19th International Conference on Principles of Knowledge Representation and Reasoning

Discovering User-Interpretable Capabilities of Black-Box Planning Agents

Pulkit Verma, Shashank Rao Marpally, Siddharth Srivastava

Arizona State University

Pulkit

Shashank

Siddharth

2022 KR

AAIR
Autonomous Agents and Intelligent Robots
ASU Arizona State University

Personalized Assessment of Taskable AI Systems

- Users can give them multiple tasks.
  - How would users know what they can do?

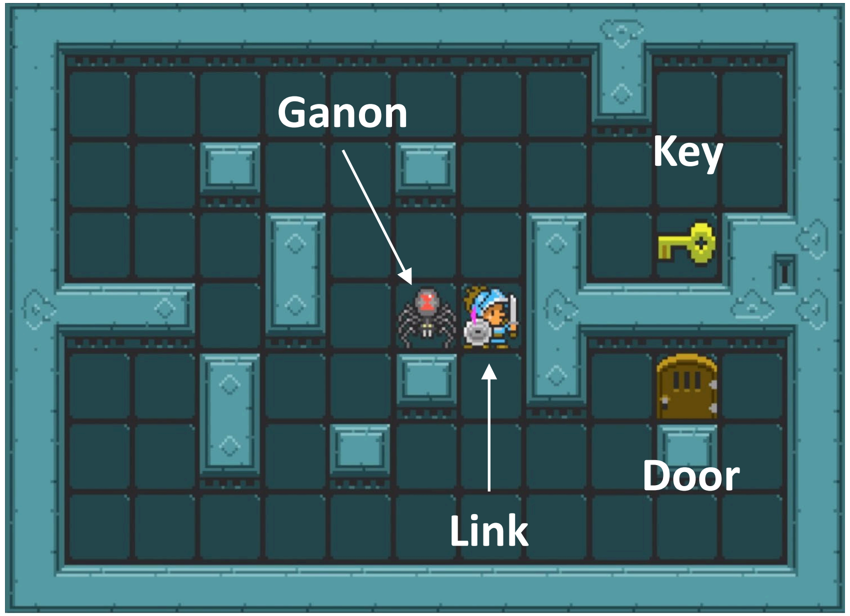
- They should make it easy for its operators to learn how to use them safely.[†]

- Should work with black-box AI systems.

[†]Srivastava S. *Unifying Principles and Metrics for Safe and Assistive AI.* In Proc. AAAI 2021.

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

User-vocabulary may be limited

Agent's State Representation	Interpretable State Representation
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)	

Might be more expressive than what the user understands

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.

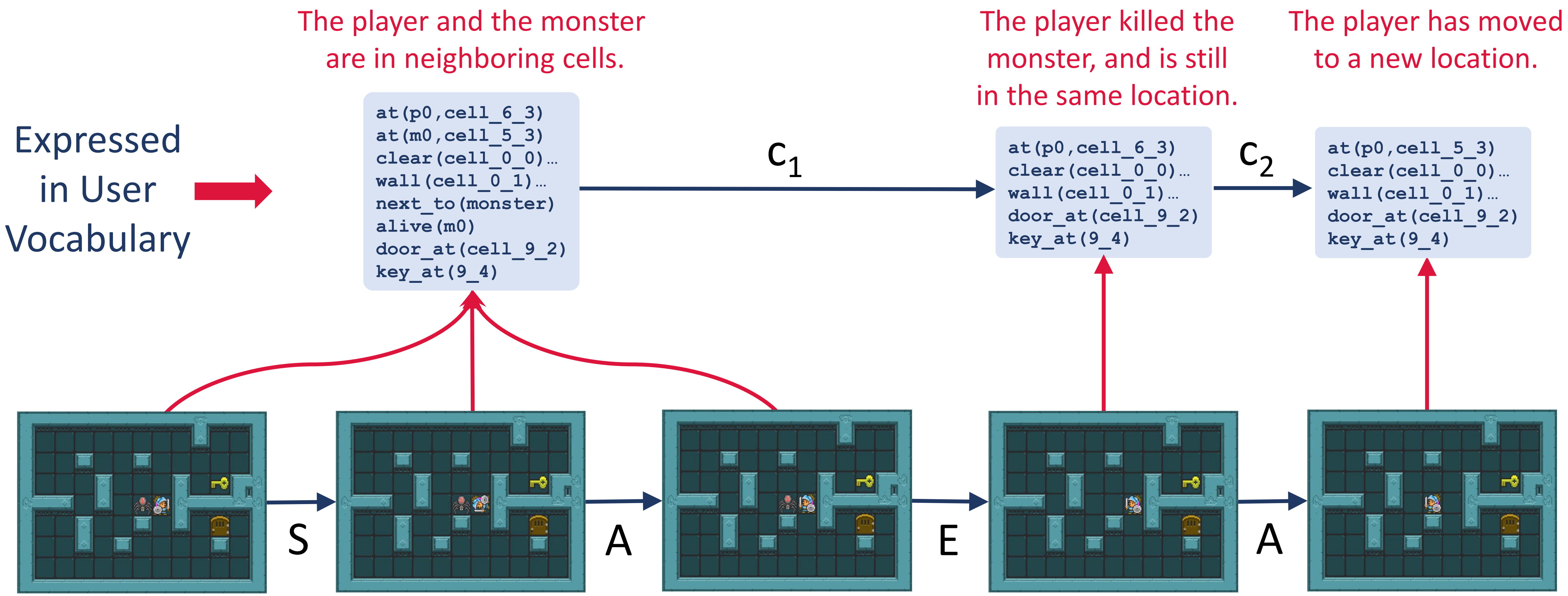
[†]Kim et al. *Interpretability beyond feature attribution: Testing with Concept Activation Vectors*. In Proc. ICML 2018.

Previous Work

Learning high-level symbolic models of AI systems using observations or interventions.

- Konidaris et al. (JAIR’18) : not interpretable, assume access to predefined options
- AIA - Verma et al. (AAAI’21): assume precise user-vocabulary
- Zhang et al. (ICML’18): Needs hand-coding of states
- Schema Networks - Kansky et al. (ICML’17), Agarwal et al. (NIPS’16): require lot of data
- LOCM - Cresswell et al. (ICAPS’09), ARMS - Yang et al. (AIJ 2007), LOUGA - Kučera and Barták (KMAIS 2018), SAM - Stern and Juba (IJCAI 2017, KR 2021), FAMA - Aineto et al. (AIJ 2019) – based on observations, works for known set of operators

Discovering Capabilities



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]

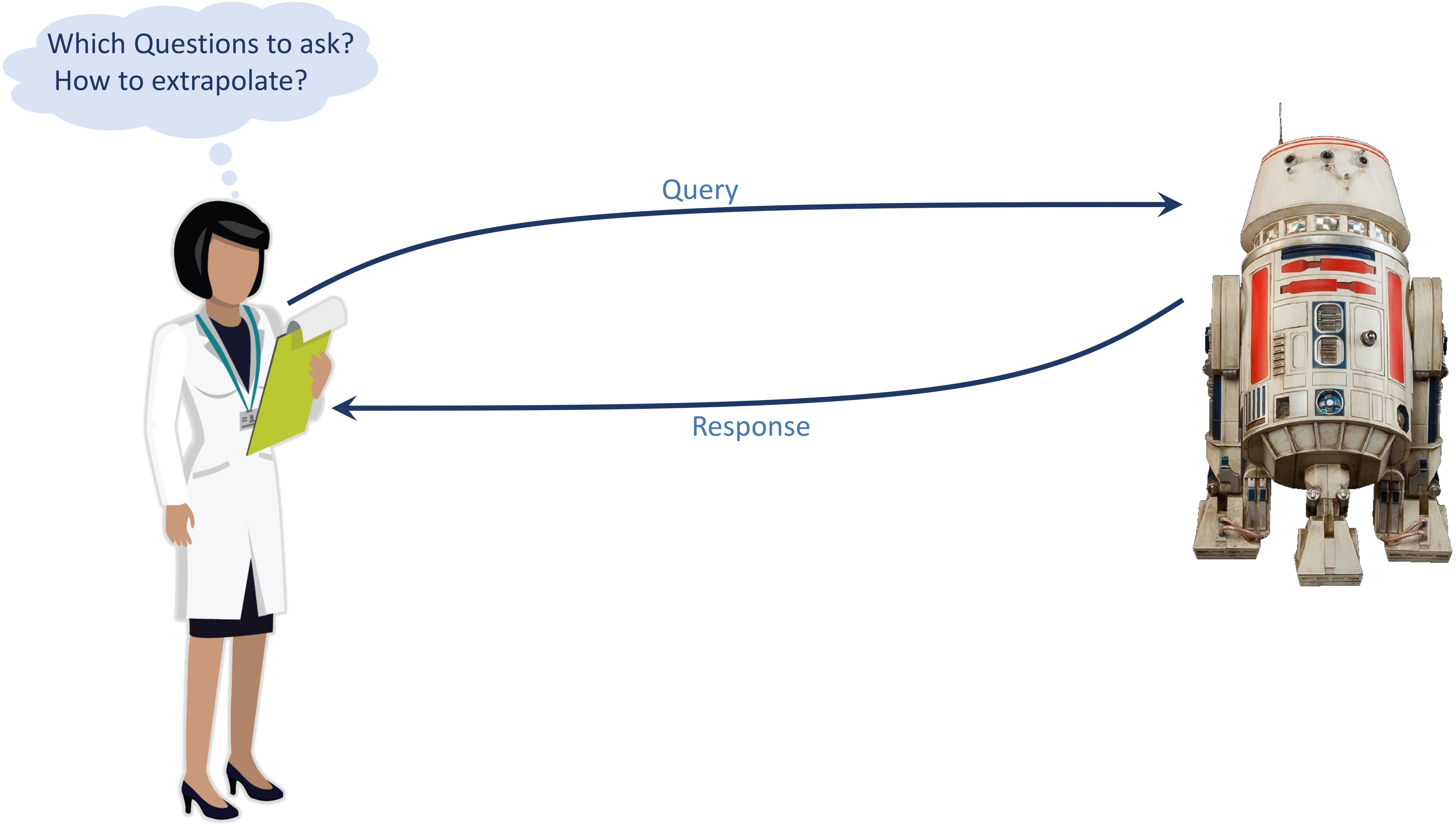
```
(:capability c4
 :parameters (?player1 ?cell1
   ?monster1 ?cell2)
 :precondition
  (and (alive ?monster1)
    (at ?player1 ?cell1)
    (at ?monster1 ?cell2)
    (next_to ?monster1))
 :effect
  (and (clear ?cell2)
   (not(alive ?monster1))
   (not(at ?monster1 ?cell2))
   (not(next_to ?monster1))))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

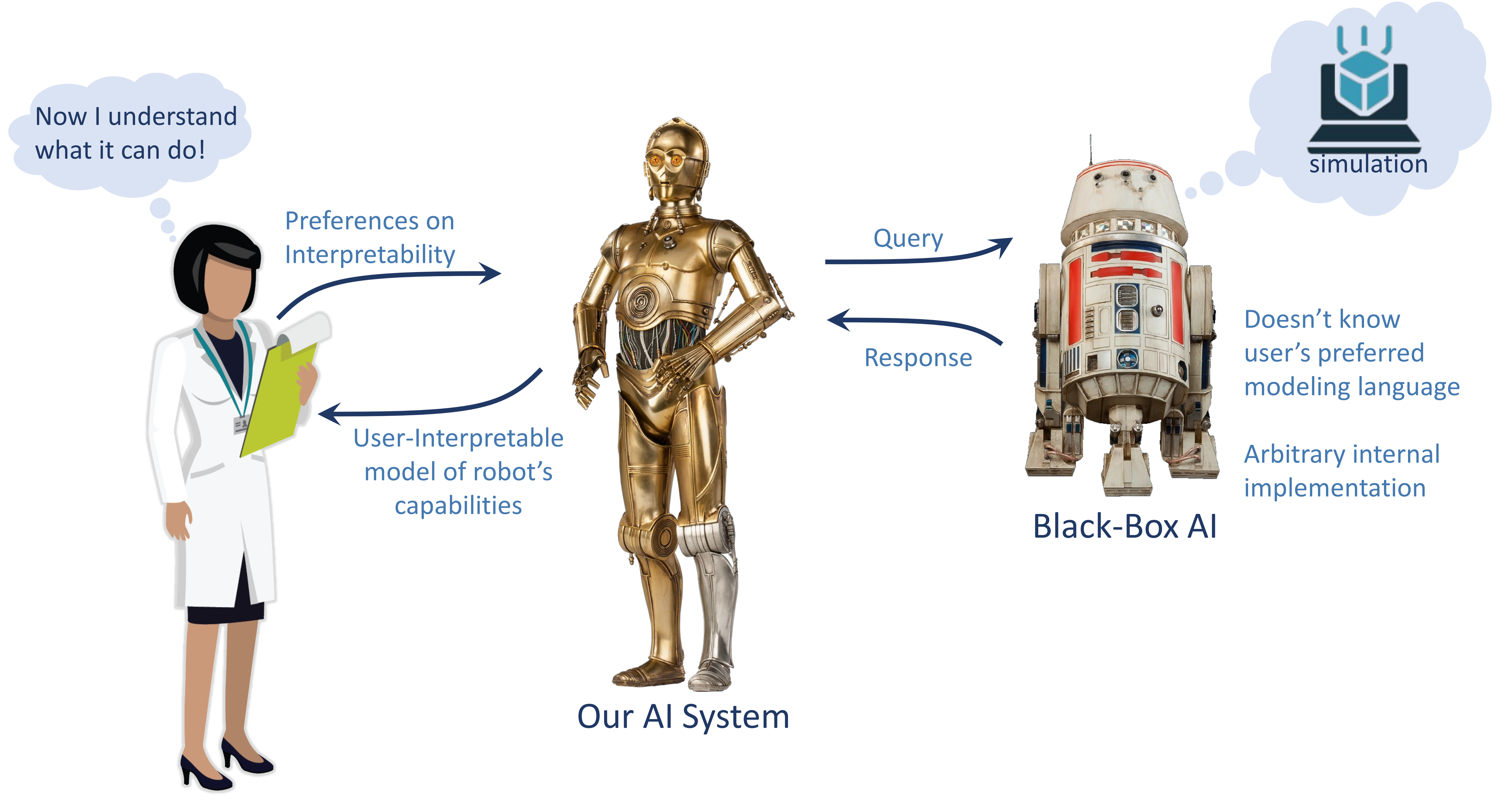
How to learn these?

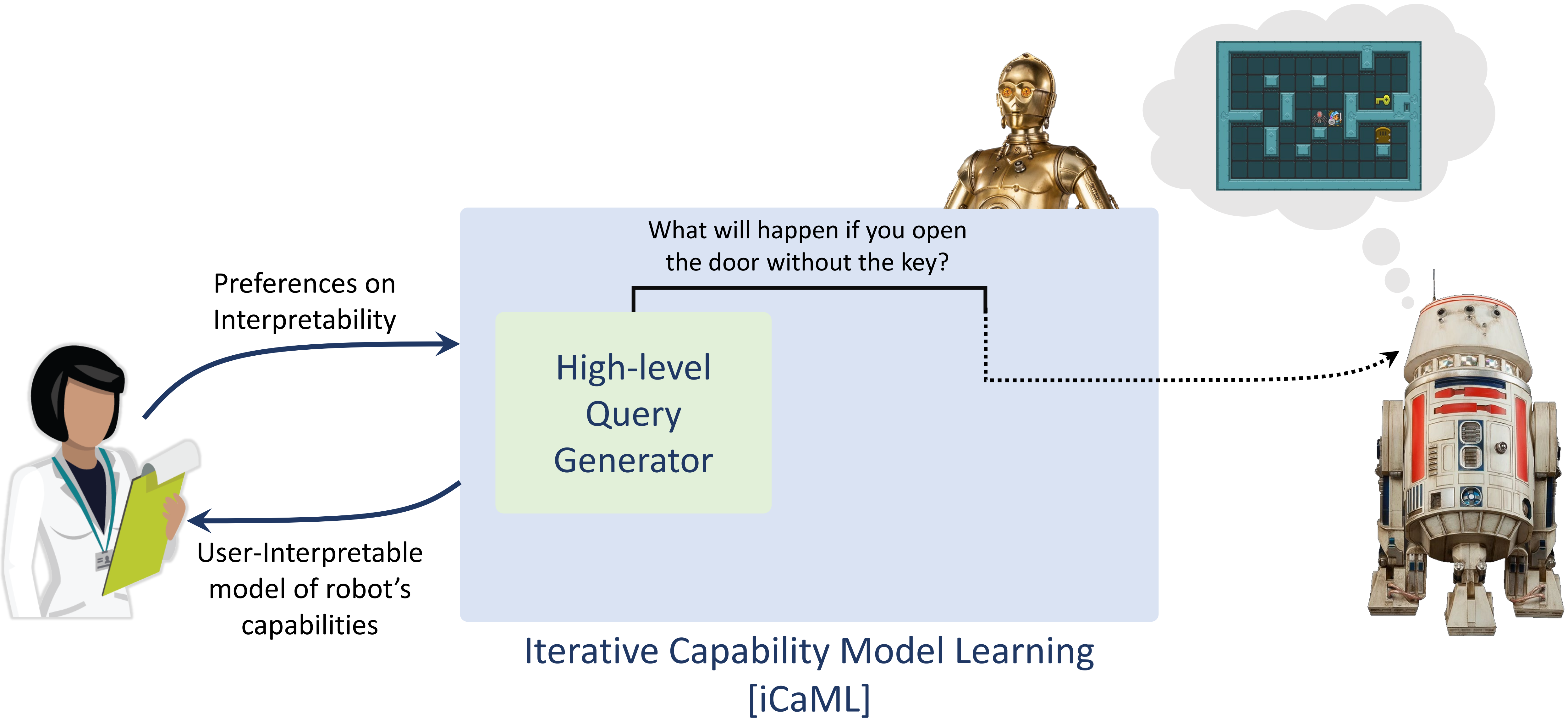
[Learned capability description]

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.


Which Questions to ask?
How to extrapolate?
Query
Response


Now I understand what it can do!
Preferences on Interpretability
User-Interpretable model of robot's capabilities
Our AI System
Query
Response
simulation
Black-Box AI
Doesn't know user's preferred modeling language
Arbitrary internal implementation


What will happen if you open
the door without the key?
Preferences on
Interpretability
High-level
Query
Generator
User-Interpretable
model of robot’s
capabilities
Iterative Capability Model Learning
[iCaML]

What is a Query?

- Every query can be viewed as a function from models to responses.

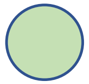
Plan Outcome Queries:

Query: $\langle \boldsymbol{s_I}, \boldsymbol{\pi} \rangle$
Initial State and Plan (in terms of capabilities)

Agent's Response: $\langle \ell, \boldsymbol{s_F} \rangle$
Length of plan that can be executed successfully and the final state.

How do we generate these queries?
How do we use them?

Algorithm for Hierarchical Query Synthesis



(:action pickup

:parameters (?ob)

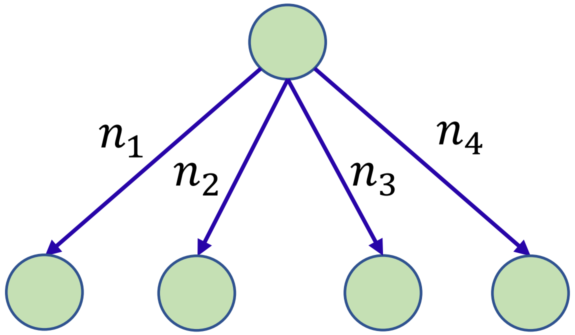
:precondition (and (+/-/∅) (handempty) n_1

(+/-/∅) (ontable ?ob)) n_2

:effect (and (+/-/∅) (handempty) n_3

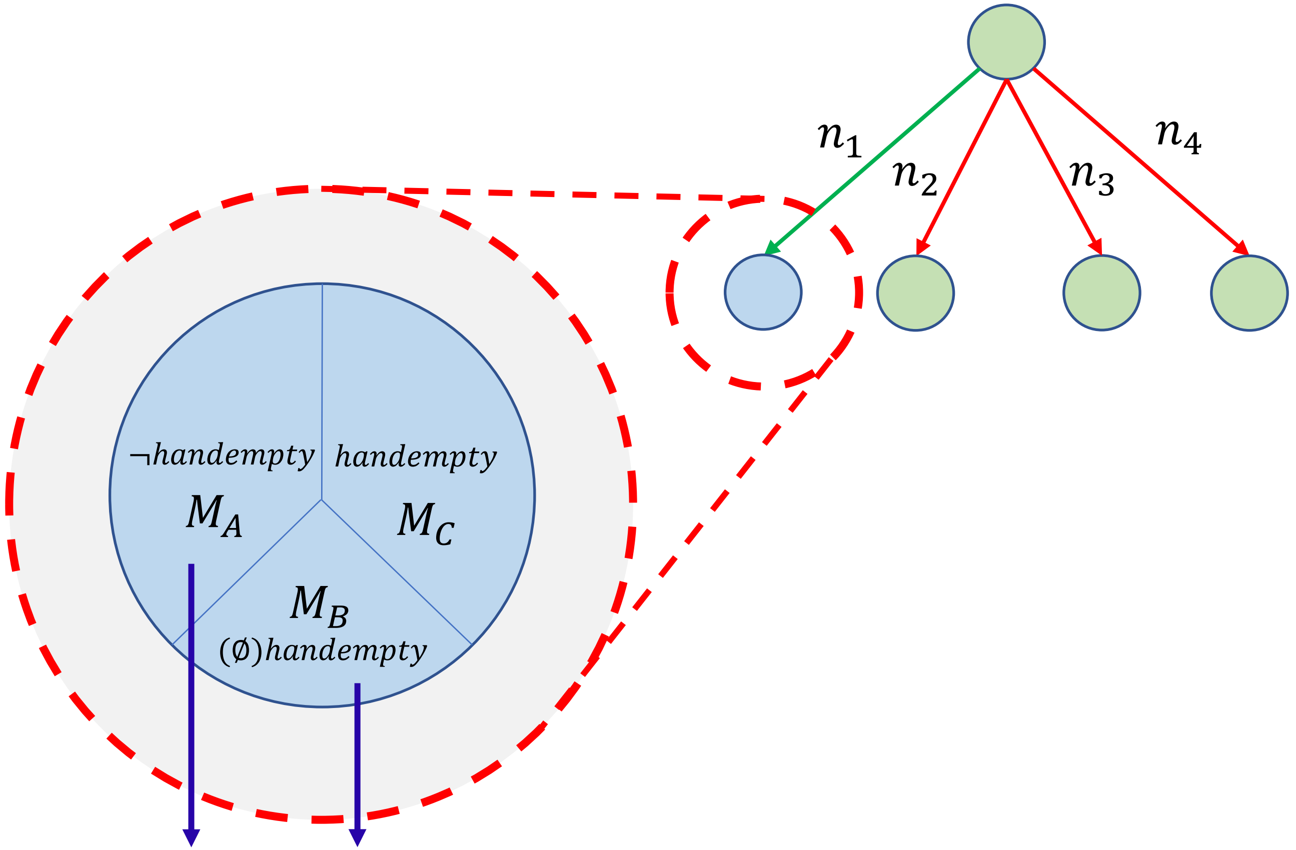
(+/-/∅) (ontable ?ob))) n_4

Algorithm for Hierarchical Query Synthesis



(:action pickup
:parameters (?ob)
:precondition (and (+/-/∅) (handempty) n_1
(+/-/∅) (ontable ?ob)) n_2
:effect (and (+/-/∅) (handempty) n_3
(+/-/∅) (ontable ?ob))) n_4

Algorithm for Hierarchical Query Synthesis



(:action pickup

:parameters (?ob)

:precondition (and (+/-/∅) (handempty)

(+/-/∅) (ontable ?ob))

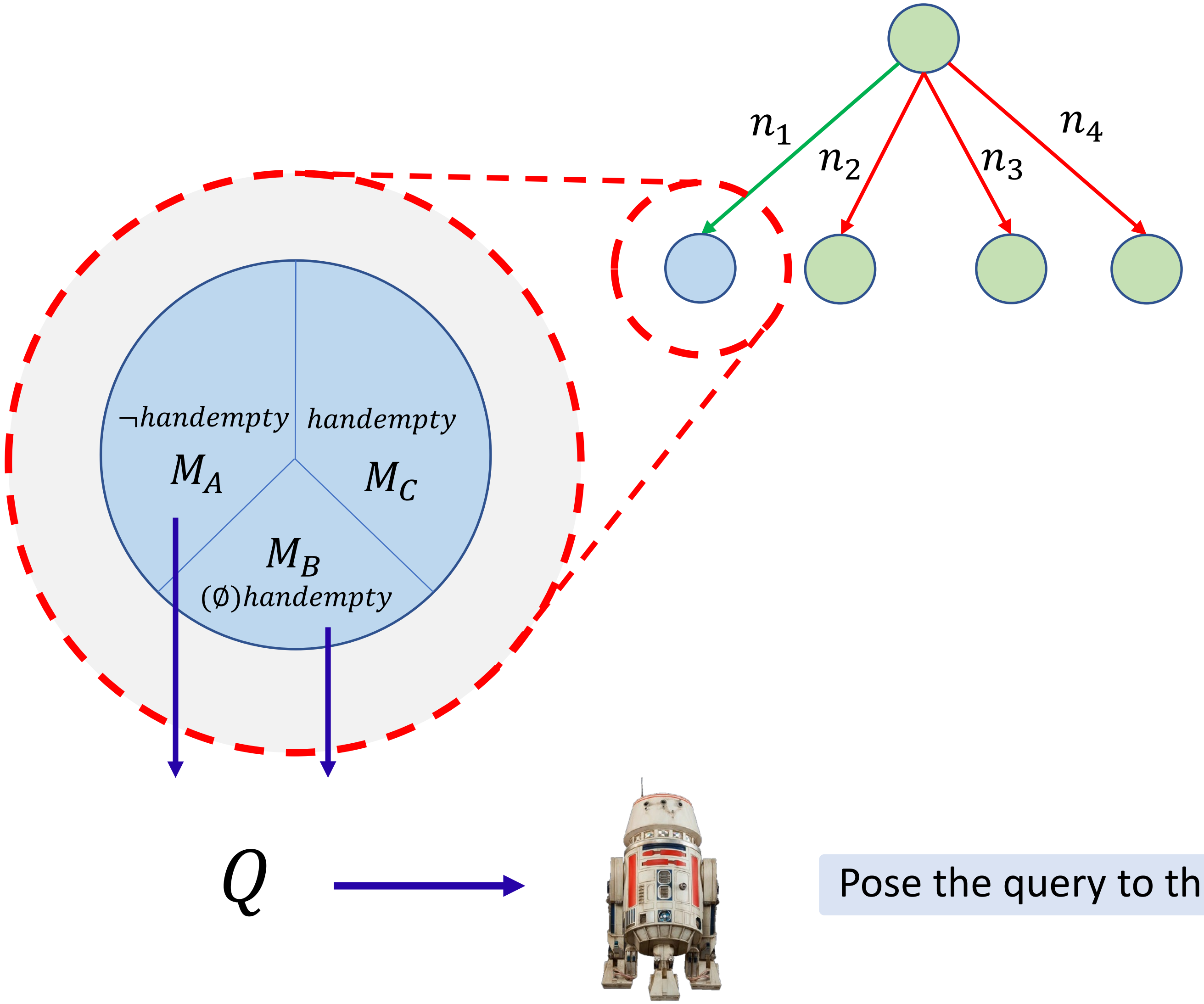
:effect (and (+/-/∅) (handempty)

(+/-/∅) (ontable ?ob)))

Generate a
distinguishing query:
Q such that $Q(M_A) \neq Q(M_B)$

Query-plan generated automatically
by reduction to planning

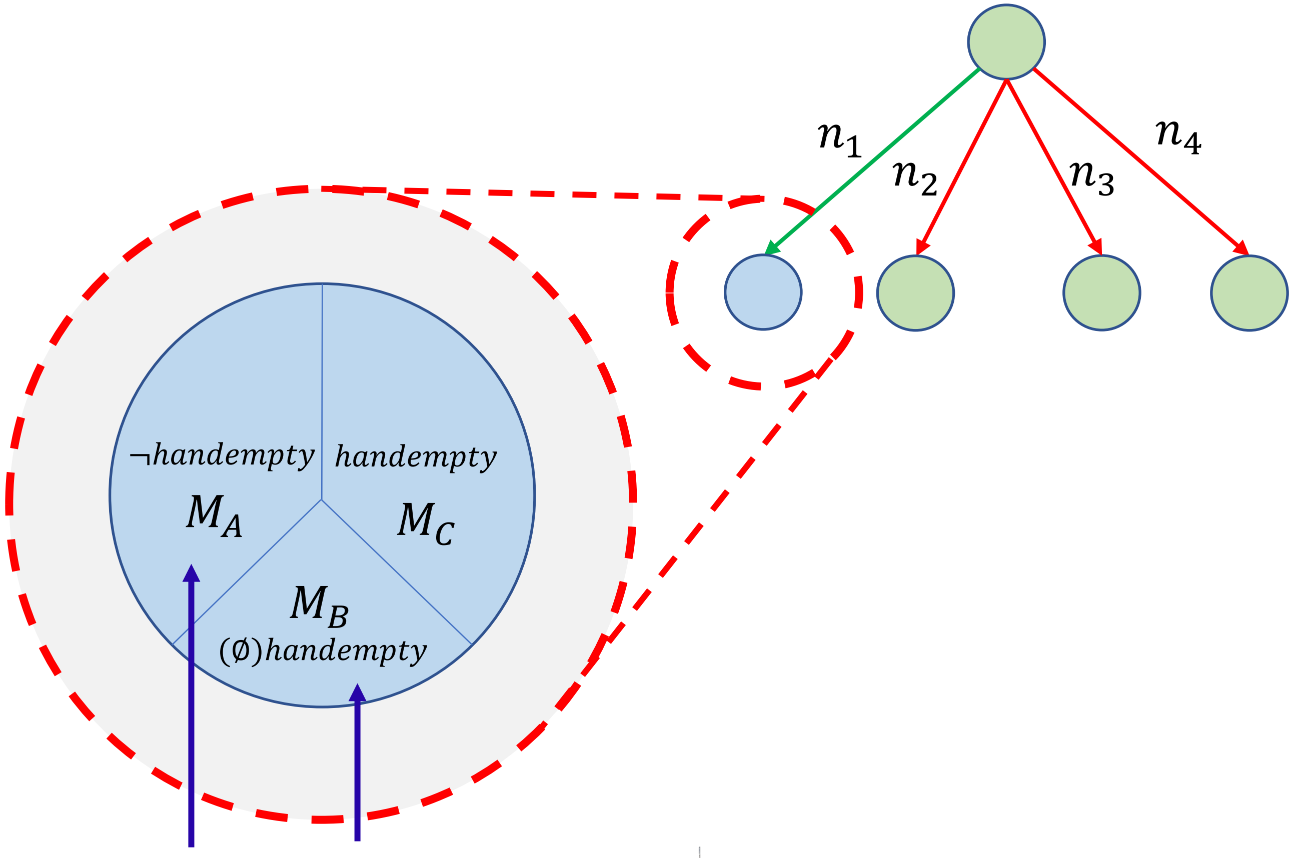
Algorithm for Hierarchical Query Synthesis



(:action pickup
:parameters (?ob)
:precondition (and (+/-/∅) (handempty)
(+/-/∅) (ontable ?ob))
:effect (and (+/-/∅) (handempty)
(+/-/∅) (ontable ?ob)))

Pose the query to the agent

Algorithm for Hierarchical Query Synthesis



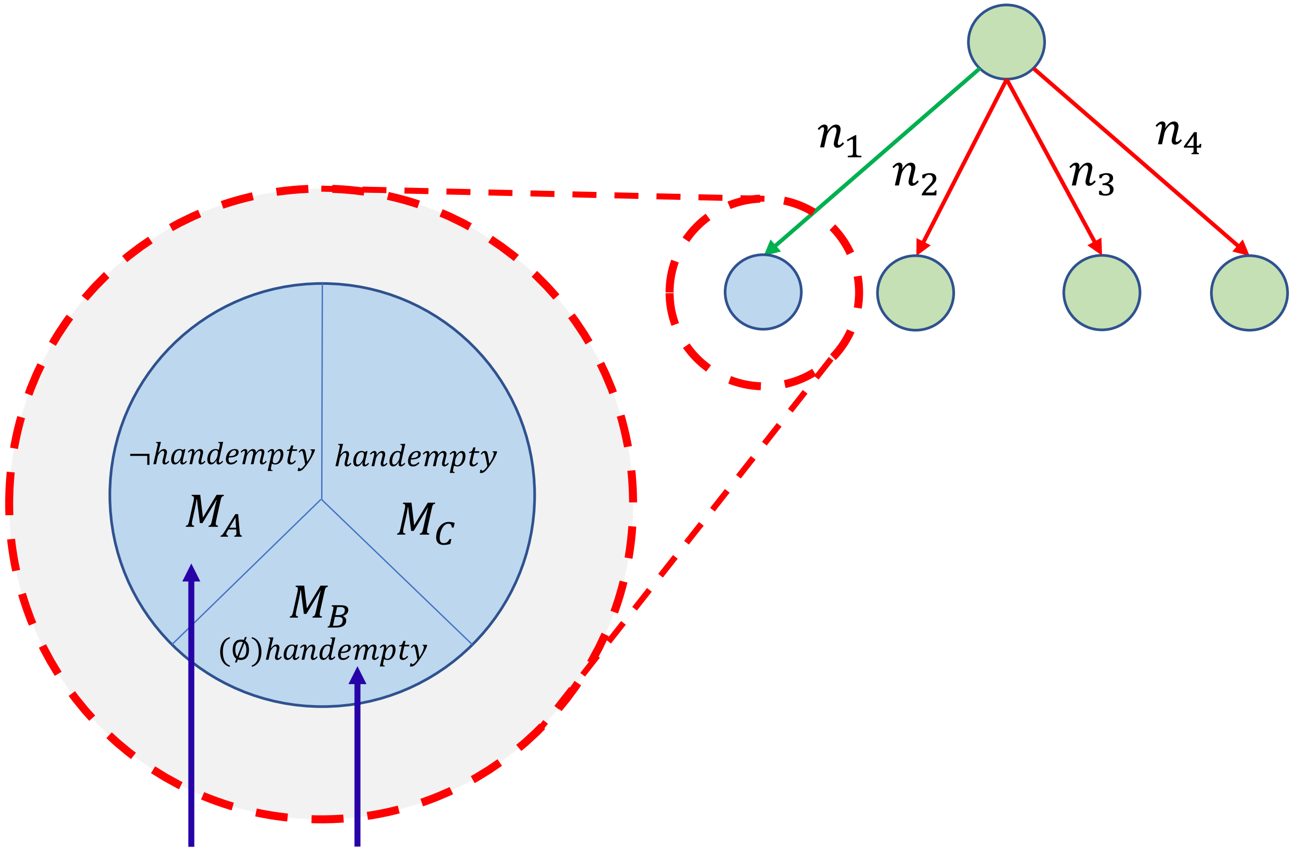
(:action pickup
:parameters (?ob)
:precondition (and (+/-/∅) (handempty)
(+/-/∅) (ontable ?ob))
:effect (and (+/-/∅) (handempty)
(+/-/∅) (ontable ?ob)))

$$\theta = Q(Agent)$$

$$Q(M_A) \neq Q(M_B)$$

Check the consistency of refinements with the agent response

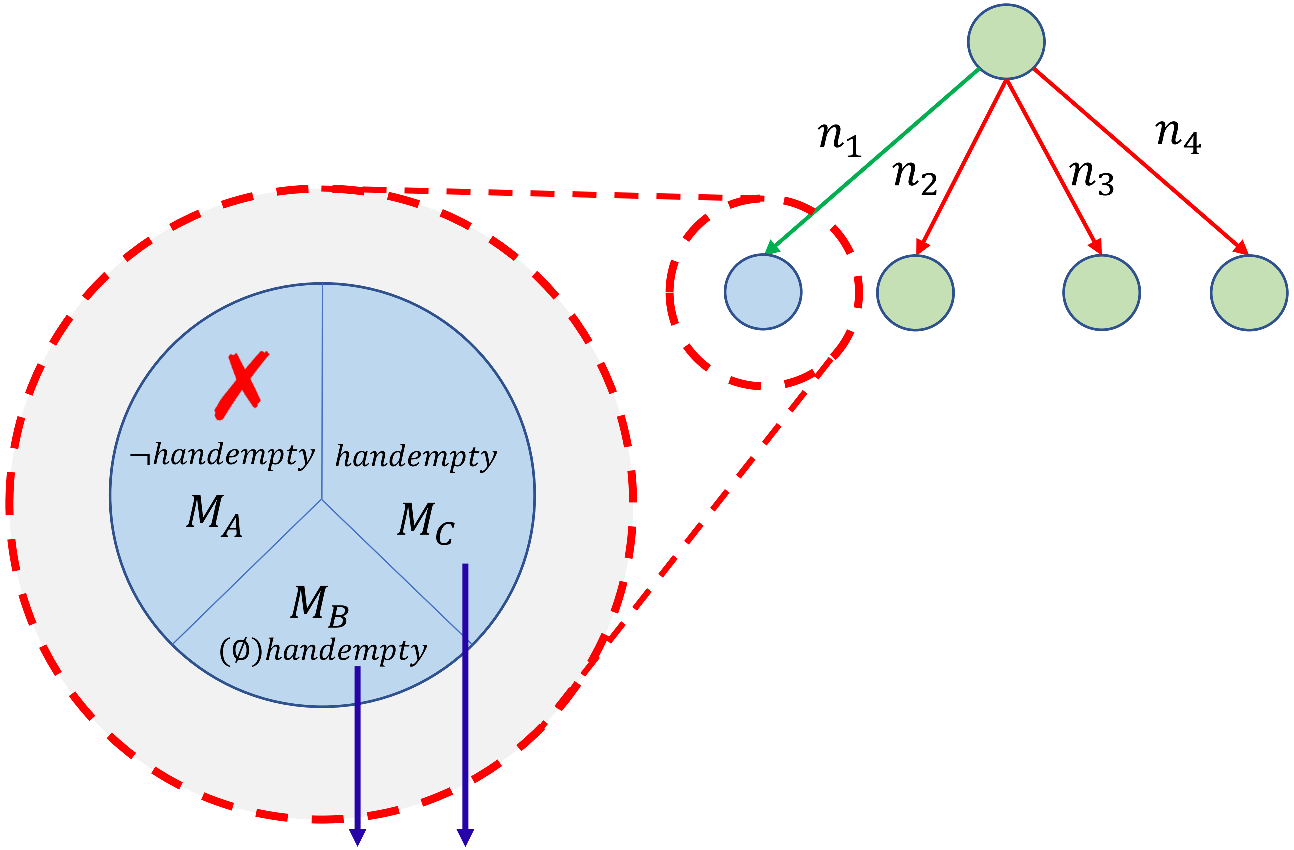
Algorithm for Hierarchical Query Synthesis



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

(:action pickup
:parameters (?ob)
:precondition (and (+/-/∅) (handempty) ←
(+/-/∅) (ontable ?ob))
:effect (and (+/-/∅) (handempty)
(+/-/∅) (ontable ?ob)))

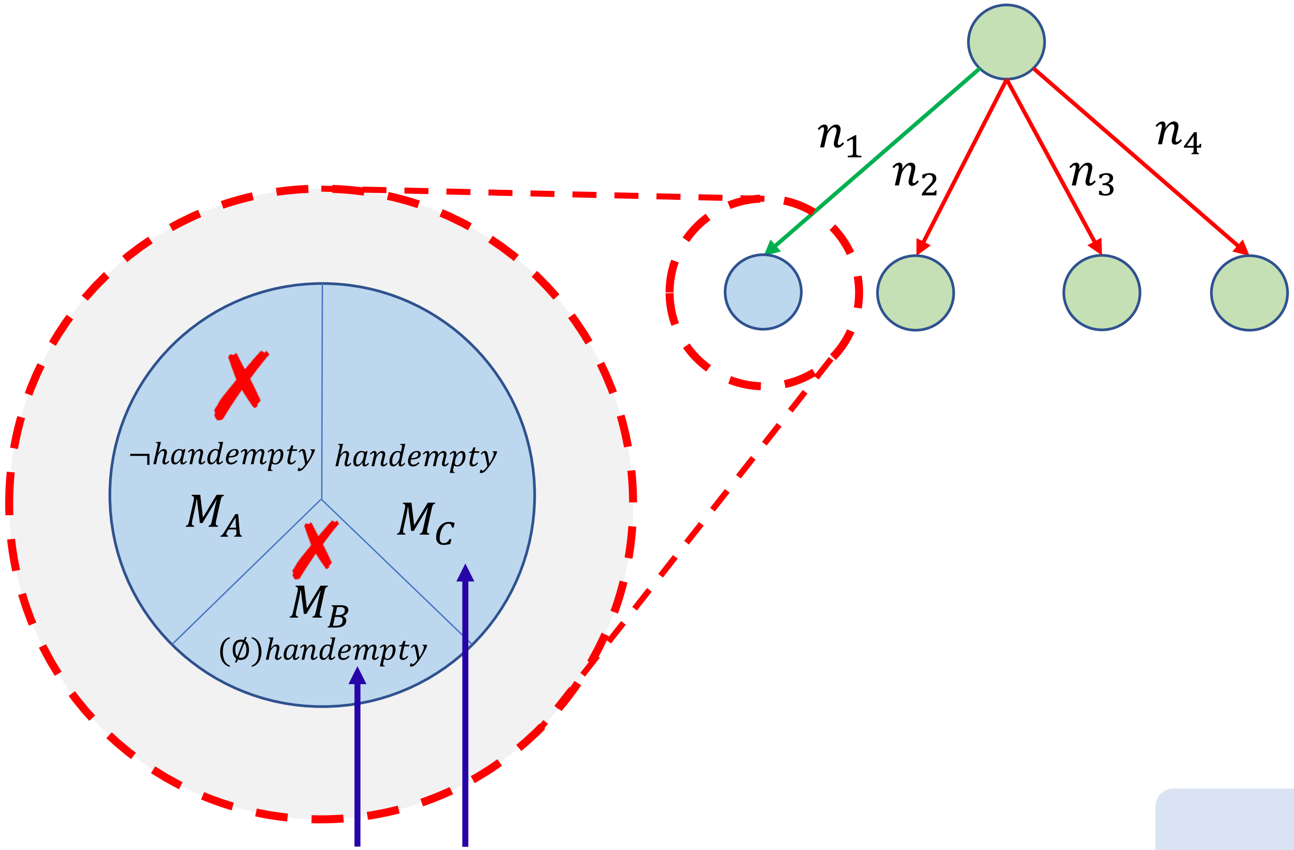
Algorithm for Hierarchical Query Synthesis



Generate a distinguishing query for these two refinements

```
(:action pickup
  :parameters (?ob)
  :precondition (and (+/-/∅) (handempty)
                     (+/-/∅) (ontable ?ob))
  :effect (and (+/-/∅) (handempty)
               (+/-/∅) (ontable ?ob)))
```

Algorithm for Hierarchical Query Synthesis



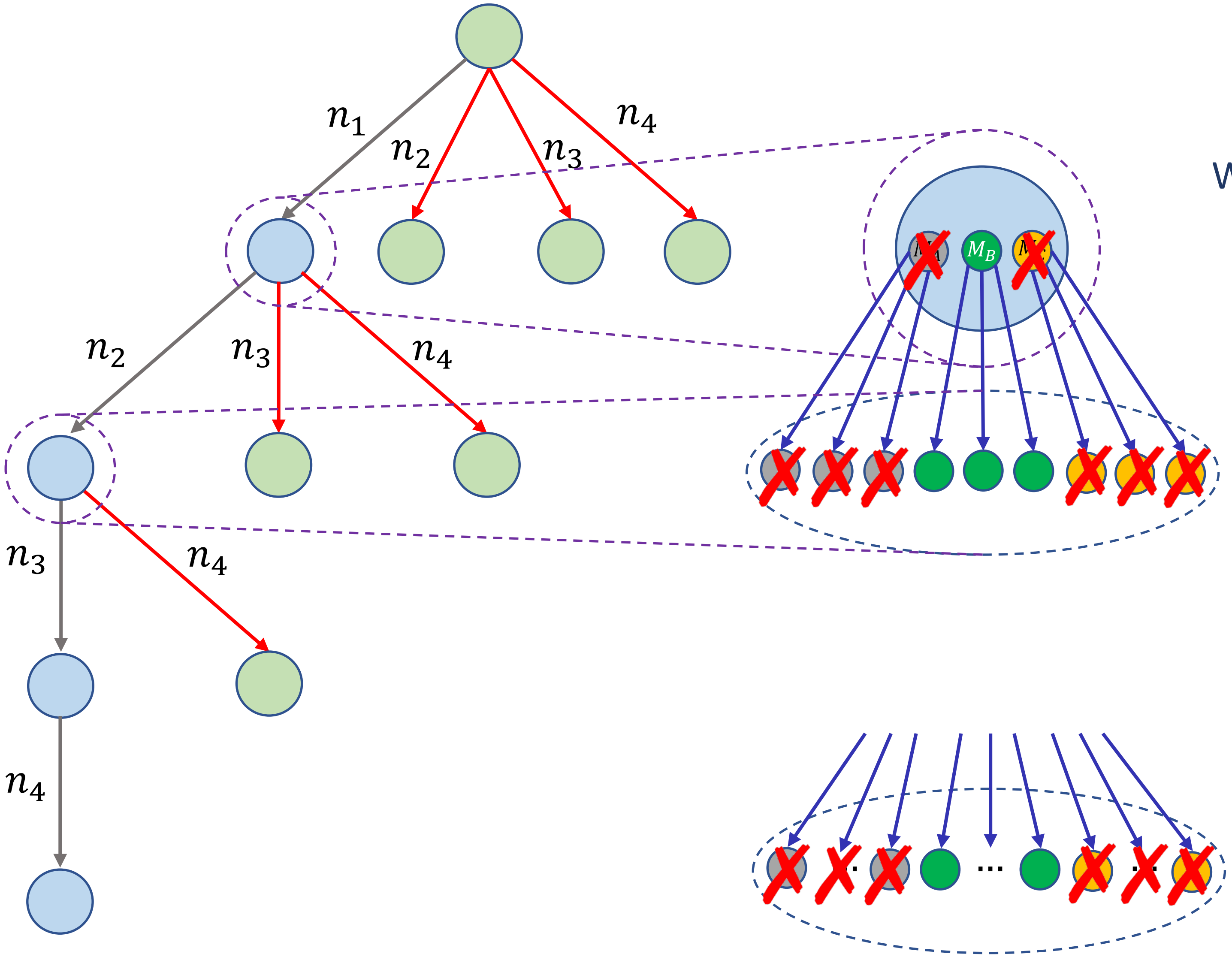
Reject the refinement
that is not consistent with the agent

(:action pickup
:parameters (?ob)
:precondition (and (+/-/∅) (handempty)
(+/-/∅) (ontable ?ob))
:effect (and (+/-/∅) (handempty)
(+/-/∅) (ontable ?ob)))

Lemma

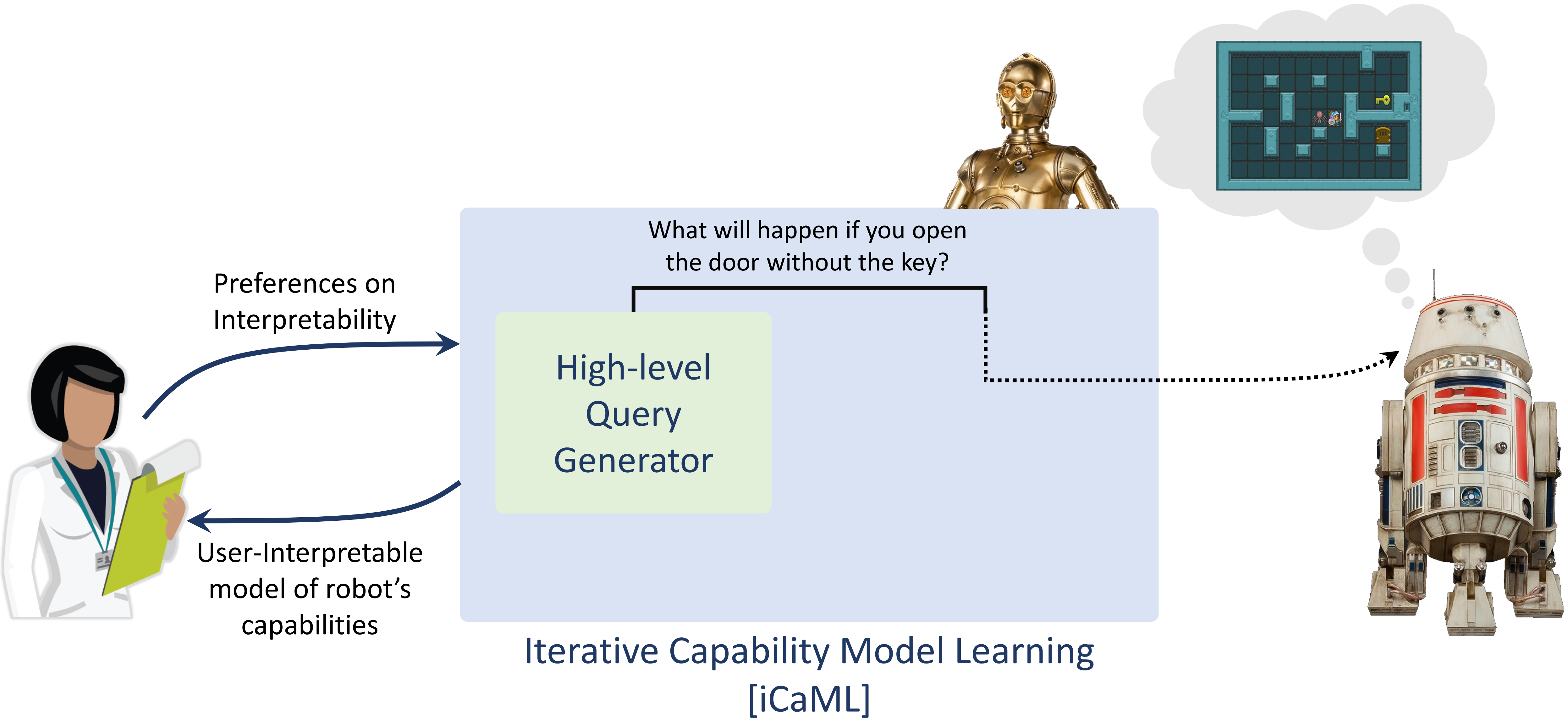
At least one of the refinements will be consistent with the agent.

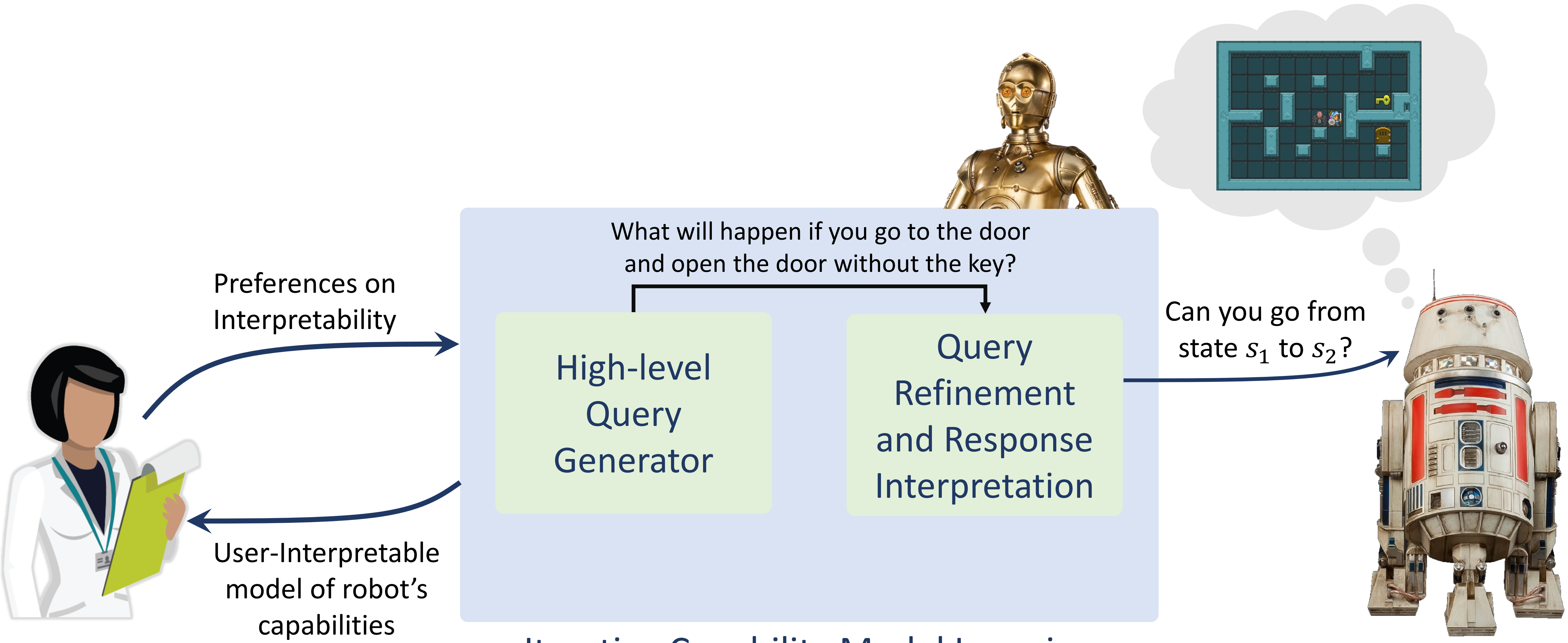
Algorithm for Hierarchical Query Synthesis



Key feature of the algorithm

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


Preferences on Interpretability
What will happen if you open the door without the key?
High-level Query Generator
User-Interpretable model of robot's capabilities
Iterative Capability Model Learning [iCaML]



Iterative Capability Model Learning [iCaML]

State Reachability Query

Query: $\langle s_I, s_G \rangle$
Initial State and Goal State

Agent's Response: $\langle Yes, No \rangle$
Whether it can reach from initial state to goal using its internal mechanism.

How do we generate these queries from plan outcome queries?

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

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

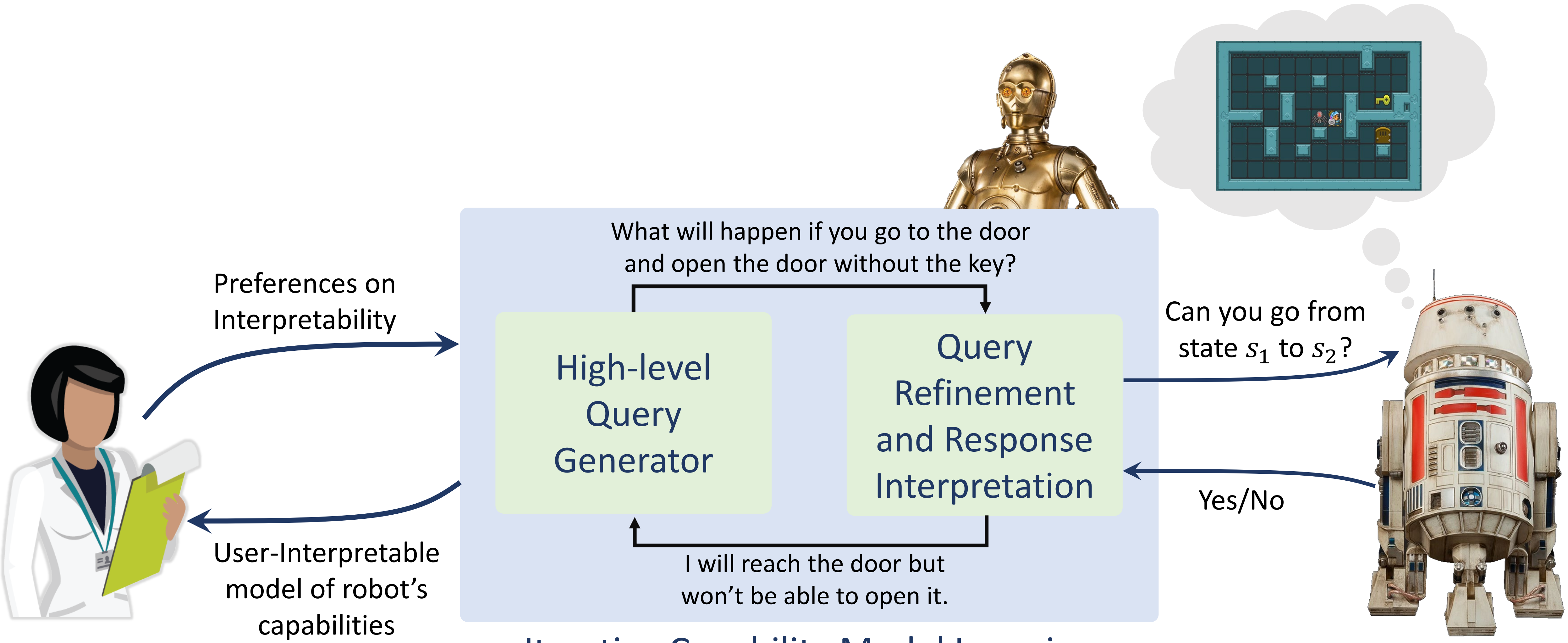
Can you reach state $\bar{s}_1$ from state $\bar{s}_0$?

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

. . .

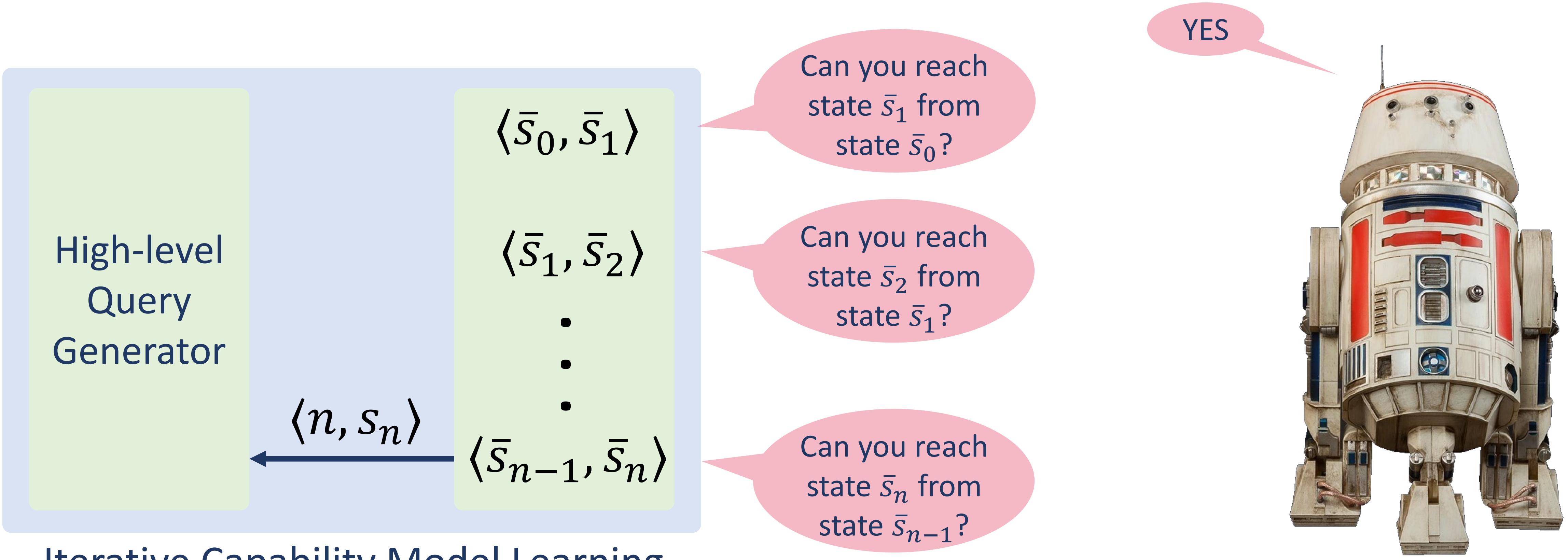
Can you reach state $\bar{s}_n$ from state $\bar{s}_{n-1}$?

State Reachability Queries



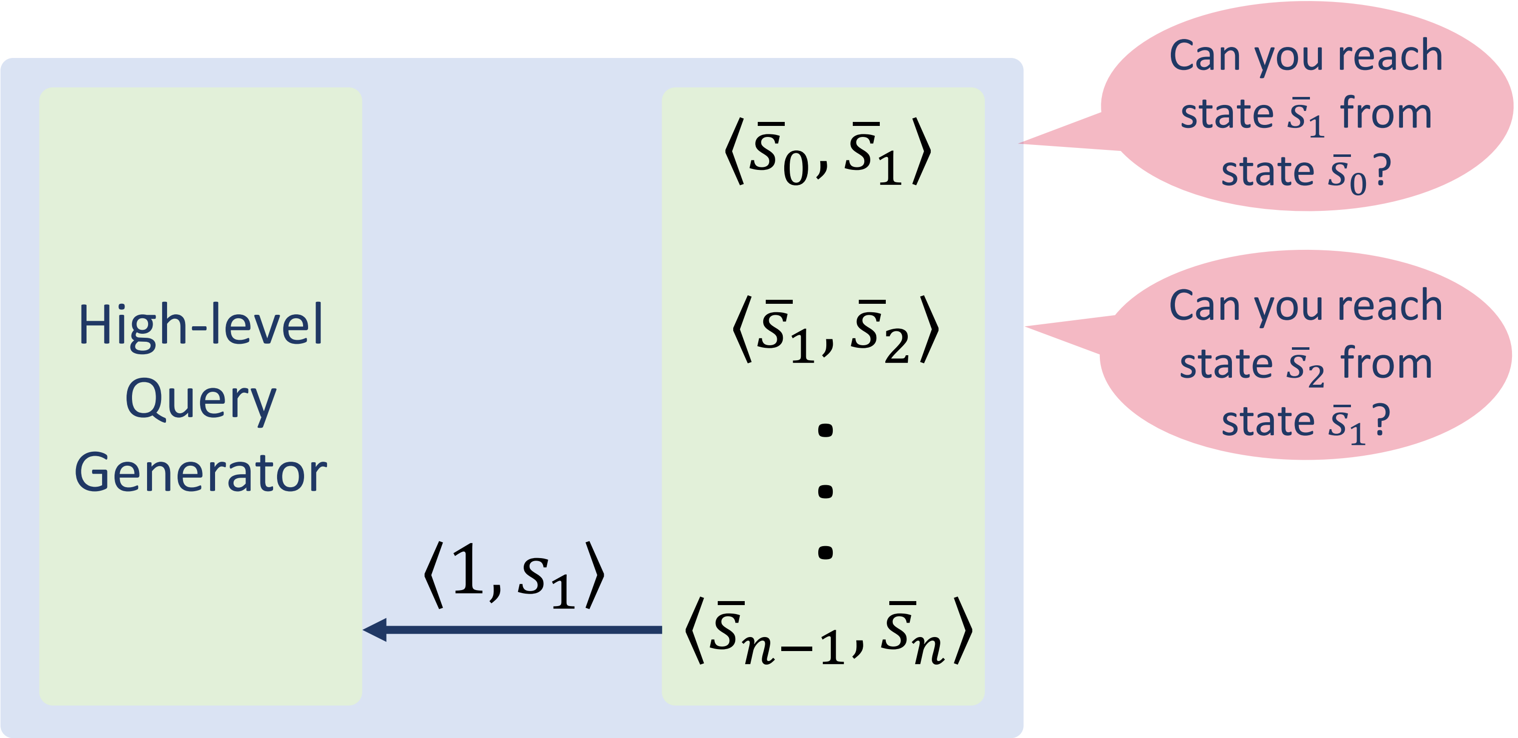
Iterative Capability Model Learning [iCaML]

Response Interpretation



Iterative Capability Model Learning
[iCaML]

Response Interpretation



Iterative Capability Model Learning [iCaML]

Formal Results

- The learned descriptions are consistent with the observations and the queries.
- 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.
- Learned capabilities are realizable, i.e., downward refinement is ensured.
- If a high-level model is expressible deterministically using the user vocabulary and local connectivity holds, then in the limit of infinite execution traces, the probability of discovering all capabilities expressible in the user vocabulary is 1.

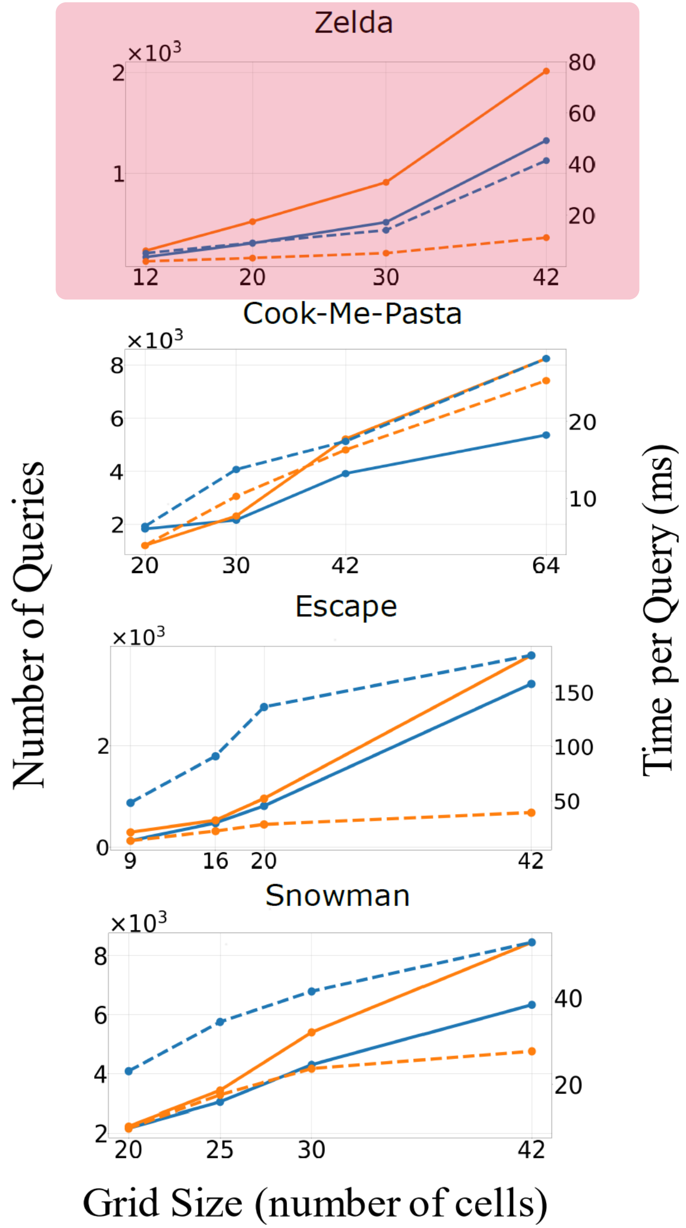
Experimental Setup

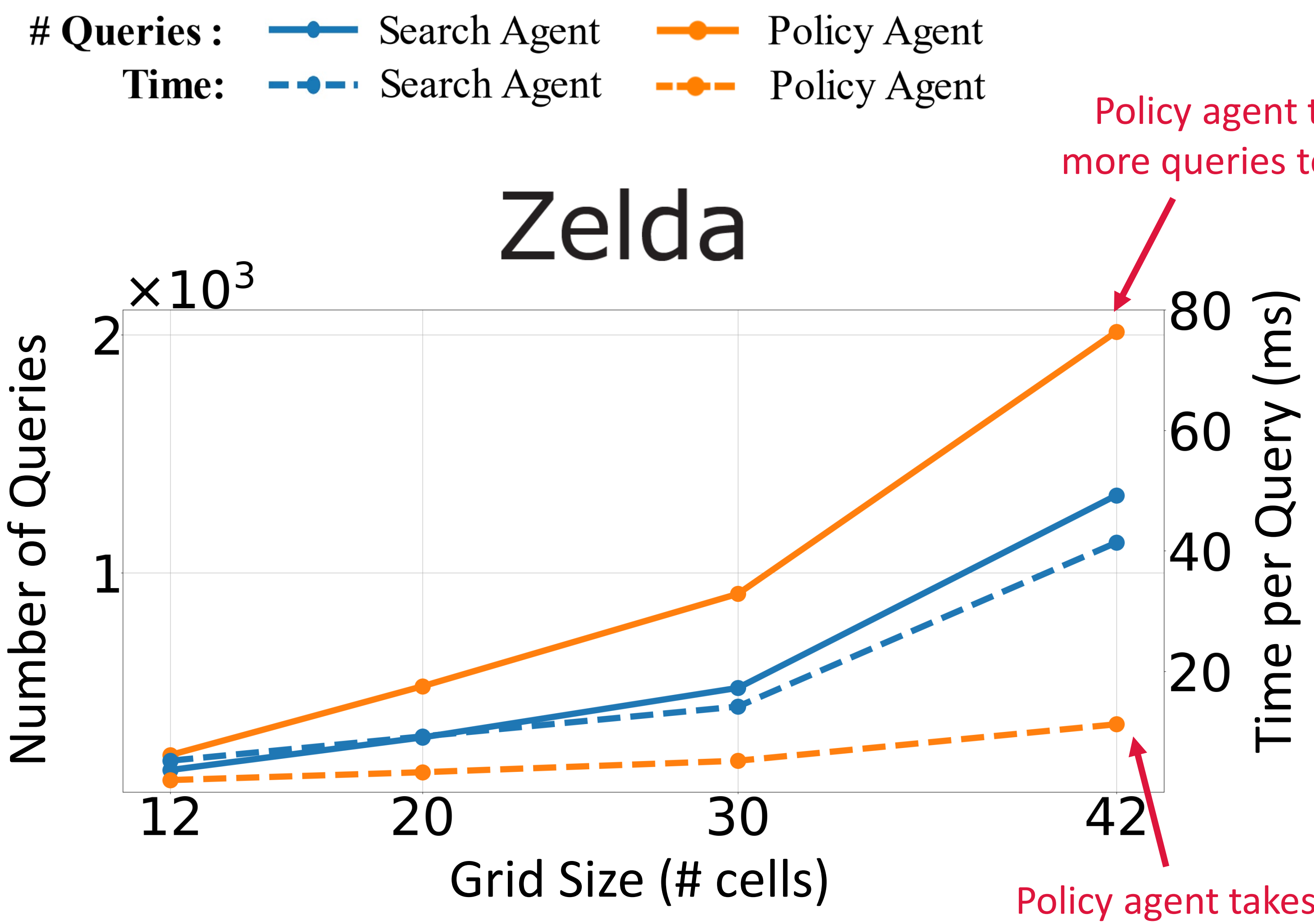
- Randomly generate an environment from one of four GVGAI Games.

- Initialize two kinds of agents –
  - Search Agent: Use search algorithms to generate plan and answer queries.
  - Policy Agent: Use black-box policies to answer queries.

- Vary grid size to see variations in number of queries and time taken per query.

Results





Policy agent takes more queries to learn

Policy agent takes less time per query

Results: Example of Learned 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))))
```

Position of Link has not changed

Ganon is not at its previous location

Ganon is not alive anymore

Link is not next to Ganon

Utility of Discovered Capability Models

- Rules of Zelda-like game explained to users.
- 108 participants split into two groups of 54 each.
- Didn't train the users, only descriptions shown in English:
  - Capability Group participants shown learned capability descriptions.
  - Functionality Group participants shown descriptions of keystrokes.

We created a single-player game like *The Legend of Zelda* that looks something like this:

The hero of the game is called *Link*. *Link* must defeat the evil spider *Ganon* and escape. The **rules of the game** are as follows:
1. *Link* can move around the grid, whereas *Ganon* cannot.
2. *Link* must defeat *Ganon* and get the key (in any order), then enter the door to win.
3. *Link* cannot go through the cells containing walls, keys, *Ganon*, or door.
4. Game ends in *Link*'s loss if *Link* moves into a cell with *Ganon*.

The empty cells are represented as ■. All other kinds of cells are impassable.

Utility of Discovered Capability Models

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

[Capability Description Example]

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

[Functionality Description Example]

Utility of Discovered Capability Models

If *Link* starts in the state shown below:

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

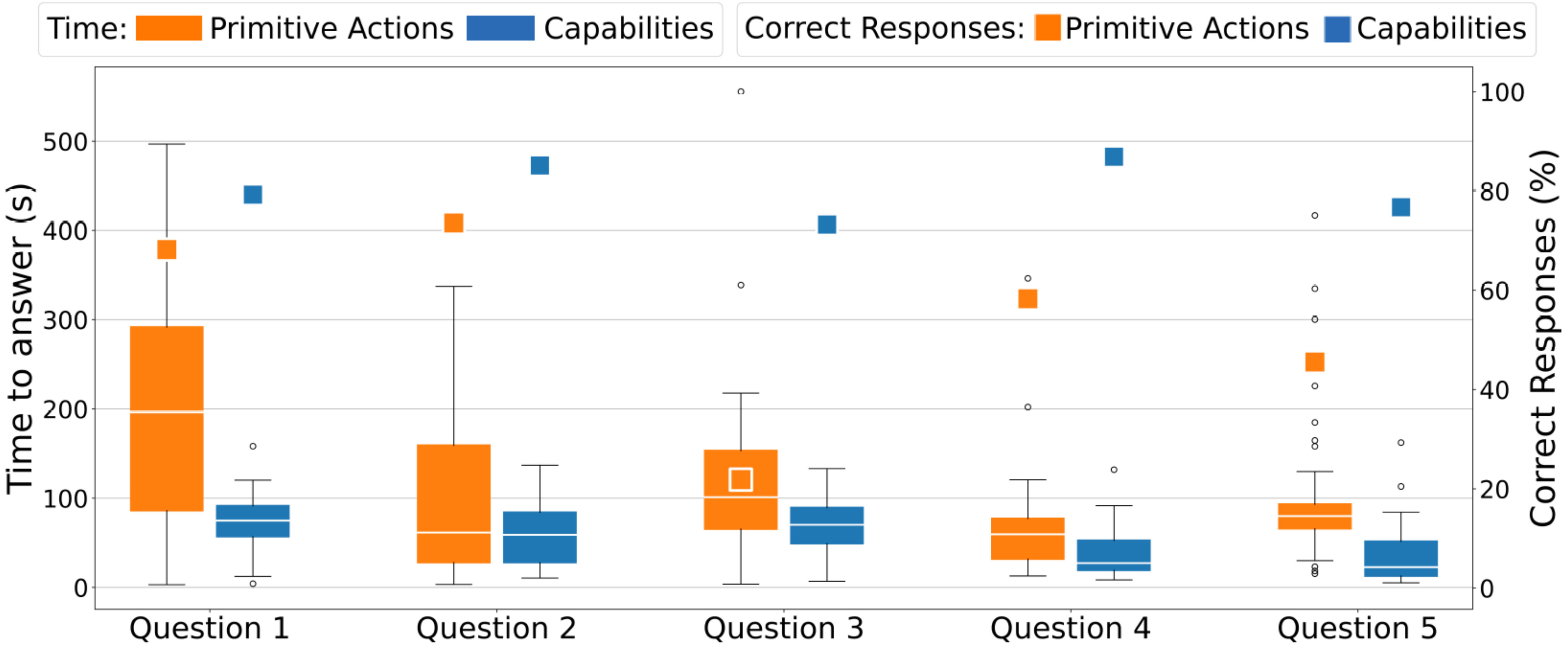
◯ C1 → C5 (Go next to Ganon → Go next to Key)

[Example of an option for Capability Group]

◯ W → W → D → D → W → W → W → D → D → D → S → S → A → E (Up → Up → Right → Right → Up → Up → Up → Right → Right → Right → Down → Down → Left → Interact)

[Example of an option for Functionality Group]

Results: Behavior Prediction Study



Key Takeaways

The proposed approach:

- Efficiently discovers capabilities of an agent in a STRIPS-like form in fully observable and deterministic settings.
- Needs no prior knowledge of the agent model.
- Only requires an agent to have rudimentary query answering capabilities.
- Learns a maximally consistent capability model accurately with a small number of queries.
- Learns capability descriptions that are interpretable as shown using a user study.

Paper

arxiv: 2107.13668

verma.pulkit@asu.edu