AAAI 2024 Spring Symposium on

User-Aligned Assessment of Adaptive AI Systems

User-Aligned Autonomous Capability Assessment of Black-Box AI Systems

Pulkit Verma and Siddharth Srivastava

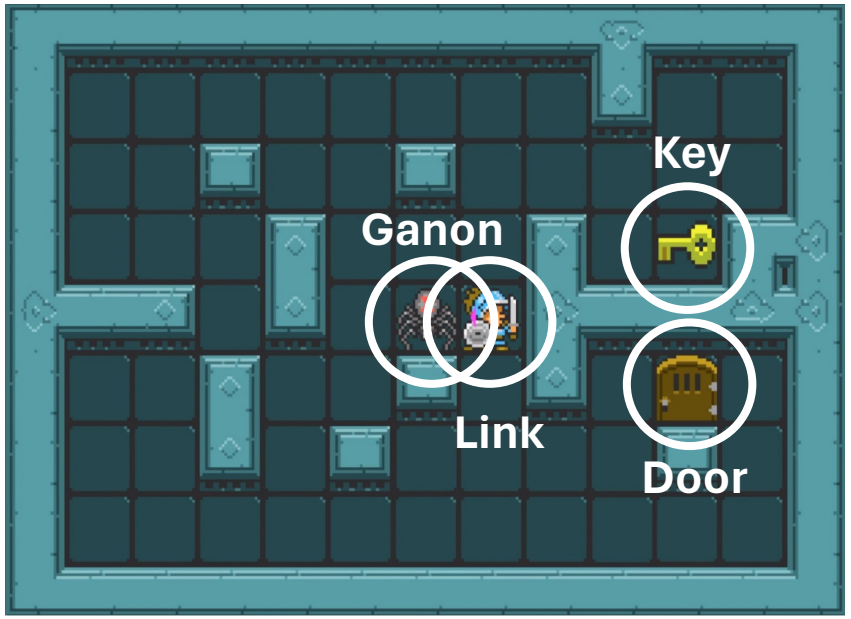
Focus: Taskable AI Systems

- Sequential Decision-Making Systems.
- User gives AI systems a task (an objective).
- AI system needs to figure out how to complete that task.

Personalized Assessment of Adaptive AI Systems

- 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 black-box AI systems.

Capability vs Functionality



Agent Functionality (Keystrokes)	Capabilities such as..
W	Defeat Ganon
A	Go to door
S	Go to key
D	Go to Ganon
E	Pick key
	Open door

Easier to reason in terms of capabilities than in terms of functionalities

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

Query

Response

Arbitrary internal implementation

Black-Box AI

Personalized AI-Assessment Module

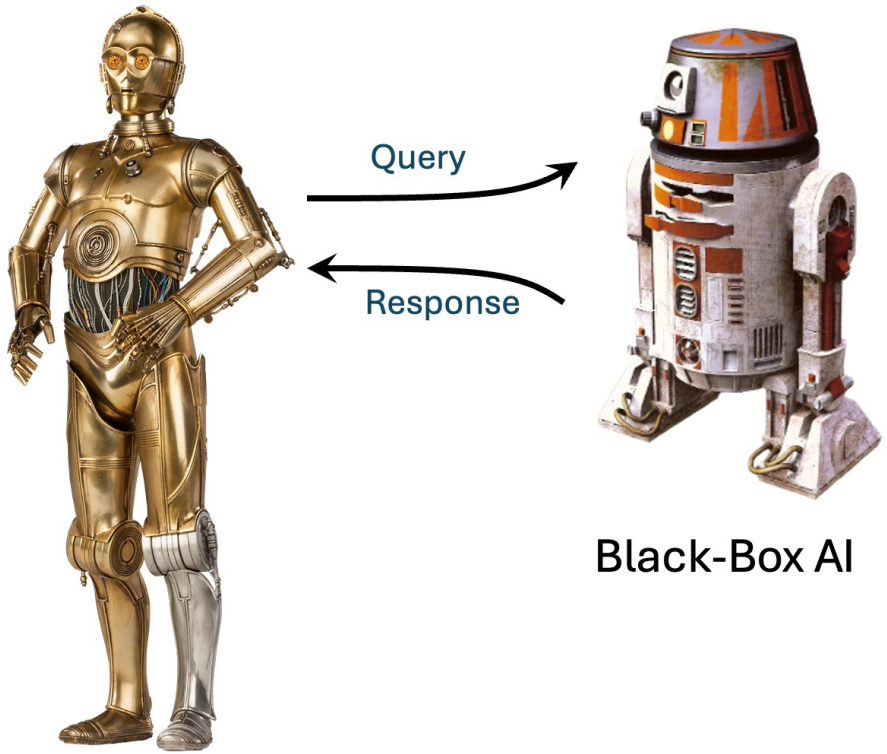
Query

Response

Black-Box AI

Arbitrary internal implementation

Black-Box AI System Interface



- Should work for a variety of taskable AI systems
- Should be easy to support

	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.

```
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

Query

Response

Black-Box AI

Arbitrary internal implementation

Doesn't know user's vocabulary

[Input]
Concepts that the user understands

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

Personalized AI-Assessment Module

Query

Response

Black-Box AI

Arbitrary internal implementation

Doesn't know user's vocabulary

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:

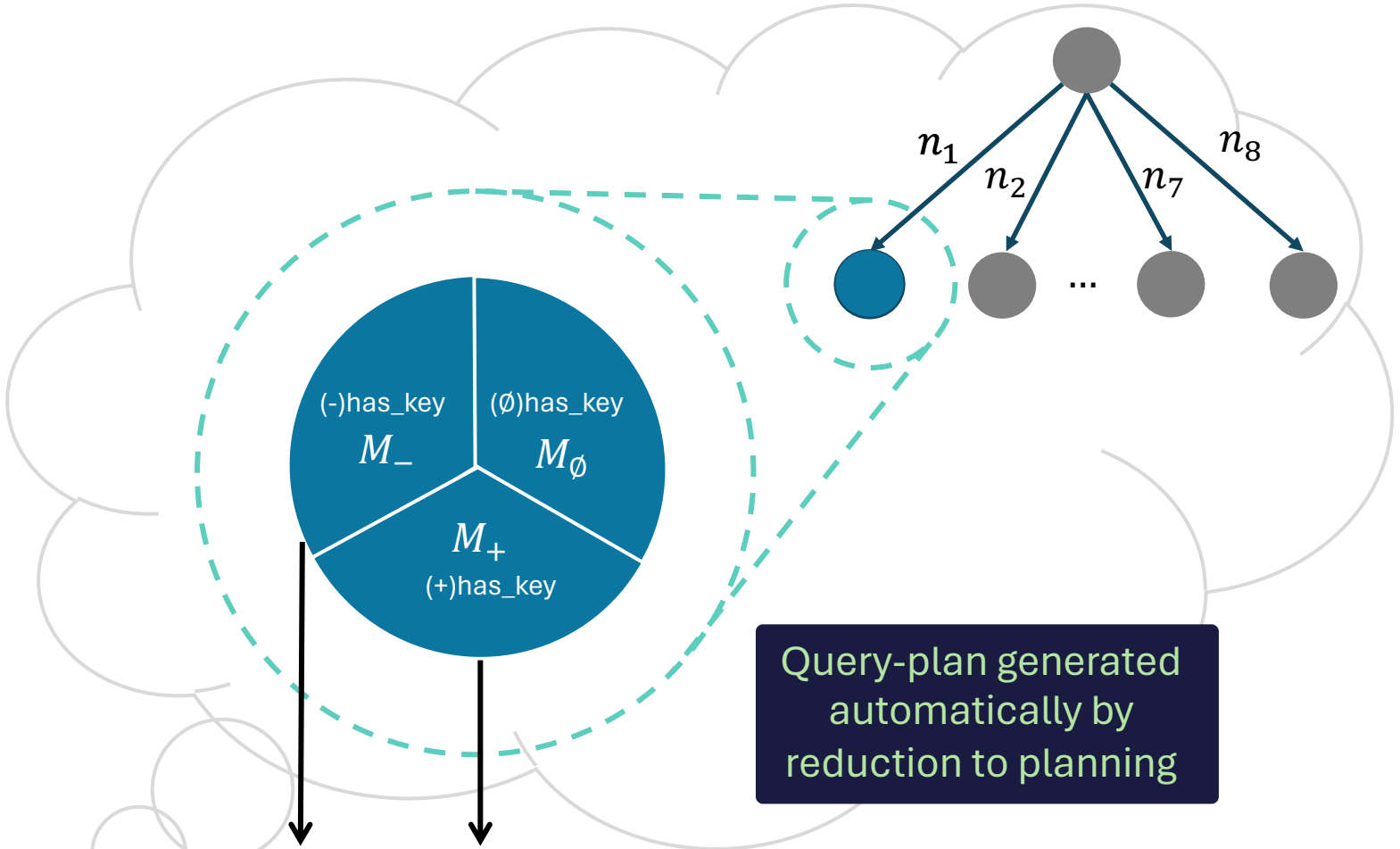
- With 95% probability, the door will open
- With 5% probability, the player will not have the key and the game will be over

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))
```

Hierarchical Query Synthesis



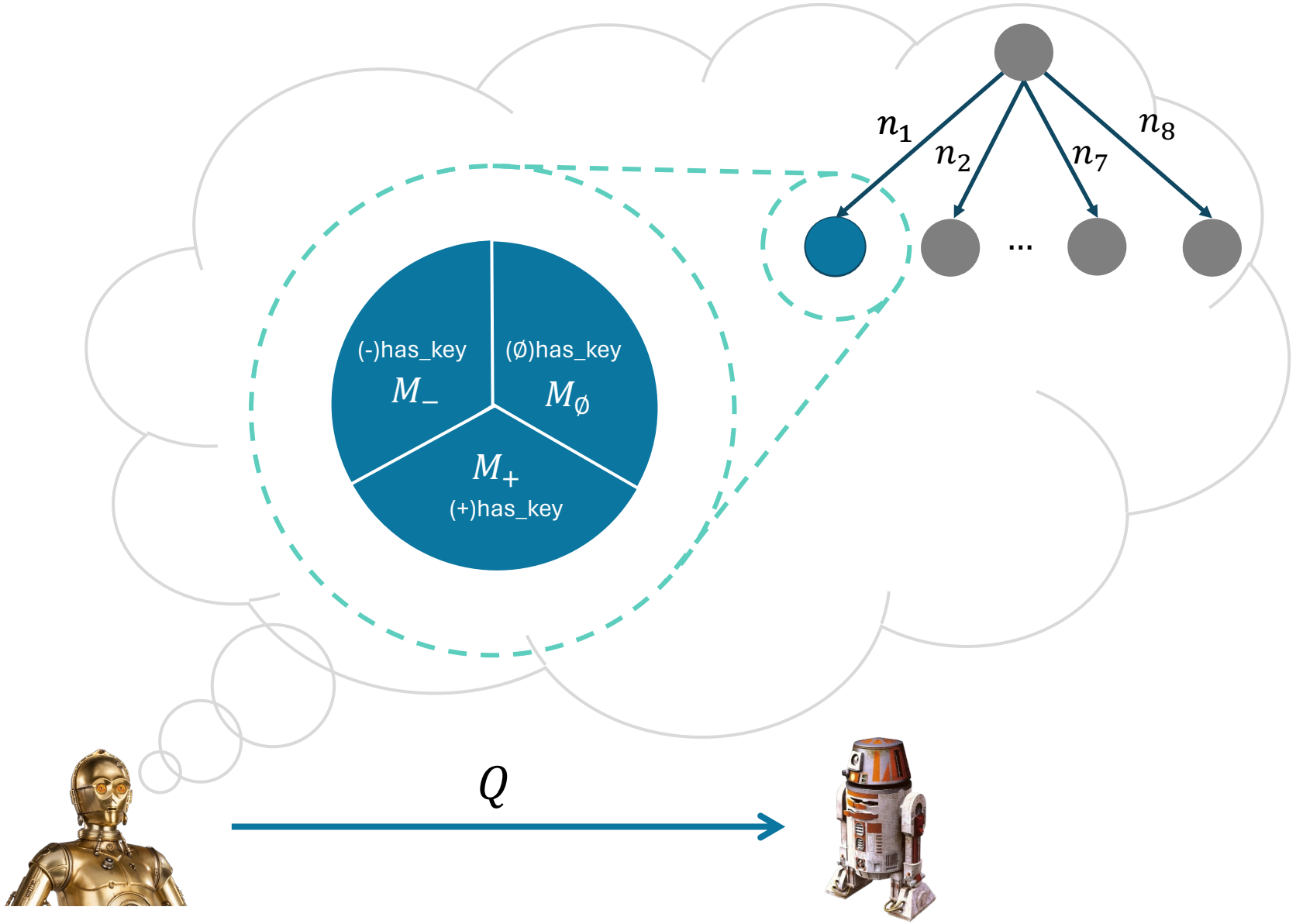
Query-plan generated automatically by reduction to planning

Generate a
distinguishing query:
Q such that $Q(M_-) \neq Q(M_+)$

```
(: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]

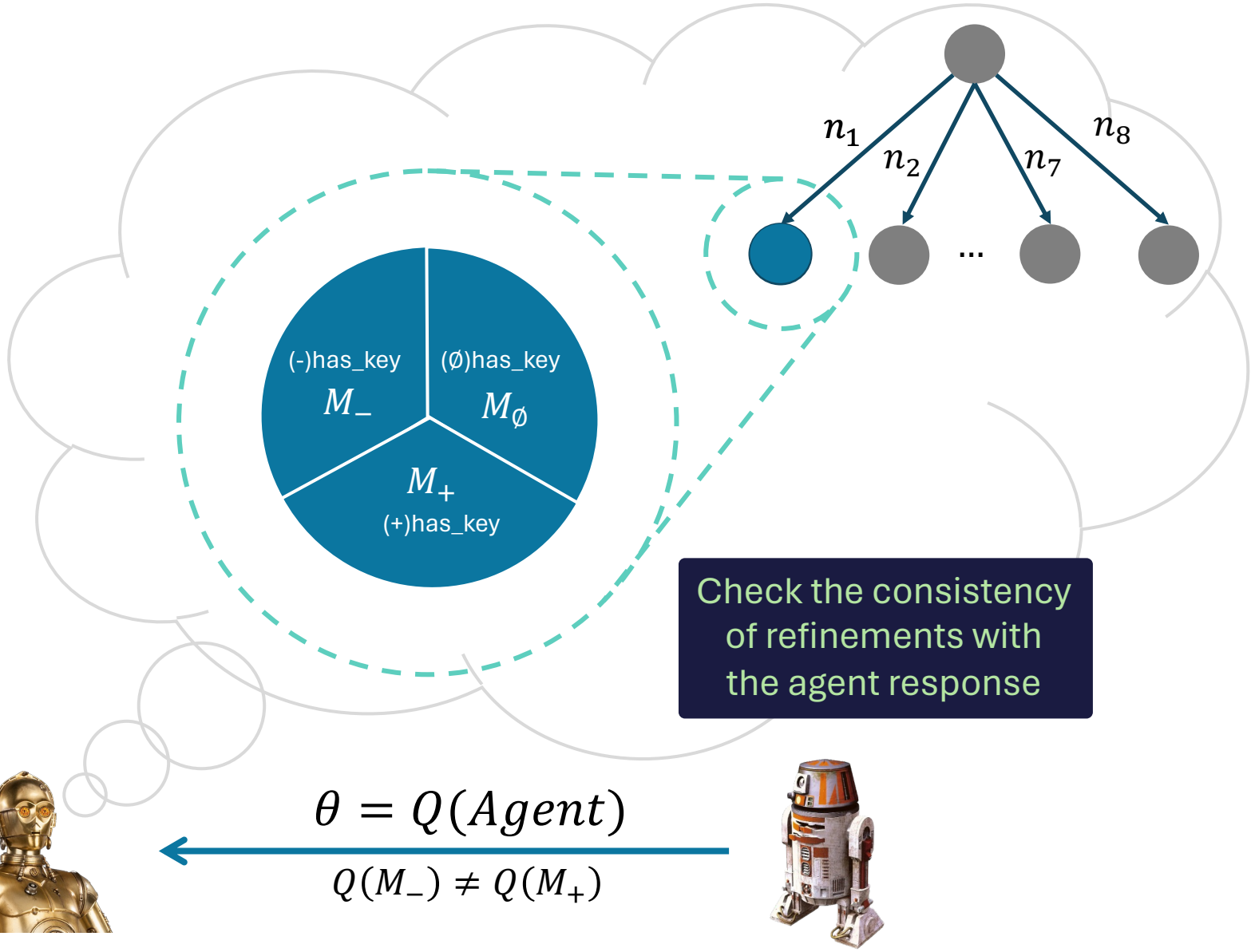
Hierarchical Query Synthesis




```
(: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]

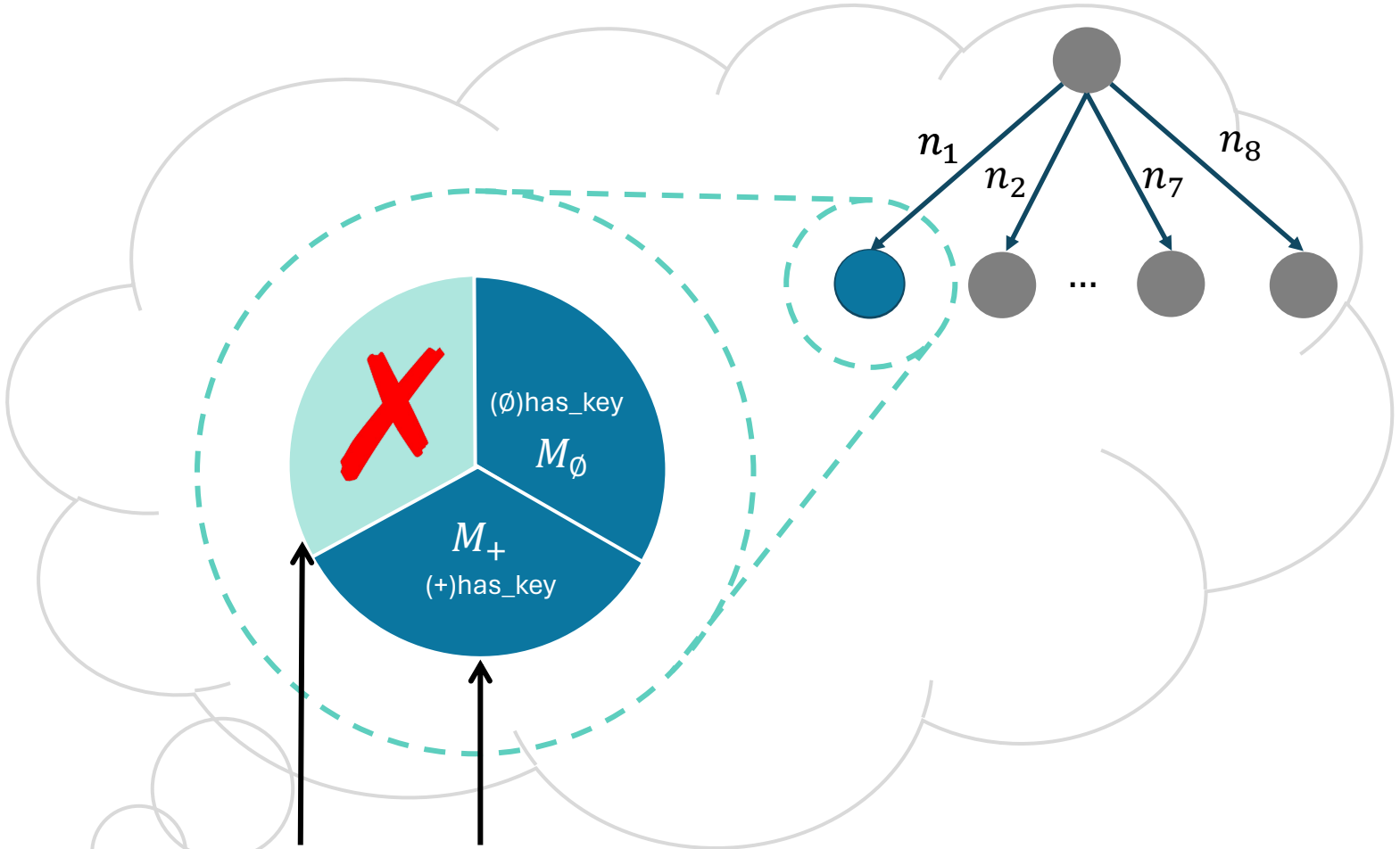
Hierarchical Query Synthesis




```
(:action open-door
  :parameters (?l1)
  :precondition (and
n1 (+/-/∅)(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

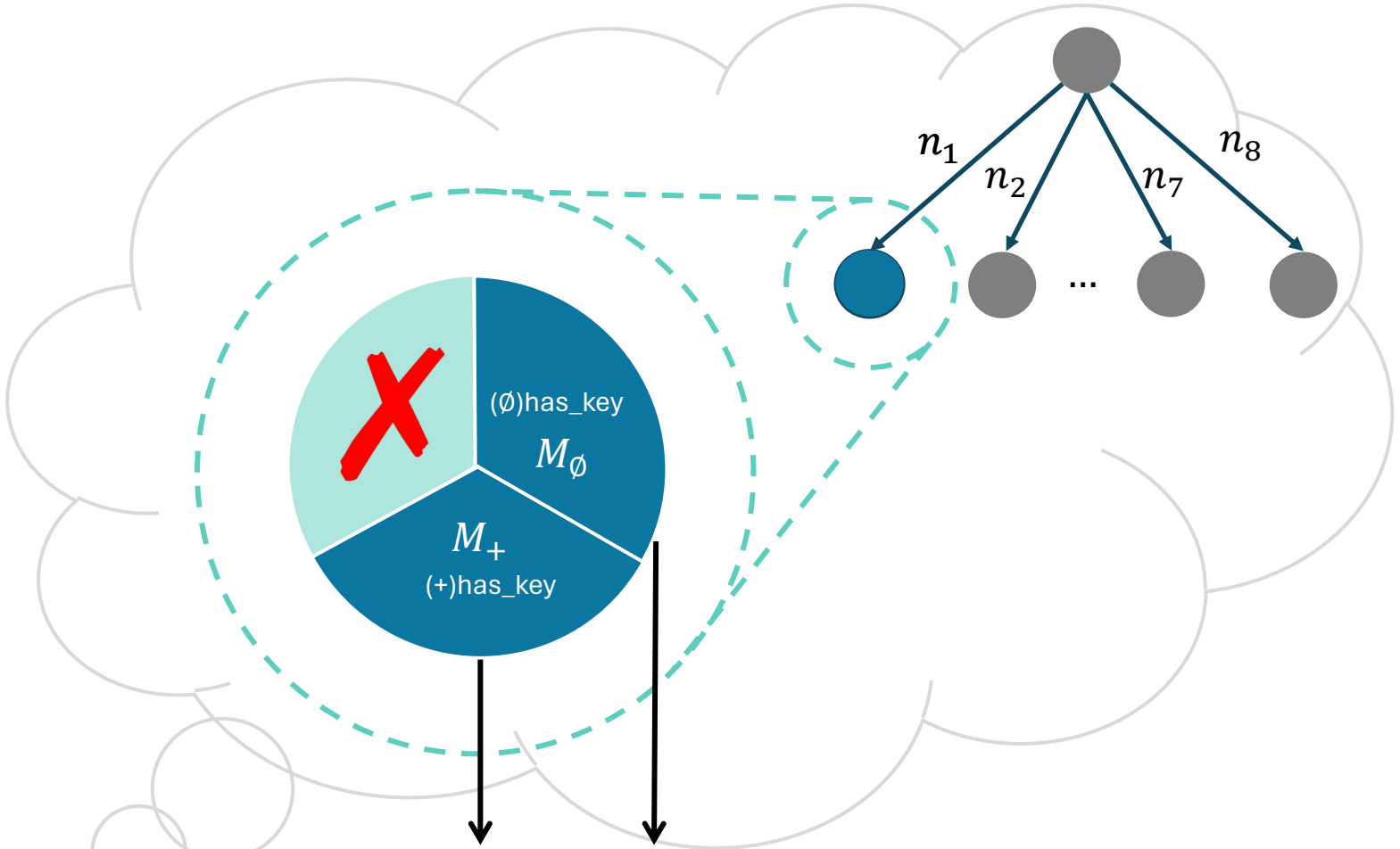


Reject refinement(s) that are not consistent with the agent

```
(: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

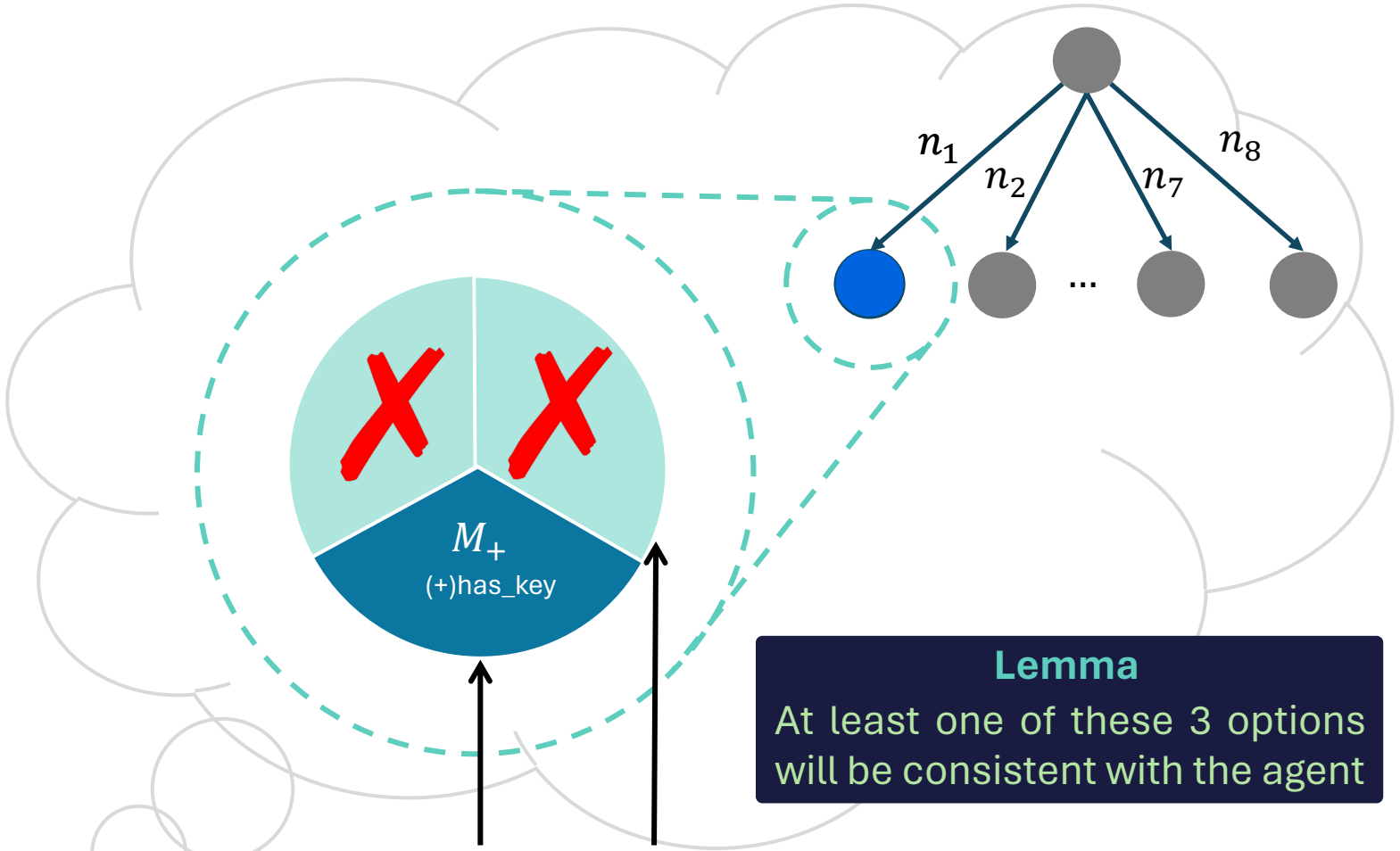



```
(:action open-door
  :parameters (?l1)
  :precondition (and
n1   (+/Ø)(has_key)
n2   (+/-/Ø)(door_open)
n3   (+/-/Ø)(door_adjacent ?l1)
n4   (+/-/Ø)(player_at ?l1))
  :effect (and
n5   (+/-/Ø)(has_key)
n6   (+/-/Ø)(door_open)
n7   (+/-/Ø)(door_adjacent ?l1)
n8   (+/-/Ø)(player_at ?l1))
```

Generate a distinguishing query for these two refinements

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

Hierarchical Query Synthesis



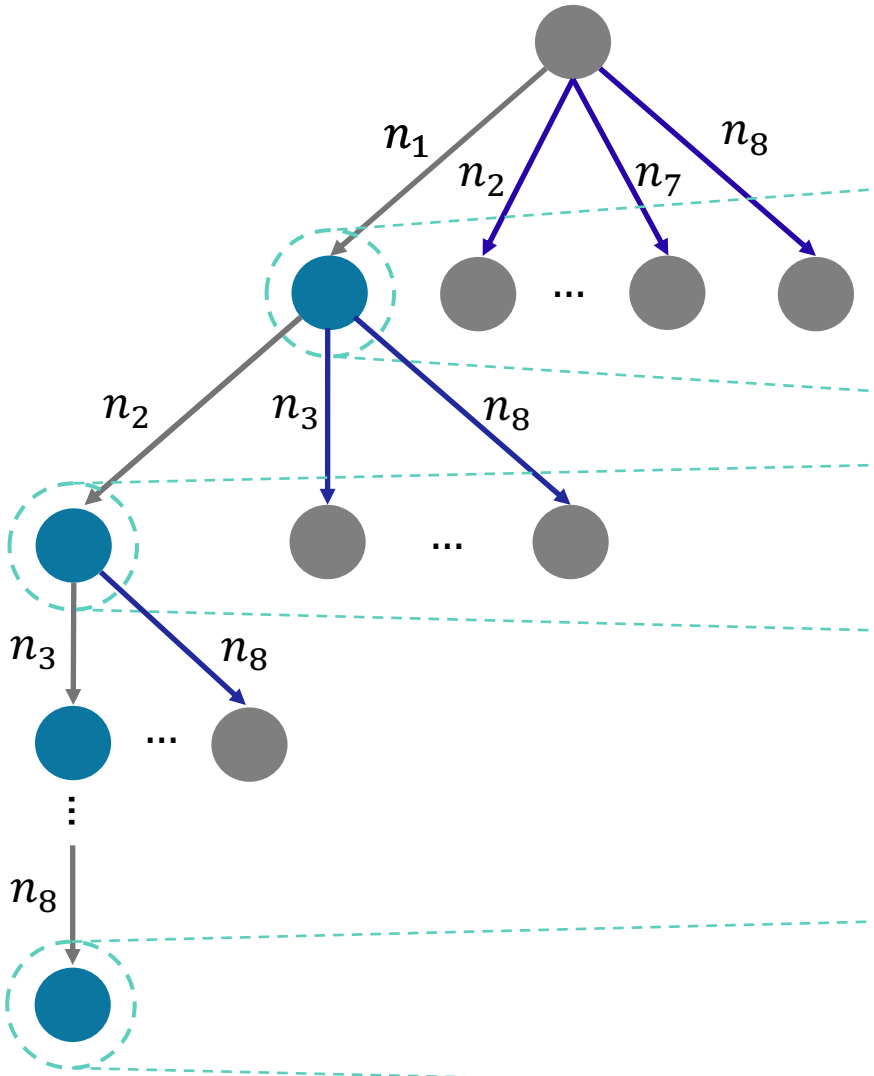
Lemma
At least one of these 3 options will be consistent with the agent

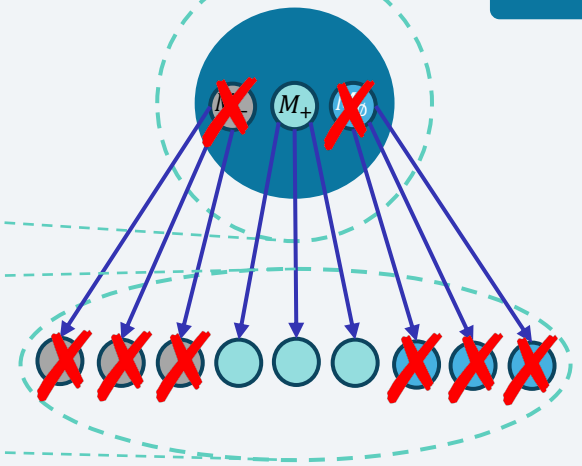
Reject the refinement
that is not consistent with the agent

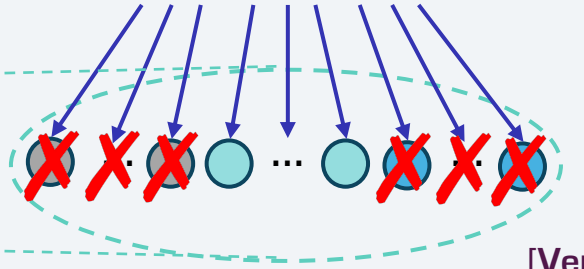
```
(:action open-door
  :parameters (?l1)
  :precondition (and
n1      (+)(has_key)
n2    (+/-/∅)(door_open)
n3    (+/-/∅)(door_adjacent ?l1)
n4    (+/-/∅)(player_at ?l1))
  :effect (and
n5    (+/-/∅)(has_key)
n6    (+/-/∅)(door_open)
n7    (+/-/∅)(door_adjacent ?l1)
n8    (+/-/∅)(player_at ?l1))
```

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

Hierarchical Query Synthesis







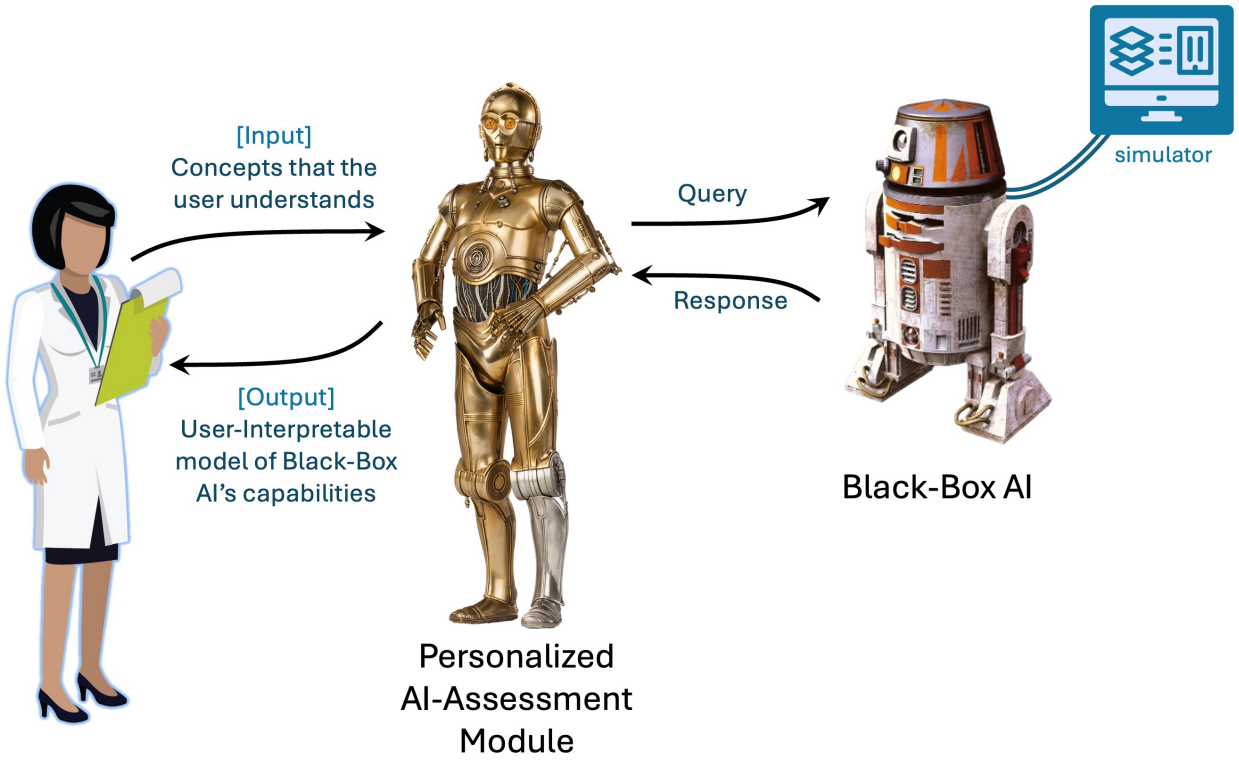
Key feature of the algorithm

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

Active Learning

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

Evaluation with Known Capabilities



Assumptions

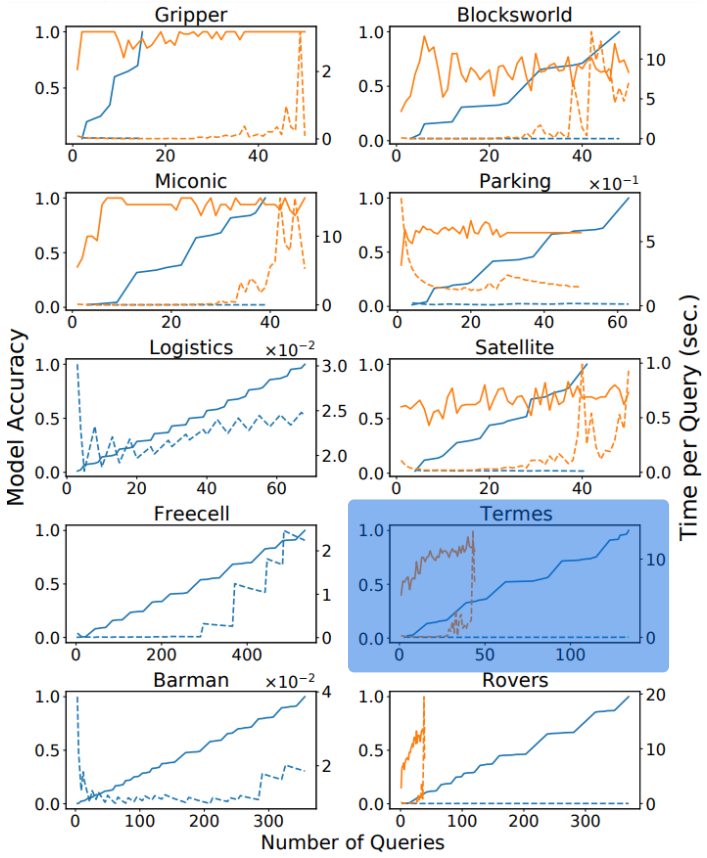
- User's vocabulary matches simulator's vocabulary.
- Black-Box AI provides a list of capabilities.
- Deterministic setting.

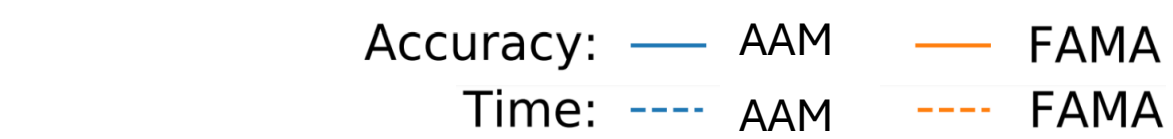
- Randomly generate an agent and environment from International Planning Competition (IPC).
- Evaluate performance of the assessment module and compare it with FAMA†.

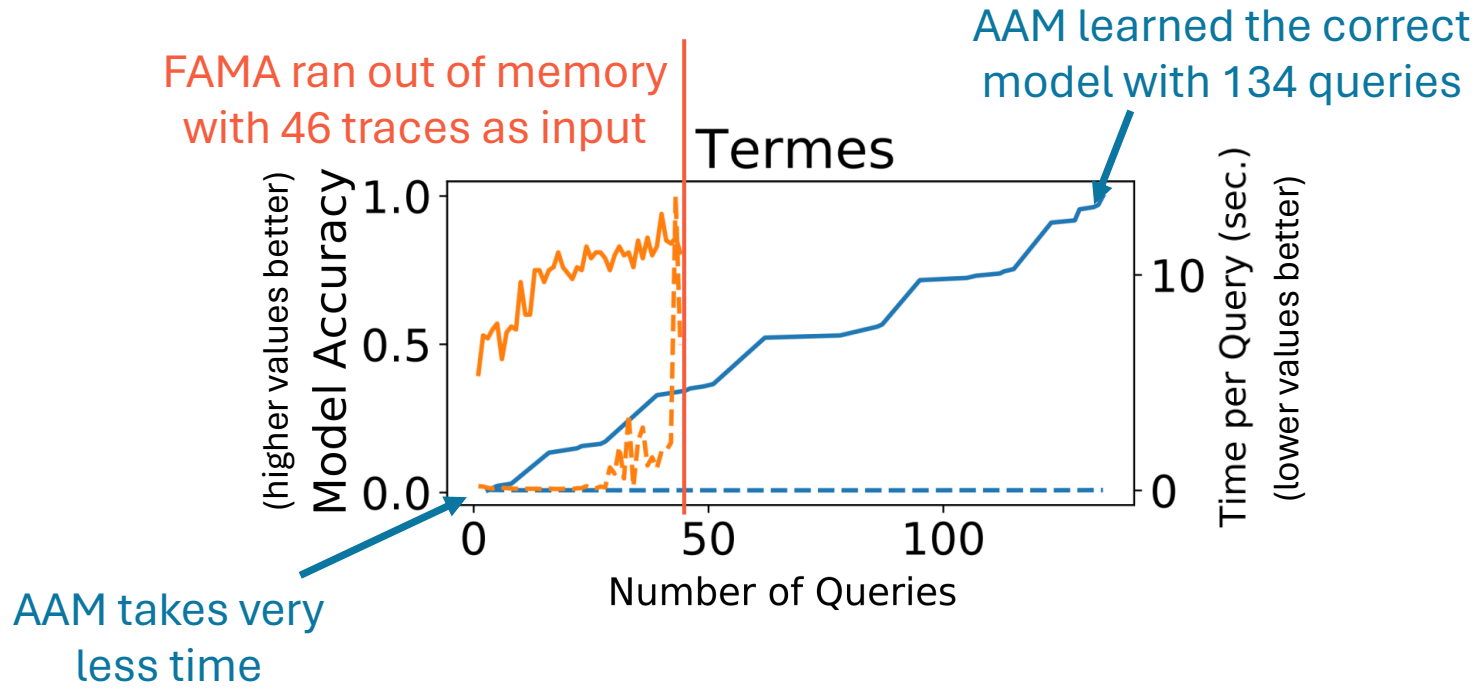
†Aineto, D.; Celorrio, S. J.; and Onaindia, E. 2019. *Learning Action Models With Minimal Observability*. Artificial Intelligence 275: 104–137.

AAM learns Accurate Model with fewer Queries

- Random traces as input to FAMA.
- Increased #traces till it ran out of memory.







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

Capability Discovery

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

Differential Assessment

Stochastic Setting

Capability Discovery

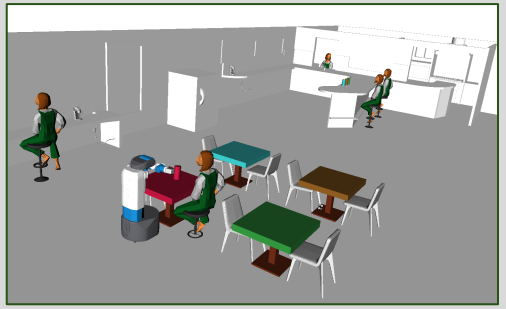
Differential Assessment

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

Stochastic Setting

Capability Discovery

Differential Assessment

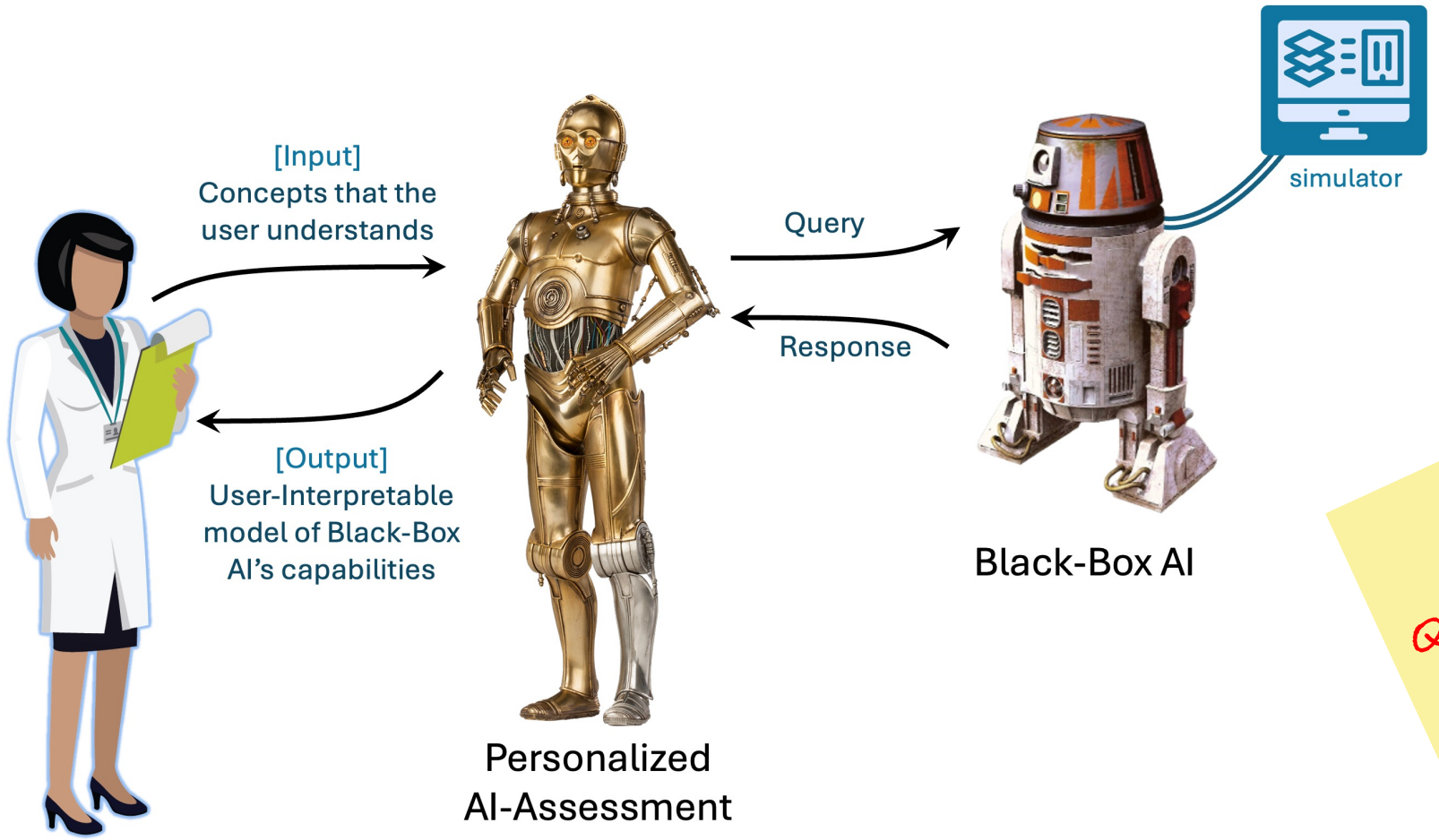


Stochastic Setting

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

User-Aligned Autonomous Capability Assessment of Black-Box AI Systems

Pulkit Verma and Siddharth Srivastava



Questions?