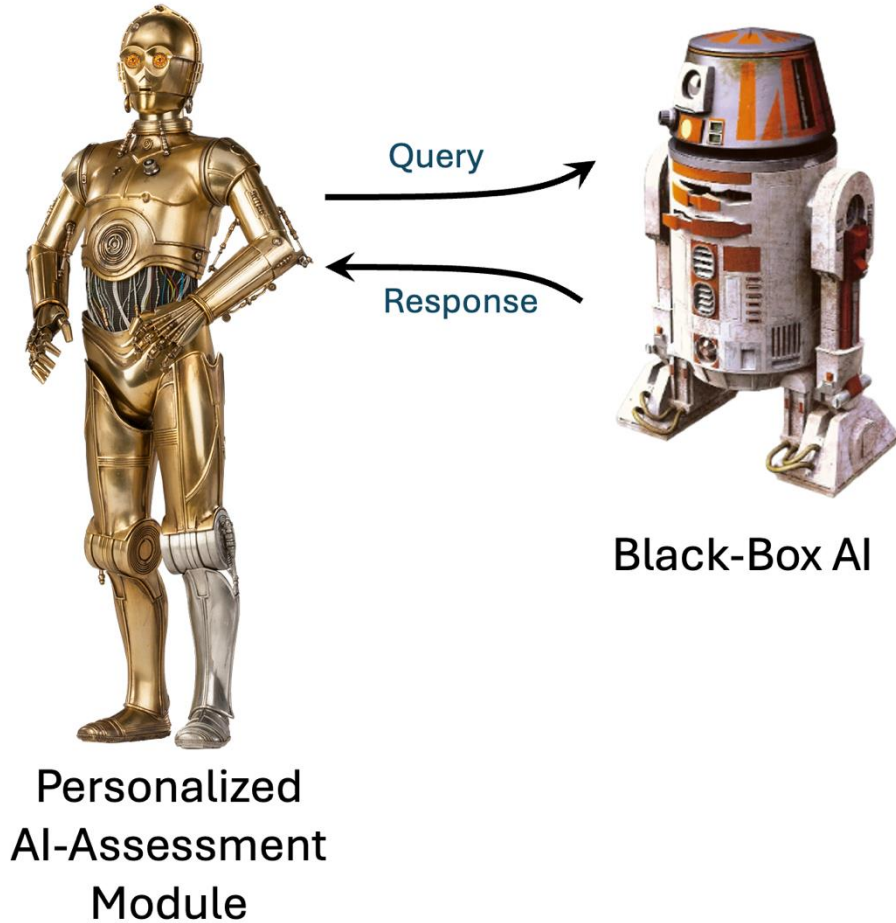


simulator

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: $\langle \text{QueryType}, \text{ResponseType}, \rho \rangle$.



[Input]
Concepts that the
user understands



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



Personalized
AI-Assessment
Module (AAM)

Query



Response



Black-Box AI



simulator

Arbitrary internal
implementation

Doesn't know
user's vocabulary

This Work

The Assessment Problem

Black-Box (Taskable) AI

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

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.

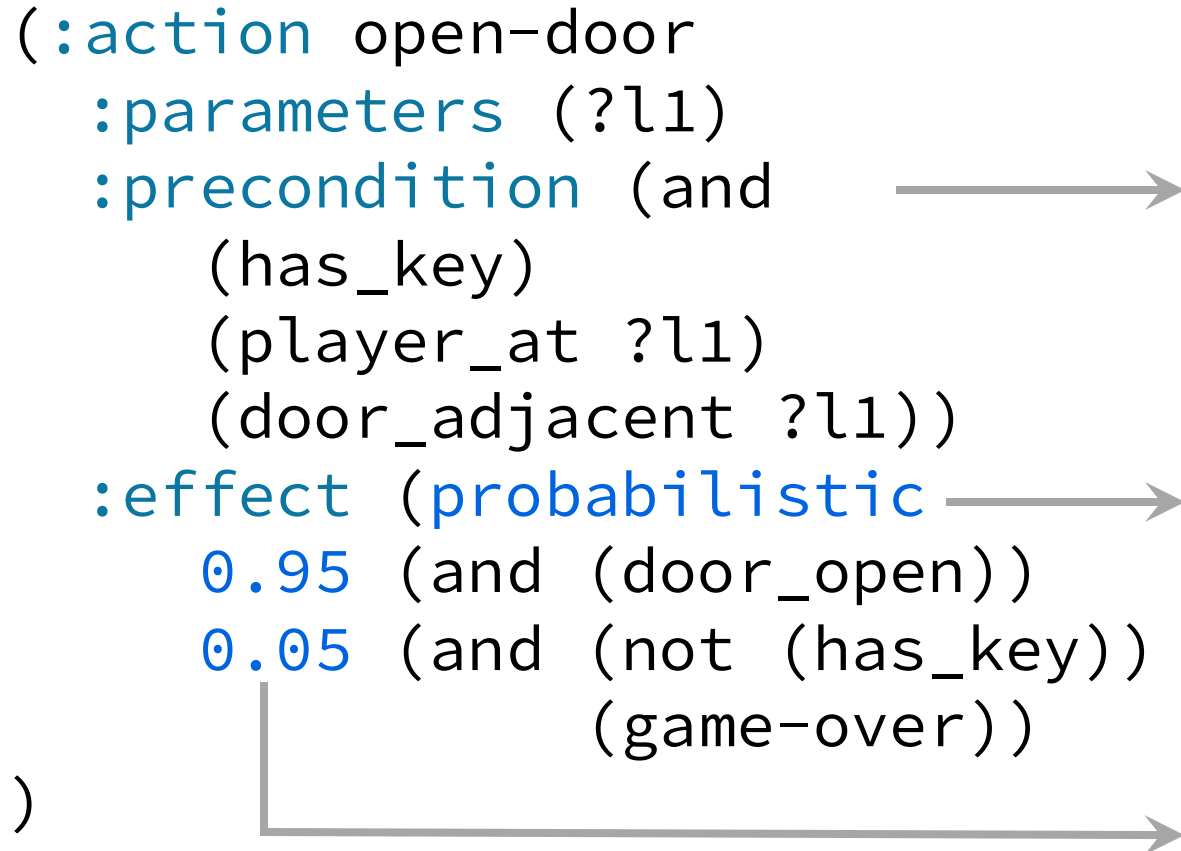


simulator

Black-Box AI

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

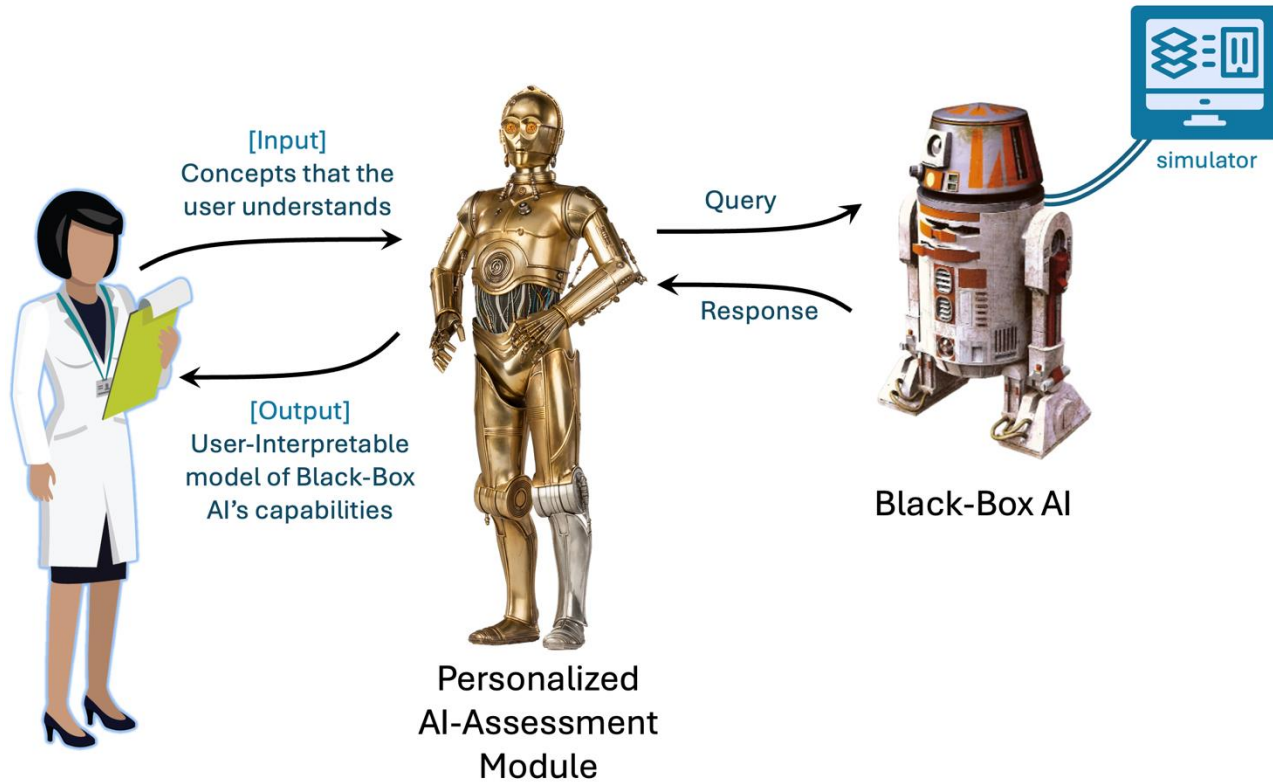
03

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)
- $9^{|C| \times |P|} = 9^{1 \times 4} = 6561$ 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)))
```

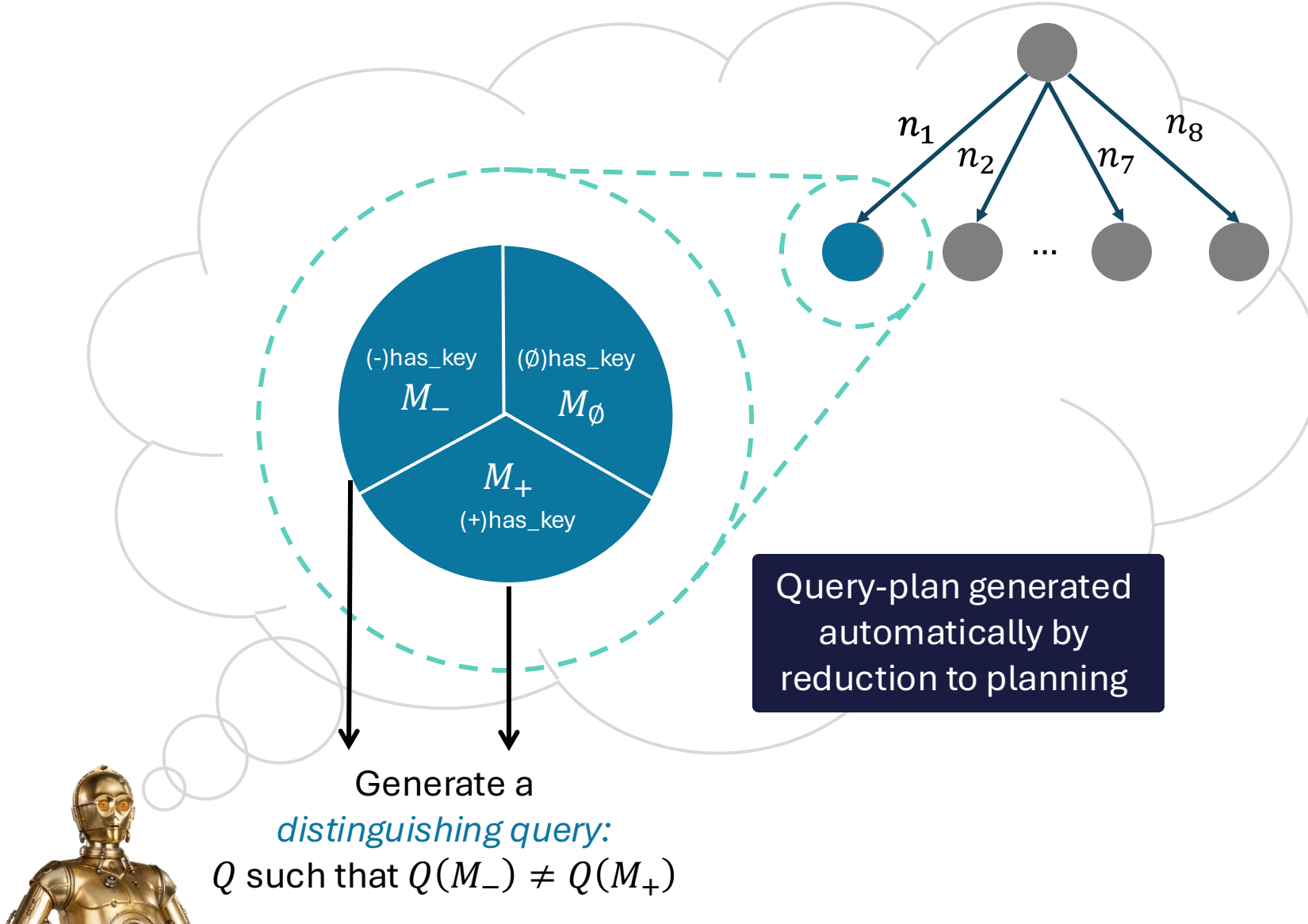
Simple Queries

Query	In state s_I , what will happen if you execute the plan $\pi = \langle c_1, \dots, c_n \rangle$?	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.
	Plan Outcome Queries	State Reachability Query

- 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.

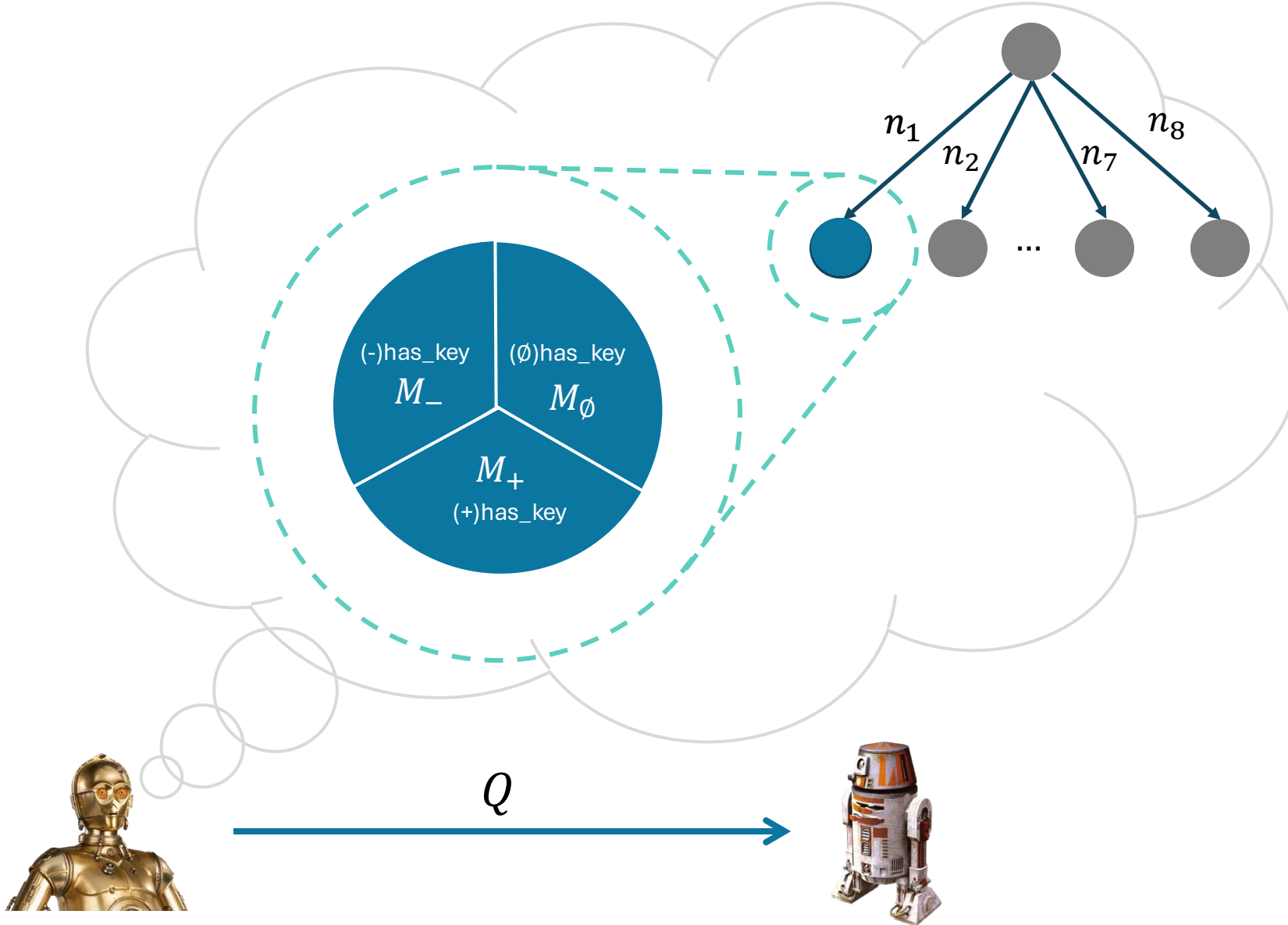
Hierarchical Query Synthesis



```
(: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)))
```

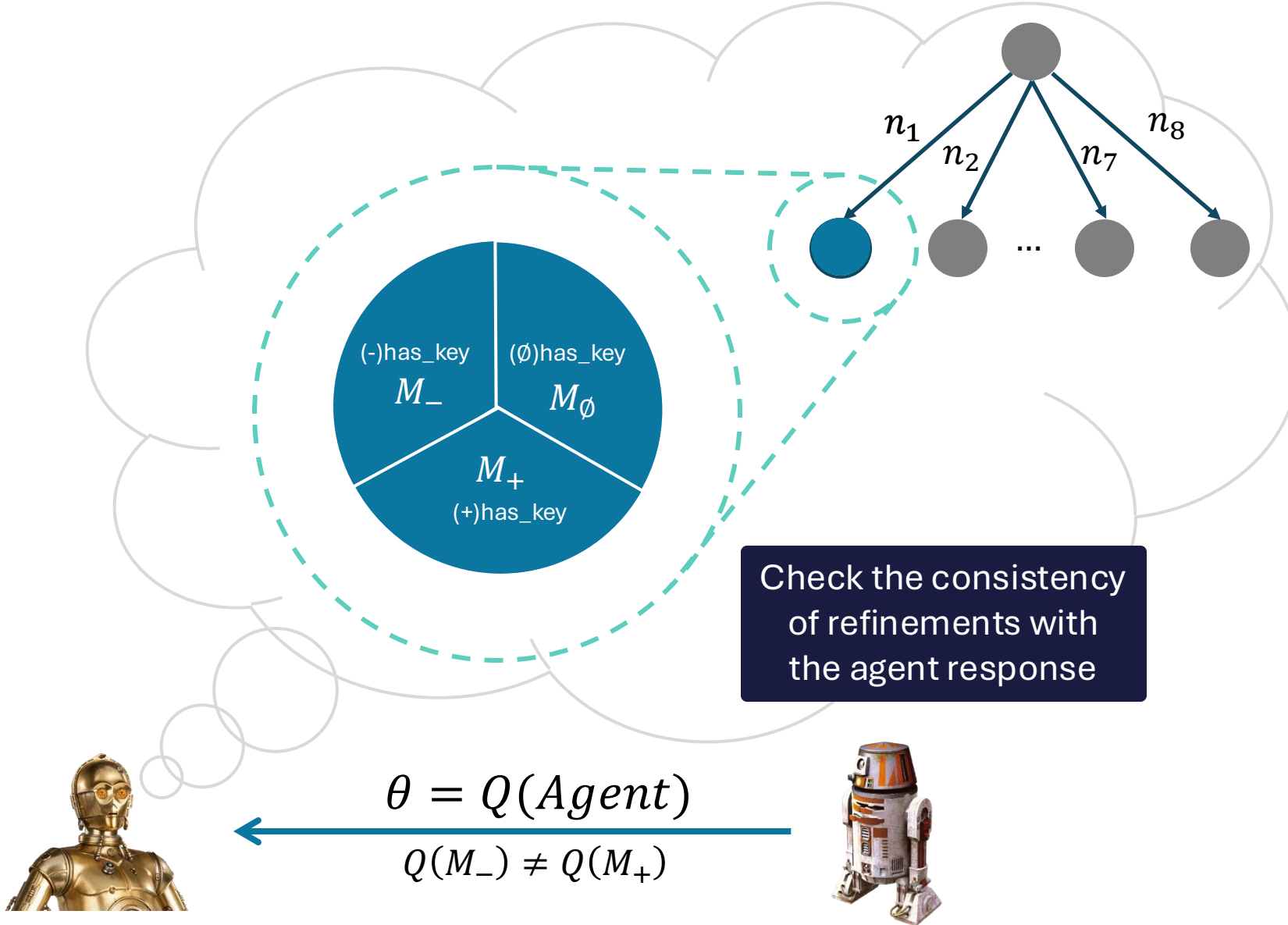


Hierarchical Query Synthesis



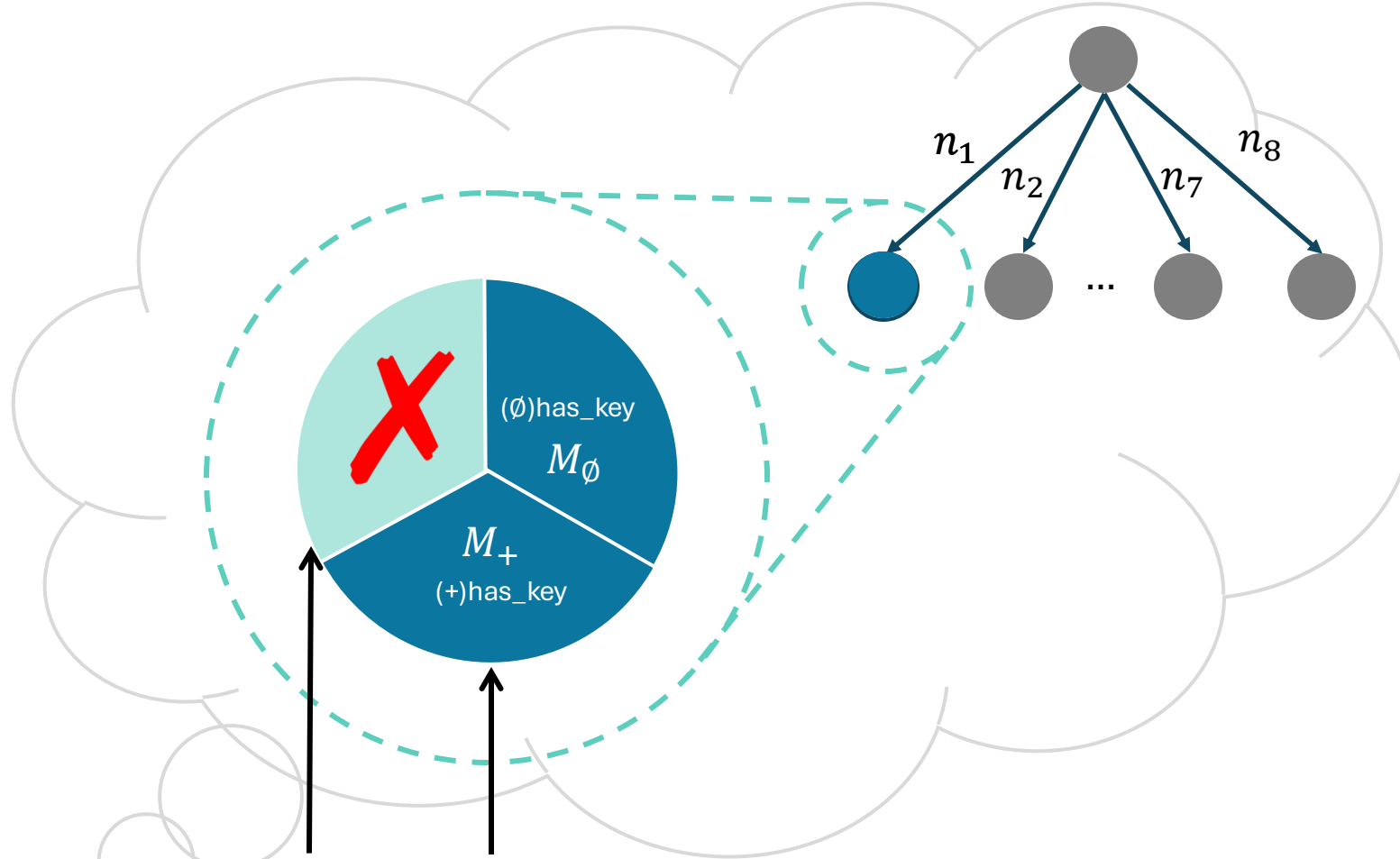
```
(:action open-door
  :parameters (?l1)
  :precondition (and
    n1 (+/-/∅)(has_key)
    n2 (+/-/∅)(door_open)
    n3 (+/-/∅)(door_adjacent ?l1)
    n4 (+/-/∅)(player_at ?l1))
  :effect (and
    n5 (+/-/∅)(has_key)
    n6 (+/-/∅)(door_open)
    n7 (+/-/∅)(door_adjacent ?l1)
    n8 (+/-/∅)(player_at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

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))
```

Hierarchical Query Synthesis

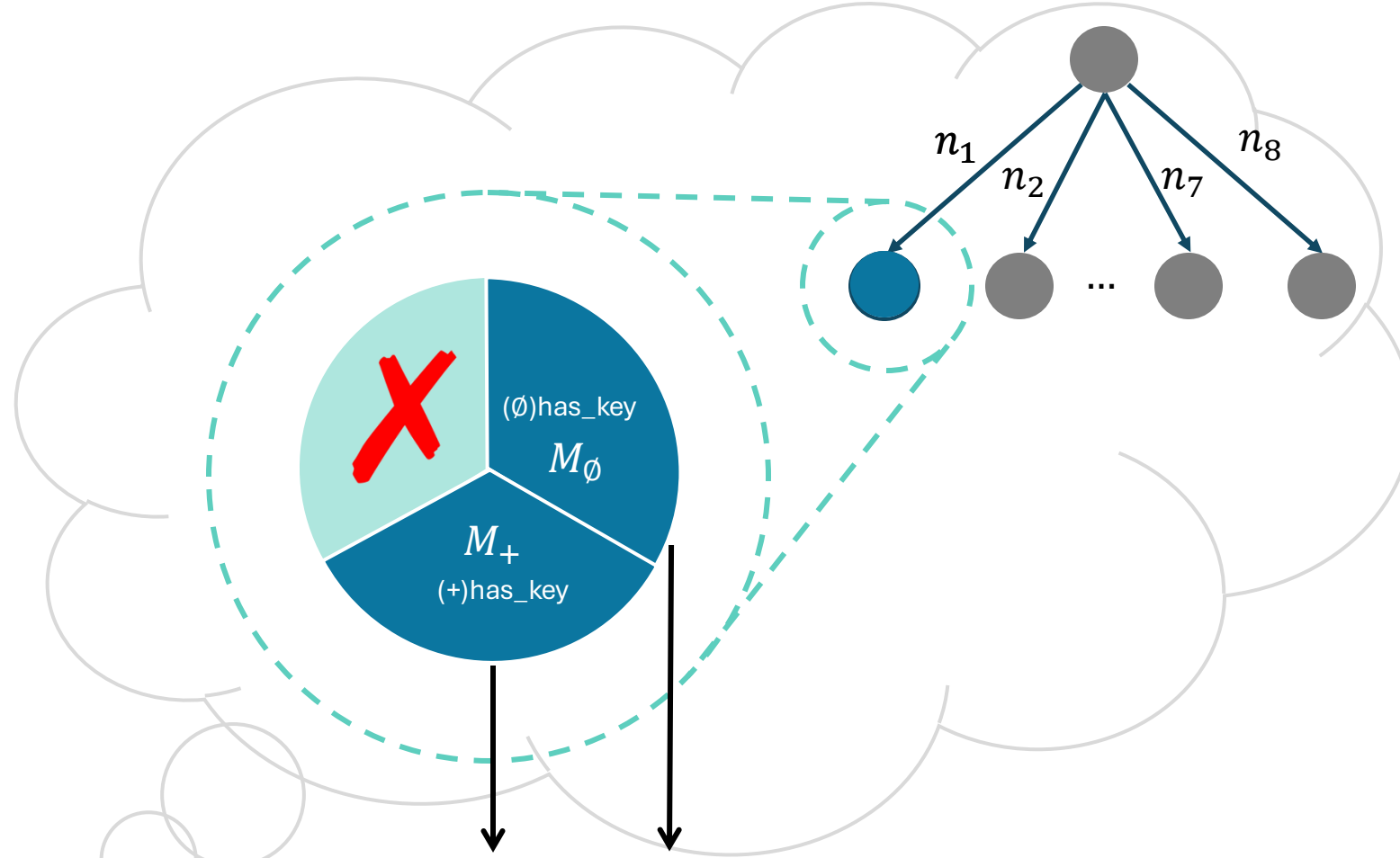


Reject abstract model(s) that are not consistent with the agent



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

Hierarchical Query Synthesis

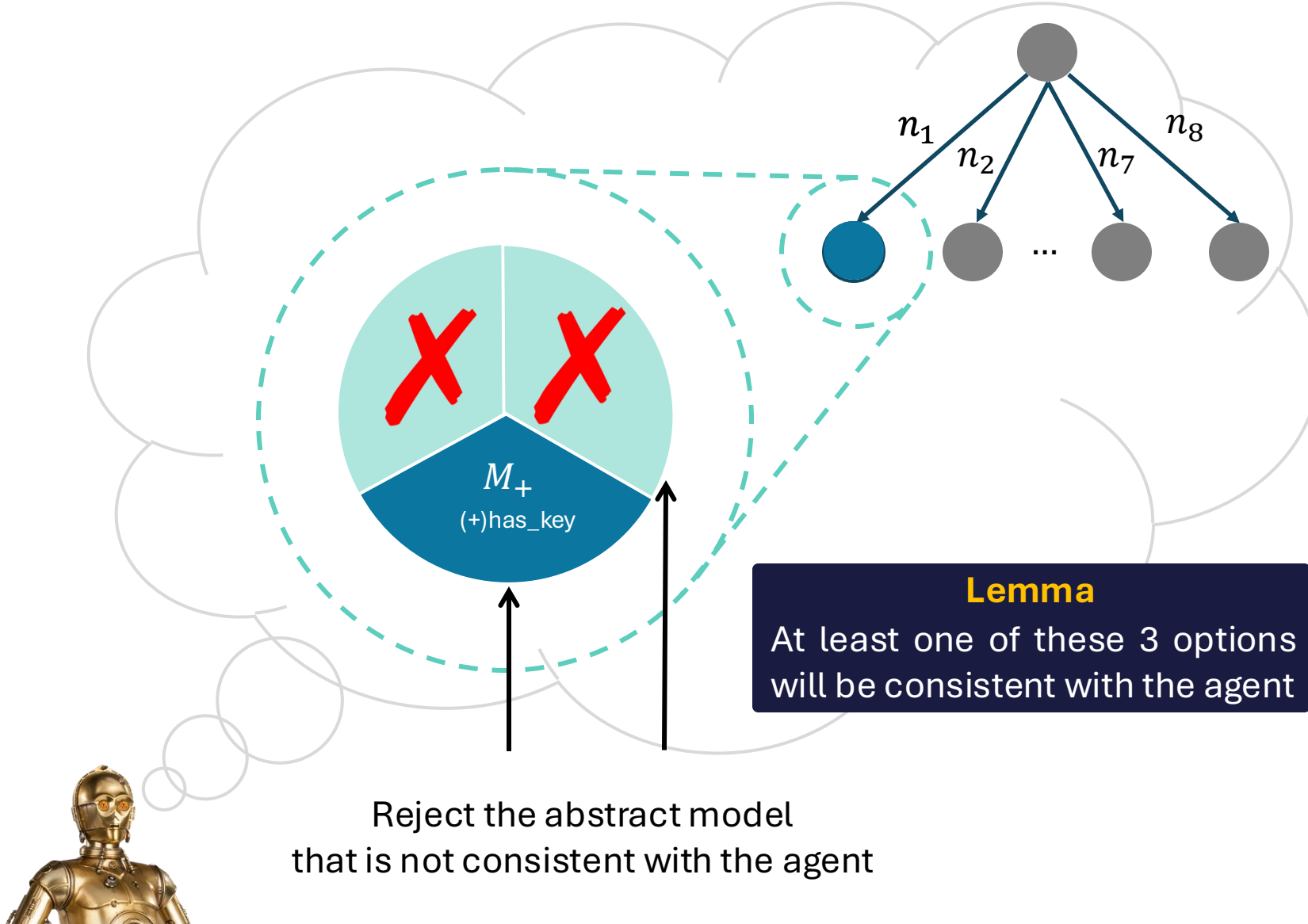


```
(:action open-door
:parameters (?l1)
:precondition (and
   $n_1$   $(+/\emptyset)(\text{has\_key})$ 
   $n_2$   $(+/-/\emptyset)(\text{door\_open})$ 
   $n_3$   $(+/-/\emptyset)(\text{door\_adjacent } ?l1)$ 
   $n_4$   $(+/-/\emptyset)(\text{player\_at } ?l1)$ )
:effect (and
   $n_5$   $(+/-/\emptyset)(\text{has\_key})$ 
   $n_6$   $(+/-/\emptyset)(\text{door\_open})$ 
   $n_7$   $(+/-/\emptyset)(\text{door\_adjacent } ?l1)$ 
   $n_8$   $(+/-/\emptyset)(\text{player\_at } ?l1)$ ))
```



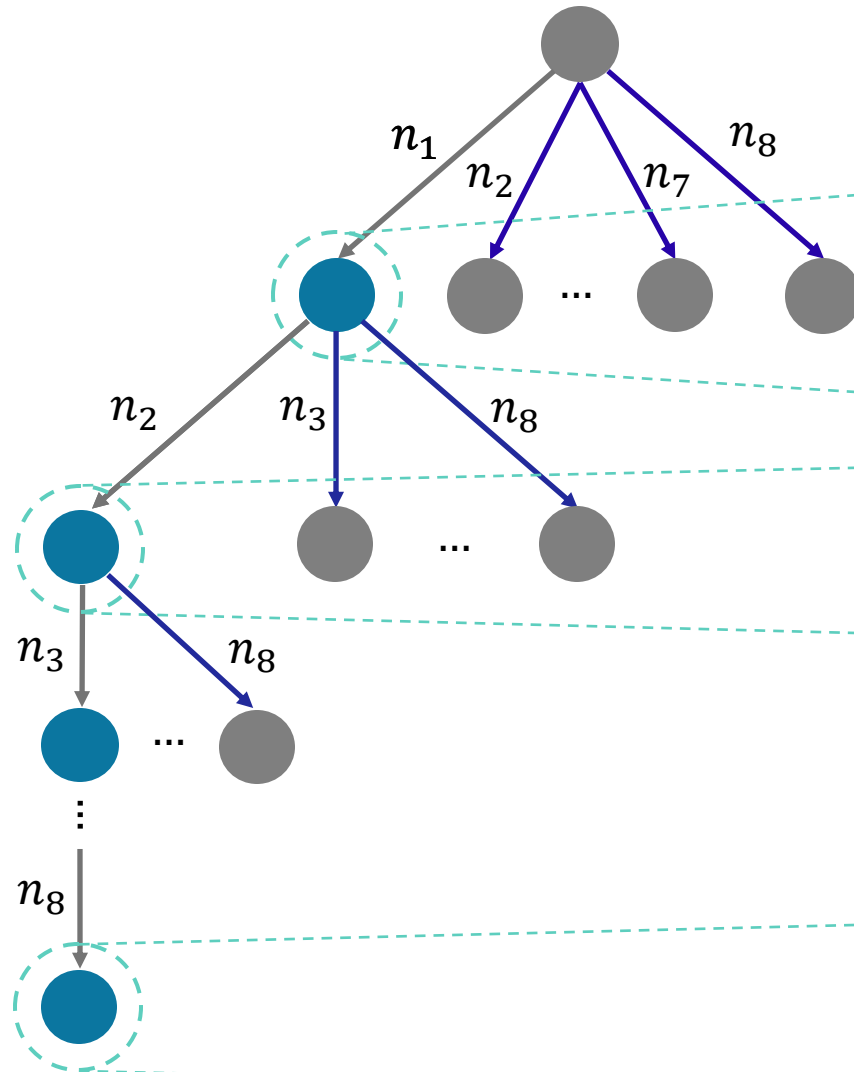
Generate a distinguishing query
for these two abstract models

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))
```

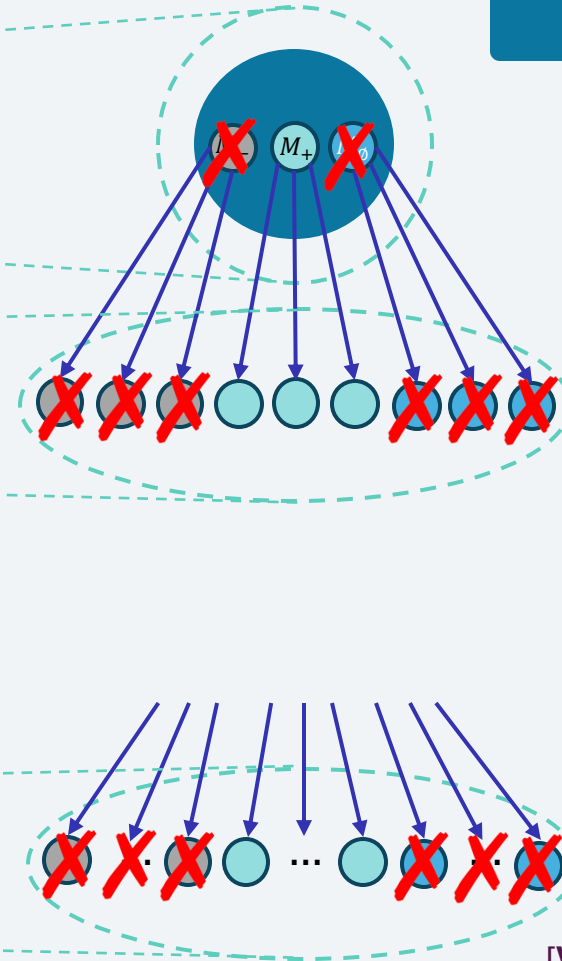
Hierarchical Query Synthesis



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.

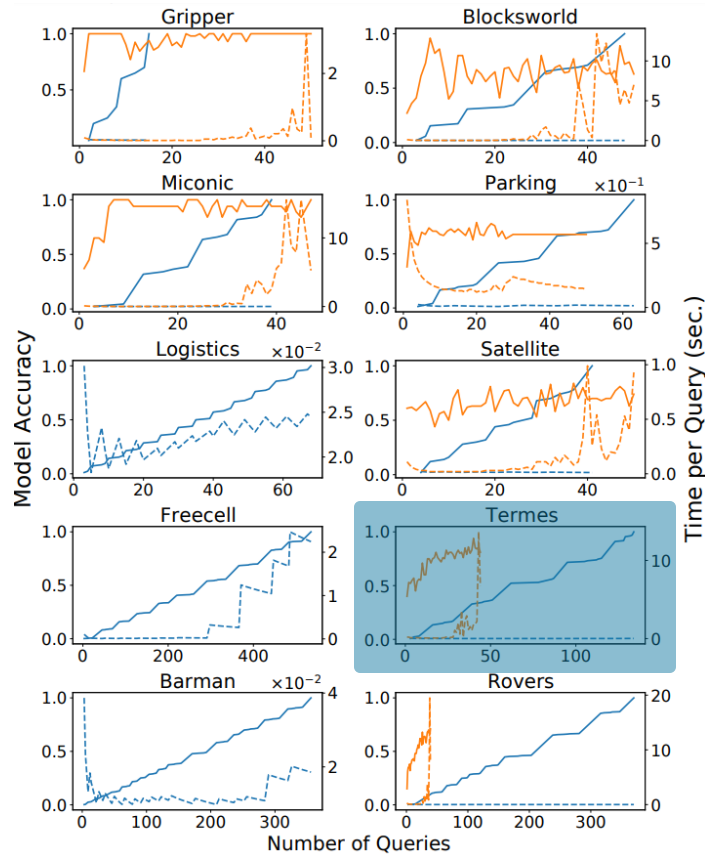
AAM learns Accurate Model with fewer Queries

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



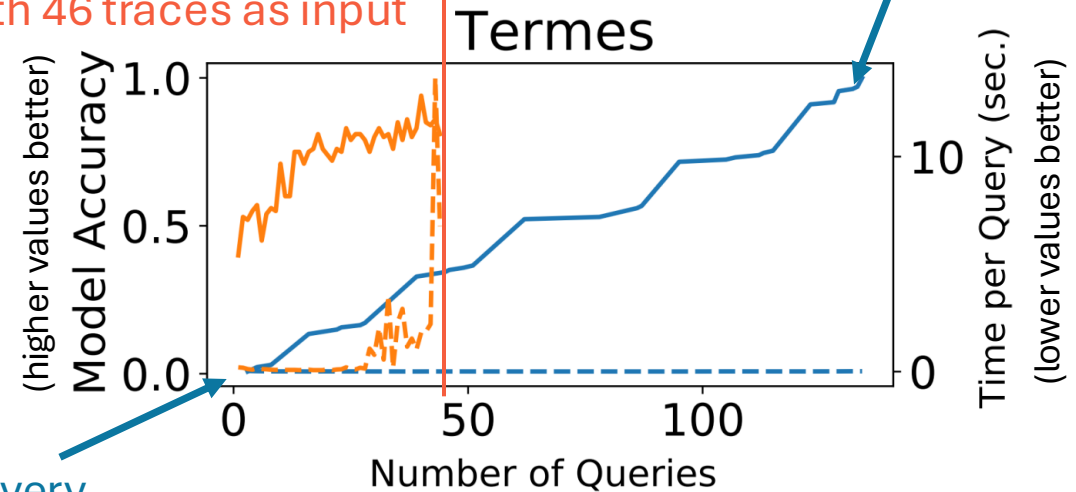
Random deterministic planning agent from IPC

Accuracy: — AAM — FAMA
Time: - - - AAM - - - FAMA



FAMA ran out of memory with 46 traces as input

AAM learned the correct model with 134 queries



AAM takes very less time

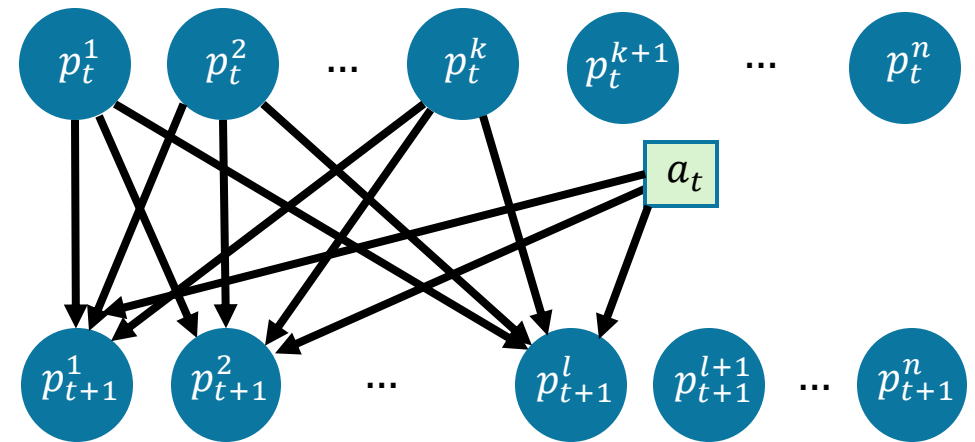
[Verma, Marpally, Srivastava; AAAI '21]

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[†].



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.

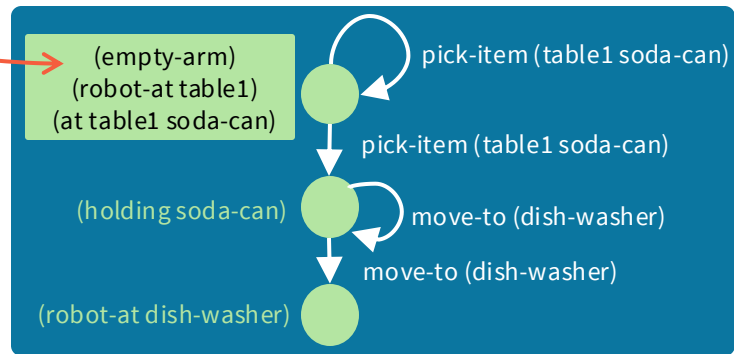
Assumptions

- 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

Initial State



*Policy: Generated Autonomously
by Reduction to Non-Deterministic
Planning*

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

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

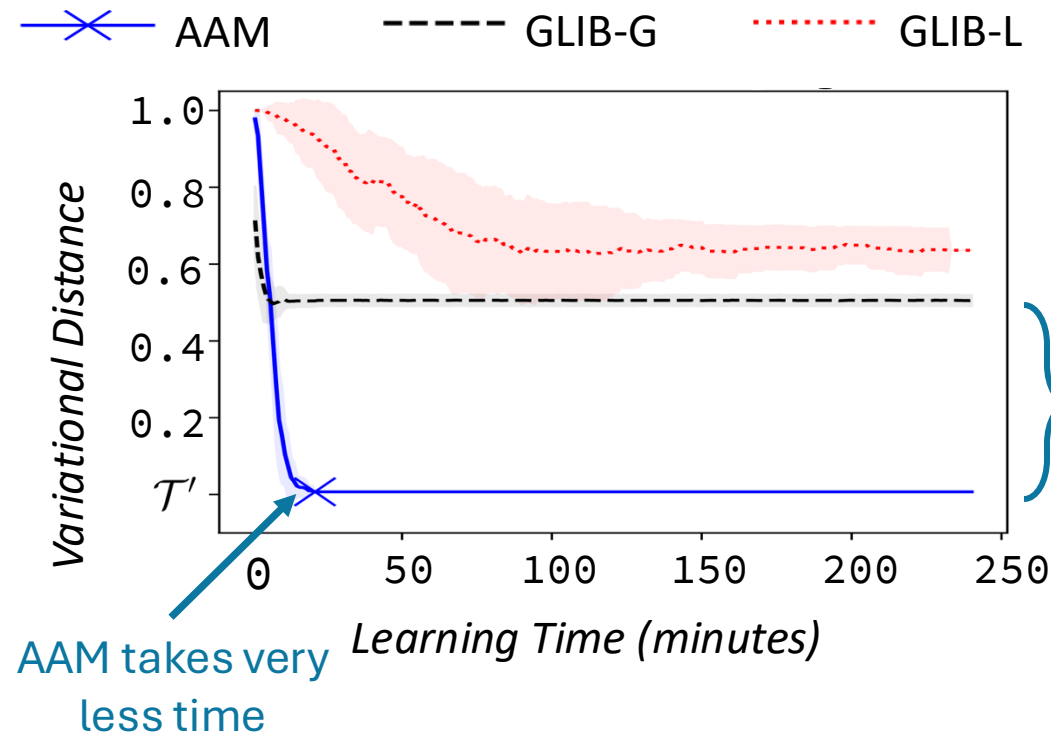
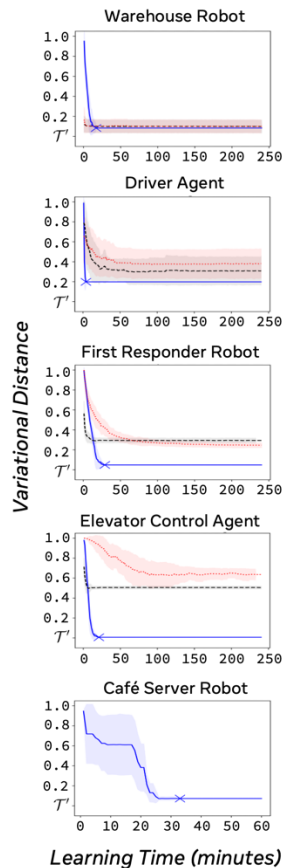
```
(:action open-door
:parameters (?l1)
:precondition (and
  (+/-/∅) (has_key)
  (+/-/∅) (door_open)
  (+/-/∅) (door_adjacent ?l1)
  (+/-/∅) (player_at ?l1))
:effect (probabilistic
  0.xx (and
    (+/-/∅) (has_key)
    (+/-/∅) (door_open)
    (+/-/∅) (door_adjacent ?l1)
    (+/-/∅) (player_at ?l1))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

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



AAM learns a much better model than GLIB

[Verma, Karia, Srivastava; NeurIPS '23]

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.



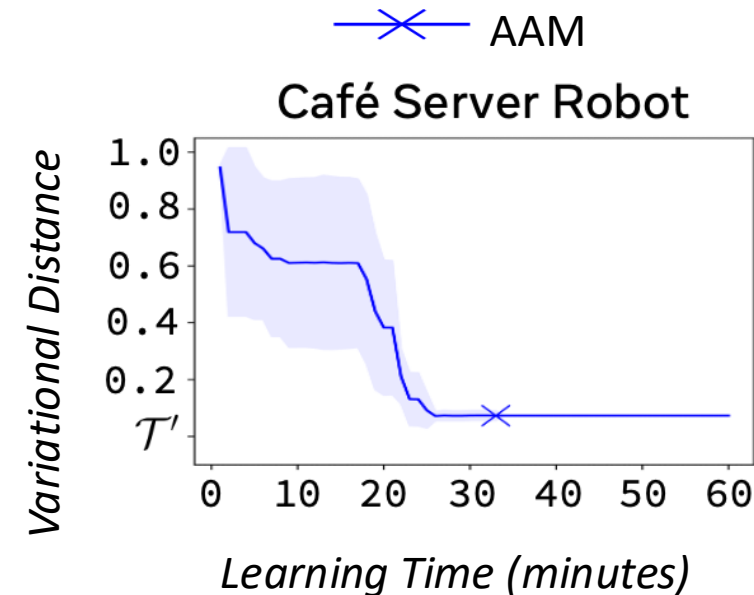
Probabilistic planning agent
using OpenRave and TMP



	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



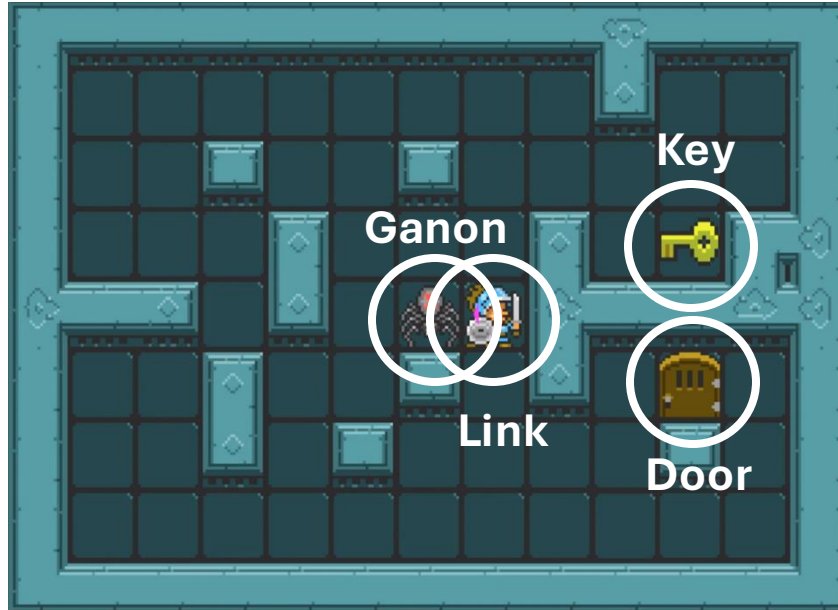
(empty-arm)
(robot-at table1)
(at table1 soda-can)



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

pixel_1_1(#42A8B3)
pixel_1_2(#42A8B3)
.
.
.
pixel_n_m(#203A3D)

State Representation in User's Vocabulary

(at ganon 5,3)
(at link 6,3)
(at key 9,4)
(at door 9,2)

Discovering Capabilities

Input

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



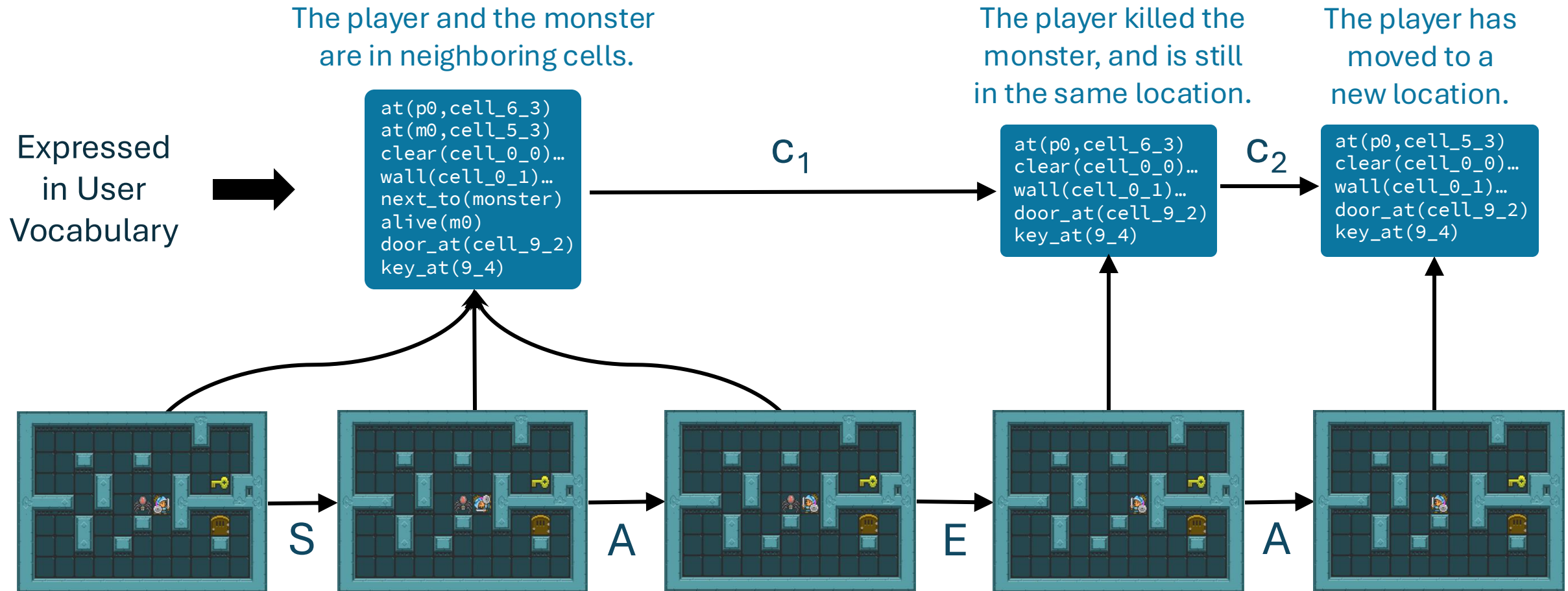
Output

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

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



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

Effects Preconditions

4. Capability C4:

The *player* can execute this capability when:

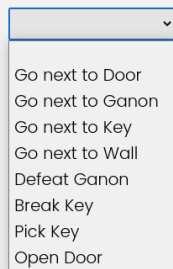
- The *monster* is not defeated.
- The *player* is in *cell1*.
- The *monster* is in *cell2*.
- The *player* is in a cell adjacent to the *monster*.

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.



A dropdown menu with a downward arrow icon. The menu is open, showing a list of options: Go next to Door, Go next to Ganon, Go next to Key, Go next to Wall, Defeat Ganon, Break Key, Pick Key, and Open Door.

Possible options
to choose from

[Capability Description Example]

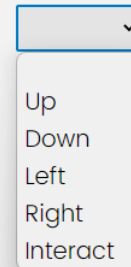
Keystroke Description

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.



A dropdown menu with a downward arrow icon. The menu is open, showing a list of options: Up, Down, Left, Right, and Interact.

Possible options
to choose from

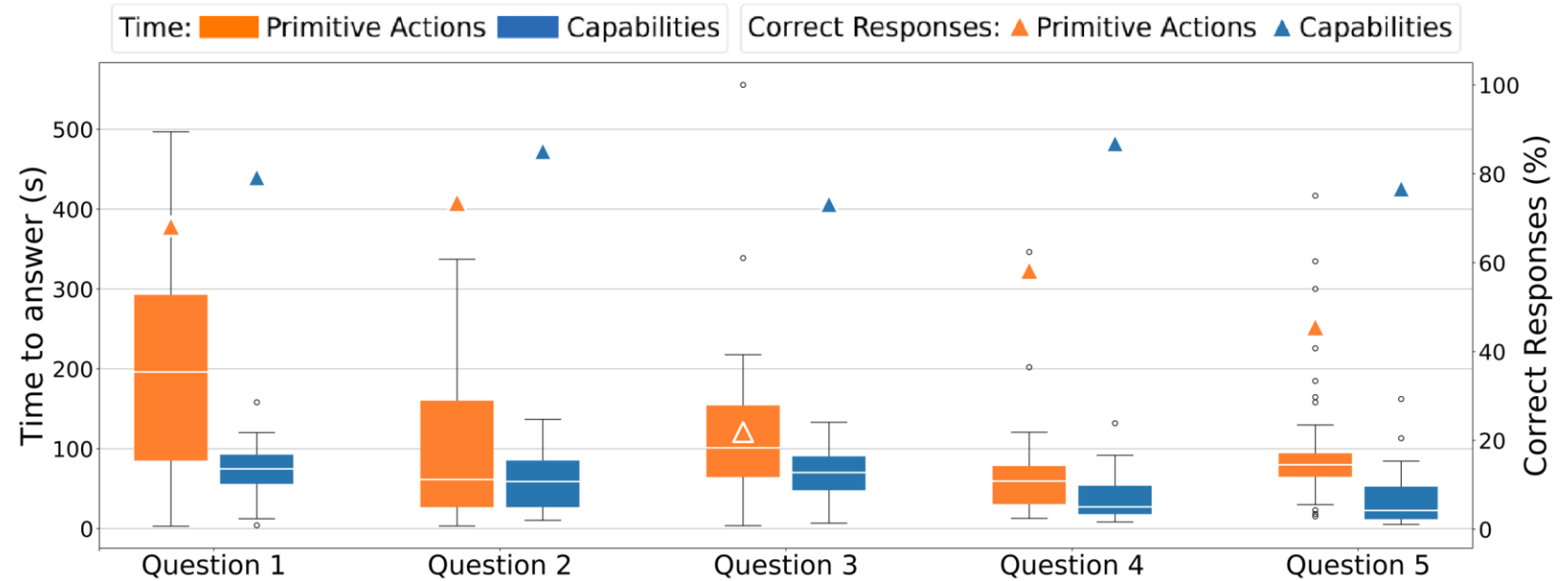
[Functionality Description Example]

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:



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.

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?

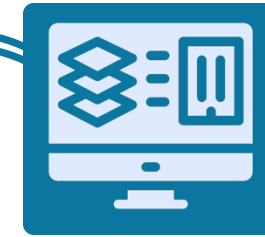
Assumptions

- User's vocabulary matches simulator's vocabulary.
- Black-Box AI provides a list of capabilities.
- ~~Stationary~~ ^{Adaptive} 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

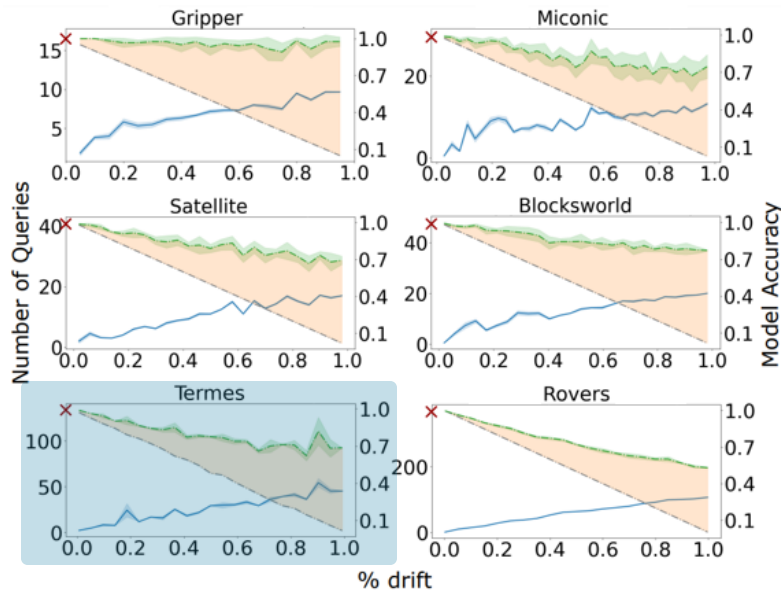


Fewer Queries Needed Compared to Learning from Scratch

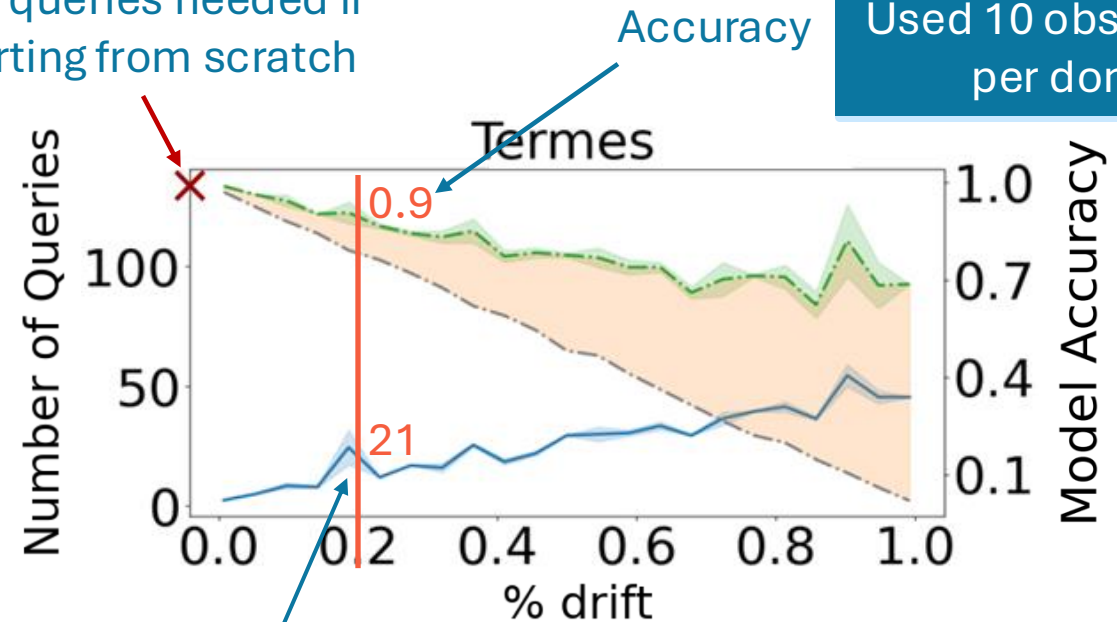


Random deterministic planning agent from IPC

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



134 queries needed if starting from scratch



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.

01

Introduction

02

Foundational
Approach

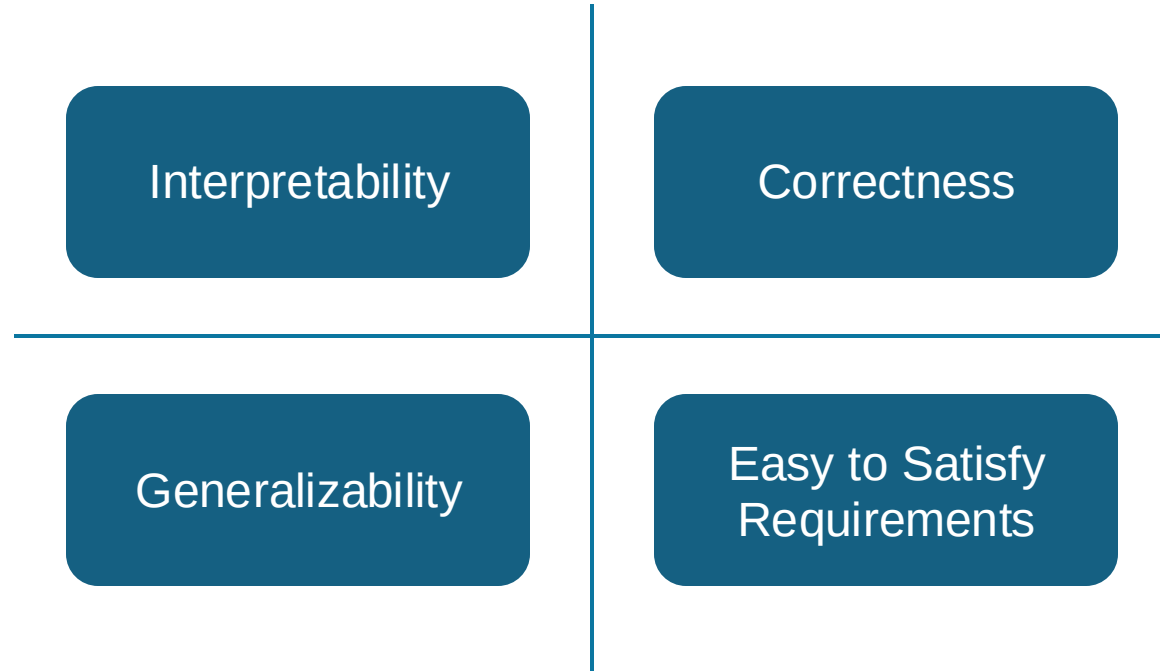
03

Generalizations

04

Conclusion and
Future Work

Desiderata of Assessment System



How well we did on Desiderata?

Interpretability

We showed with a user study that the discovered capability models and their descriptions we learn are interpretable.

Correctness

We defined how correctness can be measured for each work and proved that we can achieve it.

Generalizability

The approaches are applicable to any taskable AI that satisfies the given assumptions for each approach.

Easy to satisfy requirements

The requirements on the AI system are:

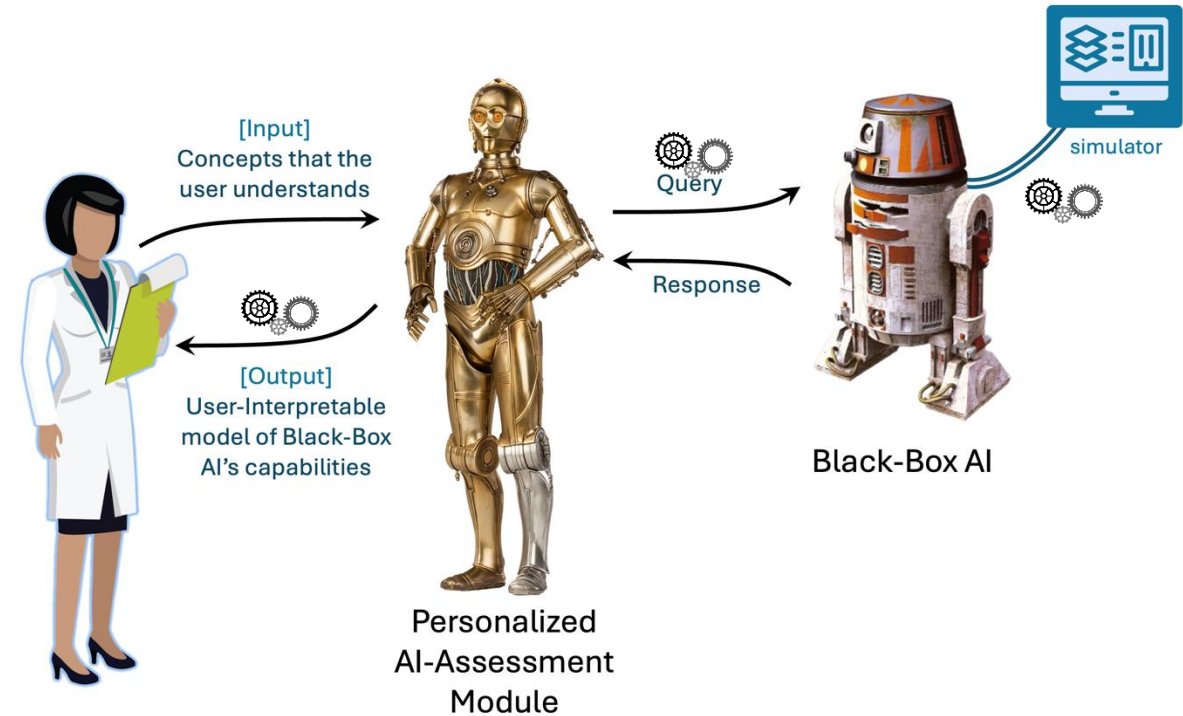
- Simulator access.
- Support for simple queries available to any SDM system.

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.

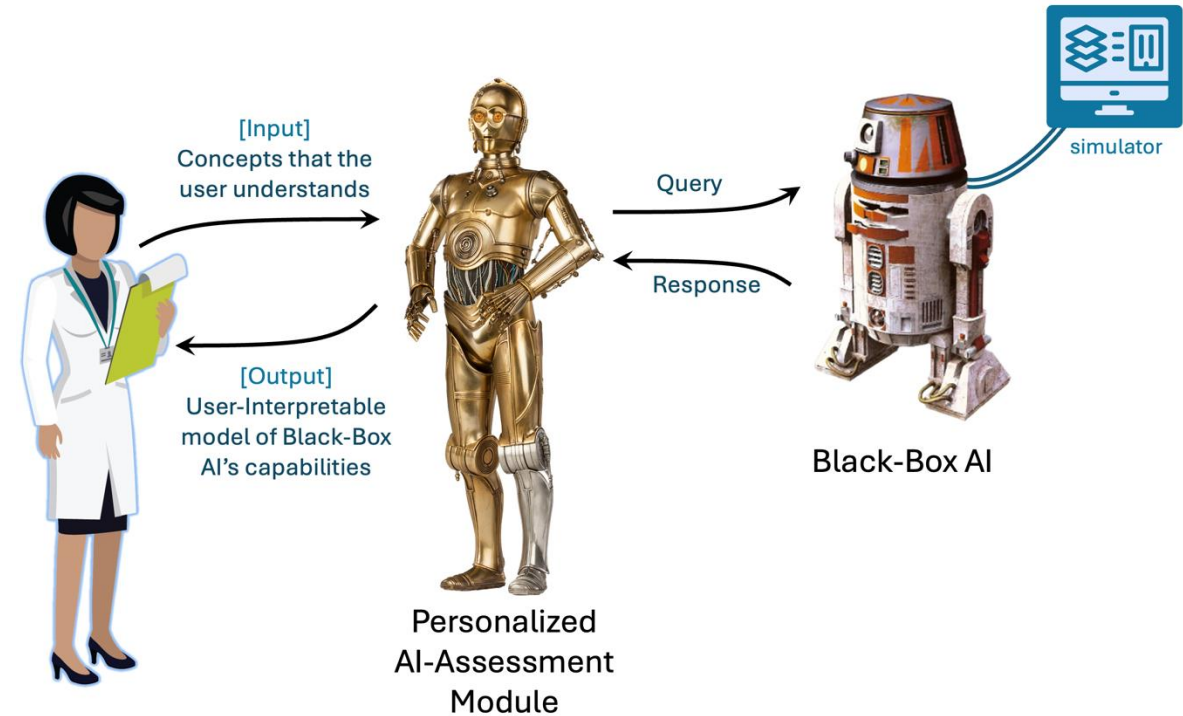
Future Work: Complexity Analysis

- 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.



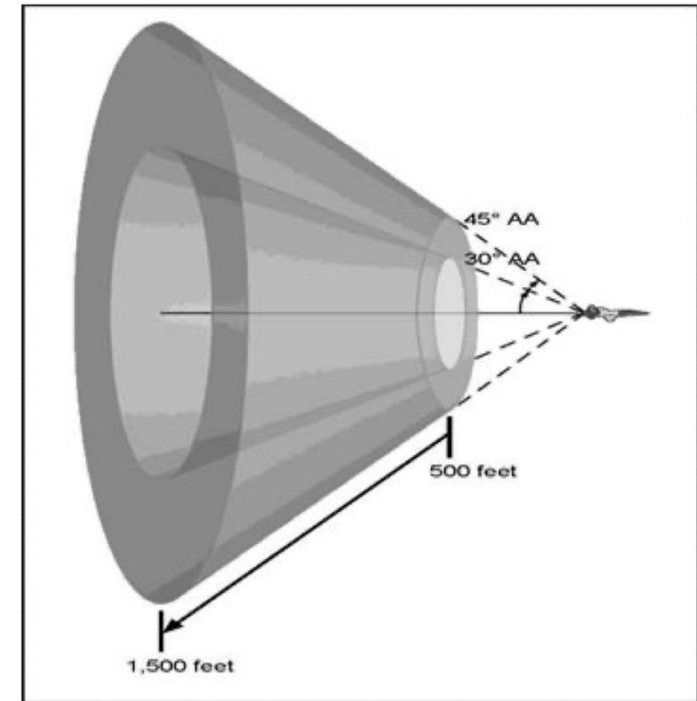
Future Work: Interpretable Representations

- Representations beyond PDDL.
 - LTL – Linear Temporal Logic
 - STL- Signal Temporal Logic
 - CDL- Crow Definition Language
- How interpretable each representation is?



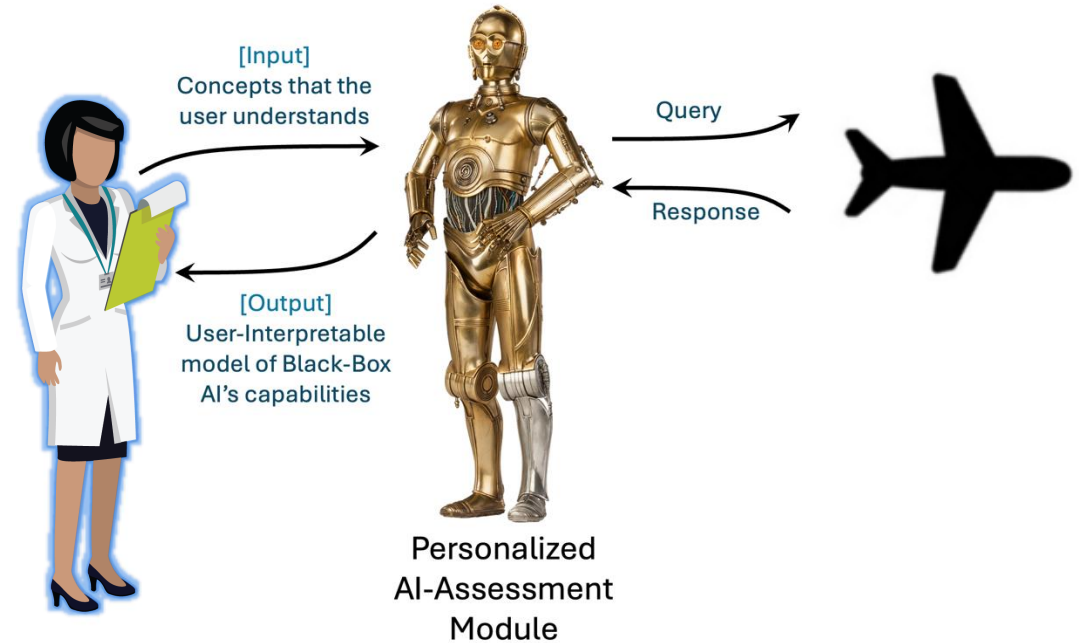
Future Work: Real World Setting

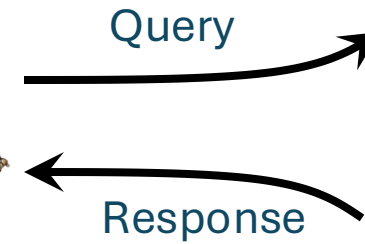
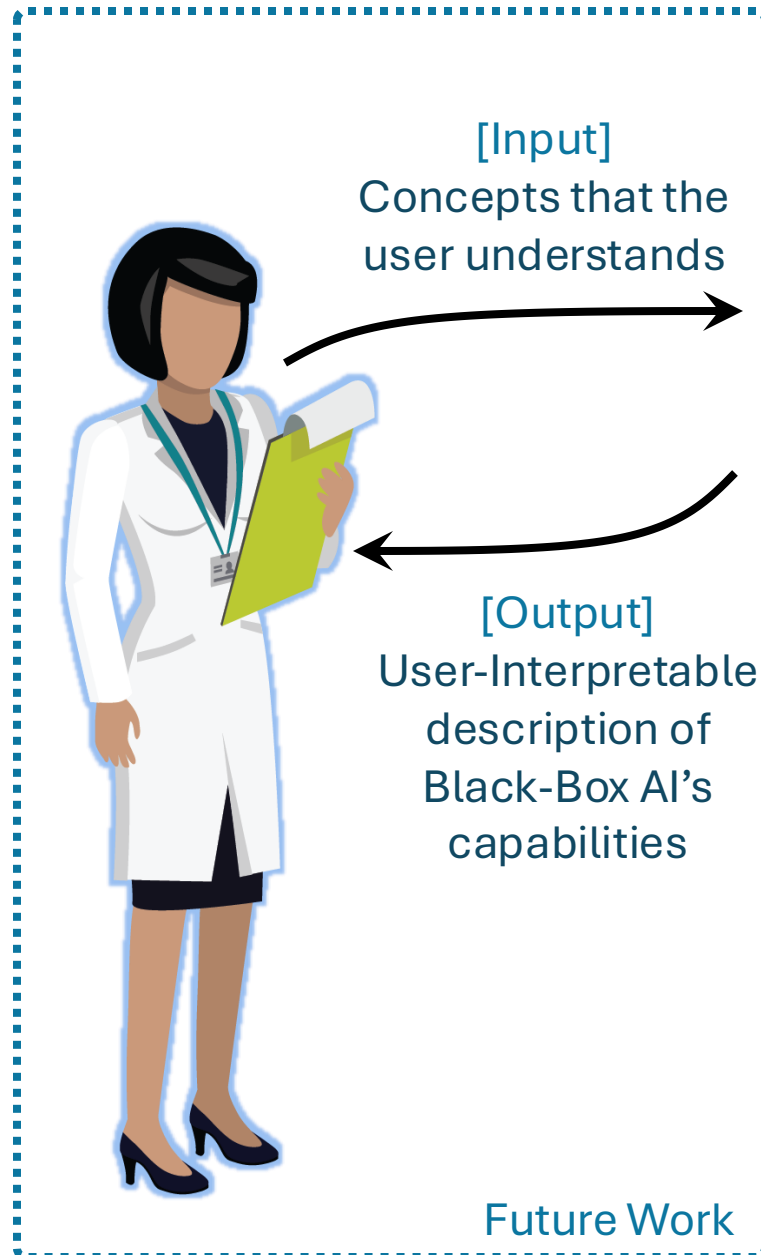
- Flying Extended Trail
- Human must collaborate with the AI-powered aircraft.



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?

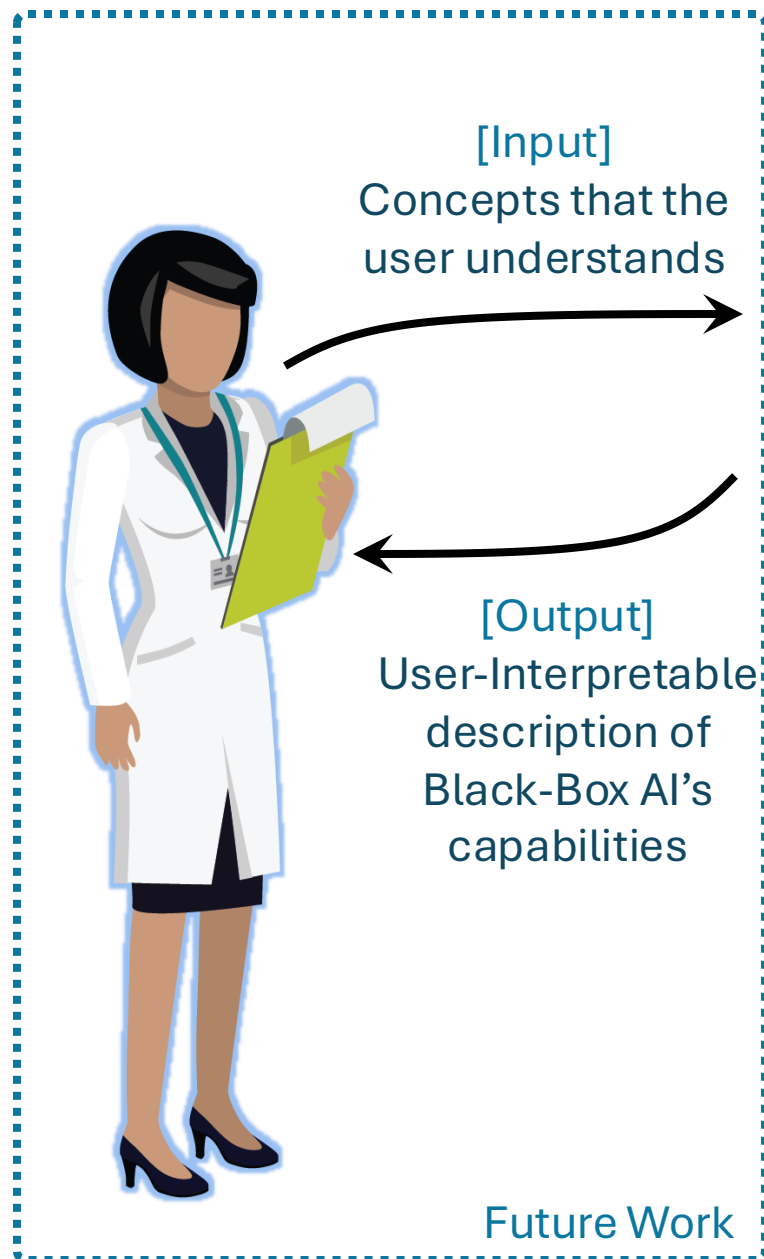




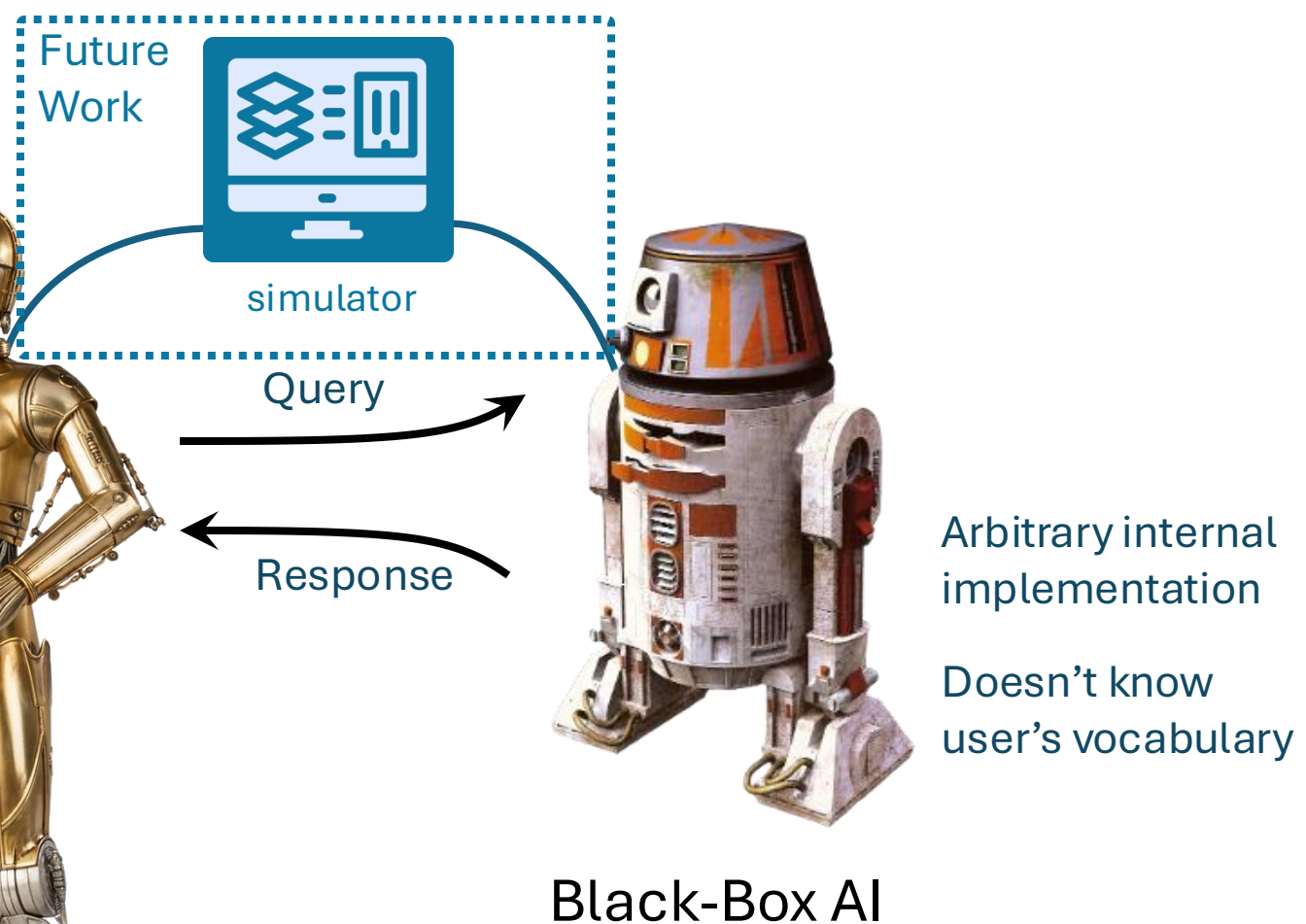
simulator

Arbitrary internal
implementation

Doesn't know
user's vocabulary



Personalized
AI-Assessment
Module (AAM)





Rushang Karia



Shashank Marpally



Rashmeet Nayyar



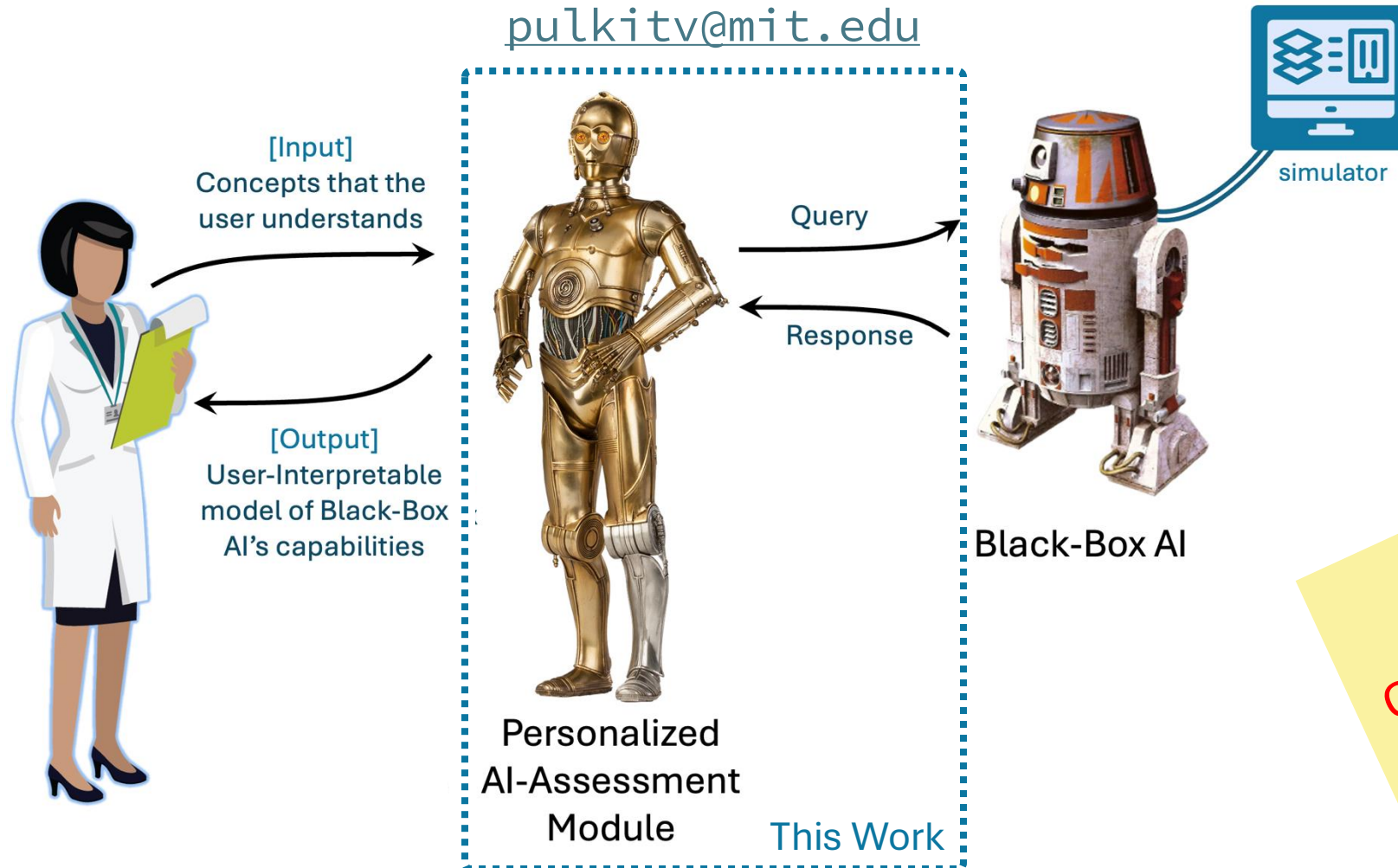
Siddharth Srivastava



Learning Interpretable Models for Personalized Assessment of Taskable AI Systems

Pulkit Verma

pulkitv@mit.edu



Questions?