AAAI 2022 Workshop on Explainable Agency in Artificial Intelligence

Discovering User-Interpretable Capabilities of Black-Box Planning Agents

Pulkit Verma, Shashank Rao Marpally, Siddharth Srivastava

Arizona State University

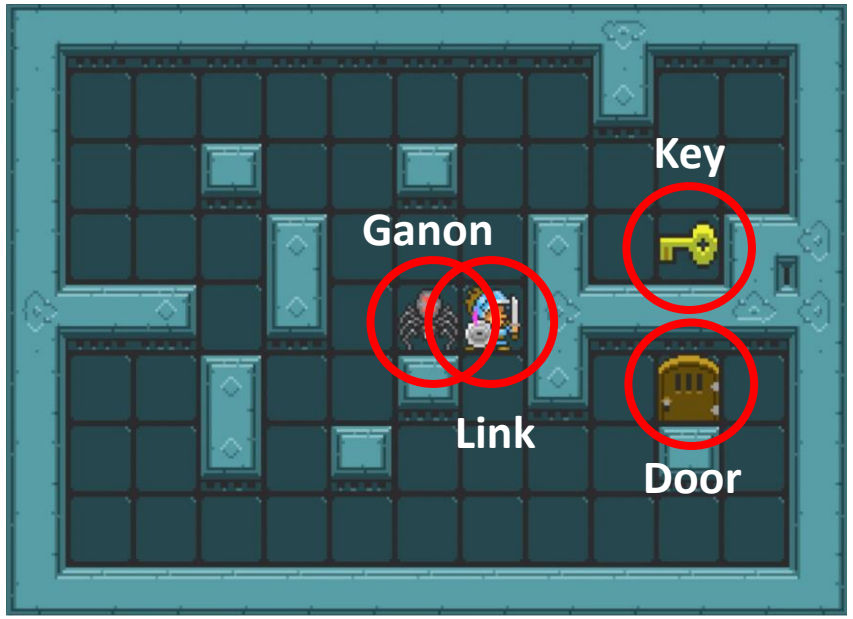
Personalized Assessment of Taskable AI Systems

- 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)

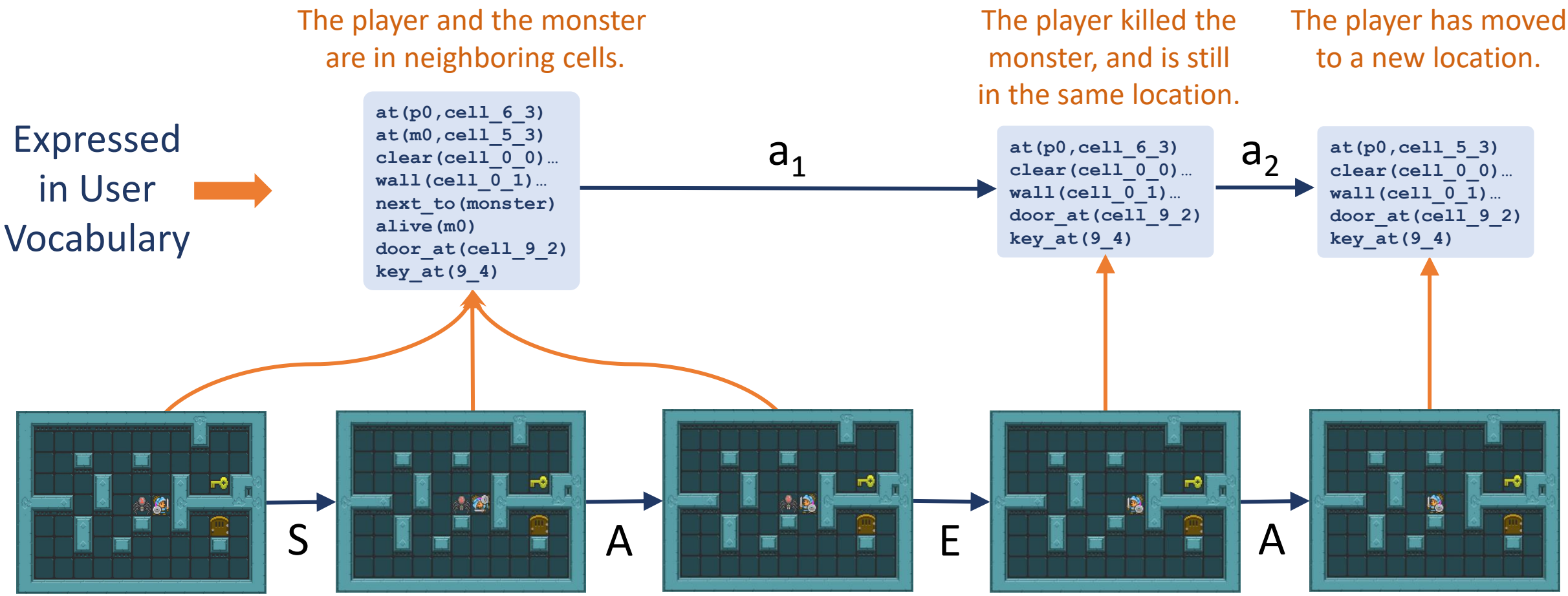
Knowledge of primitive actions might be insufficient to understand the agent's capabilities

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

State and Temporal Abstraction



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]

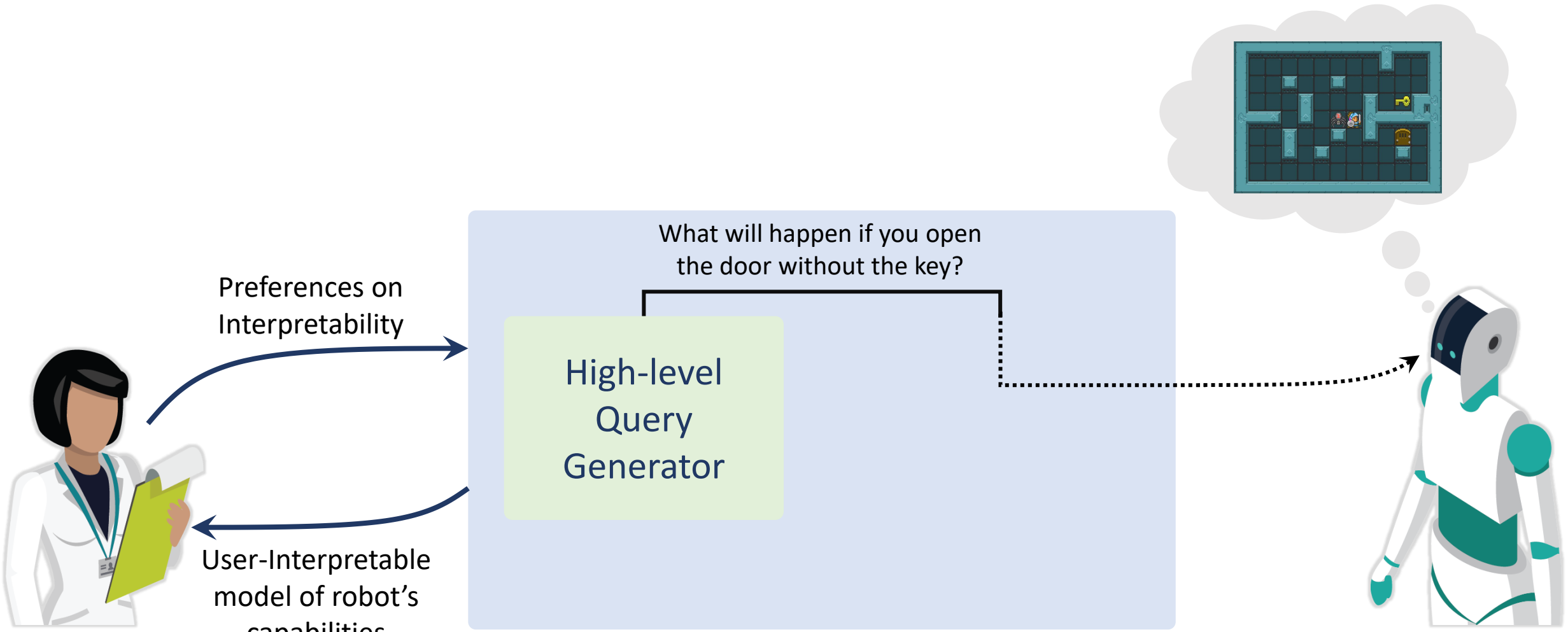
```
(:capability c4
 :parameters (?player1 ?cell1
   ?monster1 ?cell2)
 :precondition
  (and (alive ?monster1)
    (at ?player1 ?cell1)
    (at ?monster1 ?cell2)
    (next_to ?monster1))
 :effect
  (and (clear ?cell2)
   (not(alive ?monster1))
   (not(at ?monster1 ?cell2))
   (not(next_to ?monster1))))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

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.



Capability Discovery Algorithm

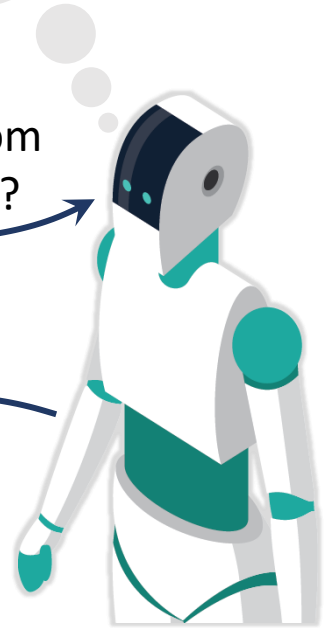
What will happen if you go to the door and open the door without the key?

Preferences on Interpretability

High-level Query Generator

Query Refinement and Response Interpretation

Can you go from state s_1 to s_2?



Yes/No

User-Interpretable model of robot's capabilities

I will reach the door but won't be able to open it.

Capability Discovery Algorithm

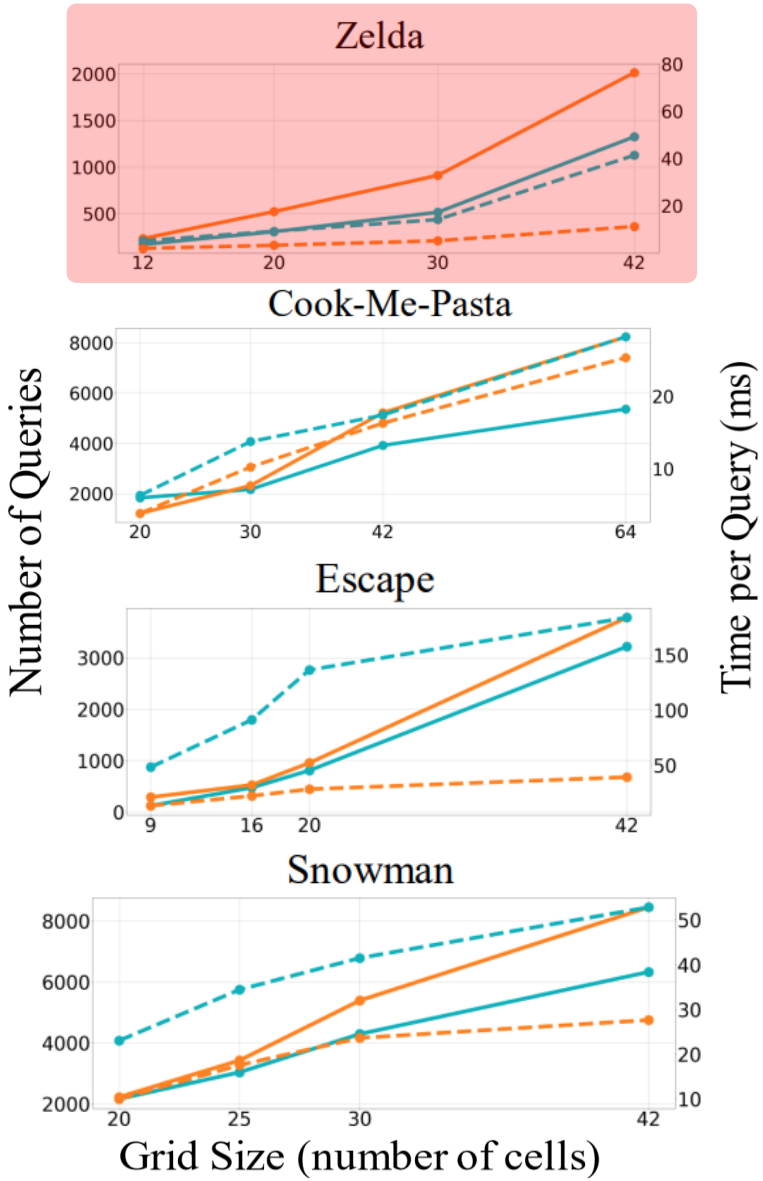
Formal Results

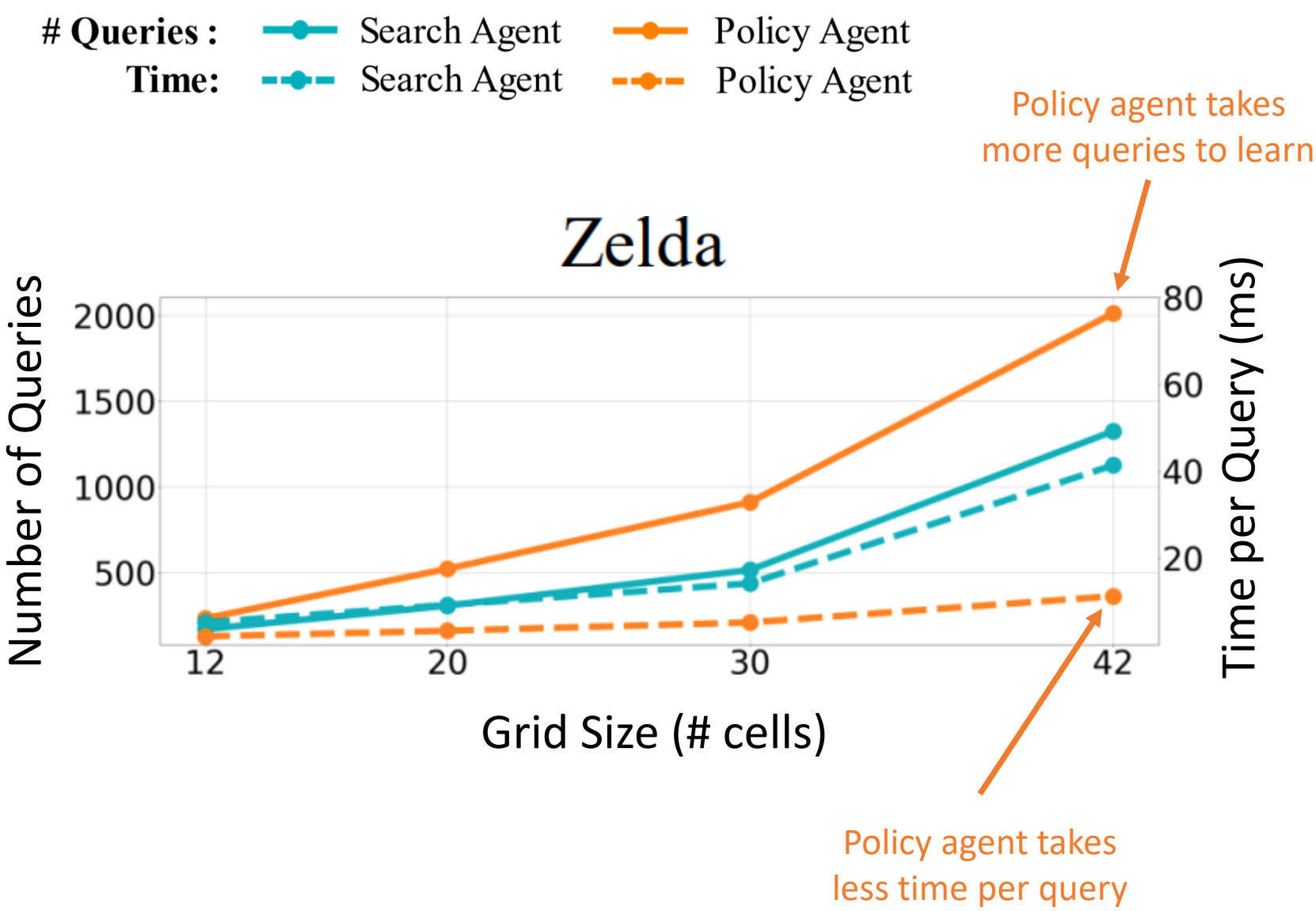
- The learned descriptions are consistent with the observations.
- This approach is maximally consistent, i.e., we cannot add any more literals to the preconditions or effects without ruling out some truly possible models.
- Learned capabilities are realizable, i.e., downward refinement is ensured.
- 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


Zelda
Cook-Me-Pasta
Escape
Snowman
Number of Queries
Time per Query (ms)
Grid Size (number of cells)


Queries : Search Agent Policy Agent
Time: Search Agent Policy Agent
Policy agent takes more queries to learn
Zelda
Number of Queries
Time per Query (ms)
Grid Size (# cells)
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

Behavior Prediction Study

If *Link* starts in the state shown below:

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

◯ C3 → C6 → C1 (Go next to Door → Open Door → Go next to Ganon)

[An example of an option for Capability Group]

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

[An example of an option for Functionality Group]

Behavior Prediction Study

- Rules of Zelda-like game explained to users.
- 108 participants split into two groups of 54 each.
- Capability Group participants shown learned capability descriptions in English.
- Functionality Group participants shown descriptions of keystrokes in English.

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.

Behavior Prediction Study

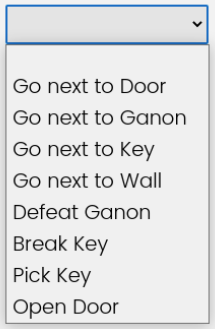
4. **Capability C4**:
The *player* can execute this capability when:

- 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.

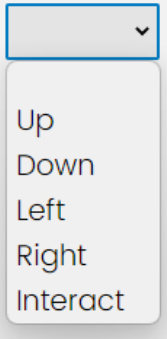


[Capability Description Example]

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.



[Functionality Description Example]

Behavior Prediction Study

If *Link* starts in the state shown below:

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

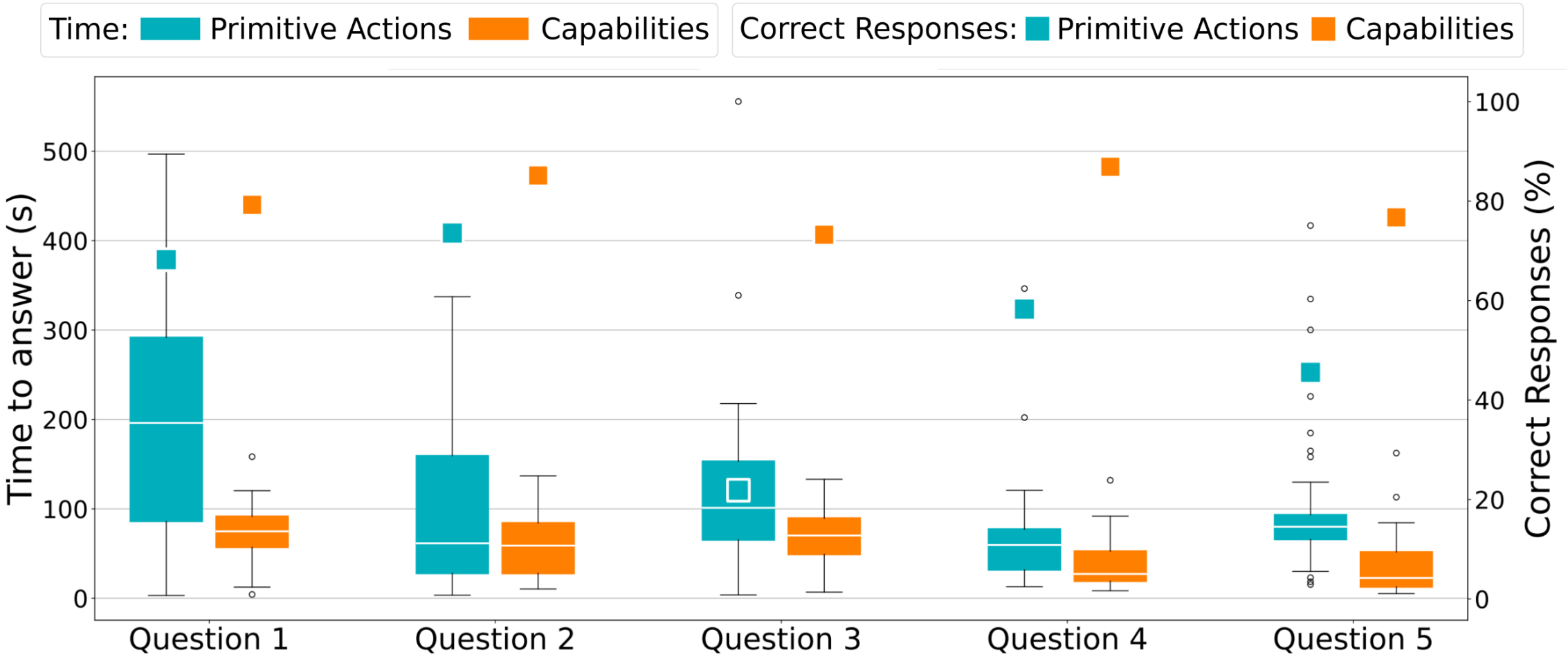
◯ C1 → C5 (Go next to Ganon → Go next to Key)

[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.

Paper

arxiv: 2107.13668

verma.pulkit@asu.edu