From Verbs to Tasks: An Integrated Account of Task Learning from Situated Interactive Instruction

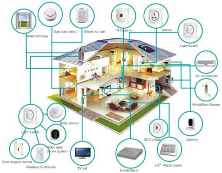
Shiwali Mohan

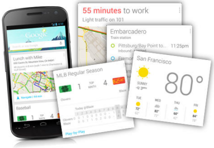
Computer Science and Engineering
University of Michigan, Ann Arbor

September 22nd, 2014

Autonomous Intelligent Collaborators

intelligent co-operative behavior in novel environments

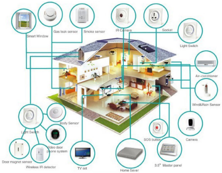


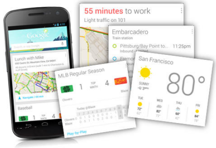


- Diverse tasks, environments, user preferences
- Users are not programmers

Autonomous Intelligent Collaborators

intelligent co-operative behavior in novel environments

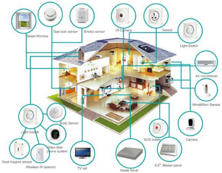


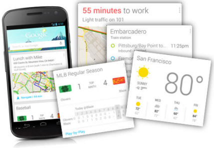


- Diverse tasks, environments, user preferences
- Users are not programmers
- How can agents be programmed by naive users?

Autonomous Intelligent Collaborators

intelligent co-operative behavior in novel environments





- Diverse tasks, environments, user preferences
- Users are not programmers
- How can agents be programmed by naive users?
- How to design agents that can extend their knowledge from natural interactions?

Interactive Task Learning

Agents that

learn new task specifications

- relevant objects, perceptual and spatial features
- goal and subgoals
- execution knowledge, policy, plan

from natural interactions

- dialog, gestures, demonstrations

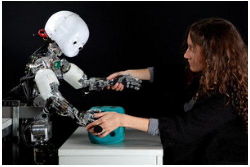
and perform these tasks in their environments.

- incorporate perceptual and cognitive reasoning for performance

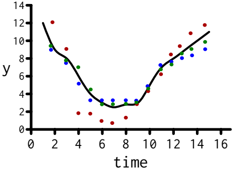
Prior Approaches

Prior Approaches

Kinesthetic Training

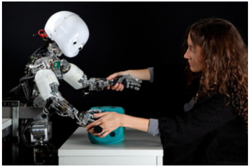


Argall et al. (2009)

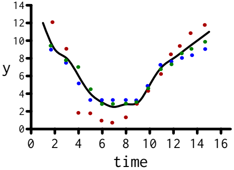


Prior Approaches

Kinesthetic Training



Argall et al. (2009)

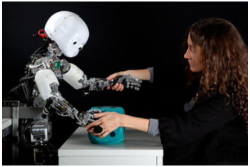


Interactive Reinforcement Learning

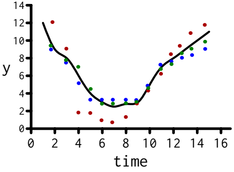
Thomaz et al. (2005)

Prior Approaches

Kinesthetic Training



Argall et al. (2009)



Interactive Reinforcement Learning

Thomaz et al. (2005)

Learning from Dialog

Mericli et al. (2014)

```
User : While landmark 1 is visible
Robot: What should I do in this loop?
User : Turn until landmark 1 is ahead
Robot: I will turn until I am facing
Landmark 1.
Robot: What should I do next?
User : Forward until 0.5 meters from
Landmark 1 max 0.2 meters
...
```

Task-oriented Dialog

Human-human task-oriented dialog
from Grosz and Sidner (1986)

```
Human1: First you have to remove the
        flywheel.
Human2: How do I remove the flywheel?
Human1: First, loosen the two allen
     setscrews holding it to the shaft,
        then pull it off.
Human2: OK. I can only find one screw.
        Where is the other one?
....
Human1: Use the wheelpuller. Do you know
        how to use it?
Human2: No.
Human1: Do you know how it looks like?
...
```

Task-oriented Dialog

Human-human task-oriented dialog
from Grosz and Sidner (1986)

```
Human1: First you have to remove the
        flywheel.
Human2: How do I remove the flywheel?
Human1: First, loosen the two allen
      setscrews holding it to the shaft,
        then pull it off.
Human2: OK. I can only find one screw.
        Where is the other one?
....
Human1: Use the wheelpuller. Do you know
        how to use it?
Human2: No.
Human1: Do you know how it looks like?
...
```

- *Situated*
  - communication occurs within the environmental and task context

Task-oriented Dialog

Human-human task-oriented dialog
from Grosz and Sidner (1986)

```
Human1: First you have to remove the
        flywheel.
Human2: How do I remove the flywheel?
Human1: First, loosen the two allen
     setscrews holding it to the shaft,
        then pull it off.
Human2: OK. I can only find one screw.
        Where is the other one?
....
Human1: Use the wheelpuller. Do you know
        how to use it?
Human2: No.
Human1: Do you know how it looks like?
...
```

- *Situated*
    - communication occurs within the environmental and task context
- *Interactive*
    - flexible dialog
    - distributed onus of learning

Task-oriented Dialog

Human-human task-oriented dialog
from Grosz and Sidner (1986)

```
Human1: First you have to remove the
        flywheel.
Human2: How do I remove the flywheel?
Human1: First, loosen the two allen
     setscrews holding it to the shaft,
        then pull it off.
Human2: OK. I can only find one screw.
        Where is the other one?
....
Human1: Use the wheelpuller. Do you know
        how to use it?
Human2: No.
Human1: Do you know how it looks like?
...
```

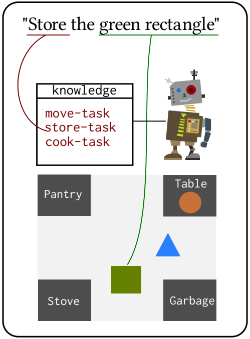
- *Situated*
  - communication occurs within the environmental and task context
- *Interactive*
  - flexible dialog
  - distributed onus of learning
- *Instruction*
  - diverse information types
  - incremental
  - situation specific and sparse

Rosie - An SII Agent

Research Problems

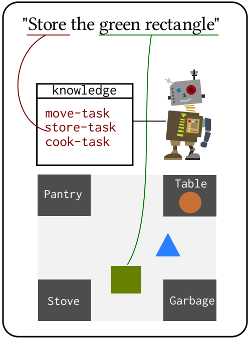
Research Problems

Situated Comprehension

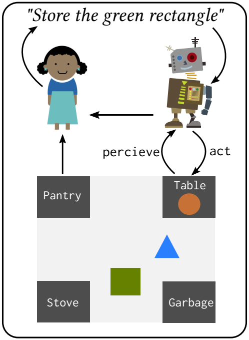


Research Problems

Situated Comprehension

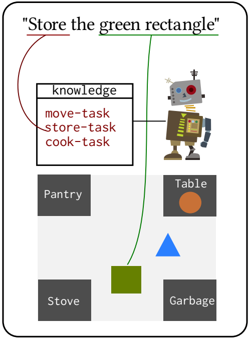


Integrative Interaction

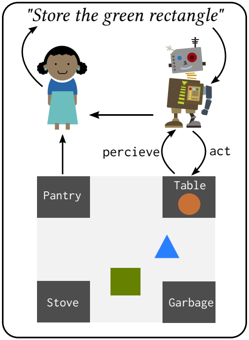


Research Problems

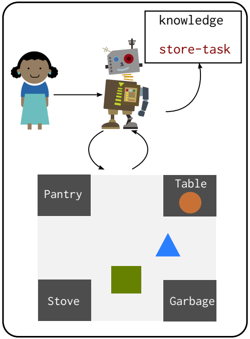
Situated Comprehension



Integrative Interaction



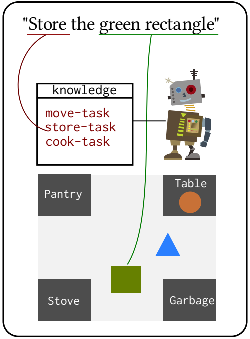
Task Learning



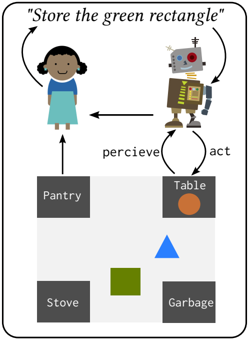
Research Problems

How can verbs be represented so that they are grounded in tasks?

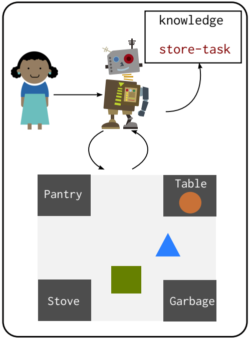
Situated Comprehension



Integrative Interaction



Task Learning



Scope of Tasks: Verbs for Domestic Chores

Cakmak, M. and Takayama, L. Towards a Comprehensive Chore List for Domestic Robots. *Proceedings of the Eighth International Conference on Human-Robot Interaction.* 2013.

Mohan, S. *Towards a Comprehensive Chore List for Kitchen Tasks.* 2013. URL: http://www.shiwali.me/kitchen-data.html.

Scope of Tasks: Verbs for Domestic Chores

- Task types
  - skill-oriented: *wipe*, *sweep*, *vacuum*
  - goal-oriented
    - functional: *cook*, *wash*
    - organizational: *set*, *organize*

Cakmak, M. and Takayama, L. Towards a Comprehensive Chore List for Domestic Robots. *Proceedings of the Eighth International Conference on Human-Robot Interaction.* 2013.

Mohan, S. *Towards a Comprehensive Chore List for Kitchen Tasks.* 2013. URL: `http://www.shiwali.me/kitchen-data.html`.

Scope of Tasks: Verbs for Domestic Chores

- Task types
  - skill-oriented: *wipe*, *sweep*, *vacuum*
  - goal-oriented [✓]
    - functional: *cook*, *wash*
    - organizational: *set*, *organize*

Cakmak, M. and Takayama, L. Towards a Comprehensive Chore List for Domestic Robots. *Proceedings of the Eighth International Conference on Human-Robot Interaction.* 2013.

Mohan, S. *Towards a Comprehensive Chore List for Kitchen Tasks.* 2013. URL: `http://www.shiwali.me/kitchen-data.html`.

Scope of Tasks: Verbs for Domestic Chores

- Task types
    - skill-oriented: *wipe*, *sweep*, *vacuum*
    - goal-oriented [✓]
        - functional: *cook*, *wash*
        - organizational: *set*, *organize*
- Goal types
    - achievement: a majority of verbs
        - prospective: *clean dishes after every meal*
    - maintenance: *keep*
    - perform

Cakmak, M. and Takayama, L. Towards a Comprehensive Chore List for Domestic Robots. *Proceedings of the Eighth International Conference on Human-Robot Interaction.* 2013.

Mohan, S. *Towards a Comprehensive Chore List for Kitchen Tasks.* 2013. URL: `http://www.shiwali.me/kitchen-data.html`.

Scope of Tasks: Verbs for Domestic Chores

- Task types
  - skill-oriented: *wipe*, *sweep*, *vacuum*
  - goal-oriented [✓]
    - functional: *cook*, *wash*
    - organizational: *set*, *organize*
- Goal types
  - achievement: a majority of verbs [✓]
    - prospective: *clean dishes after every meal*
  - maintenance: *keep*
  - perform

Cakmak, M. and Takayama, L. Towards a Comprehensive Chore List for Domestic Robots. *Proceedings of the Eighth International Conference on Human-Robot Interaction.* 2013.

Mohan, S. *Towards a Comprehensive Chore List for Kitchen Tasks.* 2013. URL: `http://www.shiwali.me/kitchen-data.html`.

Scope of Tasks: Verbs for Domestic Chores

- Task types
  - skill-oriented: *wipe*, *sweep*, *vacuum*
  - goal-oriented [✓]
    - functional: *cook*, *wash*
    - organizational: *set*, *organize*
- Goal types
  - achievement: a majority of verbs [✓]
    - prospective: *clean dishes after every meal*
  - maintenance: *keep*
  - perform
- Implicit information
  - implicit goal
    - *organize the pantry.*
  - implicit parameters
    - *put away the book (on the shelf).*
    - *put away the toys (in the box).*

Cakmak, M. and Takayama, L. Towards a Comprehensive Chore List for Domestic Robots. *Proceedings of the Eighth International Conference on Human-Robot Interaction.* 2013.

Mohan, S. *Towards a Comprehensive Chore List for Kitchen Tasks.* 2013. URL: `http://www.shiwali.me/kitchen-data.html`.

Scope of Tasks: Verbs for Domestic Chores

- Task types
  - skill-oriented: *wipe*, *sweep*, *vacuum*
  - goal-oriented [✓]
    - functional: *cook*, *wash*
    - organizational: *set*, *organize*
- Goal types
  - achievement: a majority of verbs [✓]
    - prospective: *clean dishes after every meal*
  - maintenance: *keep*
  - perform
- Implicit information
  - implicit goal [✓]
    - *organize the pantry.*
  - implicit parameters [✓]
    - *put away the book (on the shelf).*
    - *put away the toys (in the box).*

Cakmak, M. and Takayama, L. Towards a Comprehensive Chore List for Domestic Robots. *Proceedings of the Eighth International Conference on Human-Robot Interaction.* 2013.

Mohan, S. *Towards a Comprehensive Chore List for Kitchen Tasks.* 2013. URL: `http://www.shiwali.me/kitchen-data.html`.

Situated Comprehension

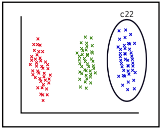
Situated Comprehension

- the Indexical Hypothesis (Glenberg and Robertson, 1999)
    - linguistic communication is reference to elements in common ground
    - language specifies the scene, domain knowledge fills up details

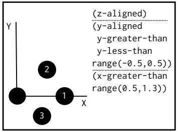
Situated Comprehension

- the Indexical Hypothesis (Glenberg and Robertson, 1999)
  - linguistic communication is reference to elements in common ground
  - language specifies the scene, domain knowledge fills up details
- Our approach: the Indexical Model

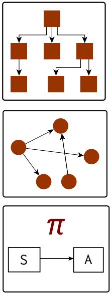
perceptual



spatial



task



attentional processes

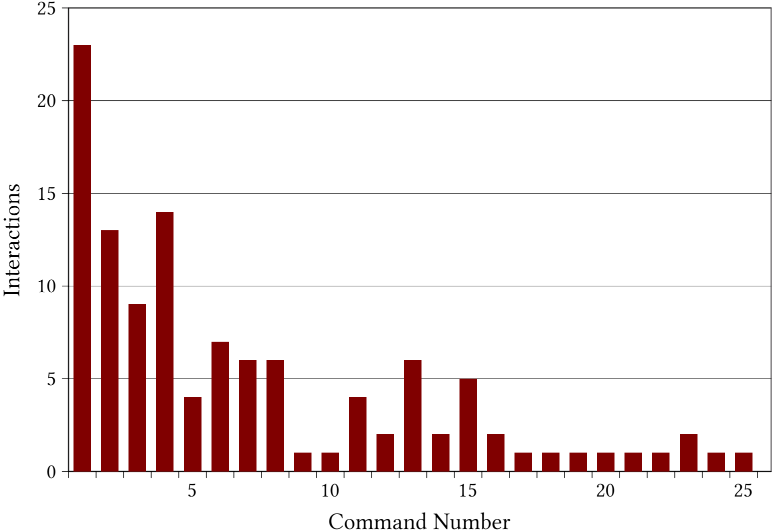
$$\ln(\sum t_j^{-d})$$

dialog

Store the green object.
What is the goal of this task?
The goal is ....
What should I do next?
Open the pantry
....

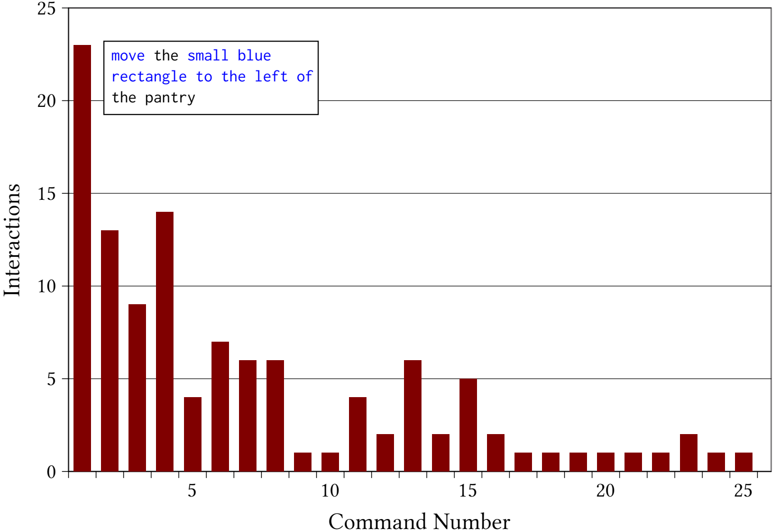
Scaling with Learning

9 nouns/adjectives, 3 prepositions, 3 verbs (100% correct execution)



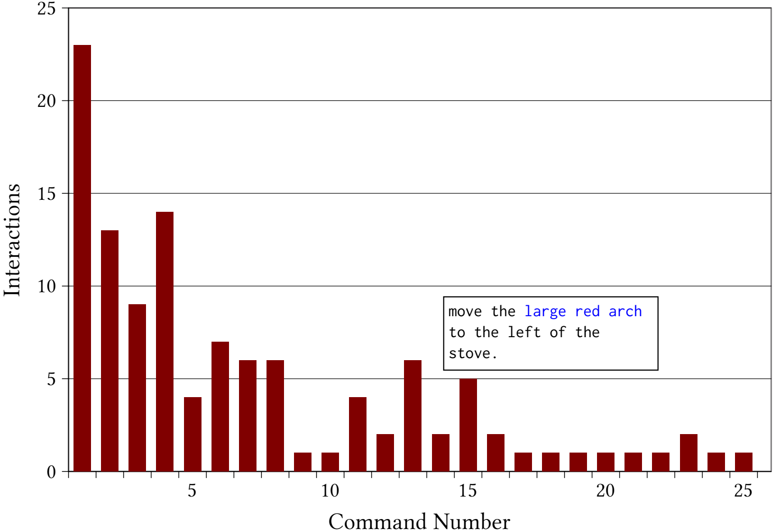
Scaling with Learning

9 nouns/adjectives, 3 prepositions, 3 verbs (100% correct execution)



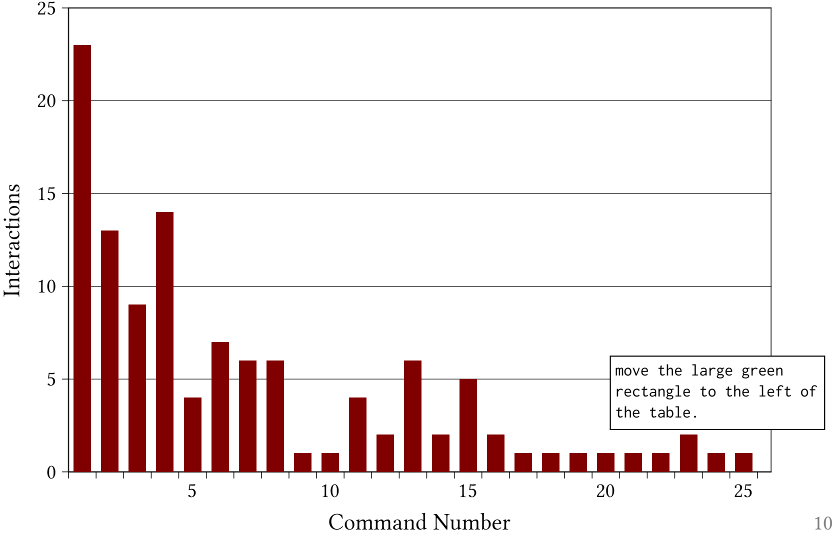
Scaling with Learning

9 nouns/adjectives, 3 prepositions, 3 verbs (100% correct execution)



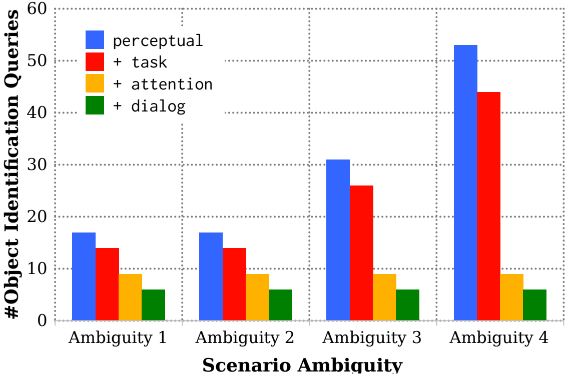
Scaling with Learning

9 nouns/adjectives, 3 prepositions, 3 verbs (100% correct execution)



Reference Resolution

- Corpus: instructional dialogs,
  12 *it*,
  4 *this*,
  3 *that cylinder*
  14 *the red cylinder*
- Queries
  Instructor: Pick it.
  Agent: Which object?
- Linguistic context only
  Stanford CoreNLP fails at 28.6% of references.



Unexpressed Argument Alternations

- move
  a. *Move the red cylinder to the right of the table.*
  b. *Move the red cylinder to the table.*
- store
  a. *Store the red cylinder in the pantry.*
  b. *Store the red cylinder.*
- cook
  a. *Cook the steak on the stove.*
  b. *Cook the steak.*

Integrative Interaction

- Collaborative Discourse Theory (Grosz and Sidner, 1986)
    - communication is goal-oriented

Integrative Interaction

- Collaborative Discourse Theory (Grosz and Sidner, 1986)
  - communication is goal-oriented
- Our approach: interaction model, extends Rich and Sidner, 1998
  - events, segments, stack

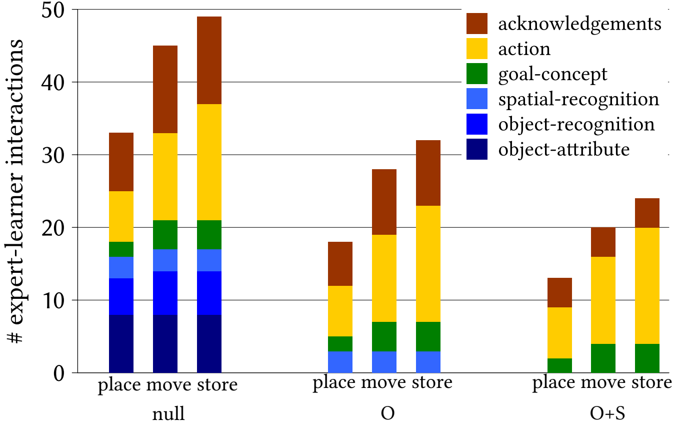
get-next-task	dialog-event	Agent: Waiting for the next command...	
verb-command	dialog-event	Human: Store the orange object.	
attribute-q	dialog-event	Agent: What kind of attribute is orange?	
attribute-ans	dialog-event	Human: A color	O11
	learning-event	Agent: (learns semantic categorization)	
	dialog-event	Agent: I don't see an orange object. Please teach me to recognize one	O12
	dialog-event	Human: (clicks on the object). This is orange.	
	learning-event	Agent: (learns perceptual symbol + class)	
get-goal	dialog-event	Agent: What is the goal of the action?	
goal-semantics	dialog-event	Human: The goal is the orange object in the pantry.	
relation-q	dialog-event	Agent: I don't know the preposition in.	G12
		Human: The orange object is in the garbage.	A1
relation-ans	learning-event	Agent: (learns semantic composition of 'in')	P121
	learning-event	Agent: (learns goal of 'store')	
	dialog-event	Agent: What action should I take next?	

P121: learn preposition *in*

G12: acquire goal for store

A1: external action store

Flexible Dialog



Instruction Strategy

Agent controlled instruction

```
Mentor: set the table
Agent: This is a new task for me. What is the goal of this task?
Mentor: the goal is the purple object on the table and
        the green object right of the purple object
Agent: I don't know the preposition right-of.
Mentor: the green object is right of the blue object
Agent: What kind of attribute is green?
Mentor: a color
...
```

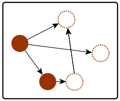
Instructor controlled instruction

```
...
Mentor: the red object is on the table
Agent: Okay.
Mentor: the blue object is right of the green object
Agent: Okay.
Mentor: place the blue object on the table
Agent: This is a new task for me. What is the goal of this task?
...
Mentor: set the table
Agent: This is a new task for me. What is the goal of this task?
...
```

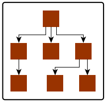
Interactive Task Learning

Interactive Task Learning

characteristics



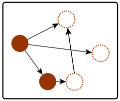
relational goals



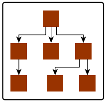
hierarchical decomposition

Interactive Task Learning

characteristics



relational goals



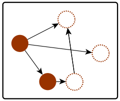
hierarchical decomposition

acquire

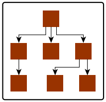
- task structure
- execution knowledge

Interactive Task Learning

characteristics



relational goals



hierarchical decomposition

acquire

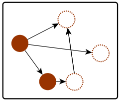
- task structure
- execution knowledge

desiderata

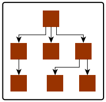
- multi-task learning
- fast generalization
- transfer
- distributed initiative

Interactive Task Learning

characteristics



relational goals



hierarchical decomposition

acquire

- task structure
- execution knowledge

desiderata

- multi-task learning
- fast generalization
- transfer
- distributed initiative

approach

- composable, hierarchical, multi-modal representation
- knowledge-intensive machine learning - EBL

Knowledge to be Learned

For the task: *store the <u>red arch</u>*

What?

- explicit parameters
  `store(object:[o])`
- implicit parameters
  `[o]:arch -> location:pantry`
- goal
  `in(O2,pantry)` $\wedge$ `closed(pantry)`
- subtasks
  `store`: `open`, `place [pick-up, put-down]`, `close`

How?

- policy
  if `[state,task]` then `execute([subtask])`
- model
  if `[state,task]` then `[next-state]`

When?

- availability
  if `[state]` then `available(store)`
- termination
  if `[state]` then `terminate(store)`

Knowledge to be Learned

For the task: *store the <u>red arch</u>*

What? *semantic memory*	• explicit parameters `store(object:[o])` • implicit parameters `[o]:arch -> location:pantry` • goal `in(O2,pantry)` ∧ `closed(pantry)` • subtasks `store`: `open`, `place [pick-up, put-down]`, `close`
How?	• policy if `[state,task]` then `execute([subtask])` • model if `[state,task]` then `[next-state]`
When?	• availability if `[state]` then `available(store)` • termination if `[state]` then `terminate(store)`

Knowledge to be Learned

For the task: *store the red arch*

What? *semantic memory*

- explicit parameters
  `store(object:[o])`
- implicit parameters
  `[o]:arch -> location:pantry`
- goal
  `in(O2,pantry)` ∧ `closed(pantry)`
- subtasks
  `store`: `open`, `place [pick-up, put-down]`, `close`

How? *procedural memory*

- policy
  if `[state,task]` then `execute([subtask])`
- model
  if `[state,task]` then `[next-state]`

When? *procedural memory*

- availability
  if `[state]` then `available(store)`
- termination
  if `[state]` then `terminate(store)`

Overview

2 step process

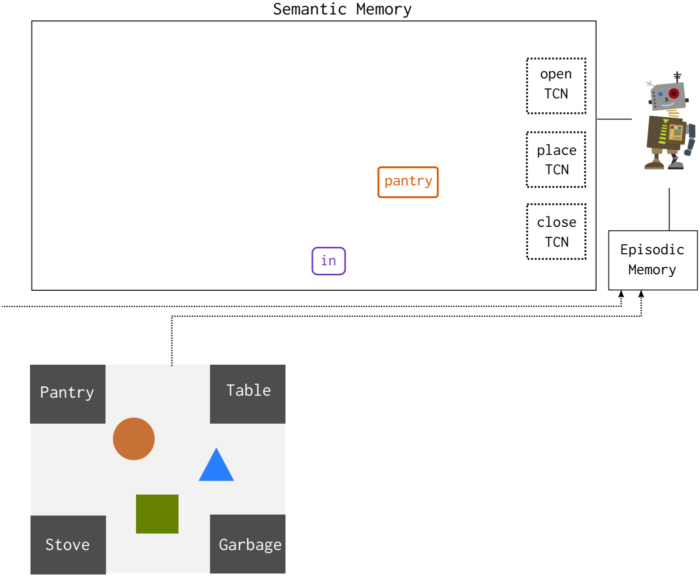
1. interactive execution
   - accumulate examples
   - explore autonomously
2. retrospective explanation
   - generalize specific experience
   - assimilate instructions

Looking ahead

- learn task execution knowledge
  - goals, hierarchy, policy
- learn implicit parameters
  - associative default values

Interactive Execution

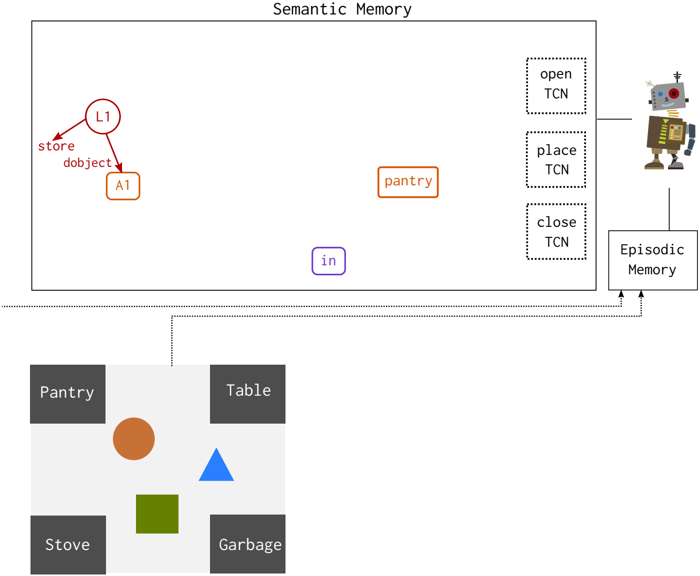
Interaction trace
Instructor: Store the green rectangle.



Interactive Execution

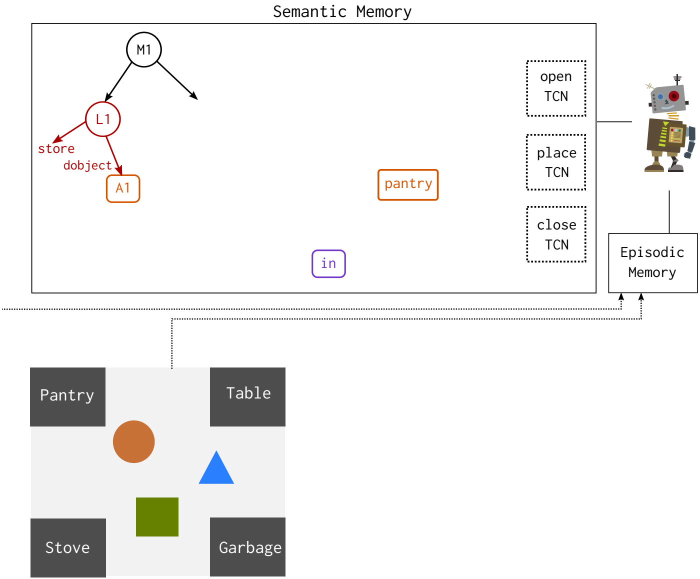
Interaction trace

Instructor: Store the green rectangle.



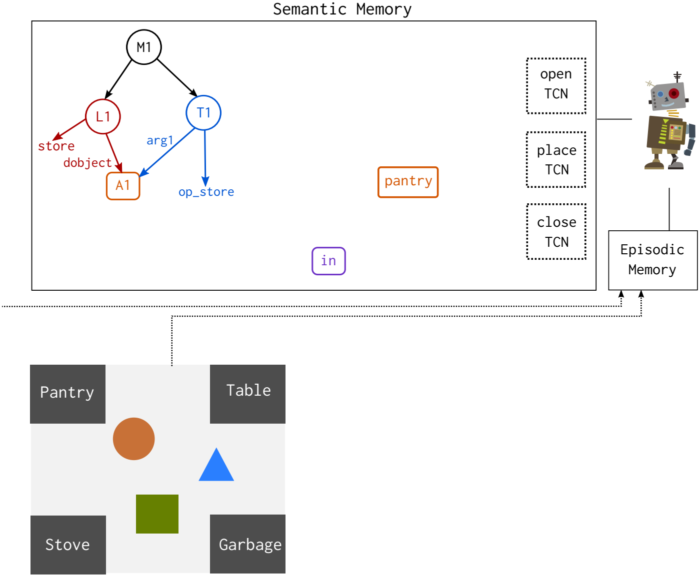
Interactive Execution

<u>Interaction trace</u>
Instructor: Store the green rectangle.



Interactive Execution

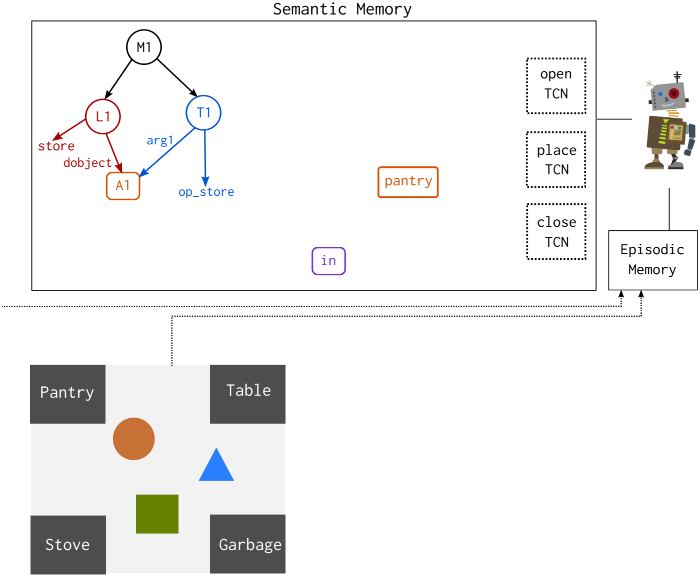
<u>Interaction trace</u>
Instructor: Store the green rectangle.



Interactive Execution

Interaction trace
Instructor: Store the green rectangle.

Agent: What is the goal of the action?



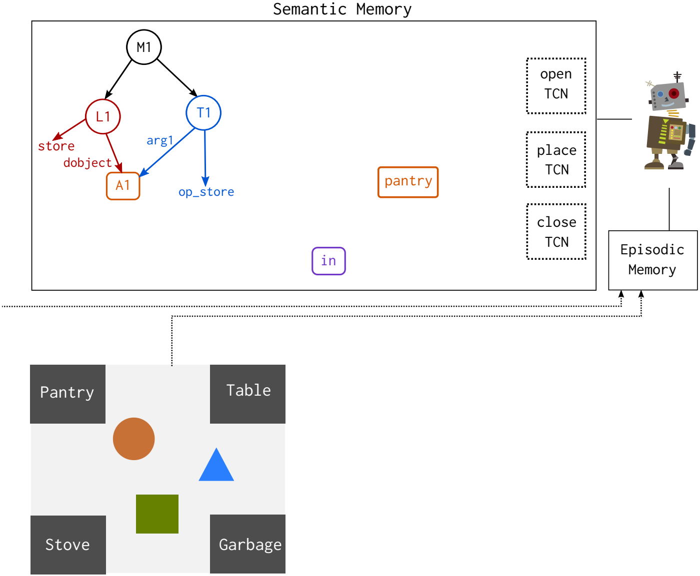
Interactive Execution

<u>Interaction trace</u>

Instructor: Store the green rectangle.

Agent: What is the goal of the action?

Instructor: The goal is the green rectangle in the pantry and the pantry closed.



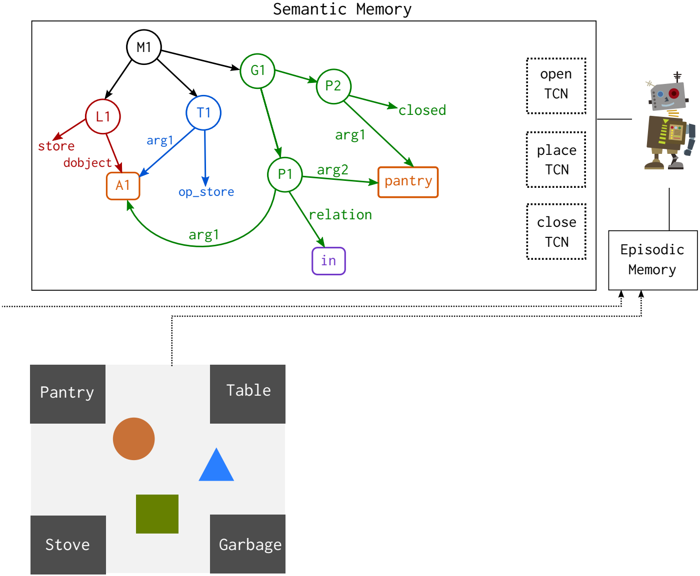
Interactive Execution

Interaction trace

Instructor: Store the green rectangle.

Agent: What is the goal of the action?

Instructor: The goal is the green rectangle in the pantry and the pantry closed.

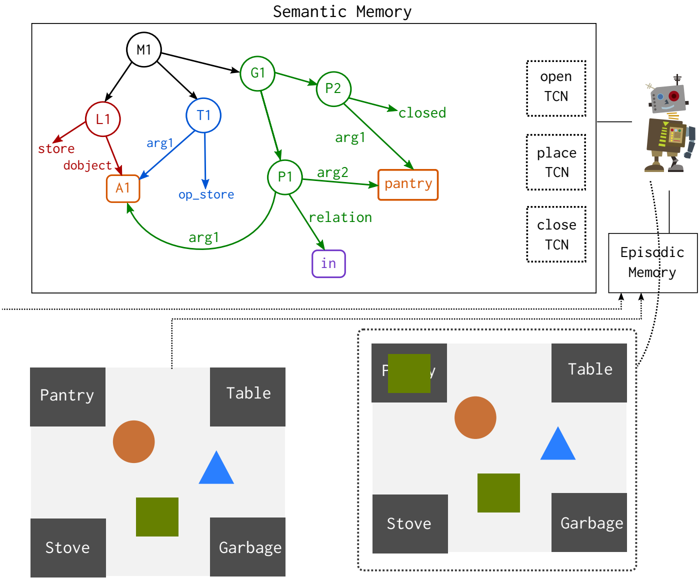


Interactive Execution

Interaction trace
Instructor: Store the green rectangle.

Agent: What is the goal of the action?

Instructor: The goal is the green rectangle in the pantry and the pantry closed.



Interactive Execution

<u>Interaction trace</u>

Instructor: Store the green rectangle.

Agent: What is the goal of the action?

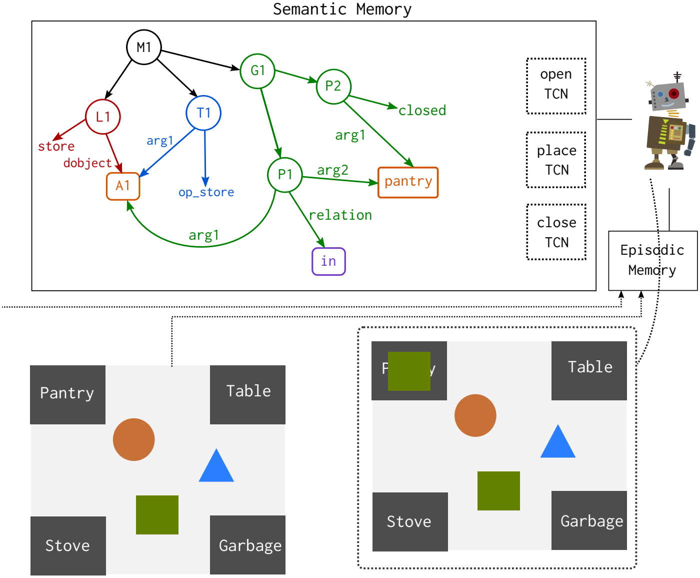
Instructor: The goal is the green rectangle in the pantry and the pantry closed.

Agent: hypothetical explanation

elaborate-goal store:
If store([x]) -->
desired: IN([x],PANTRY) and
CLOSED(PANTRY)

---

terminate store:
If store([x]) and desired and
IN([x],PANTRY) and
CLOSED(PANTRY)
-->
terminate store[x]



Interactive Execution

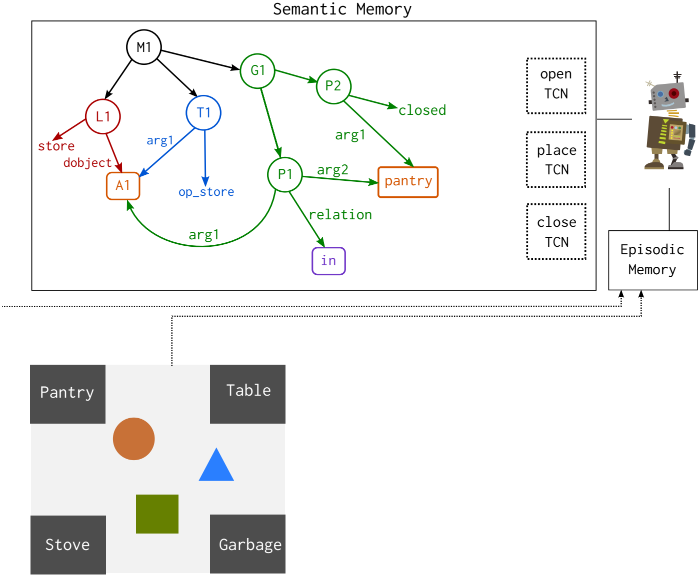
Interaction trace
Instructor: Store the green rectangle.

Agent: What is the goal of the action?

Instructor: The goal is the green rectangle in the pantry and the pantry closed.

Agent: hypothetical explanation

Agent: explore(depth = 2)



Interactive Execution

<u>Interaction trace</u>

Instructor: Store the green rectangle.

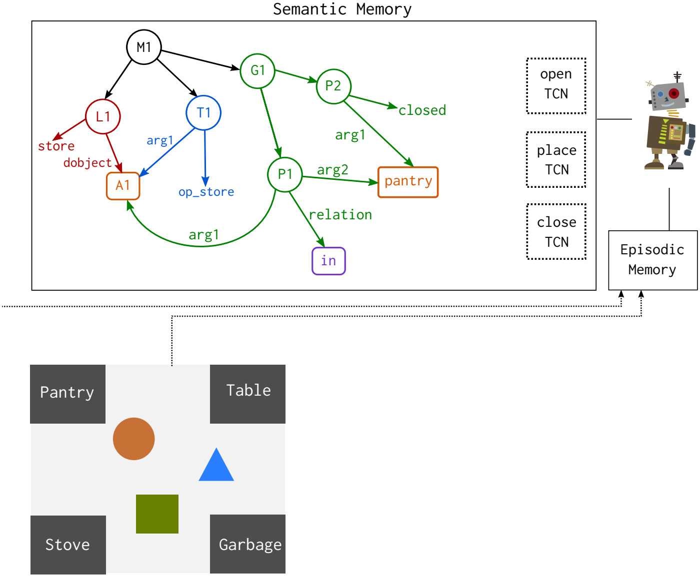
Agent: What is the goal of the action?

Instructor: The goal is the green rectangle in the pantry and the pantry closed.

Agent: `hypothetical explanation`

Agent: `explore(depth = 2)`

Agent: Which action should I take?



Interactive Execution

<u>Interaction trace</u>

Instructor: Store the green rectangle.

Agent: What is the goal of the action?

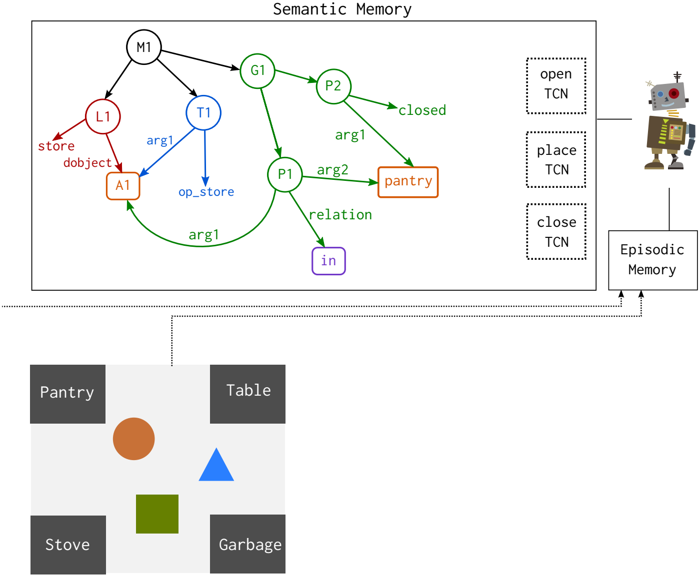
Instructor: The goal is the green rectangle in the pantry and the pantry closed.

Agent: `hypothetical explanation`

Agent: `explore(depth = 2)`

Agent: Which action should I take?

Instructor: Open the pantry.



Interactive Execution

Interaction trace

Instructor: Store the green rectangle.

Agent: What is the goal of the action?

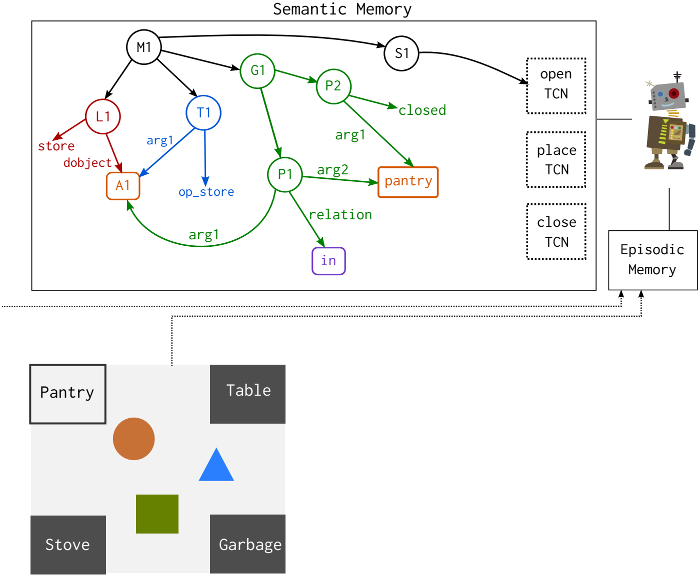
Instructor: The goal is the green rectangle in the pantry and the pantry closed.

Agent: hypothetical explanation

Agent: explore(depth = 2)

Agent: Which action should I take?

Instructor: Open the pantry.



Interactive Execution

<u>Interaction trace</u>

Instructor: Store the green rectangle.

Agent: What is the goal of the action?

Instructor: The goal is the green rectangle in the pantry and the pantry closed.

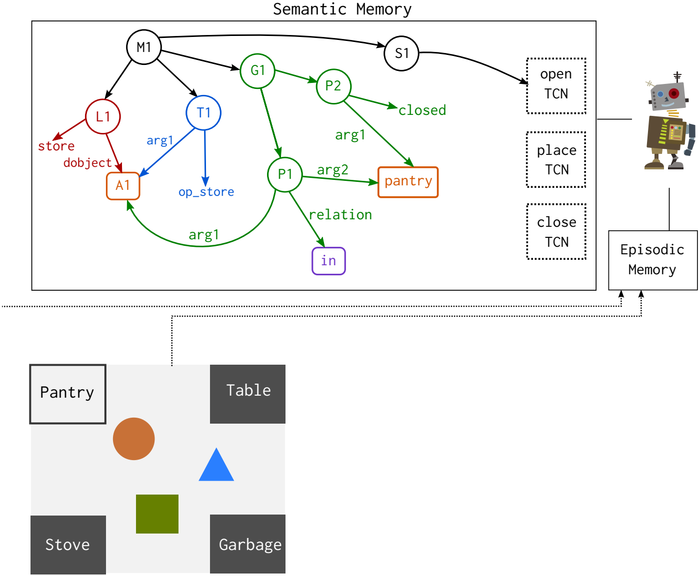
Agent: `hypothetical explanation`

Agent: `explore(depth = 2)`

Agent: Which action should I take?

Instructor: Open the pantry.

Agent: `explore(depth = 2)`



Interactive Execution

Interaction trace

Instructor: Store the green rectangle.

Agent: What is the goal of the action?

Instructor: The goal is the green rectangle in the pantry and the pantry closed.

Agent: hypothetical explanation

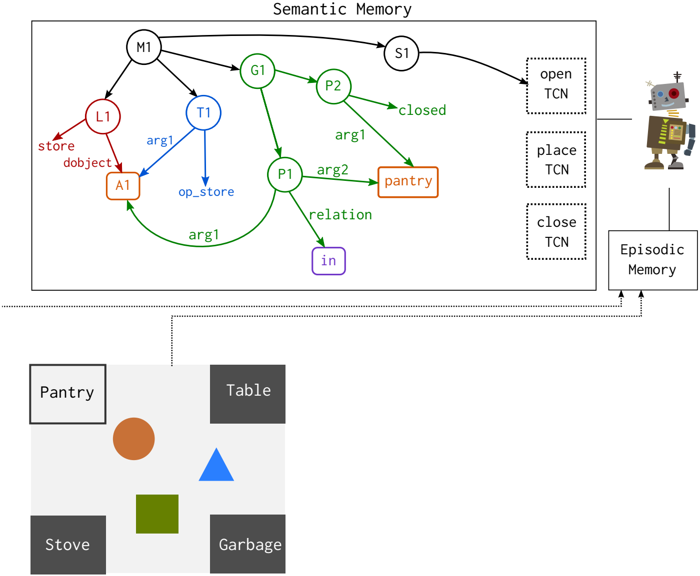
Agent: explore(depth = 2)

Agent: Which action should I take?

Instructor: Open the pantry.

Agent: explore(depth = 2)

Agent: Which action should I take?



Interactive Execution

Interaction trace

Instructor: Store the green rectangle.

Agent: What is the goal of the action?

Instructor: The goal is the green rectangle in the pantry and the pantry closed.

Agent: hypothetical explanation

Agent: explore(depth = 2)

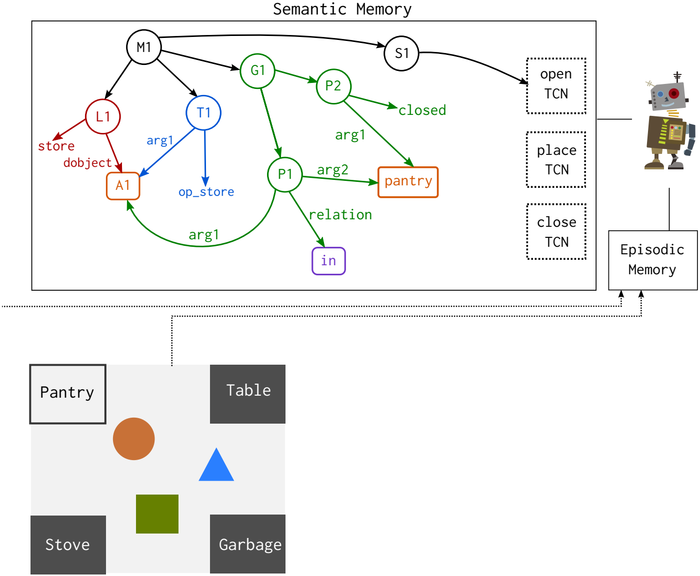
Agent: Which action should I take?

Instructor: Open the pantry.

Agent: explore(depth = 2)

Agent: Which action should I take?

Instructor: Place the object in the pantry.



Interactive Execution

Interaction trace
Instructor: Store the green rectangle.

Agent: What is the goal of the action?

Instructor: The goal is the green rectangle in the pantry and the pantry closed.

Agent: `hypothetical explanation`

Agent: `explore(depth = 2)`

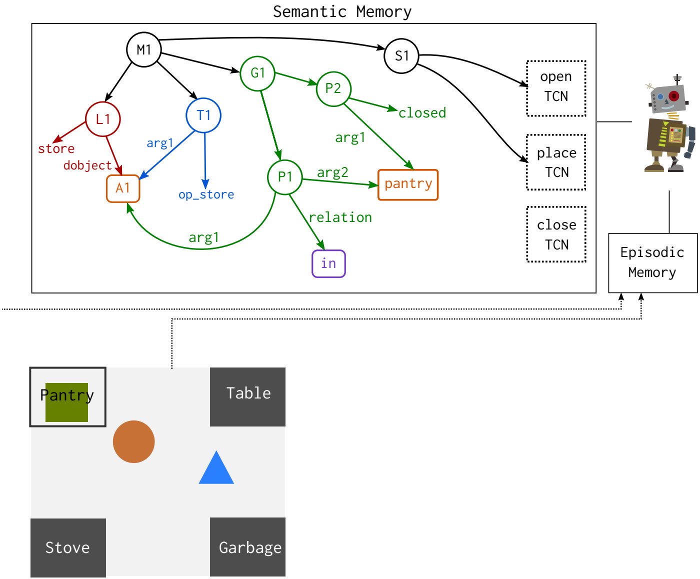
Agent: Which action should I take?

Instructor: Open the pantry.

Agent: `explore(depth = 2)`

Agent: Which action should I take?

Instructor: Place the object in the pantry.



Interactive Execution

Interaction trace

Instructor: Store the green rectangle.

Agent: What is the goal of the action?

Instructor: The goal is the green rectangle in the pantry and the pantry closed.

Agent: `hypothetical explanation`

Agent: `explore(depth = 2)`

Agent: Which action should I take?

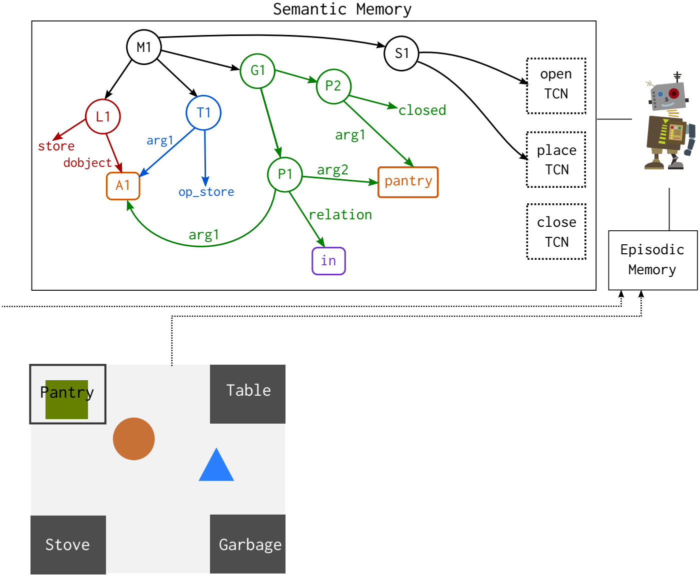
Instructor: Open the pantry.

Agent: `explore(depth = 2)`

Agent: Which action should I take?

Instructor: Place the object in the pantry.

Agent: `explores(depth = 2)`



Interactive Execution

<u>Interaction trace</u>

Instructor: Store the green rectangle.

Agent: What is the goal of the action?

Instructor: The goal is the green rectangle in the pantry and the pantry closed.

Agent: `hypothetical explanation`

Agent: `explore(depth = 2)`

Agent: Which action should I take?

Instructor: Open the pantry.

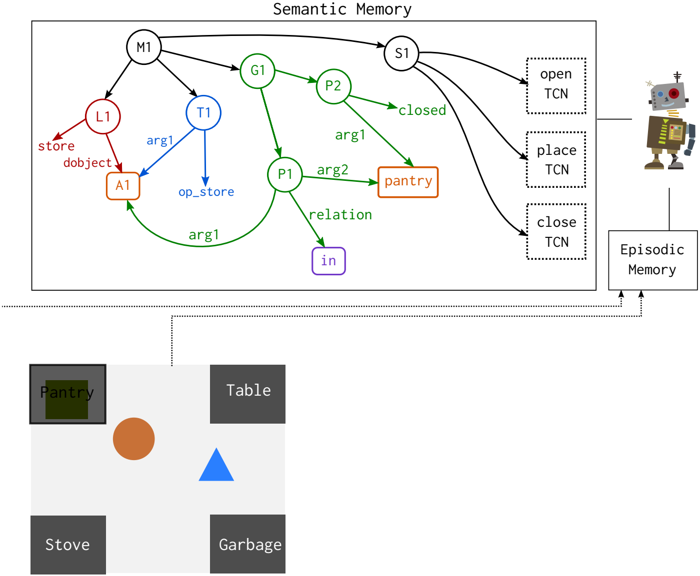
Agent: `explore(depth = 2)`

Agent: Which action should I take?

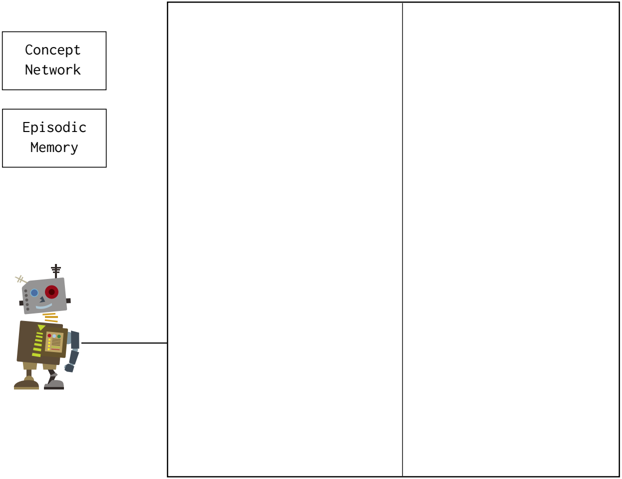
Instructor: Place the object in the pantry.

Agent: `explores(depth = 2)`

Agent: `closes the pantry`



Retrospective Explanation



terminate store:
If `store([x])` and `desired`
and `IN([x],PANTRY)` and
`CLOSED(PANTRY)`
`-->`
terminate `store[x]`

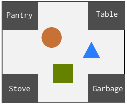
Retrospective Explanation

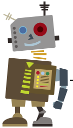
terminate store:
If store([x]) and desired and IN([x],PANTRY) and CLOSED(PANTRY)
-->
terminate store[x]

Concept Network

Episodic Memory

1. Recreate original state.





Retrospective Explanation

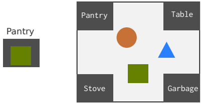
terminate store:
If store([x]) and desired
and IN([x],PANTRY) and
CLOSED(PANTRY)
-->
terminate store[x]

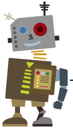
Concept Network

Episodic Memory

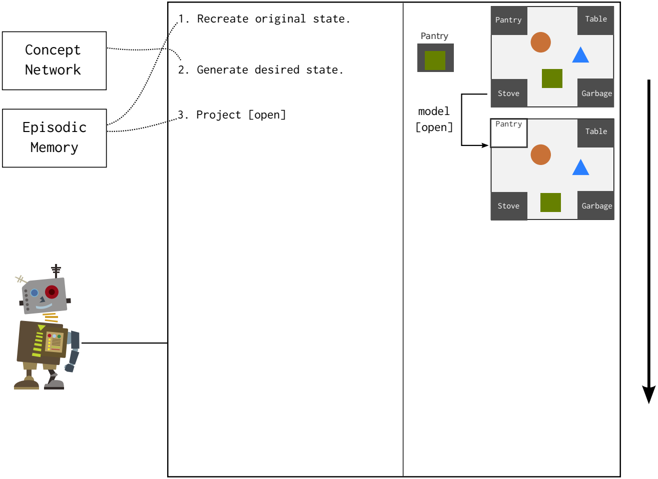
1. Recreate original state.

2. Generate desired state.



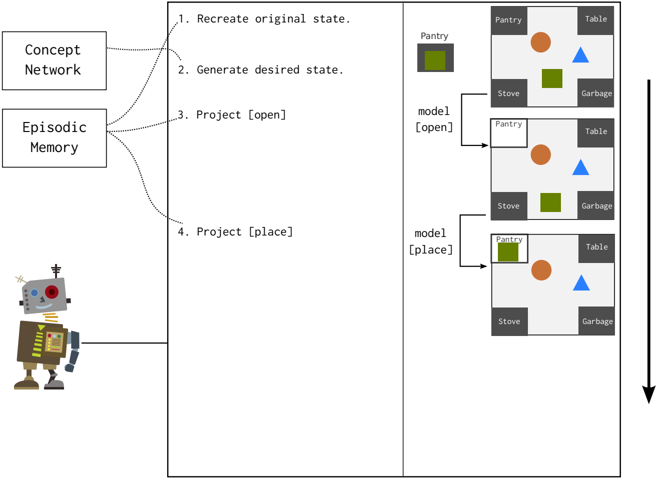


Retrospective Explanation



terminate store:
If store([x]) and desired and IN([x],PANTRY) and CLOSED(PANTRY)
-->
terminate store[x]

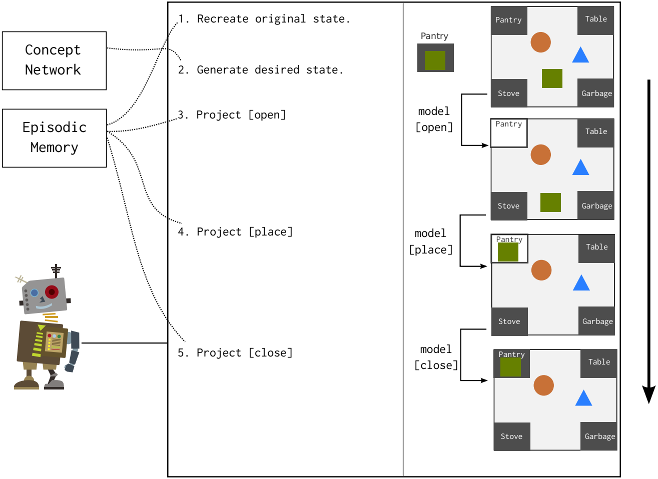
Retrospective Explanation



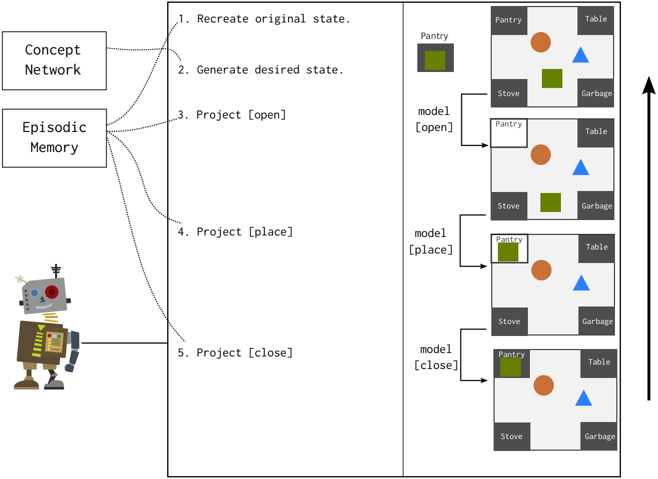
terminate store:
If store([x]) and desired and IN([x],PANTRY) and CLOSED(PANTRY)
-->
terminate store[x]

Retrospective Explanation

terminate store:
If store([x]) and desired
and IN([x],PANTRY) and
CLOSED(PANTRY)
-->
terminate store[x]



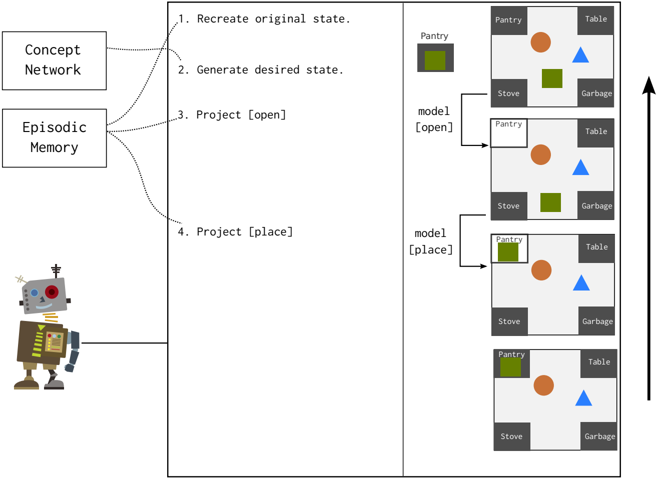
Retrospective Explanation



terminate store:
If store([x]) and desired
and IN([x],PANTRY) and
CLOSED(PANTRY)
-->
terminate store[x]

select close:
If store([x]) and desired
and IN([x],PANTRY) and
OPEN(PANTRY)
-->
select close(PANTRY)

Retrospective Explanation



terminate store:
If store([x]) and desired and IN([x],PANTRY) and CLOSED(PANTRY)
-->
terminate store[x]

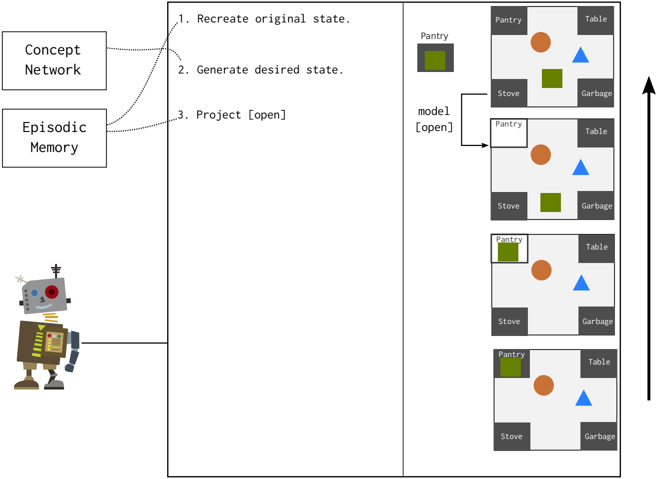
---

select close:
If store([x]) and desired and IN([x],PANTRY) and OPEN(PANTRY)
-->
select close(PANTRY)

---

select place:
If store([x]) and desired and -IN([x],PANTRY) and OPEN(PANTRY)
-->
select place([x],IN,PANTRY)

Retrospective Explanation



terminate store:
If store([x]) and desired and IN([x],PANTRY) and CLOSED(PANTRY)
-->
terminate store[x]

---

select close:
If store([x]) and desired and IN([x],PANTRY) and OPEN(PANTRY)
-->
select close(PANTRY)

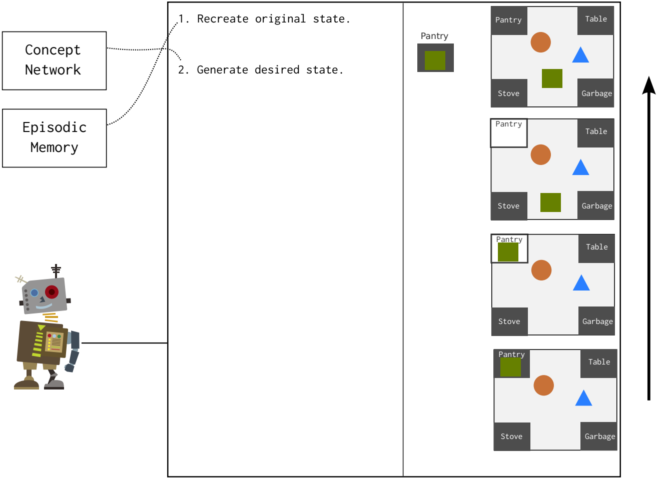
---

select place:
If store([x]) and desired and -IN([x],PANTRY) and OPEN(PANTRY)
-->
select place([x],IN,PANTRY)

---

select open:
If store([x]) and desired and -IN([x],PANTRY) and CLOSED(PANTRY)
-->
select open(PANTRY)

Retrospective Explanation



terminate store:
If store([x]) and desired and IN([x],PANTRY) and CLOSED(PANTRY)
-->
terminate store[x]

select close:
If store([x]) and desired and IN([x],PANTRY) and OPEN(PANTRY)
-->
select close(PANTRY)

select place:
If store([x]) and desired and -IN([x],PANTRY) and OPEN(PANTRY)
-->
select place([x],IN,PANTRY)

select open:
If store([x]) and desired and -IN([x],PANTRY) and CLOSED(PANTRY)
-->
select open(PANTRY)

available store:
If -IN([x],PANTRY) or OPEN(PANTRY)
-->
available store([x])

Multi-task Learning

organizational

- `place([x],[rel],[y])`
  place the red object on the table.
- `move([x],[y])`
  move the arch to the table.
- `stack([x],[y],[z])`
  stack the arch, the cylinder, and the cube.
- `discard([x])`
  discard the arch.
- `store([x])`
  store the arch, store the cube
- `set`
  set the table

functional

- `cook([x])`
  cook the steak.
- `serve([x])`
  serve the steak.

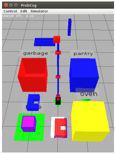
Generality

Learns general representations of tasks from few (~2-3) instances

Generality

Learns general representations of tasks from few (~2-3) instances

abstraction



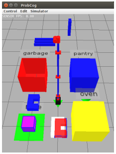
`~~obj1: centroid: 5, 3; obj2: centroid 5, 3~~`

`on(purple object, table)`

Generality

Learns general representations of tasks from few (~2-3) instances

abstraction



~~`obj1: centroid: 5, 3; obj2: centroid 5, 3`~~

`on(purple object, table)`

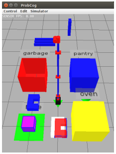
predicate selection

select open:
If `store(O1)` and `-IN(O1,PANTRY)` and `CLOSED(PANTRY)` and ~~`CLOSED(STOVE)`~~ and ~~`OFF(STOVE)`~~ and ~~`-ON(O2,STOVE)`~~ and ...
-->
select `open(PANTRY)`

Generality

Learns general representations of tasks from few (~2-3) instances

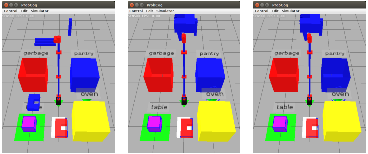
abstraction



~~`obj1: centroid: 5, 3; obj2: centroid 5, 3`~~

`on(purple object, table)`

causal analysis



~~`#examples: 3`~~

`#examples: 1`

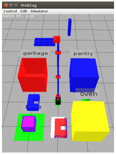
predicate selection

select open:
If `store(O1)` and `-IN(O1,PANTRY)` and `CLOSED(PANTRY)` and ~~`CLOSED(STOVE)`~~ and ~~`OFF(STOVE)`~~ and ~~`-ON(O2,STOVE)`~~ and ...
-->
select `open(PANTRY)`

Generality

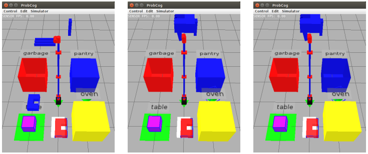
Learns general representations of tasks from few (~2-3) instances

abstraction



~~`obj1: centroid: 5, 3; obj2: centroid 5, 3`~~
`on(purple object, table)`

causal analysis



~~`#examples: 3`~~
`#examples: 1`

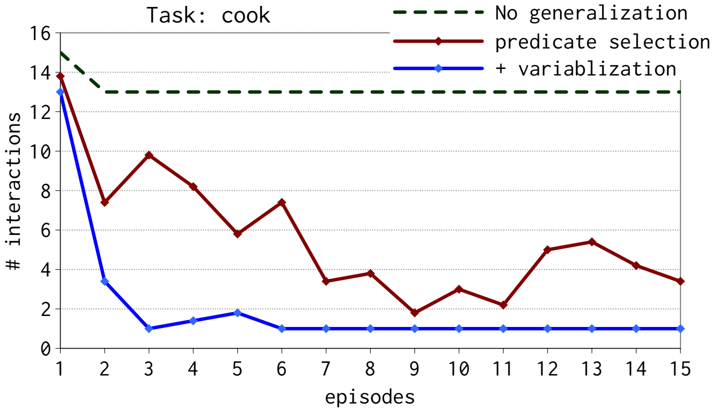
predicate selection

select open:
If `store(O1)` and `-IN(O1,PANTRY)` and `CLOSED(PANTRY)` and ~~`CLOSED(STOVE)`~~ and ~~`OFF(STOVE)`~~ and ~~`-ON(O2,STOVE)`~~ and ...
-->
select `open(PANTRY)`

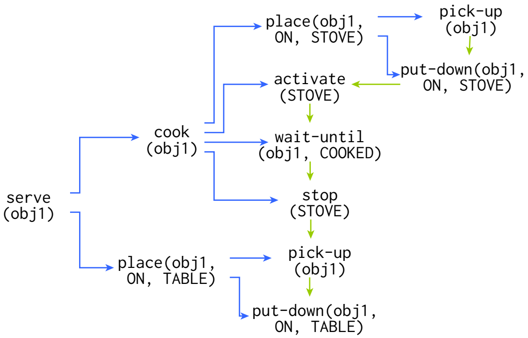
variablization

Store the green rectangle.
The goal is the green rectangle in the pantry and the pantry is closed.
Open the pantry.
Move the green rectangle to the pantry.
...

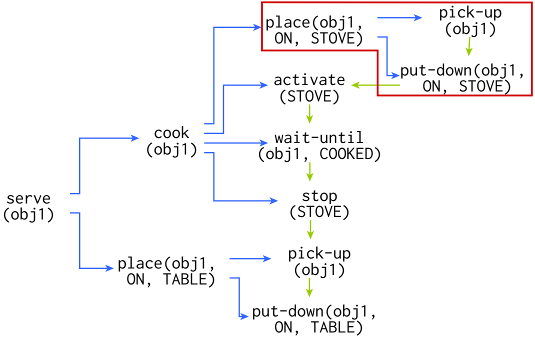
Generality


Task: cook
No generalization
predicate selection
+ variablization
# interactions
episodes
0
2
4
6
8
10
12
14
16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

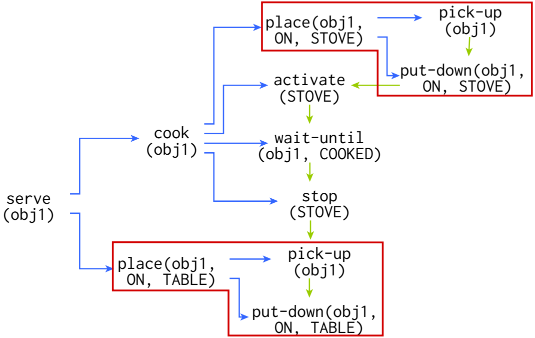
Hierarchical Learning



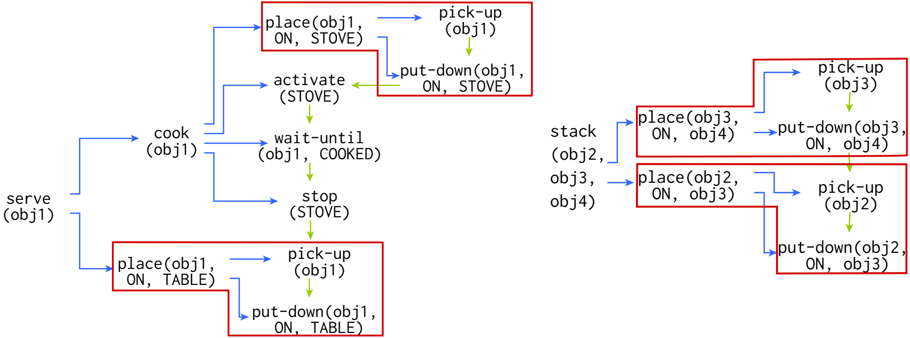
Hierarchical Learning



Hierarchical Learning

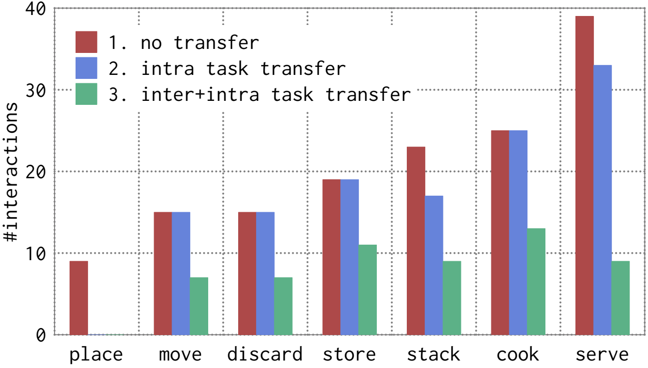


Hierarchical Learning



Transfer

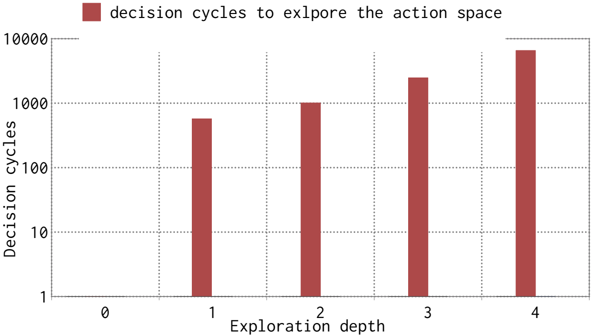
Exploits the common policy space for instruction-aided transfer.



Distributed Initiative

Integrates agent-driven exploration and instruction-guided exploitation

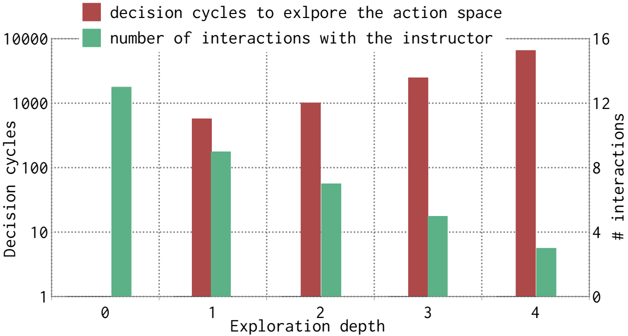
Learning the task *store*.



Distributed Initiative

Integrates agent-driven exploration and instruction-guided exploitation

Learning the task *store*.



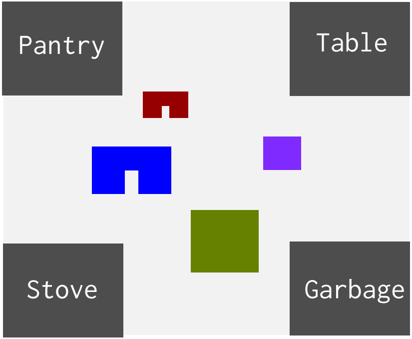
Learning Implicit Parameters

- Implicit parameters depend on explicit parameters
  - *put away the books*
  - *put away the groceries*
- Incomplete domain theory
  - action models only include physical affordances
  - no semantics related to home organization
    - books are *usually* kept on the shelf

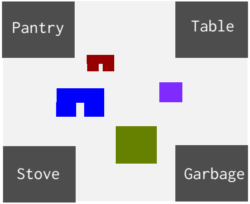
New variation of *store*

store implies

- if an *arch*, place it in the pantry, close pantry
- if an *cube*, place it on the table

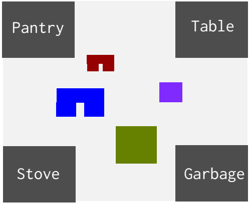


Associative Default Values



Store the red object.

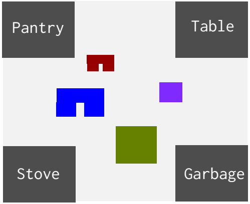
Associative Default Values



Store the red object.

The goal is the red object in the pantry and the pantry closed.

Associative Default Values

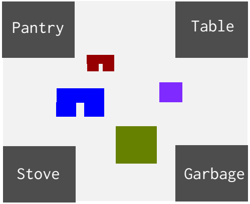


Store the red object.

The goal is the red object in the pantry and the pantry closed.

elaborate-goal store:
If `store(obj)` and
`obj:red,arch,small` and
`loc:pantry,container`
`-->`
desired: `IN(obj,loc)` and
`CLOSED(loc)`

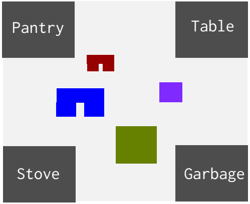
Associative Default Values



Store the blue object.

elaborate-goal store:
If `store(obj)` and
`obj:red,arch,small` and
`loc:pantry,container`
`-->`
desired: `IN(obj,loc)` and
`CLOSED(loc)`

Associative Default Values

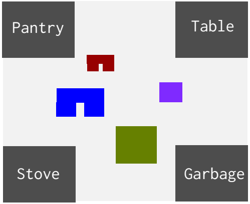


Store the blue object.

The goal is the blue object in the pantry and the pantry closed.

elaborate-goal store:
If `store(obj)` and
`obj:red,arch,small` and
`loc:pantry,container`
`-->`
desired: `IN(obj,loc)` and
`CLOSED(loc)`

Associative Default Values

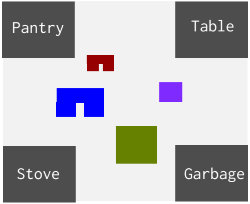


Store the blue object.

The goal is the blue object in the pantry and the pantry closed.

elaborate-goal store:
If `store(obj)` and
`obj:~~red~~,arch,~~small~~` and
`loc:pantry,container`
`-->`
desired: `IN(obj,loc)` and
`CLOSED(loc)`

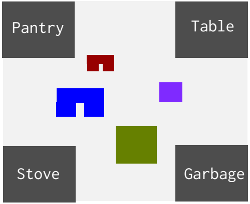
Associative Default Values



Store the green object.

elaborate-goal store:
If `store(obj)` and
`obj:~~red~~,arch,~~small~~` and
`loc:pantry,container`
`-->`
desired: `IN(obj,loc)` and
`CLOSED(loc)`

Associative Default Values

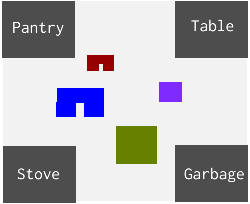


Store the green object.

The goal is the green object on the table.

elaborate-goal store:
If `store(obj)` and
`obj:`~~`red`~~`,arch,`~~`small`~~ and
`loc:pantry,container`
`-->`
desired: `IN(obj,loc)` and
`CLOSED(loc)`

Associative Default Values



Store the green object.

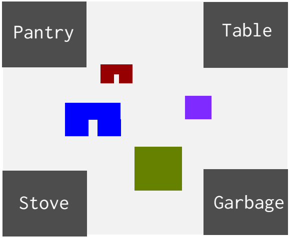
The goal is the green object on the table.

elaborate-goal store:
If `store(obj)` and
`obj:green,cube,medium` and
`loc:table,flat`
`-->`
desired: `ON(obj,loc)`

If `store(obj)` and
`obj:~~red~~,arch,~~small~~` and
`loc:pantry,container`
`-->`
desired: `IN(obj,loc)` and
`CLOSED(loc)`

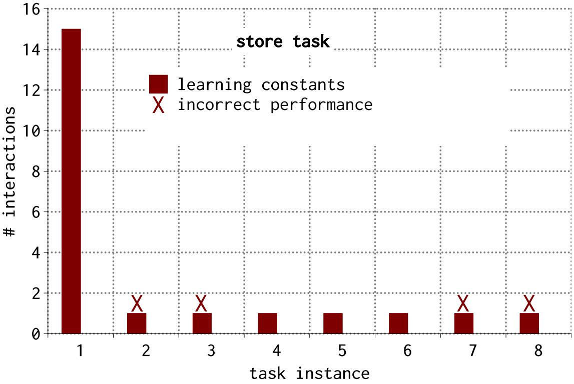
Generality and Performance

- *store*
  - arch → in pantry
  - cube → on table
- *move*
  - o , pantry → in
  - o , table → on
- *set*
  - medium → on table,
    small → right-of medium



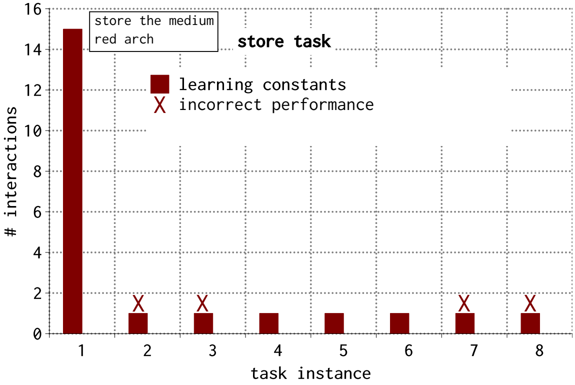
Generality and Performance

1. Learn constants (AAAI 2014)
`store([o], IN, PANTRY)`



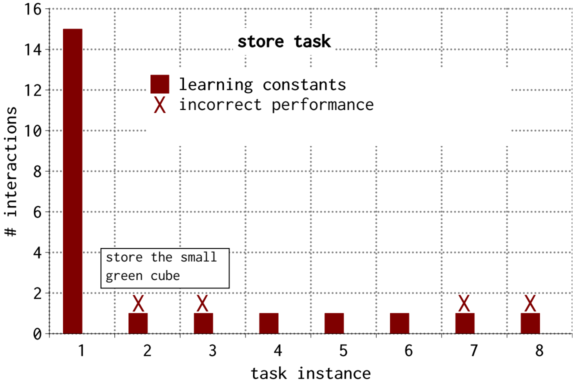
Generality and Performance

1. Learn constants (AAAI 2014)
`store([o], IN, PANTRY)`



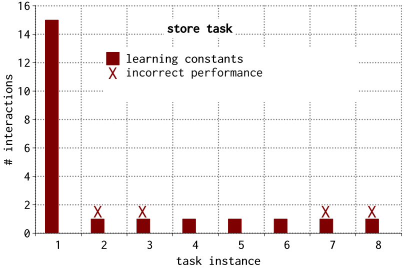
Generality and Performance

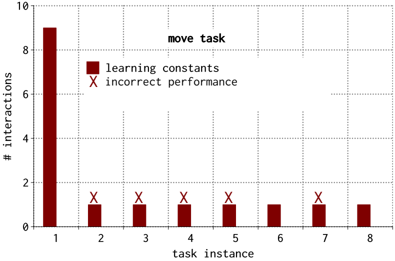
1. Learn constants (AAAI 2014)
`store([o], IN, PANTRY)`

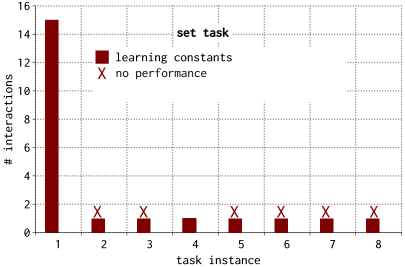


Generality and Performance

1. Learn constants (AAAI 2014)
   `store([o], IN, PANTRY)`

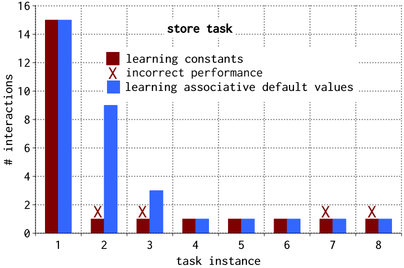


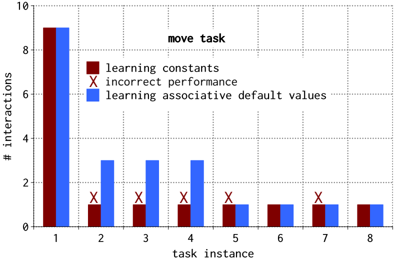


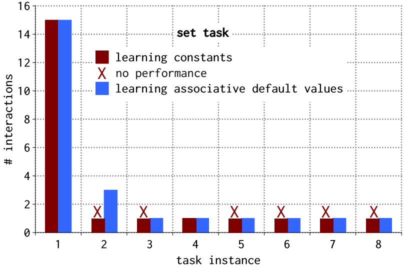


Generality and Performance

1. Learn constants (AAAI 2014)
   `store([o], IN, PANTRY)`
2. Learn associative default values







Challenges

- developing a learnable representation that can be employed for situated dialog as well as task performance.
- developing computational models from psycholinguistic, linguistic theories.
- designing the interactive learning paradigm
  - online interaction and learning

Contributions

Contributions

- *How can verbs be grounded in task goals and execution knowledge?*
  - *approach*: a mixed-modality representation
  - encompasses lexical, semantic, task-oriented aspects
  - learnable, modular, generalizes quickly
  - provides information useful in comprehension

Contributions

- *How can verbs be grounded in task goals and execution knowledge?*
    - *approach*: a mixed-modality representation
    - encompasses lexical, semantic, task-oriented aspects
    - learnable, modular, generalizes quickly
    - provides information useful in comprehension
- *How can task commands be understood?*
    - *approach*: the Indexical Model
    - formulates language comprehension as a search over common ground
    - exploits non-linguistic contexts for reference resolution, unexpressed argument alternations
    - scales with learning

Contributions

- *How can verbs be grounded in task goals and execution knowledge?*
  - *approach*: a mixed-modality representation
  - encompasses lexical, semantic, task-oriented aspects
  - learnable, modular, generalizes quickly
  - provides information useful in comprehension
- *How can task commands be understood?*
  - *approach*: the Indexical Model
  - formulates language comprehension as a search over common ground
  - exploits non-linguistic contexts for reference resolution, unexpressed argument alternations
  - scales with learning
- *How can task-oriented linguistic interaction be sustained?*
  - *approach*: the interaction model based on collaborative discourse theory
  - integrates with comprehension and learning
  - allows for flexible dialog, flexible instruction strategy

Contributions

- *How can verbs be grounded in task goals and execution knowledge?*
  - *approach*: a mixed-modality representation
  - encompasses lexical, semantic, task-oriented aspects
  - learnable, modular, generalizes quickly
  - provides information useful in comprehension
- *How can task commands be understood?*
  - *approach*: the Indexical Model
  - formulates language comprehension as a search over common ground
  - exploits non-linguistic contexts for reference resolution, unexpressed argument alternations
  - scales with learning
- *How can task-oriented linguistic interaction be sustained?*
  - *approach*: the interaction model based on collaborative discourse theory
  - integrates with comprehension and learning
  - allows for flexible dialog, flexible instruction strategy
- *How can task goals, structure, and execution knowledge be learned interactively?*
  - *approach*: interactive-variation of EBG
  - learns several tasks, quickly generalizes, transfers knowledge, accommodates exploration
  - used for learning how to play games (Kirk and Laird 2014)

Contributions

- Human-robot/agent interaction
  - grounding language in perception, control, knowledge, and experience
- Interactive learning
  - complimentary to programming/learning by demonstration
- Cognitive architectures
  - integration of diverse capabilities

Future Work (1)

Scope

- applicability in real domestic environments
- instruction for other tasks: information acquisition, performance, skill-oriented

HRI Experiments

- are such dialogs easy for humans?
- how do humans describe goals?
- do humans teach tasks hierarchically?

Future Work (2)

Corrective Instruction

- on-performance agent-initiated corrections
- concept correction, policy correction

Non-situated Comprehension

- extensions to the Indexical Model
- learning game-play, appliance operation from manuals; tasks from WikiHow

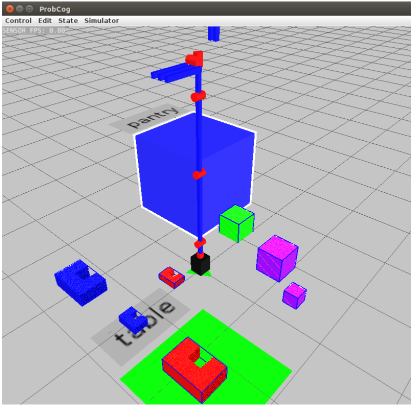
Thanks!

Handling Uncertainty

- Action models do not encode uncertainty explicitly
- Policy representation ensures robust behavior
  - `place(obj, rel, loc)`
  - `if clear(obj) and not-grabbed(obj) and not-predicate(obj, rel, loc)` $\rightarrow$ `pick-up(obj)`
  - `if grabbed(obj) and not-predicate(obj, rel, loc)` $\rightarrow$ `put-down(obj, rel, loc)`

Task Knowledge and Comprehension

place the red object on the table.



Hierarchical Learning v/s Flat Learning

```
serve(obj)
```

Interactive Learning

Explanation-based Learning

Given: concept definition
Given: a positive example
Given: domain theory (prior beliefs, inference)

concept definition:
`is-cup(x)?`

example:
`OBJ1: color(OBJ1,red)` ∧ `is-concave(OBJ1)` ∧ `weight(OBJ1,light)`

domain theory:

[ ] `drinkable-from(x)` → `cup(x)`

[ ] `liftable(x)` ∧ `open(x)` → `drinkable-from(x)`

[ ] `has-part(x, flat-bottom)` → `stable(x)`

[ ] `is-concave(x)` → `open(x)`

[ ] `weight(x,light)` → `liftable(x)`

[ ] `color(x,red)` → `loves(Rosie,x)` ...

Mitchell, T. M., Keller, R. M., and Kedar-Cabelli, S. T. Explanation-based Generalization: A Unifying View. 1986.

Explanation-based Learning

Given: concept definition
Given: a positive example
Given: domain theory (prior beliefs, inference)

1. construct explanations from examples
   - inference over domain theory
   - why does the example fit the concept definition?

concept definition:
`is-cup(x)?`

example:
`OBJ1: color(OBJ1,red)` ∧
`is-concave(OBJ1)` ∧
`weight(OBJ1,light)`

domain theory:
[ ] `drinkable-from(x)` → `cup(x)`
[ ] `liftable(x)` ∧ `open(x)` → `drinkable-from(x)`
[ ] `has-part(x, flat-bottom)` → `stable(x)`
[ ] `is-concave(x)` → `open(x)`
[ ] `weight(x,light)` → `liftable(x)`
[ ] `color(x,red)` → `loves(Rosie,x)` ...

Mitchell, T. M., Keller, R. M., and Kedar-Cabelli, S. T. Explanation-based Generalization: A Unifying View. 1986.

Explanation-based Learning

Given: concept definition
Given: a positive example
Given: domain theory (prior beliefs, inference)

1. construct explanations from examples
   - inference over domain theory
   - why does the example fit the concept definition?

concept definition:
`is-cup(x)?`

example:
`OBJ1: color(OBJ1,red)` $\wedge$
`is-concave(OBJ1)` $\wedge$
`weight(OBJ1,light)`

domain theory:
[✓] `drinkable-from(x)` $\rightarrow$ `cup(x)`
[✓] `liftable(x)` $\wedge$ `open(x)` $\rightarrow$ `drinkable-from(x)`
[ ] `has-part(x, flat-bottom)` $\rightarrow$ `stable(x)`
[✓] `is-concave(x)` $\rightarrow$ `open(x)`
[✓] `weight(x,light)` $\rightarrow$ `liftable(x)`
[ ] `color(x,red)` $\rightarrow$ `loves(Rosie,x)` ...

Mitchell, T. M., Keller, R. M., and Kedar-Cabelli, S. T. Explanation-based Generalization: A Unifying View. 1986.

Explanation-based Learning

Given: concept definition
Given: a positive example
Given: domain theory (prior beliefs, inference)

1. construct explanations from examples
   - inference over domain theory
   - why does the example fit the concept definition?
2. generalization
   - remove unused constraints & features
   - replace constants with variables

concept definition:
`is-cup(x)?`

example:
`OBJ1: color(OBJ1,red)` ∧ `is-concave(OBJ1)` ∧ `weight(OBJ1,light)`

domain theory:

[✓] `drinkable-from(x)` → `cup(x)`

[✓] `liftable(x)` ∧ `open(x)` → `drinkable-from(x)`

[ ] `has-part(x, flat-bottom)` → `stable(x)`

[✓] `is-concave(x)` → `open(x)`

[✓] `weight(x,light)` → `liftable(x)`

[ ] `color(x,red)` → `loves(Rosie,x)` ...

Mitchell, T. M., Keller, R. M., and Kedar-Cabelli, S. T. Explanation-based Generalization: A Unifying View. 1986.

Explanation-based Learning

Given: concept definition
Given: a positive example
Given: domain theory (prior beliefs, inference)

1. construct explanations from examples
   - inference over domain theory
   - why does the example fit the concept definition?
2. generalization
   - remove unused constraints & features
   - replace constants with variables

concept definition:
`is-cup(x)?`

example:
`OBJ1:` ~~`color(OBJ1,red)`~~ ∧ `is-concave(OBJ1)` ∧ `weight(OBJ1,light)`

domain theory:

[✓] `drinkable-from(x)` → `cup(x)`

[✓] `liftable(x)` ∧ `open(x)` → `drinkable-from(x)`

[ ] `has-part(x, flat-bottom)` → `stable(x)`

[✓] `is-concave(x)` → `open(x)`

[✓] `weight(x,light)` → `liftable(x)`

[ ] `color(x,red)` → `loves(Rosie,x)` ...

Mitchell, T. M., Keller, R. M., and Kedar-Cabelli, S. T. Explanation-based Generalization: A Unifying View. 1986.

Explanation-based Learning

Given: concept definition
Given: a positive example
Given: domain theory (prior beliefs, inference)

1. construct explanations from examples
   - inference over domain theory
   - why does the example fit the concept definition?
2. generalization
   - remove unused constraints & features
   - replace constants with variables

concept definition:
`is-cup(x)?`

example:
`OBJ1:` ~~`color(OBJ1,red)`~~ ∧ `is-concave([x1])` ∧ `weight([x1],light)`

domain theory:
[✓] `drinkable-from(x)` → `cup(x)`
[✓] `liftable(x)` ∧ `open(x)` → `drinkable-from(x)`
[ ] `has-part(x, flat-bottom)` → `stable(x)`
[✓] `is-concave(x)` → `open(x)`
[✓] `weight(x,light)` → `liftable(x)`
[ ] `color(x,red)` → `loves(Rosie,x)` …

Mitchell, T. M., Keller, R. M., and Kedar-Cabelli, S. T. Explanation-based Generalization: A Unifying View. 1986.

Explanation-based Learning

Given: concept definition
Given: a positive example
Given: domain theory (prior beliefs, inference)

1. construct explanations from examples
   - inference over domain theory
   - why does the example fit the concept definition?
2. generalization
   - remove unused constraints & features
   - replace constants with variables

concept definition:
`is-cup(x)?`

example:
`OBJ1:` ~~`color(OBJ1,red)`~~ ∧
<u>`is-concave([x1])` ∧</u>
<u>`weight([x1],light)`</u>

domain theory:
[✓] `drinkable-from(x)` → `cup(x)`
[✓] `liftable(x)` ∧ `open(x)` → `drinkable-from(x)`
[ ] `has-part(x, flat-bottom)` → `stable(x)`
[✓] `is-concave(x)` → `open(x)`
[✓] `weight(x,light)` → `liftable(x)`
[ ] `color(x,red)` → `loves(Rosie,x)` …

Mitchell, T. M., Keller, R. M., and Kedar-Cabelli, S. T. Explanation-based Generalization: A Unifying View. 1986.

Explanation-based Learning

Given: concept definition
Given: a positive example
Given: domain theory (prior beliefs, inference)

1. construct explanations from examples
   - inference over domain theory
   - why does the example fit the concept definition?
2. generalization
   - remove unused constraints & features
   - replace constants with variables

powerful generalization, *small*-data

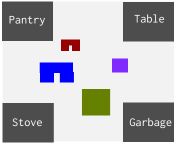
concept definition:
`is-cup(x)?`

example:
`OBJ1:` ~~`color(OBJ1,red)`~~ ∧
`is-concave([x1])` ∧
`weight([x1],light)`

domain theory:
[✓] `drinkable-from(x)` → `cup(x)`
[✓] `liftable(x)` ∧ `open(x)` → `drinkable-from(x)`
[ ] `has-part(x, flat-bottom)` → `stable(x)`
[✓] `is-concave(x)` → `open(x)`
[✓] `weight(x,light)` → `liftable(x)`
[ ] `color(x,red)` → `loves(Rosie,x)` ...

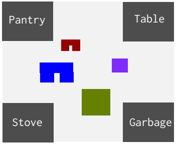
Mitchell, T. M., Keller, R. M., and Kedar-Cabelli, S. T. Explanation-based Generalization: A Unifying View. 1986.

Associative Default Values

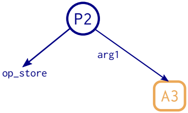


Store the red object.

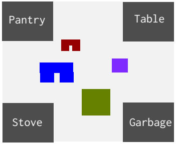
Associative Default Values



Store the red object.

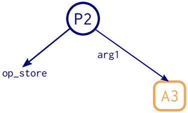


Associative Default Values

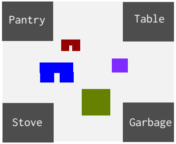


Store the red object.

The goal is the red object in the pantry and the pantry closed.

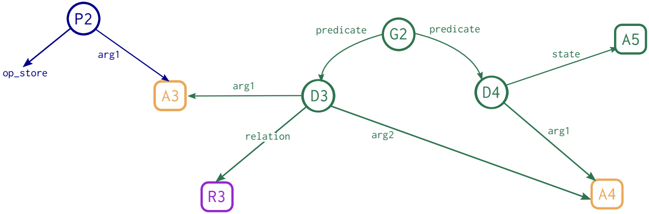


Associative Default Values

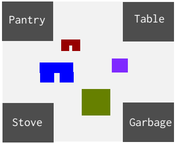


Store the red object.

The goal is the red object in the pantry and the pantry closed.

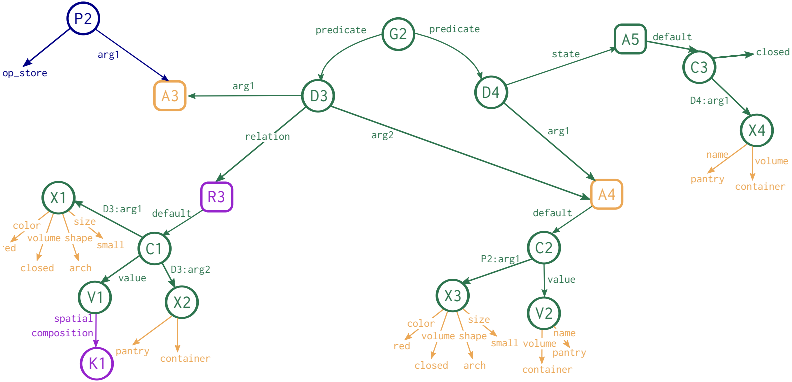


Associative Default Values

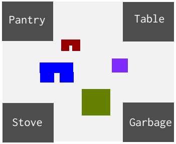


Store the red object.

The goal is the red object in the pantry and the pantry closed.



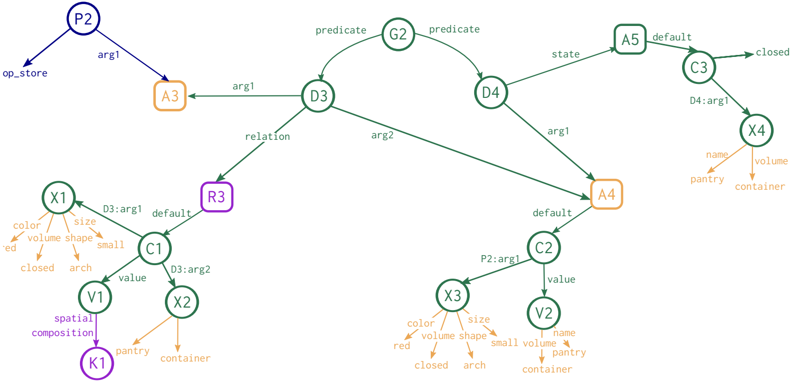
Associative Default Values



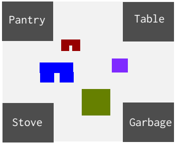
Store the red object.

The goal is the red object in the pantry and the pantry closed.

elaborate-goal store:
If `store([x])` and `[x]:red,closed,arch,small` and `[l]:pantry,container`
`-->`
desired: `IN([x],[l])` and `CLOSED([l])`

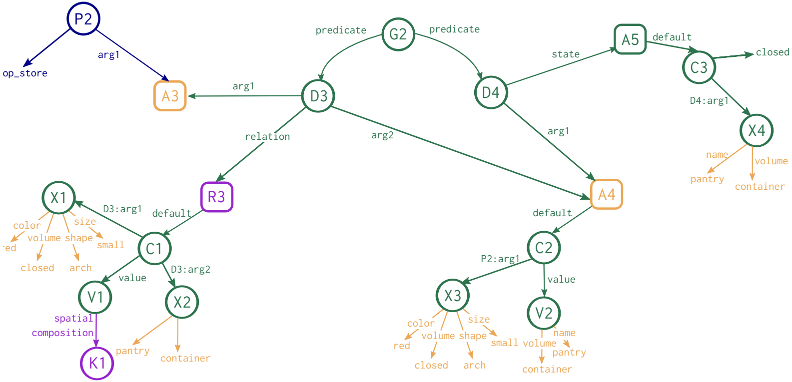


Associative Default Values

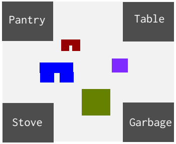


Store the blue object.

elaborate-goal store:
If `store([x])` and `[x]:red,closed,arch,small` and
`[l]:pantry,container`
`-->`
desired: `IN([x],[l])` and `CLOSED([l])`

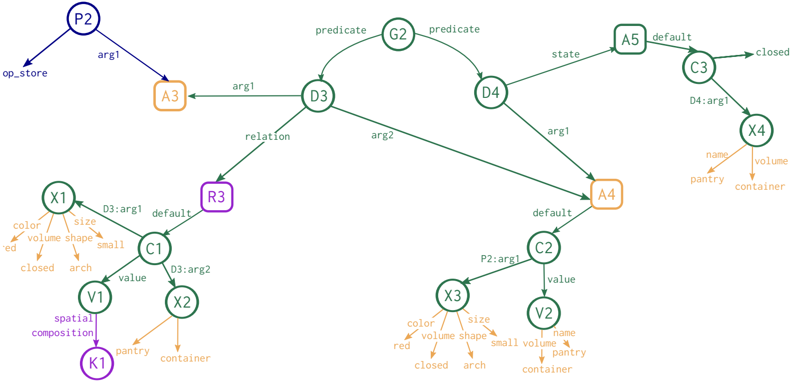


Associative Default Values

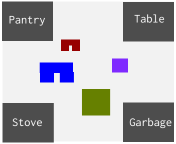


Store the blue object.

The goal is the blue object in the pantry and the pantry closed.

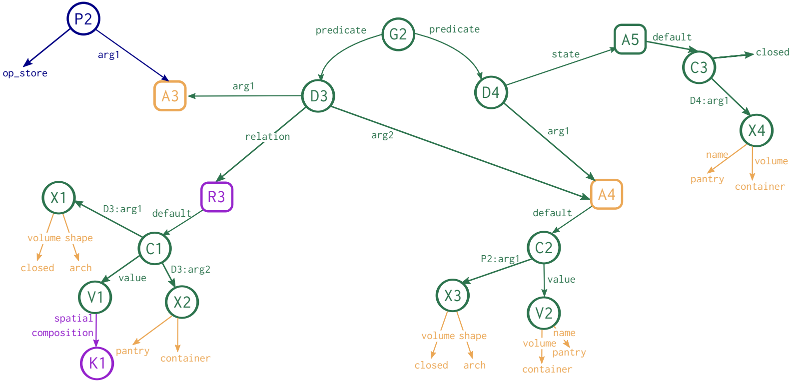


Associative Default Values

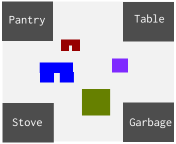


Store the blue object.

The goal is the blue object in the pantry and the pantry closed.



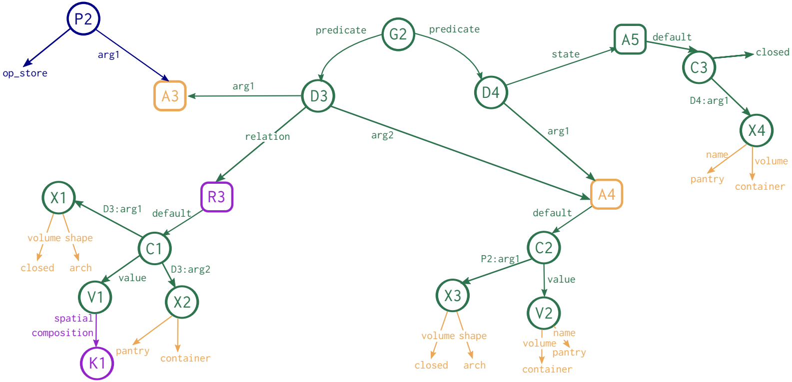
Associative Default Values



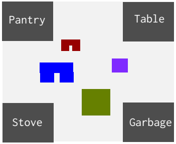
Store the blue object.

The goal is the blue object in the pantry and the pantry closed.

elaborate-goal store:
If `store([x])` and `[x]:closed,arch` and `[l]:pantry,container`
`-->`
desired: `IN([x],[l])` and `CLOSED([l])`

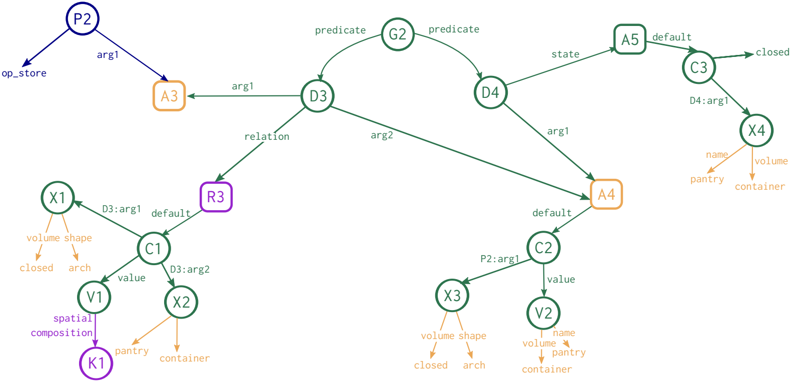


Associative Default Values

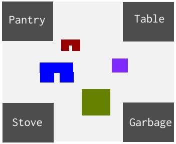


Store the green object.

elaborate-goal store:
If store([x]) and [x]:closed,arch and
[l]:pantry,container
-->
desired: IN([x],[l]) and CLOSED([l])

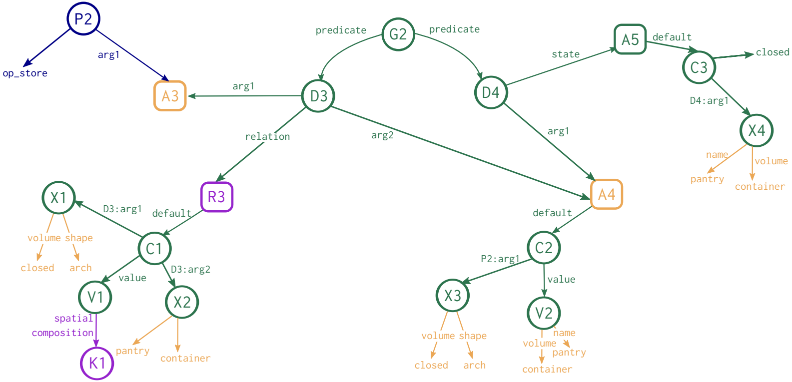


Associative Default Values

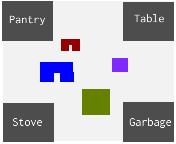


Store the green object.

The goal is the green object on the table.

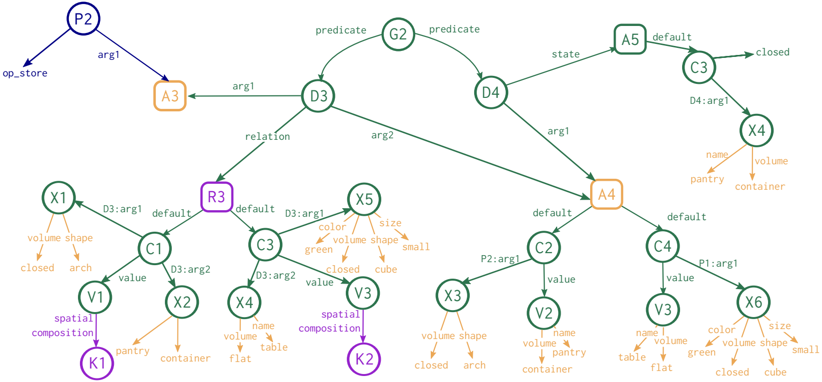


Associative Default Values

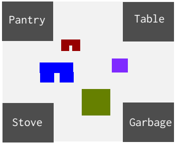


Store the green object.

The goal is the green object on the table.



Associative Default Values



Store the green object.

The goal is the green object on the table.

elaborate-goal store:
If `store([x])` and `[x]:green,closed,cube,medium` and `[l]:table,flat`
`-->`
desired: `ON([x],[l])`