AAAI 2021 Workshop on Explainable Agency in Artificial Intelligence

Asking the Right Questions: Active Action-Model Learning

Pulkit Verma, Shashank Rao Marpally, Siddharth Srivastava

Arizona State University

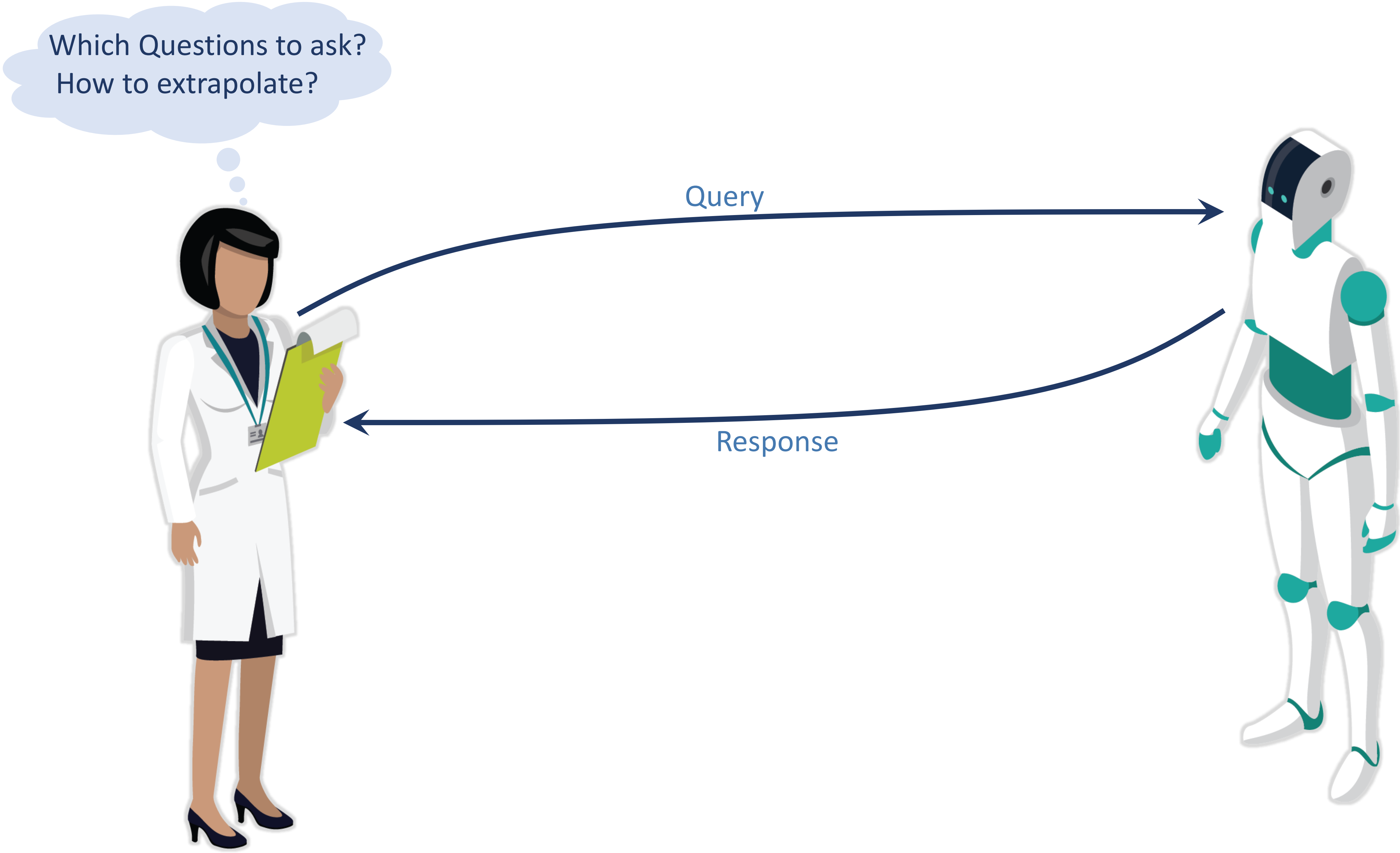
08 February 2021

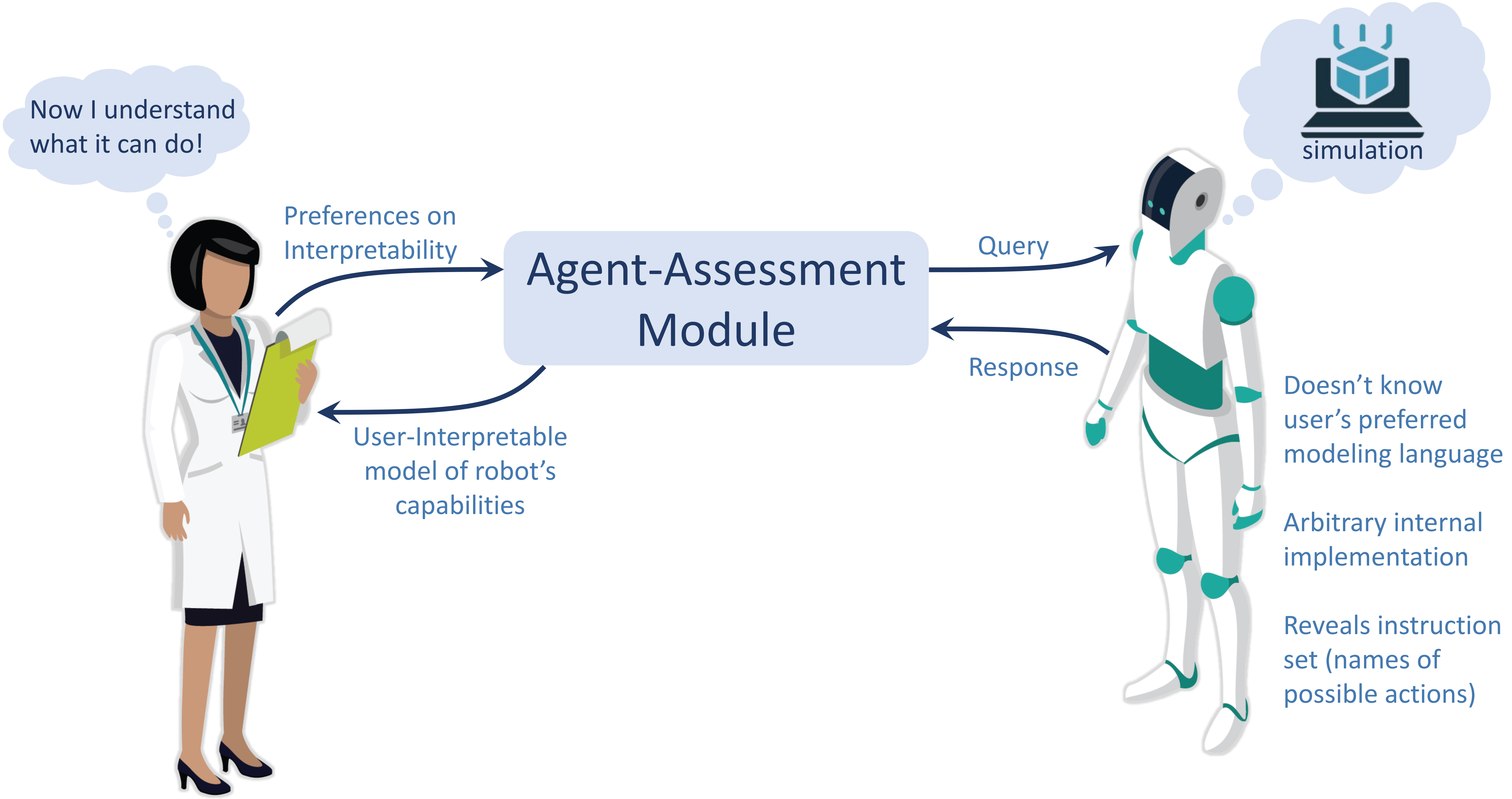
AAAI Autonomous Agents and Intelligent Robots
ASU Arizona State University

How Would End Users Assess Their AI Systems?

- How would a lay user determine whether an AI agent will be safe/reliable for a certain task?
- More challenging in settings where agent's internal code is not available (black-box).
- Can we get insights from how we assess humans in such situations?

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


Which Questions to ask?
How to extrapolate?
Query
Response


Now I understand what it can do!
Preferences on Interpretability
Agent-Assessment Module
User-Interpretable model of robot's capabilities
Query
Response
simulation
Doesn't know user's preferred modeling language
Arbitrary internal implementation
Reveals instruction set (names of possible actions)

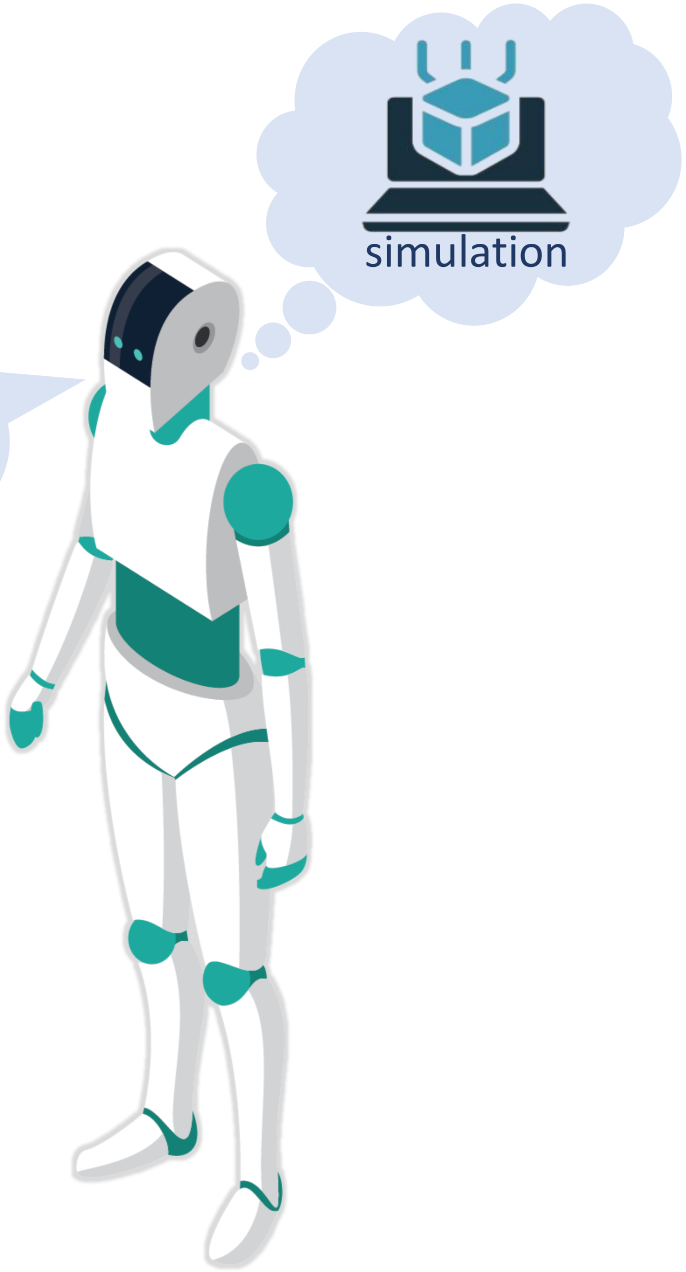
Example of an Interaction

Preferences on Interpretability

Agent-Assessment Module

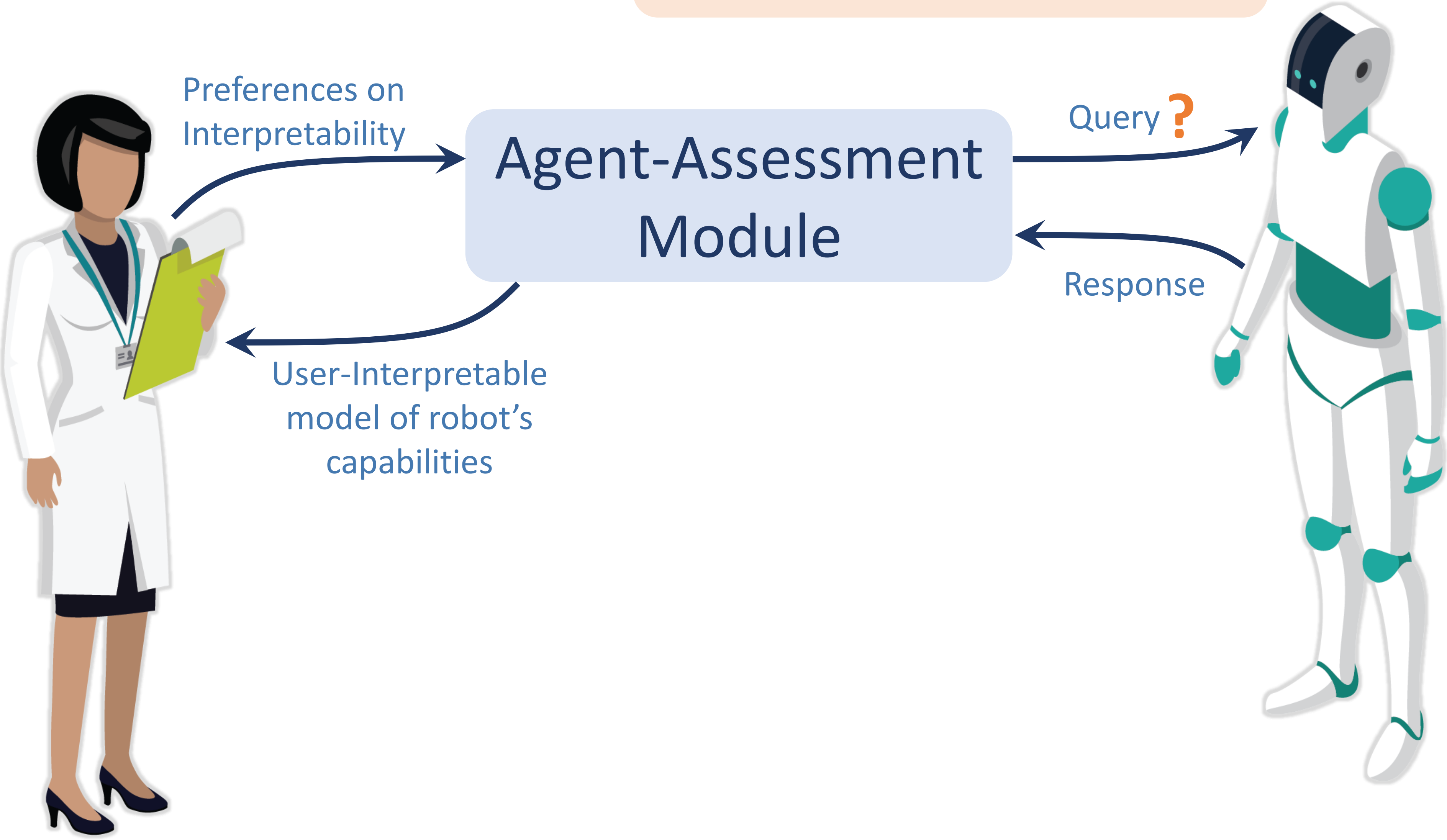
I can execute only the first step. After this both hands will be holding b1.

What do you think will happen if your hands were empty and you pickup beaker b1, then pickup beaker b2?



Key Challenges

- Which queries to ask?
- In which order to ask them?



Our Approach: Autonomous, User-Specific Agent Assessment

- Generate an interrogation policy.
- Use agent’s answers to queries to estimate a user-interpretable relational model.
- In this work: user-interpretable means STRIPS-like model.

Why STRIPS-like Modeling Language?

Advantages

- Support counterfactuals, assessment of causality
- Easy to convert into natural language text

Disadvantages

- Too many possible models ($9^{|\mathbb{P}| \times |\mathbb{A}|}$)
- $\mathbb{P}$: variable-instantiated predicates
- $\mathbb{A}$: parameterized actions

```
(:action pickup
  :parameters (?ob)
  :precondition (and (handempty)
                     (onrack ?ob))
  :effect (and (not (handempty))
               (not (onrack ?ob))
               (holding ?ob)))
```

[Deterministic, Fully Observable]

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

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?

Example of Model Abstraction

```
(:action pickup
  :parameters (?ob)
  :precondition (and (handempty)
                     (ontable ?ob))
  :effect (and (not (handempty))
               (not (ontable ?ob)))))
```

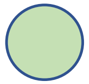
Concrete model

abstraction

```
(:action pickup
  :parameters (?ob)
  :precondition (and (handempty)
                     (+/-/Ø)(ontable ?ob))
  :effect (and (not (handempty))
               (not (ontable ?ob)))))
```

Abstract model

Algorithm for Hierarchical Query Synthesis



(:action pickup

:parameters (?ob)

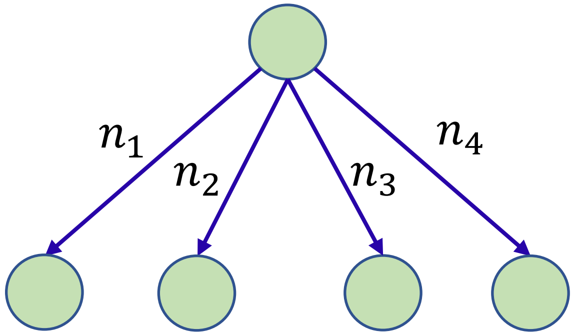
:precondition (and (+/-/∅) (handempty) n_1

(+/-/∅) (ontable ?ob)) n_2

:effect (and (+/-/∅) (handempty) n_3

(+/-/∅) (ontable ?ob))) n_4

Algorithm for Hierarchical Query Synthesis



(:action pickup

:parameters (?ob)

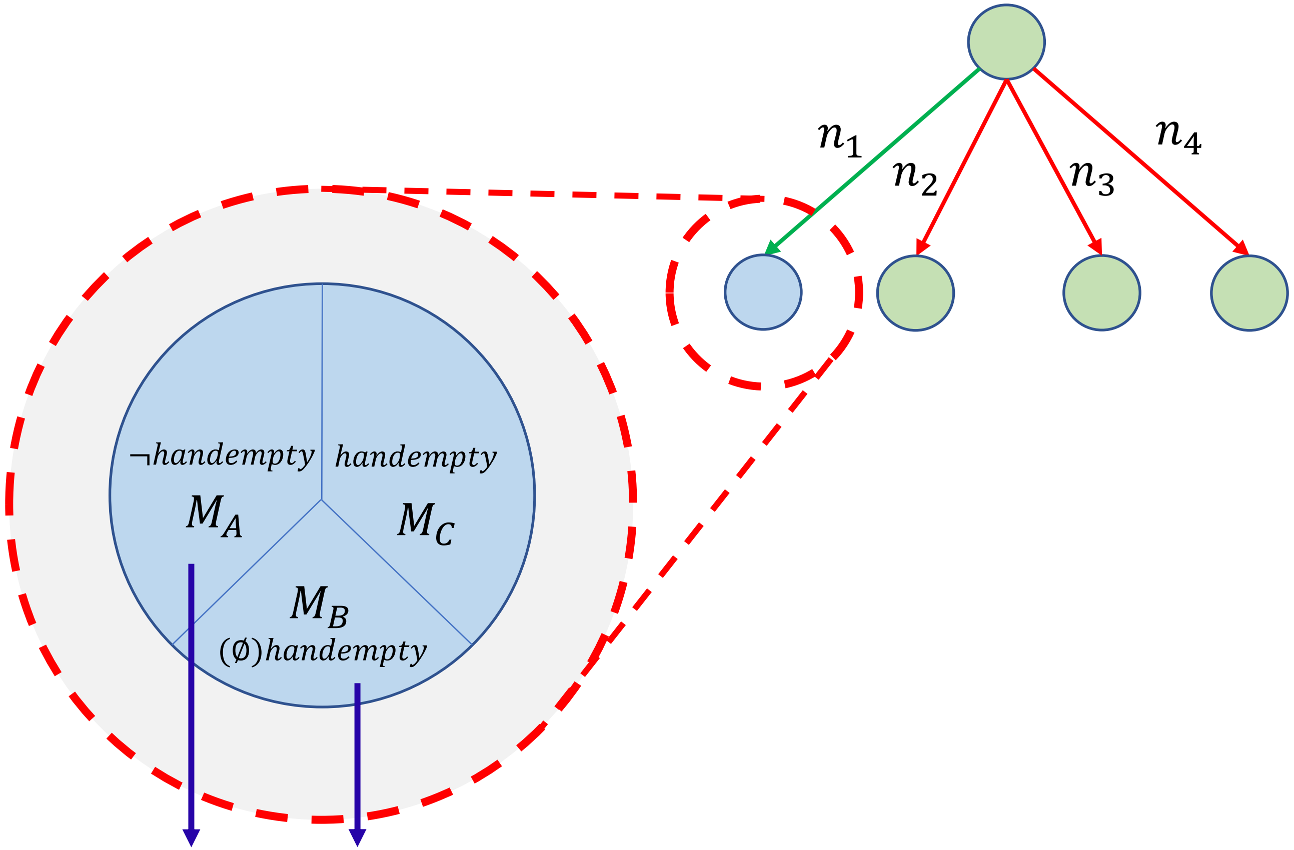
:precondition (and (+/-/∅) (handempty) n_1

(+/-/∅) (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))

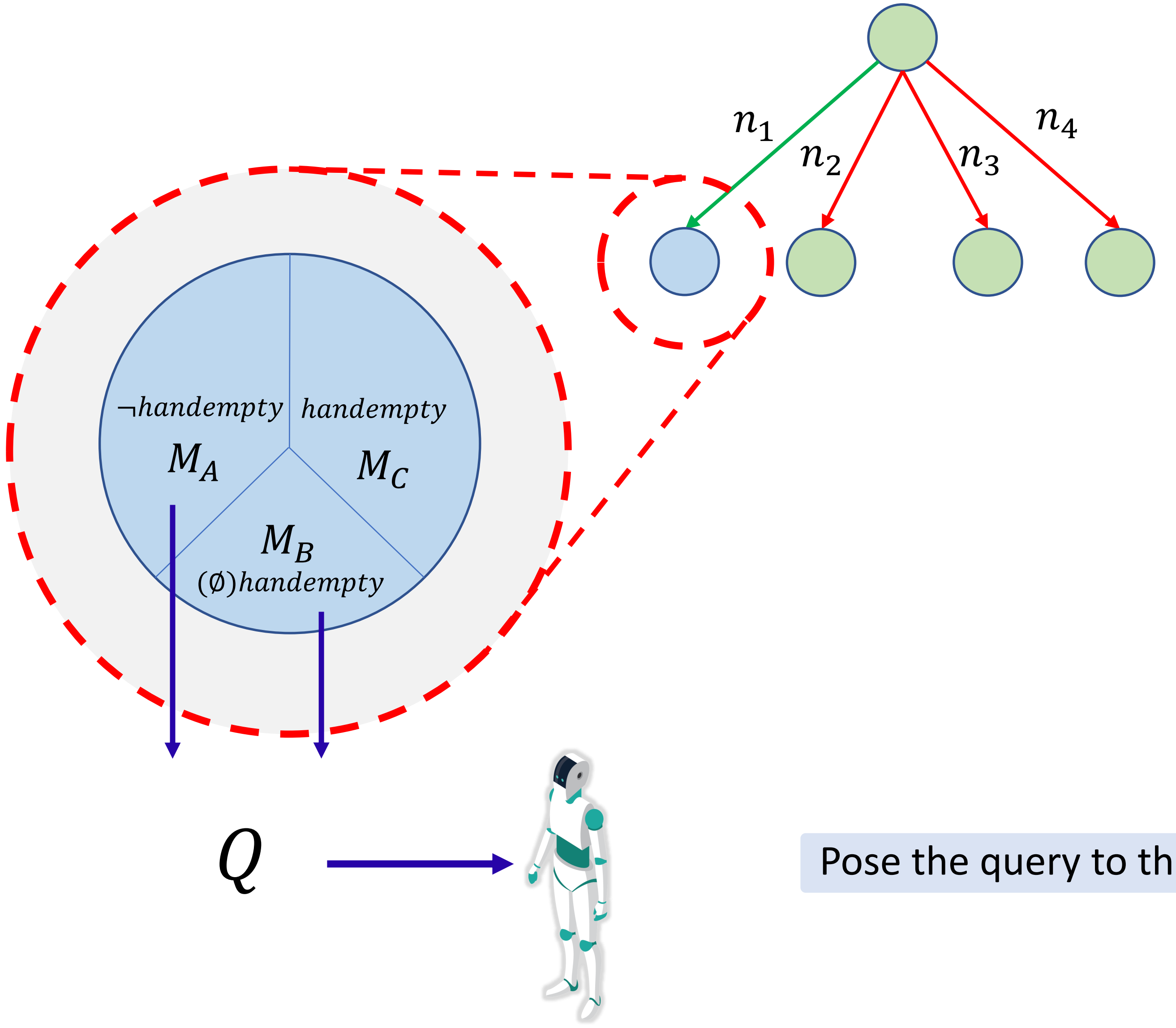
:effect (and (+/-/∅) (handempty)

(+/-/∅) (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)

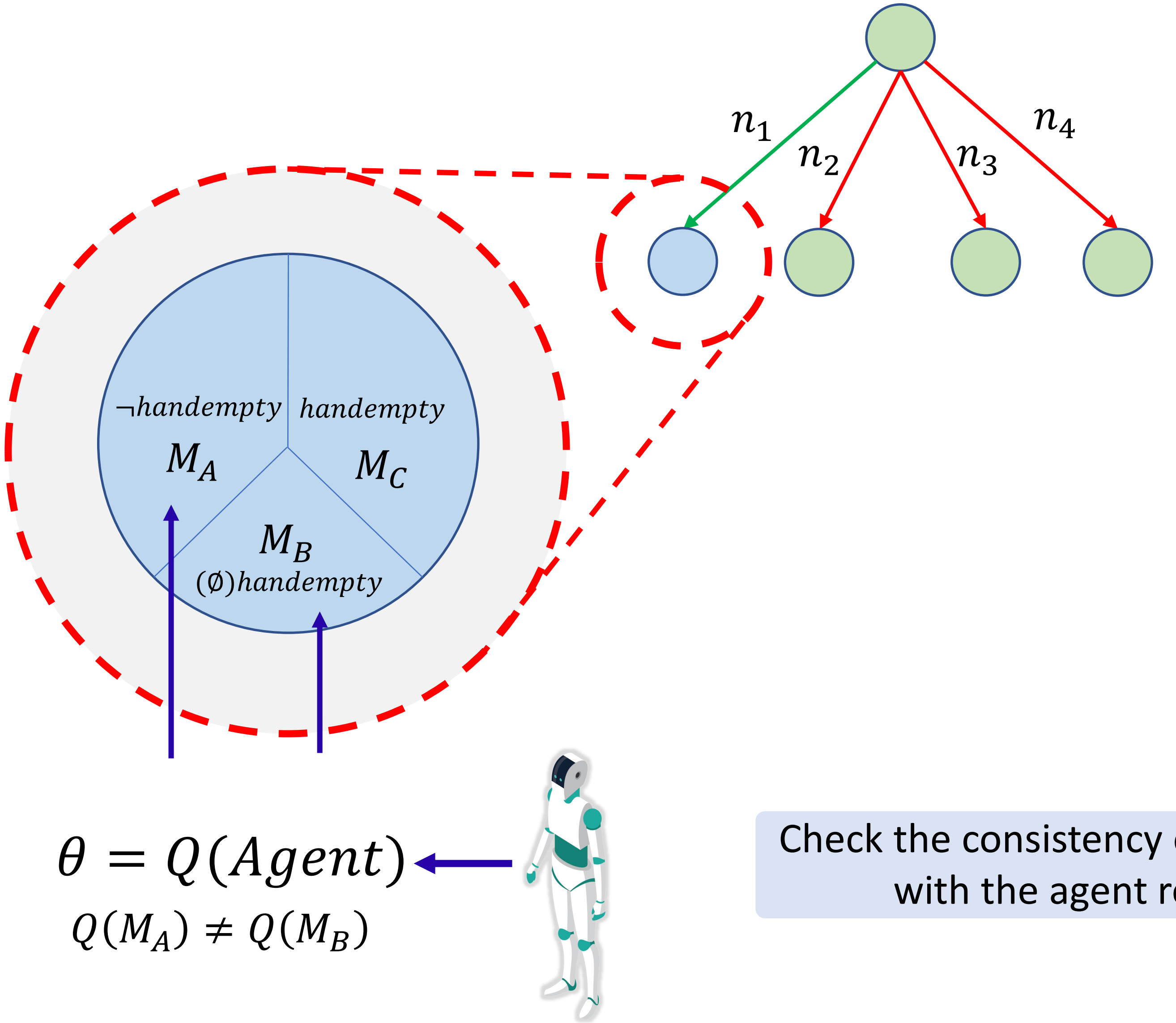
(+/-/∅) (ontable ?ob))

:effect (and (+/-/∅) (handempty)

(+/-/∅) (ontable ?ob)))

Pose the query to the agent

Algorithm for Hierarchical Query Synthesis



(:action pickup

:parameters (?ob)

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

(+/-/Ø) (ontable ?ob))

:effect (and (+/-/Ø) (handempty)

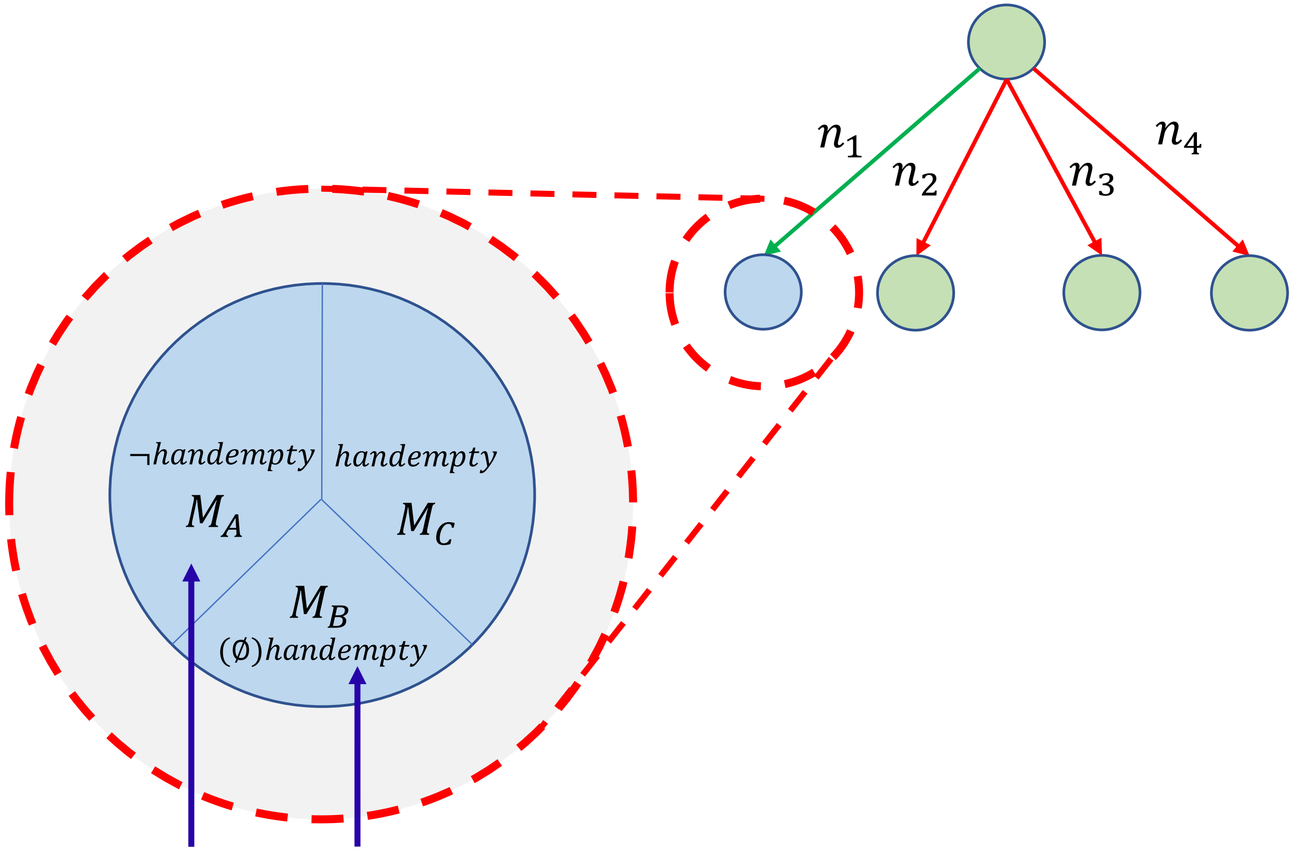
(+/-/Ø) (ontable ?ob)))

$\theta = Q(Agent)$

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

Check the consistency of refinements with the agent response

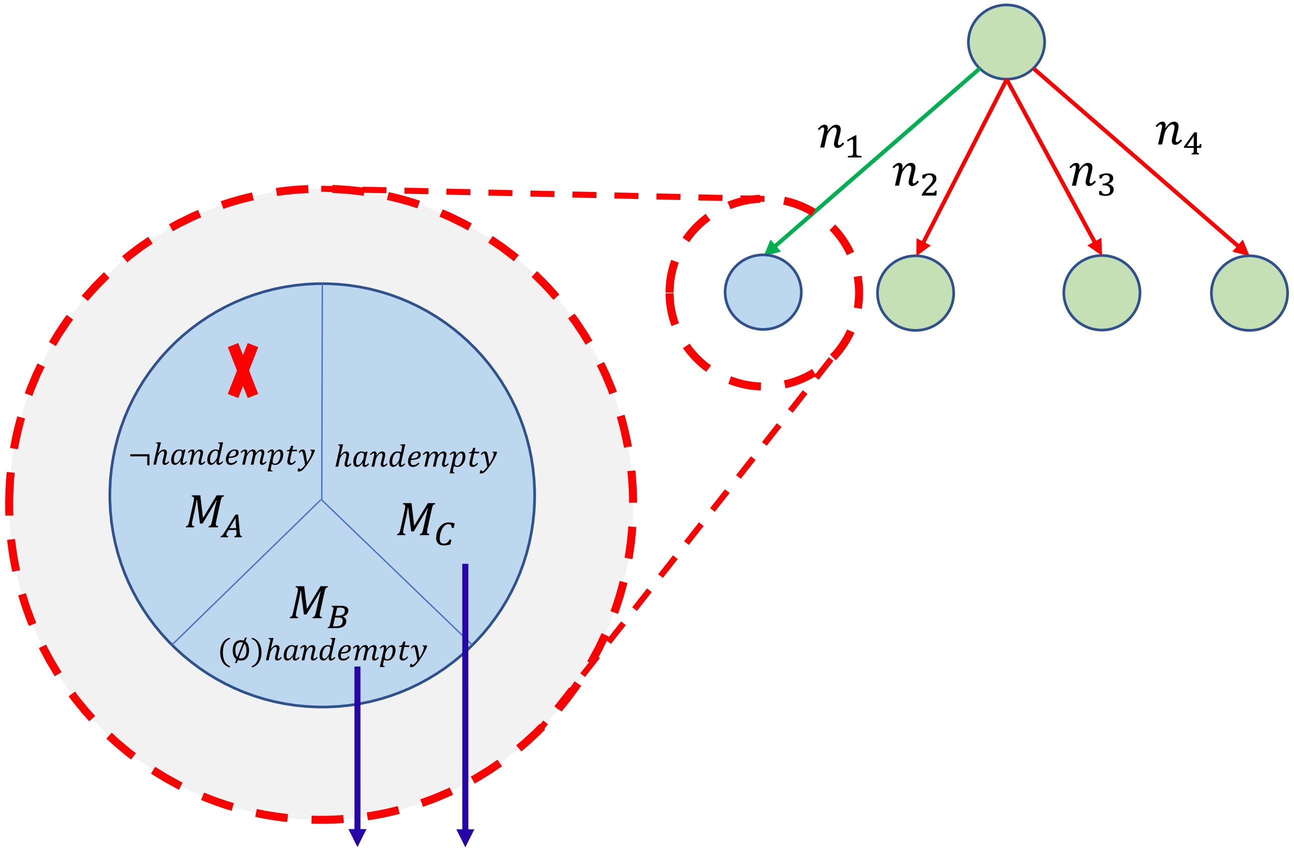
Algorithm for Hierarchical Query Synthesis



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

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

Algorithm for Hierarchical Query Synthesis



Generate a distinguishing query for these two refinements

(:action pickup

:parameters (?ob)

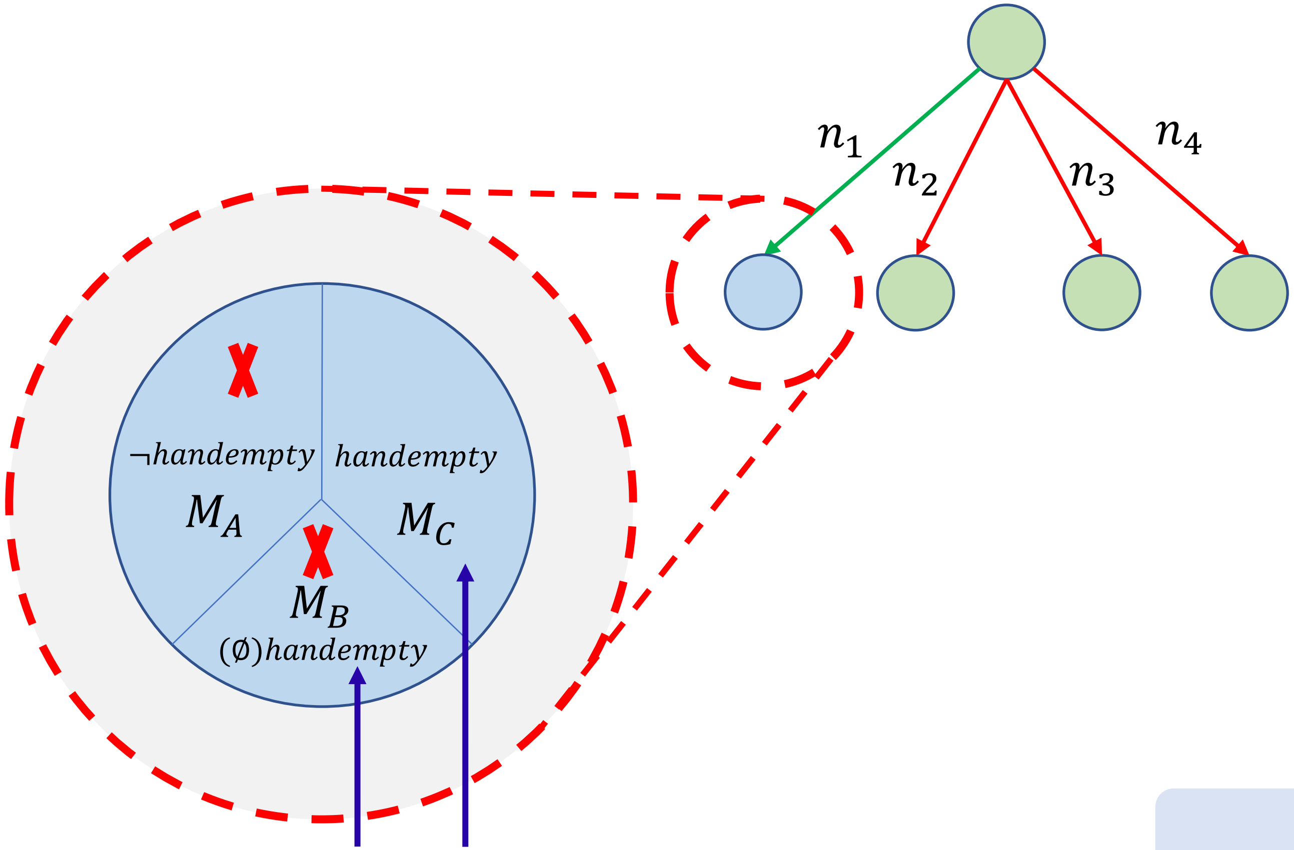
:precondition (and (+/-/∅) (handempty)

(+/-/∅) (ontable ?ob))

:effect (and (+/-/∅) (handempty)

(+/-/∅) (ontable ?ob)))

Algorithm for Hierarchical Query Synthesis



(:action pickup

:parameters (?ob)

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

(+/-/∅) (ontable ?ob))

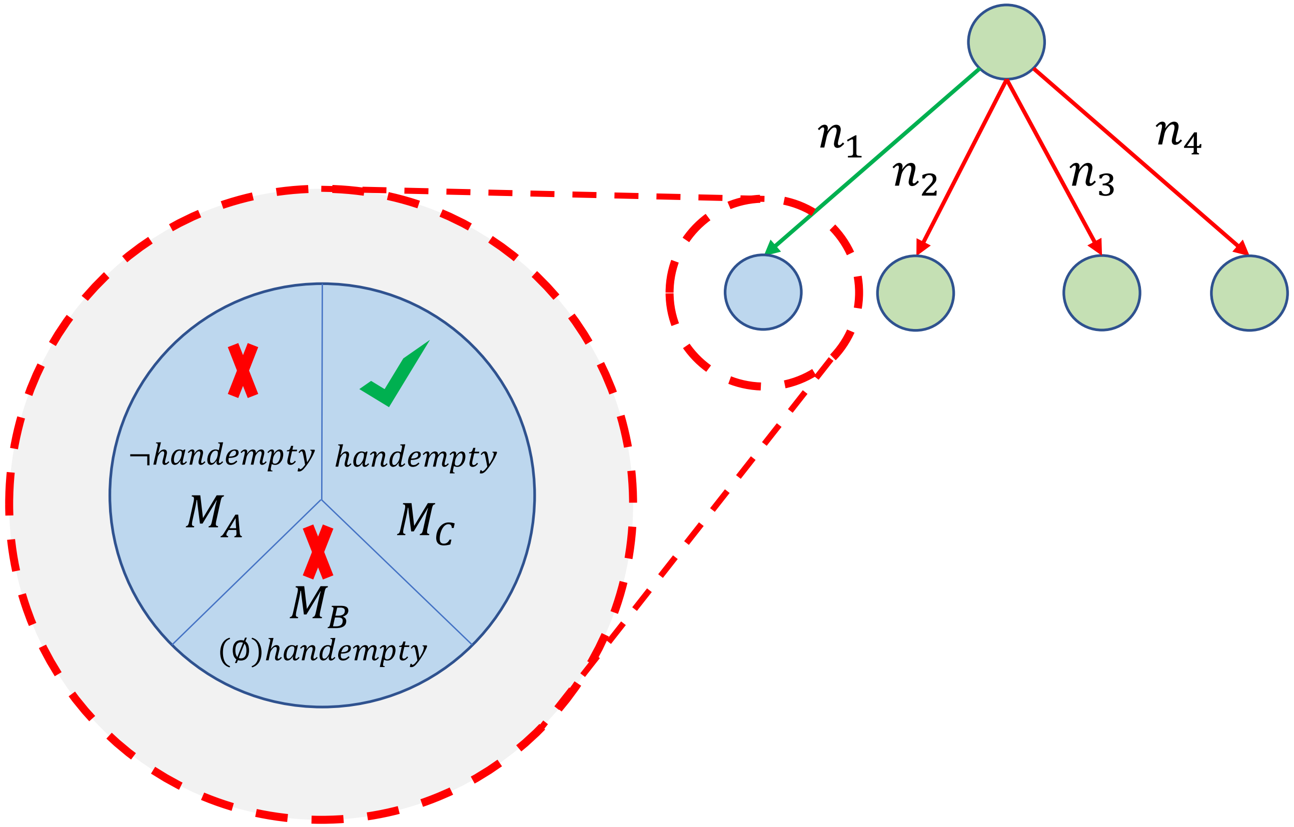
:effect (and (+/-/∅) (handempty)

(+/-/∅) (ontable ?ob)))

Reject the refinement
that is not consistent with the agent

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

Algorithm for Hierarchical Query Synthesis



(:action pickup

:parameters (?ob)

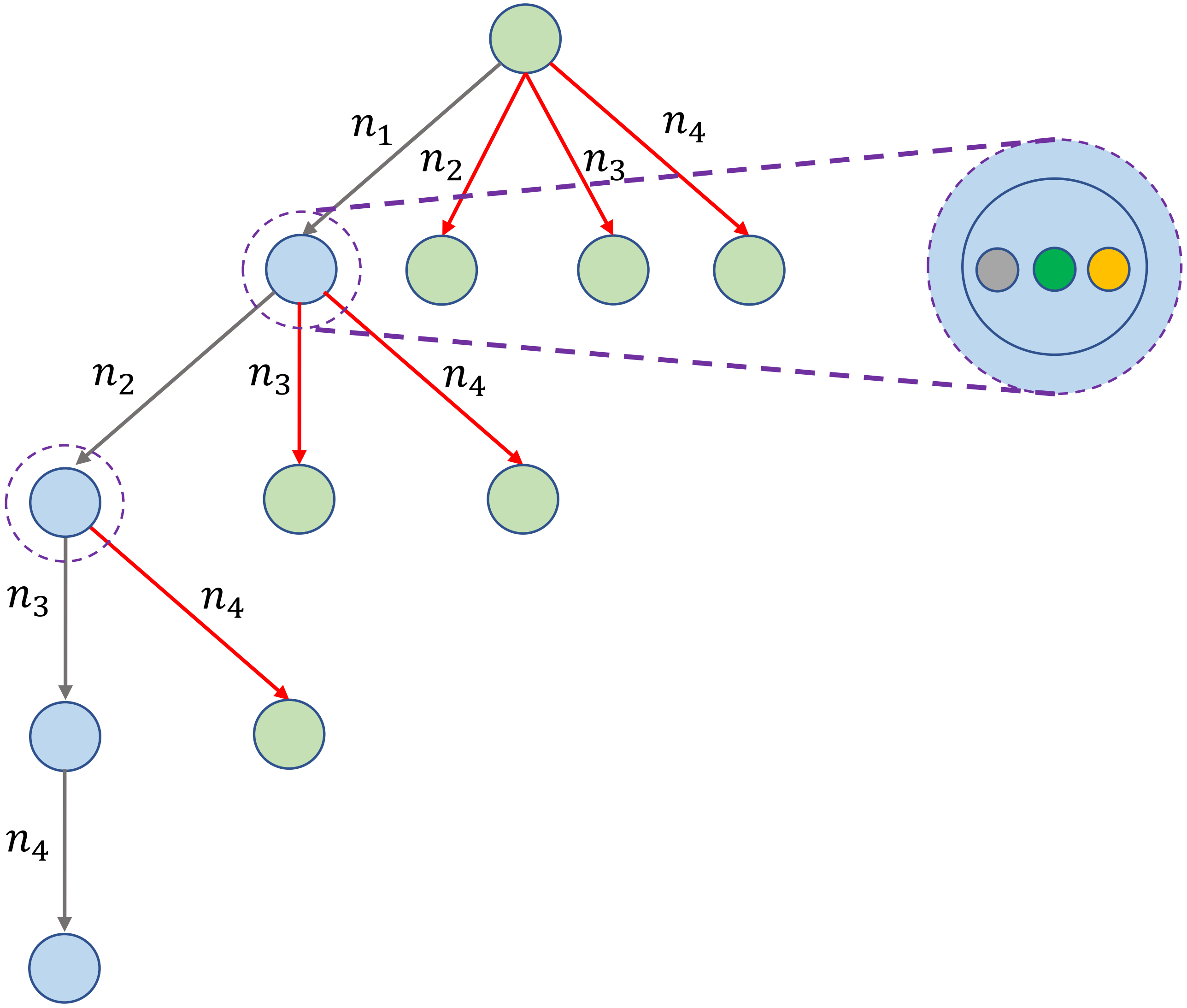
:precondition (and (handempty)

(+/-/∅) (ontable ?ob))

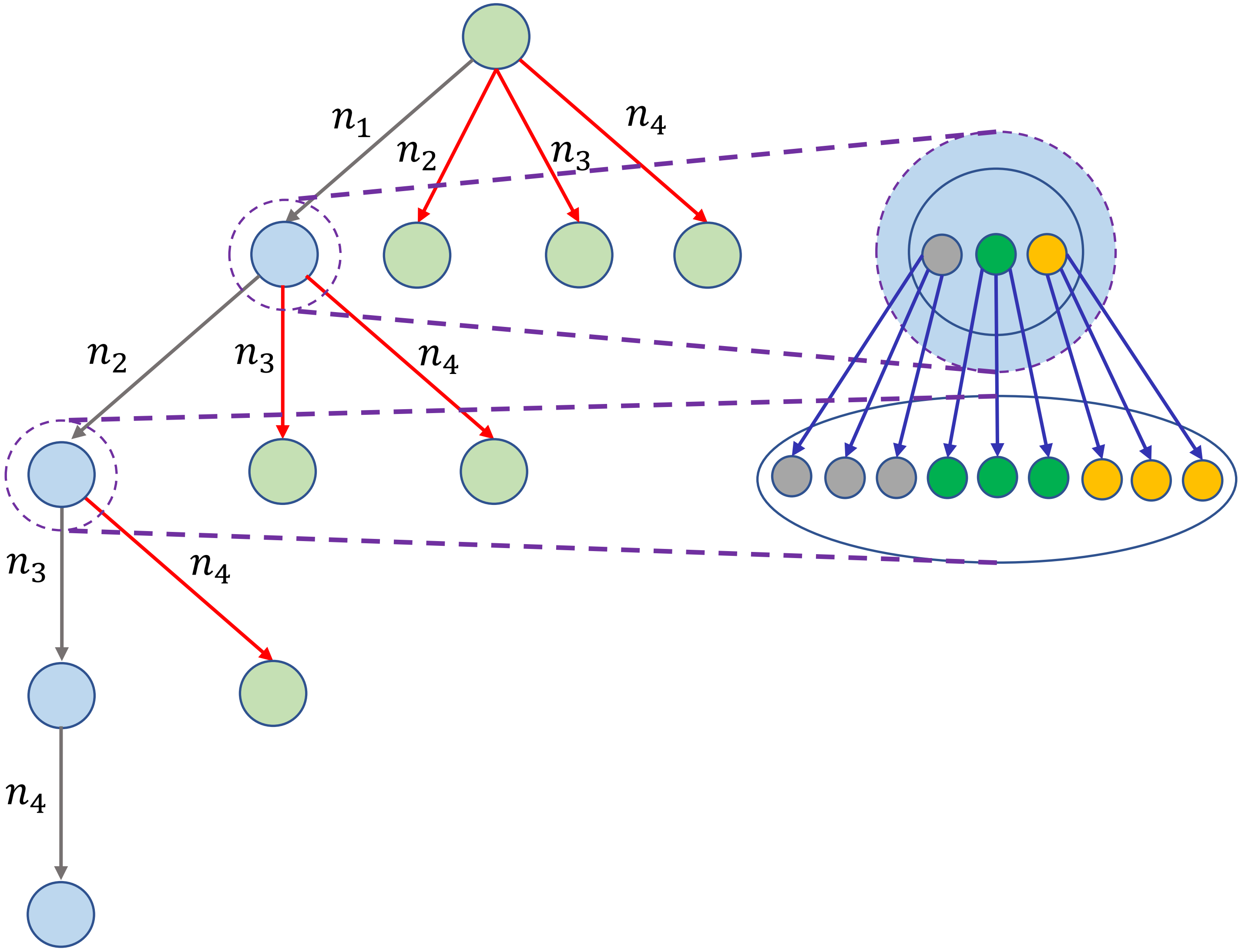
:effect (and (+/-/∅) (handempty)

(+/-/∅) (ontable ?ob)))

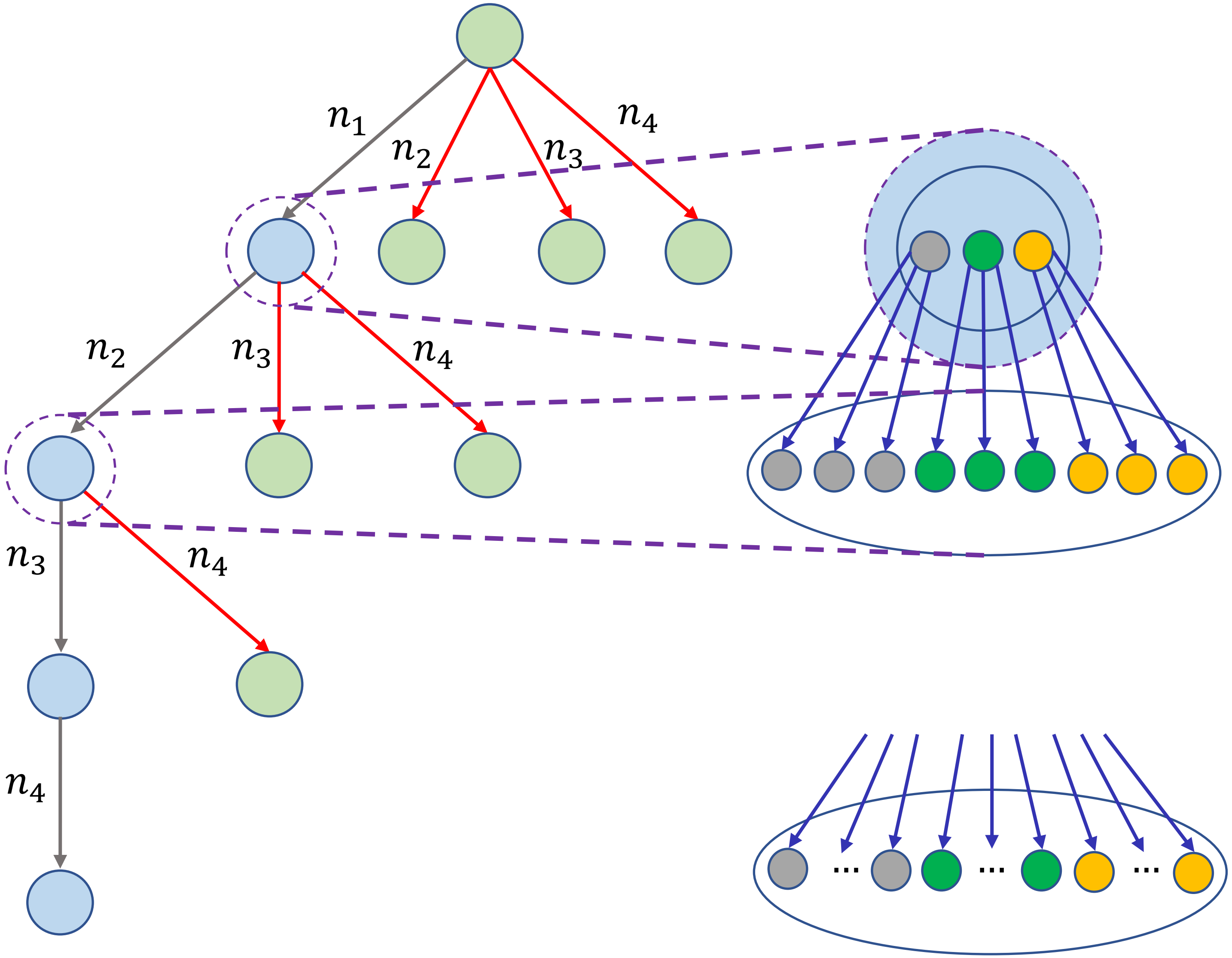
Algorithm for Hierarchical Query Synthesis



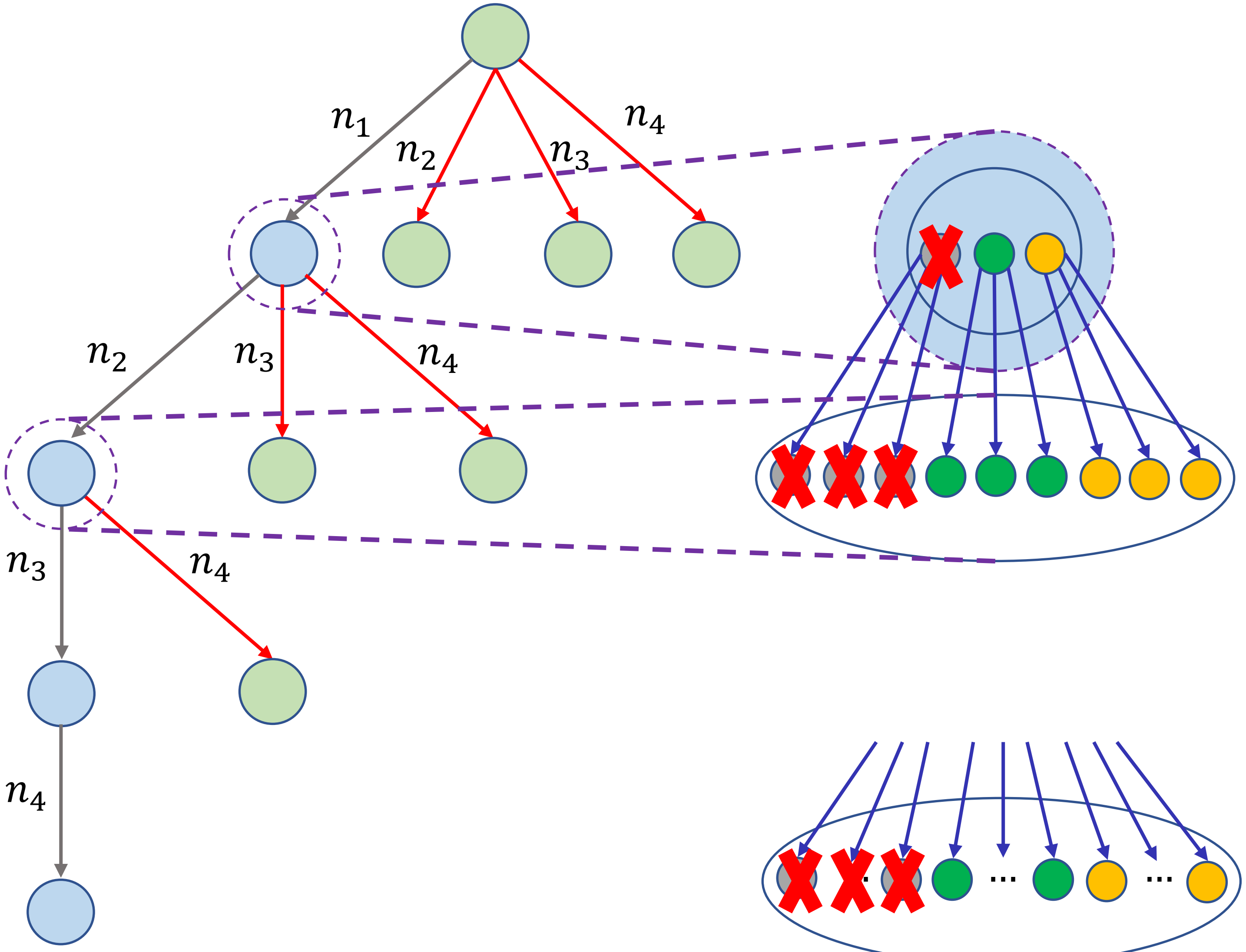
Algorithm for Hierarchical Query Synthesis



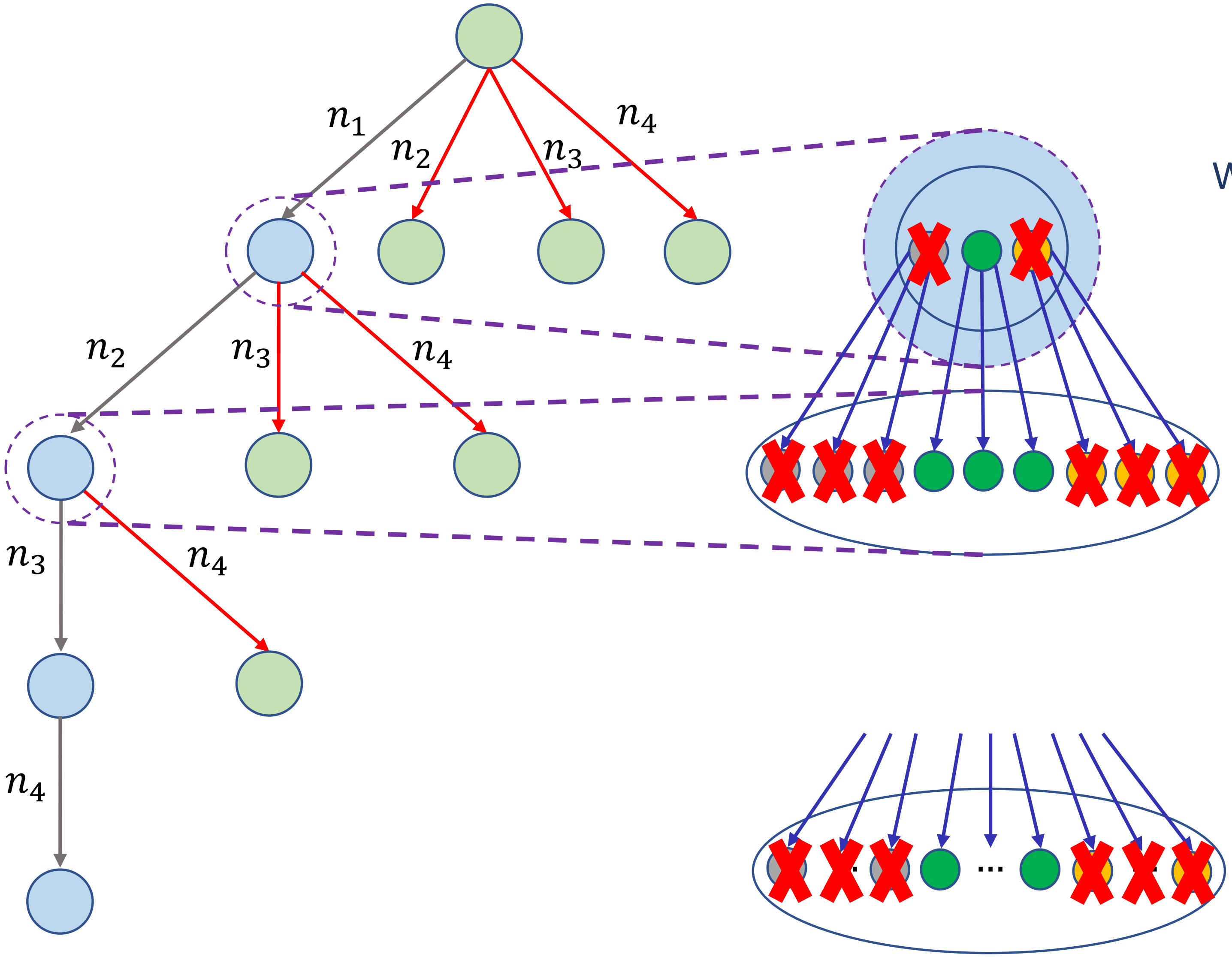
Algorithm for Hierarchical Query Synthesis



Algorithm for Hierarchical Query Synthesis



Algorithm for Hierarchical Query Synthesis



Key feature of the algorithm

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

Related Approaches

Several approaches have been developed for inferring interpretable models:

- Based on passively observed agent behavior.
    - E.g., ARMS – Yang et al. (AIJ 2007), LOCM - Cresswell et al. (ICAPS 2009), LOUGA - Kučera and Barták (KMAIS 2018), FAMA - Aineto et al. (AIJ 2019)
  - These approaches are susceptible to unsafe model inference.

- Based on explicit specification of the entire transition graph.
    - E.g., Bonet and Geffner (ECAI 2020)
  - This work requires no symbolic information but is more suited for smaller problems.

Experimental Setup

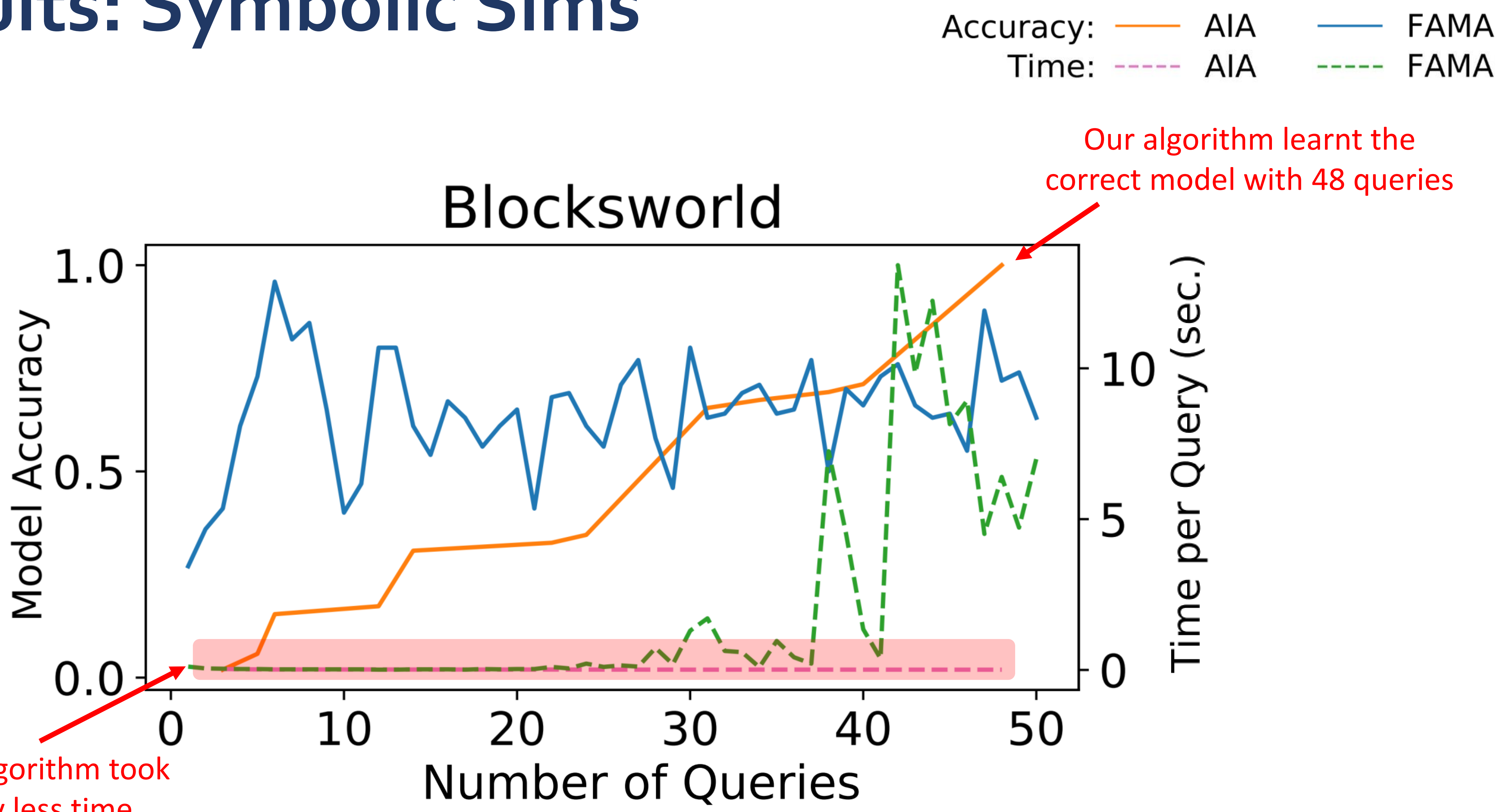
- 2 types of agents:
  - Symbolic sims: Agents that use symbolic, analytical models as simulators.
  - Black-box sims: Agents that use black-box models as simulators.
- Used FAMA[†] as baseline.

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

How we Evaluate Symbolic Sims?

- Randomly generate an agent and environment from IPC benchmark suite.
- Agent assessment module doesn’t get this information.
- Agent reports results as logic-based states (lists of grounded predicates that are true).
- Evaluate performance of agent assessment module and compare it with FAMA.

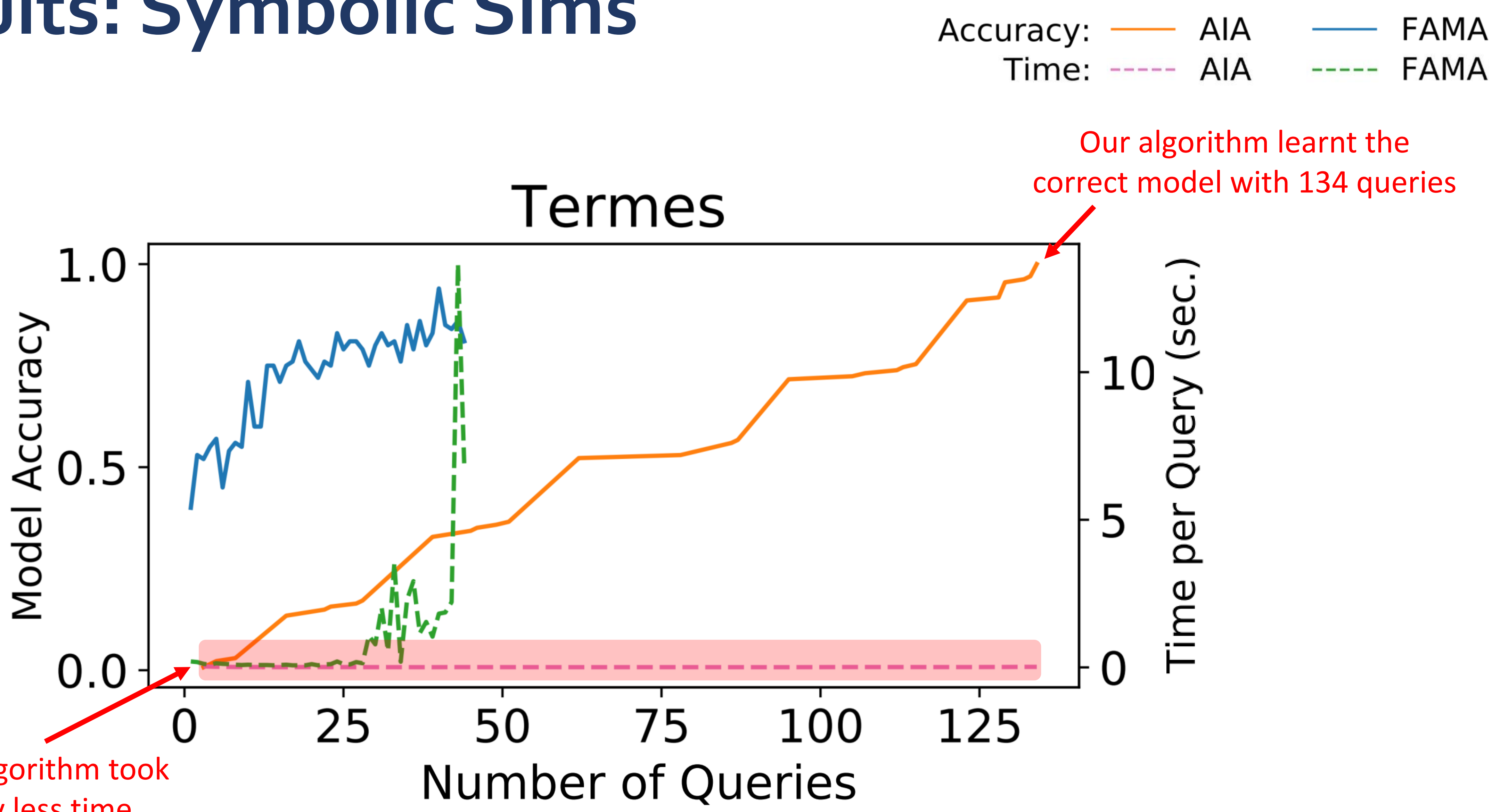
Results: Symbolic Sims



Our algorithm learnt the correct model with 48 queries

Our algorithm took very less time

Results: Symbolic Sims



Our algorithm learnt the correct model with 134 queries

Our algorithm took very less time

Results: Symbolic Sims

Domain	$\vert\mathbb{P}\vert$	$\vert\mathbb{A}\vert$	$\vert\hat{Q}\vert$	t_μ (ms)	t_σ (μs)
Gripper	5	3	17	18.0	0.2
Blocksworld	9	4	48	8.4	36
Miconic	10	4	39	9.2	1.4
Parking	18	4	63	16.5	806
Logistics	18	6	68	24.4	1.73
Satellite	17	5	41	11.6	0.87
Termes	22	7	134	17.0	110.2
Rovers	82	9	370	5.1	60.3
Barman	83	17	357	18.5	1605
Freecell	100	10	535	2242	3340

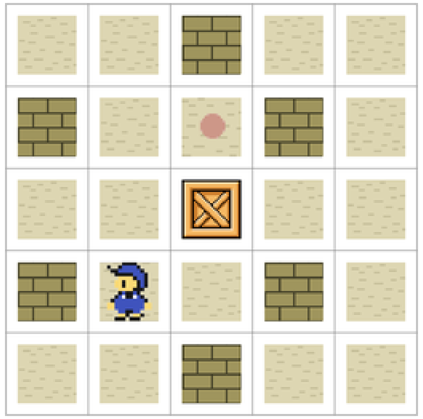
Results with FD planner:

- $|\mathbb{P}|$: Number of instantiated predicates in the domain
- $|\mathbb{A}|$: Number of actions in the domain
- $|\hat{Q}|$: Number of queries posed by the assessment module
- $|t_\mu|$: Average time per query
- $|t_\sigma|$: Variance in average time per query

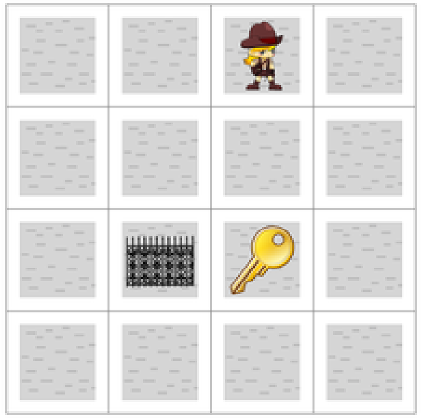
Average and variance are calculated for 10 runs of the algorithm, each on a separate problem.

How we Evaluate Black-Box Sims?

- Agent uses PDDLGym† to simulate the query plan.
- Reports results as images from the sim.
- Uses image classifiers for each predicate.
- Correctly inferred the model with:
  - 217 queries for Sokoban, and
  - 188 queries for Doors.



Sokoban



Doors

†Silver, T.; and Chitnis, R. 2020. *PDDLGym: Gym Environments from PDDL Problems*. In ICAPS 2020 PRL Workshop.

Conclusions

The proposed approach:

- Efficiently learns internal model of an agent in a STRIPS-like form.
- Needs no prior knowledge of the agent model.
- Only requires an agent to have rudimentary query answering capabilities.
- Learns the model accurately with a small number of queries.
- Can be extended :
  - to handle noisy image classifiers; or
  - to replace communication interface with natural language.

Conclusions

The proposed approach:

- Efficiently learns internal model of an agent in a STRIPS-like form.
- Needs no prior knowledge of the agent model.
- Only requires an agent to have rudimentary query answering capabilities.
- Learns the model accurately with a small number of queries.
- Can be extended :
  - to handle noisy image classifiers; or
  - to replace communication interface with natural language.

verma.pulkit@asu.edu

smarpall@asu.edu

bit.ly/3a0cmDI

Paper

bit.ly/3p4cVRu

AAAI Version†

†Verma, P.; Marpally S. R.; and Srivastava, S. *Asking the Right Questions: Learning Interpretable Action Models through Query Answering*. In AAAI 2021.