Situated Mapping of Sequential Instructions to Actions with Single-step Reward Observation

Alane Suhr and Yoav Artzi
Executing Context-Dependent Instructions

Task: map a sequence of instructions to actions

Existing Work
- Symbolic Representations
- Modeling Context

Today
- System Actions
- Learning from Exploration
Executing a Sequence of Instructions

1. **Empty out the leftmost beaker of purple chemical**

2. **Then, add the contents of the first beaker to the second**

3. **Mix it**

4. **Then, drain 1 unit from it**

5. **Same for 1 more unit**
Executing a Sequence of Instructions

1. Empty out the leftmost beaker of purple chemical.
2. Then, add the contents of the first beaker to the second.
3. Mix it.
4. Then, drain 1 unit from it.
5. Same for 1 more unit.
Executing a Sequence of Instructions

Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

Mix it

Then, drain 1 unit from it

Same for 1 more unit
Executing a Sequence of Instructions

1. Empty out the leftmost beaker of purple chemical.
2. Then, add the contents of the first beaker to the second.
3. Mix it.
4. Then, drain 1 unit from it.
5. Same for 1 more unit.
Executing a Sequence of Instructions

1. Empty out the leftmost beaker of purple chemical
2. Then, add the contents of the first beaker to the second
3. Mix it
4. Then, drain 1 unit from it
5. Same for 1 more unit
Executing a Sequence of Instructions

- Empty out the leftmost beaker of purple chemical
- Then, add the contents of the first beaker to the second
- Mix it
- Then, drain 1 unit from it
- Same for 1 more unit
Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

Mix it

Then, drain 1 unit from it

Same for 1 more unit
Empty out the leftmost beaker of purple chemical.

Then, add the contents of the first beaker to the second.

Mix it.

Then, drain 1 unit from it.

Same for 1 more unit.
Problem Setup

• Task: follow sequence of instructions
• Learning from instructions and corresponding world states

Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

Mix it

Then, drain 1 unit from it

Same for 1 more unit
Problem Setup

• Task: follow sequence of instructions

• Learning from instructions and corresponding world states

Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

Mix it

Then, drain 1 unit from it

Same for 1 more unit
Problem Setup

- Task: follow sequence of instructions
- Learning from instructions and corresponding world states

Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

Mix it

Then, drain 1 unit from it

Same for 1 more unit
Problem Setup

• Task: follow sequence of instructions
• Learning from instructions and corresponding world states

Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

Mix it

Then, drain 1 unit from it

Same for 1 more unit
Problem Setup

• Task: follow sequence of instructions

• Learning from instructions and corresponding world states

Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

Mix it

Then, drain 1 unit from it

Same for 1 more unit
Problem Setup

• Task: follow sequence of instructions
• Learning from instructions and corresponding world states

<table>
<thead>
<tr>
<th>Empty out the leftmost beaker of purple chemical</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Then, add the contents of the first beaker to the second</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mix it</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image3.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Then, drain 1 unit from it</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image4.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Same for 1 more unit</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image5.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Related Work

- Context-dependent language understanding
  - Static environments (e.g., large database)
  - Environments that change over time while instructions are given
- Following instructions in isolation; varying levels of supervision

Miller et al. 1996, Zettlemoyer and Collins 2009, Suhr et al. 2018
Long et al. 2016, Guu et al. 2017, Fried et al. 2018
Today

1. Attention-based model for generating sequences of system actions that modify the environment

2. Exploration-based learning procedure that avoids biases learned early in training
System Actions

- Each beaker is a stack
- Actions are `pop` and `push`

```
pop 2;
pop 2;
push 2 brown;
push 2 brown;
push 2 brown;
```
Meaning Representation

### High-level Program

Mix it

1. Mix it
2. pop 2; pop 2; pop 2;
3. push 2 brown;
4. push 2 brown;
5. push 2 brown;

### System Actions

- pop 2; pop 2; pop 2;
- push 2 brown;
- push 2 brown;
- push 2 brown;

### Representation Engineering

vs.

### Learning Abstractions
Meaning Representation

Mix it

High-level Program

\texttt{mix(\text{prevArg2}(2))}

System Actions

\texttt{pop 2; pop 2; pop 2;}
\texttt{push 2 brown;}
\texttt{push 2 brown;}
\texttt{push 2 brown;}

Representation Engineering vs.
Learning Abstractions
Model

Previous instructions
Throw out first beaker
Pour sixth beaker into last one

Current instruction
It turns brown

Initial state

Current state

- Four inputs
- Output: a sequence of actions
- Attend over each input when generating actions
Throw out first beaker

Previous instructions

Pour sixth beaker into last one

Current instruction

It turns brown

Model

Initial state

Current state

Encode instructions
Model

Previous instructions
- Throw out first beaker
- Pour sixth beaker into last one

Current instruction
- It turns brown

Initial state

Current state

Encode states
Current instruction

Model

Previous instructions

Throw out first beaker

Pour sixth beaker into last one

Current state

Initial state

Initialize decoder
Model

Previous instructions

*Throw out first beaker*

*Pour sixth beaker into last one*

Current instruction

*It turns brown*

Initial state

Current state

Attend over current instruction

Decoder state

Current instruction
Model

Previous instructions
- Throw out first beaker
- Pour sixth beaker into last one

Current instruction
- It turns brown

Initial state

Current state

Attend over previous instructions
Model

Previous instructions
- Throw out first beaker

Current instruction
- Pour sixth beaker into last one
- It turns brown

Initial state

Current state

Attention

Attend over initial state
Current state

Initial state

Previous instructions

Current instruction

Attend over current state

Throw out first beaker

Pour sixth beaker into last one

It turns brown

Decoder state

Current instruction

Previous instructions

Initial state

Attention

Current state
Model

Previous instructions
Throw out first beaker
Pour sixth beaker into last one

Current instruction
It turns brown

Initial state

Current state

Decoder state
pop 7

Current instruction

Previous instructions

Initial state

Current state

Predict action
Model

Previous instructions
- Throw out first beaker
- Pour sixth beaker into last one

Current instruction
- It turns brown

Initial state

Current state

Execute action, update state
Throw out first beaker

Pour sixth beaker into last one

It turns brown

Previous instructions

Current instruction

Initial state

Current state

pop 7

Attention

Attend over new state
Model

Previous instructions

- Throw out first beaker
- Pour sixth beaker into last one

Current instruction

- It turns brown

Initial state

Current state

Action decoder

(pop 7)

(pop 7)
**Model**

- **Previous instructions**
  - *Throw out first beaker*
  - *Pour sixth beaker into last one*

- **Current instruction**
  - *It turns brown*

- **Initial state**

- **Current state**

- **Action decoder**
  - `pop 7`
  - `pop 7`
  - `pop 7`
Model

Previous instructions

Throw out first beaker
Pour sixth beaker into last one

Current instruction

It turns brown

Initial state

Current state

Action decoder

pop 7
pop 7
pop 7
push 7 brown
Model

**Previous instructions**

*Throw out first beaker*  
*Pour sixth beaker into last one*

**Current instruction**

*It turns brown*

**Initial state**

**Current state**

**Action decoder**

- pop 7
- pop 7
- push 7 brown
- push 7 brown
Model

Previous instructions

Throw out first beaker

Pour sixth beaker into last one

Current instruction

It turns brown

Initial state

Current state

Action decoder

pop 7

pop 7

pop 7

push 7 brown

push 7 brown

push 7 brown
Learning from World State Annotation

• Goal: learn a policy that maps from instructions and environment states to actions

Empty out the leftmost beaker of purple chemical

Then, add the contents of the first beaker to the second

Mix it

Then, drain 1 unit from it

Same for 1 more unit
Learning from World State Annotation

• Goal: learn a policy that maps from instructions and environment states to actions

• Approach
  • Learn through exploring the environment and observing rewards
  • Policy gradient with contextual bandit

• Challenge: overcome biases acquired early during learning
Reward Function

$R(s, a, s') = P(s, a, s') + \phi(s') - \phi(s)$
Reward Function

\[ R(s, a, s') = P(s, a, s') + \phi(s') - \phi(s) \]

- +1 if action \( a \) stops the sequence and \( s' \) is the goal state
- -1 if action \( a \) stops the sequence and \( s' \) is **not** the goal state
Reward Function

\[ R(s, a, s') = P(s, a, s') + \phi(s') - \phi(s) \]

+1 if \( s' \) is closer to the goal state than \( s \) (moved closer)
-1 if \( s \) is closer to the goal state than \( s' \) (moved further)
Learning Example

Iteration #1
Add the third beaker to the first

Action Rewards:

<table>
<thead>
<tr>
<th>Action</th>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>push</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Start:  
Goal:  
Rollout:  
Rewards:
Learning Example

Iteration #1
Add the third beaker to the first

<table>
<thead>
<tr>
<th>Action</th>
<th>Rewards:</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop</td>
<td>+ 0 0</td>
</tr>
<tr>
<td>push</td>
<td>0 0</td>
</tr>
</tbody>
</table>

Start: 
Goal: 

Rollout:
pop 2; 
push 1 green; 
pop 3; 
push 1 yellow;

Rewards:
-1 
-1 
+1 
-1
Learning Example

Iteration #1

Add the third beaker to the first

Start: 

Goal: 

Action
Rewards:  

<table>
<thead>
<tr>
<th>Action</th>
<th>Rewards</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop</td>
<td>+ 1</td>
</tr>
<tr>
<td>push</td>
<td>0 2</td>
</tr>
</tbody>
</table>

No positive reward for push actions

Rollout: 

Rewards: 

pop 2; -1
push 1 green; -1
pop 3; +1
push 1 yellow; -1
Learning Example

Iteration #2

Add the third beaker to the first

Start:

Goal:

<table>
<thead>
<tr>
<th>Action</th>
<th>Rewards</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop</td>
<td>1 1</td>
</tr>
<tr>
<td>push</td>
<td>0 2</td>
</tr>
</tbody>
</table>

Rollout: Rewards:
Learning Example

**Iteration #2**

*Add the third beaker to the first*

<table>
<thead>
<tr>
<th>Action</th>
<th>Rewards</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop</td>
<td>+1 1</td>
</tr>
<tr>
<td>push</td>
<td>0 2</td>
</tr>
</tbody>
</table>

**Start:**

- 1
- 2
- 3

**Goal:**

- 1
- 2
- 3

**Rollout:**

- pop 3;
- push 1 green;
- pop 1;
- push 1 green;

**Rewards:**

- +1
- -1
- +1
- -1
Learning Example

Iteration #2
Add the third beaker to the first

Start: 🥤🥤🥤
Goal: 🥤🥤🥤

<table>
<thead>
<tr>
<th>Action</th>
<th>Rewards</th>
<th>Action</th>
<th>Rewards</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop</td>
<td>+1</td>
<td>push</td>
<td>-1</td>
</tr>
<tr>
<td>push</td>
<td>0</td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

No positive reward for push actions

Rollout:
- pop 3; +1
- push 1 green; -1
- pop 1; +1
- push 1 green; -1
Learning Example

Iteration #3
Add the third beaker to the first

<table>
<thead>
<tr>
<th>Action</th>
<th>Rewards:</th>
</tr>
</thead>
<tbody>
<tr>
<td>pop</td>
<td>+ 3 1</td>
</tr>
<tr>
<td>push</td>
<td>0 4</td>
</tr>
</tbody>
</table>

Start:  
Goal: 

Rollout:
- pop 3;
- pop 3;
- pop 1;

Rewards:
- +1
- +1
- -1

Quickly learned a strong bias against push actions
Learned Biases

• Early during learning, model learns it can get positive reward by predicting the pop actions

• Less likely to get positive reward with push action

• Becomes biased against push — during later exploration, push is never sampled!

• Compounding effect: never learns to generate push actions
Single-step Reward Observation

- **Our approach:** observe reward of all actions by looking one step ahead during exploration

- Observe reward for actions like **push**
Learning Algorithm

For each training example:

1. Rollout: sample sequence of state-action pairs from the current policy

2. For each state visited in the rollout,

   A. For each possible action, execute action and observe reward

3. Update parameters based on observed rewards for all states and actions
Simple Exploration

Start state

Only observe states along sampled trajectory

Single-step Reward Observation

Start state

Observe sampled states **and** single-step ahead
Simple Exploration

Start state

Only observe states along sampled trajectory

Single-step Reward Observation

Start state

Observe sampled states and single-step ahead
Simple Exploration

Start state

Only observe states along sampled trajectory

Single-step Reward Observation

Start state

Observe sampled states and single-step ahead
Simple Exploration

Start state

Only observe states along sampled trajectory

Single-step Reward Observation

Start state

Observe sampled states and single-step ahead
Simple Exploration

Start state

Only observe states along sampled trajectory

Single-step Reward Observation

Start state

Observe sampled states and single-step ahead
Simple Exploration

Start state

Only observe states along sampled trajectory

Single-step Reward Observation

Start state

Observe sampled states **and** single-step ahead
Simple Exploration

Start state

Only observe states along sampled trajectory

Single-step Reward Observation

Start state

Observe sampled states and single-step ahead
Single-step Observation

Iteration #4

Add the third beaker to the first

Start: □ □ □

Goal: 1 2 3
Iteration #4

Add the third beaker to the first

Rollout:

pop 3;
pop 3;
pop 1;

Current State:

Single-step Observation:
Single-step Observation

Iteration #4

Add the third beaker to the first

Rollout:
pop 3;
pop 3;
pop 1;

Current State:

Goal:

Start: 

Single-Step Actions:

-1 pop 1;
Iteration #4

Add the third beaker to the first

Rollout:
pop 3;
pop 3;
pop 1;

Current State:

Single-step Observation:

Goal:

Start:

Single-Step Actions:
-1 pop 1;
-1 pop 2;
Iteration #4
Add the third beaker to the first

Rollout:
pop 3;
pop 3;
pop 1;

Current State:

Single-step Observation:

Single-Step Actions:
-1 pop 1;
-1 pop 2;
+1 pop 3;

Start: 

Goal: 

1 2 3
Add the third beaker to the first

Current State:

Rollout:
pop 3;
pop 3;
pop 1;

Single-step Observation:

Single-Step Actions:
-1 pop 1;
-1 pop 2;
+1 pop 3;
+1 push 1 orange;
Single-step Observation

Iteration #4
Add the third beaker to the first

Rollout: pop 3;
        pop 3;
        pop 1;

Current State:

Goal:

Start:

Single-Step Actions:
Single-step Observation

Iteration #4

Add the third beaker to the first

Rollout:

pop 3;
pop 3;
pop 1;

Current State:

Start: 

Goal: 

Single-Step Actions:

-1 pop 1;
-1 pop 2;
+1 pop 3;
+1 push 1 orange
Add the third beaker to the first

Iteration #4

Start: 
Goal: 

Single-step Observation

Rollout:

Current State:

Single-Step Actions:
Single-step Observation

Iteration #4
Add the third beaker to the first

Rollout:  
pop 3;  
\[\longrightarrow\] pop 3;  
pop 1;

Current State:  

Start:  

Goal:  

Single-Step Actions:

-1 pop 1;  
-1 pop 2;  
-1 pop 3;  
\[\vdots\]  
+1 push 1 orange
Single-step Observation

Iteration #4

*Add the third beaker to the first*

Rollout:  

Current State:

\[
\text{pop 3; } \quad \text{pop 3; } \quad \text{pop 3; } \quad \text{pop 1; }
\]
Add the third beaker to the first

Iteration #4

Current State:
pop 3;
pop 3;
→ pop 1;

Start:
Goal:

Single-Step Actions:
-1 pop 1;
-1 pop 2;
-1 pop 3;
⋮
-1 push 1 orange
+1 push 1 yellow
Experimental Setup

• SCONE (Long et al. 2016): Alchemy, Scene, Tangrams

• Training data: start state and a sequence of instructions and goal states

• Standard evaluation metric: after following a sequence of instructions, is the world state correct?
Alchemy

Mix it

pop 2;
pop 2;
pop 2;
push 2 brown;
push 2 brown;
push 2 brown;
The person with a red shirt and a blue hat moves to the right end.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>remove_person</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>remove_hat</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add_person</td>
<td>10 red</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add_hat</td>
<td>10 blue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tangrams

Swap the third and fourth figures

remove 4
insert 3 boat
Results

- Outperform previous methods by up to 25%, while mapping directly to system actions.

Test Results

- Alchemy
- Scene
- Tangrams

Final state accuracy
Results

- Outperform previous methods by up to 25%, while mapping directly to system actions
- Performance is comparable to direct supervision
Learning Methods

- Single-step observations overcome biases that get model stuck

Final state accuracy Development Results

Alchemy | Scene | Tangrams

<table>
<thead>
<tr>
<th></th>
<th>SESTRA</th>
<th>Policy Gradient</th>
<th>Contextual Bandit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alchemy</td>
<td>71.8</td>
<td>52.3</td>
<td>52.6</td>
</tr>
<tr>
<td>Scene</td>
<td>56.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tangrams</td>
<td>60.3</td>
<td>52.3</td>
<td>52.6</td>
</tr>
</tbody>
</table>
Ablations

• Need access to previous instructions
• Need access to world state

Final state accuracy
Development Results
• Attention-based model for generating sequences of atomic actions that modify the environment

• Exploration-based learning procedure that avoids biases learned early in training

https://github.com/clic-lab/scone