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

Alane Suhr and **Yoav Artzi**
Department of Computer Science and Cornell Tech
Cornell University
New York, NY, 10044
{suhr, yoav}@cs.cornell.edu

A Domain-Specific Implementation Details

For each domain ALCHEMY, SCENE, and TANGRAMS, we describe the world state representation, state distance function, transition function, and the state encoder. For all states s, $s = T(s, \texttt{STOP})$.

ALCHEMY The world state in ALCHEMY is a sequence of beakers $\langle \bar{b}_1, \bar{b}_2, ..., \bar{b}_N \rangle$ of fixed length $N = 7$. Each beaker $\bar{b}_i = \langle c_{i,1}, c_{i,2}, ...c_{i,|\bar{b}_i|} \rangle$ is a variable length sequence containing chemical units c, each one of six possible colors. The distance between two world states is the sum over distances for each corresponding beaker pair. The distance between two beakers is the edit distance of the list of chemical units in each. The action space of ALCHEMY includes two action types, `POP` and `PUSH`. The `POP` action takes one argument: `N` $\in \{1, \ldots, N\}$ denoting the beaker to pop a chemical unit from. The `PUSH` action takes two arguments: `N` and `C`, one of six colors. The transition function T is defined by two cases: (a) $T(s, a =$ `PUSH N C`$)$ will return a state where `C` is added to the beaker with index `N`; and (b) $T(s, a =$ `POP N` $)$ will remove the top element from the beaker with index `N`, or if the beaker with index `N` is empty, the input state s is returned. The state encoding function ENC is parameterized by (a) ϕ^c, an embedding function for each color; (b) ϕ^p, a positional embedding function for each beaker position; and (c) LSTM^B, a forward RNN used to encode each beaker. We encoder each beaker $\bar{b}_i$ with an RNN:

$$\mathbf{h}^b_{i,j} = \text{LSTM}^B \left(\phi^c(c_{i,j}); \mathbf{h}^b_{i,j-1} \right) \ .$$

ENC returns a set of N vectors $\{\mathbf{h}_i\}_{i=1}^N$, where each $\mathbf{h}_i = [\mathbf{h}^b_{i,|\bar{b}_i|}; \phi^p(i)]$ represents a beaker.

SCENE The world state in SCENE is a sequence of positions $S = \langle p_1, p_2, ..., p_N \rangle$ of fixed length $N = 10$. Each position is a tuple $p_i = \langle s_i, h_i \rangle$, where s_i is a shirt color h_i is a hat color. There are six colors, and a special `NULL` marker indicating no shirt or hat is present. The distance between two world states is the sum over positions of the number of steps required to modify two corresponding positions to be the same given the domain actions space. The action space of SCENE includes four action types: `APPEAR_PERSON`, `APPEAR_HAT`, `REMOVE_PERSON`, and `REMOVE_HAT`. `APPEAR_PERSON` and `APPEAR_HAT` take two arguments: a position index `N` and a color `C`. `REMOVE_PERSON` and `REMOVE_HAT` take one argument: a position index `N`. The transition function T is defined by four cases: (a) $T(s, a =$ `APPEAR_PERSON N C`$)$ returns a state where position `N` contains shirt color `C` if the shirt color in position `N` is `NULL`, otherwise the action is invalid and the input state s is returned; (b) $T(s, a =$ `APPEAR_HAT N C`$)$ is defined analogously to `APPEAR_PERSON`; (c) $T(s, a =$ `REMOVE_PERSON N`$)$ returns a state where the shirt color at position `N` is set to `NULL` if there is a color at position `N`, otherwise the action is invalid and the input state s is returned; and (d) $T(s, a =$ `REMOVE_HAT N`$)$ is defined analogously to `REMOVE_PERSON`. The state encoding function ENC is parameterized by (a) ϕ^c, an embedding function for shirt and hat colors; (b) ϕ^p, a positional embedding for each position in the scene; and (c) LSTM^S, a bidirectional RNN over all positions in order. Each position is embedded using a function $\phi'(p_i) = [\phi^c(s_i); \phi^c(h_i); \phi^p(i)]$. We compute a sequence of forward hidden states:

$$\overrightarrow{\mathbf{h}}^s_i = \overrightarrow{\text{LSTM}^S} \left(\phi'(p_i); \overrightarrow{\mathbf{h}}^s_{i-1} \right) \ .$$

The backward RNN is equivalent. ENC returns the set $\{\mathbf{h}_i\}_{i=1}^N$, where $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}^s_i; \overleftarrow{\mathbf{h}}^s_i; \phi'(p_i)]$ rep-

		1	2	3	4
Coref.	Beaker	24	7	2	0
	Action	3	0	0	0
	Action + Arguments	1	0	0	0
Ellipsis	Beaker	0	0	3	1

Table 1: Count of phenomena in ALCHEMY.

		1	2	3	4
Coref.	Person	42	16	5	3
	Hat	2	0	0	0
	Action + Arguments	3	0	0	0
	Position	2	0	0	0

Table 2: Count of phenomena in SCENE.

		1	2	3	4
Coref.	Object	13	0	0	0
	Object via Arguments	6	10	2	1
	Position	0	2	0	0
	Action	5	2	0	0
	Action + Arguments	1	0	0	0
Ellipsis	Position	3	0	0	0
	Action + Arguments	1	0	0	0

Table 3: Count of phenomena in TANGRAMS.

resents a position.

TANGRAMS The world state in TANGRAMS is a list of positions $T = \langle p_1, p_2, ..., p_n \rangle$ of a variable length n. Each position contains one of five unique shapes. The distance function between states is the edit distance between the lists, with a cost of two for substitutions. The action space of TANGRAMS includes two action types, `INSERT` and `REMOVE`. The `INSERT` action takes two arguments: a position `N` $\in \{1, \cdots, M\}$, where M is the maximum length of a state in the TANGRAMS dataset, and a shape type `T`, which is one the five possible shapes. The `REMOVE` action takes a single argument: a position `N`. The transition function T is defined by two cases: (a) $T(s, a = \texttt{INSERT N T})$ returns a state where the shape `T` is in position `N` and all objects to its right shifted by one position if `T` is not already in the state, otherwise the action is invalid and s is returned; and (b) $T(s, a = \texttt{REMOVE N})$ returns a state where the object in position `N` was removed if `N` $\leq n$, otherwise the action is invalid and s is returned. The state encoding function ENC is parameterized by (a) $\mathbf{h}_{NULL}$, a vector used when $n = 0$; (b) ϕ^s, an embedding function for the shapes; and (c) ϕ^p, a positional embedding of the position i. ENC returns a set $\{\mathbf{h}_i\}_{i=1}^n$, where $\mathbf{h}_i = [\phi^p(i); \phi^s(p_i)]$ is the position encoding, or it returns $\{\mathbf{h}_{NULL}\}$ if the state contains no objects.

B Data Analysis

We analyze SCONE to identify the frequency of various discourse phenomena in the three domains, including explicit coreference and ellipsis, which is implicit reference to previous entities. We observe references to previous objects (e.g., beakers in ALCHEMY), actions, locations (e.g., positions in SCENE), and world states. We analyze thirty development set interactions for each domain for presence of these references. We define the age of each referent as the number of turns since it was last explicitly mentioned. This illustrates the extent to which this dataset challenges models for context-dependent reasoning.

ALCHEMY Table 1 shows phenomena counts in ALCHEMY. Each interaction contains on average 1.4 references dependent on the interaction history. Each non-first utterance contains on average 0.3 references. The most common form of reference is explicit coreference (Coref.) to previously-mentioned beakers, for example *mix it*. Other references are to previous actions, referring to the action only (e.g., *same with the last beaker*) or the action as well as the arguments (e.g., *same for one more unit*, referring to draining one unit from a previously-used beaker). Ellipsis occurred four times in the thirty evaluated interactions, for example *then, drain 1 unit*, implicitly referring to a specific beaker to drain from.

SCENE Table 2 shows phenomena counts in SCENE. Each interaction contains on average 2.4 references dependent on the interaction history. Each non-first utterance contains on average 0.6 references. The most common form of reference is explicit coreference (Coref.) to previously-mentioned people, for example *he moves to the left end*. Coreference also occurs on hat colors (e.g., *he gives it back*), actions along with their arguments (e.g., *they did it again* referring to trading specific hats), and positions (e.g., *he moves back*).

TANGRAMS Table 3 shows phenomena counts in TANGRAMS. Each interaction contains on average 1.7 references dependent on the interaction history. Each non-first utterance contains on average 0.4 references. The most common form of reference is on objects via reference to a previous step, for example *put the item you just removed in the second spot*. This requires recalling actions taken in previous turns, including the actions' ar-

guments and the previous world state. Coreference (Coref.) also occurs for positions (e.g., *...where the last deleted figure was*), actions (e.g., *do the same with the second to last figure and one before it*), and actions along with the previously-used arguments (e.g., *repeat the first step*). Ellipsis occurs for positions (e.g., *add it again*, implicitly referring to the item's previous location) and actions along with their arguments (e.g., *undo the last step*).

C Attention Analysis

Figure 1 shows attention distributions for a hand-picked example in ALCHEMY. We show the attention probabilities (α in Section 4) for the current and previous utterances, initial state, and current state throughout execution. In this example, the previous-instruction attention puts most of the weight on *brown one* during generation, which is the referent of *it* in the current instruction. The initial and current state attentions are placed heavily on the beaker being manipulated. However, for randomly selected examples, we observe that the attention distribution does not always correspond to intuitions about what should be attended on. Figures 2, 3, and 4 show examples of attention distributions for three random instructions in the development sets of the three domains where the action sequence was predicted correctly.

D Architecture and Training Details

Model Architecture We use an embedding size of 50 for words and action types and arguments. Action embedding is a concatenation of the embeddings of each part, including the action type and the two arguments; an embedded action is a vector of size 150. Embeddings of colors in ALCHEMY and SCENE, and shapes in TANGRAMS, are of size 10. Positional embeddings are of size 10. $\mathbf{W}^d$ and $\mathbf{W}^a$ are square matrices. All matrices are initialized by sampling from the uniform distribution $U\left(\left[-\sqrt{\frac{6}{M+N}}, \sqrt{\frac{6}{M+N}}\right]\right)$ (Glorot and Bengio, 2010), where M and N are the matrix dimensionality. All RNNs are single-layer LSTMs. For the main model, both the instruction encoder and action sequence decoder use a hidden size of 100 in each direction. The action sequence decoder is initialized by first setting the hidden state and cell memory to zero-vectors, and passing in a zero-vector to update the states, after which attention is computed for the first time. For ALCHEMY, the world state encoder has a hidden size of 20. For SCENE, the world state encoder has a hidden size of 5.

Training We apply dropout in three places: (a) in each attention computation after multiplying by $\mathbf{W}$; (b) after computing $\mathbf{h}_k$, the input to each decoder step; and (c) for all attention keys except for the current utterance. For POLICYGRADIENT, CONTEXTUALBANDIT, and our approach, we optimize parameters using RMSPROP (Tieleman and Hinton, 2012). For supervised learning, we use ADAM (Kingma and Ba, 2014) for optimization. We use a learning rate of 0.001 for all experiments. Our validation set is a held-out subset containing 7.0% of the training data. We stop training by observing the instruction-level reward on the validation set. We use patience for early stopping. We reset patience to $50 \cdot 1.005^x$ the x-th time the reward has improved on the validation set, decrease by one each epoch reward does not improve, and stop when patience runs out. Regardless of patience, we terminate training after 200 epochs. We tune λ, δ, and M on the development set. In ALCHEMY, $\lambda = 0.1$, $\delta = 0.15$, and $M = 7$. In SCENE, $\lambda = 0.07$, $\delta = 0.2$, and $M = 5$. In TANGRAMS, $\lambda = 0.1$, $\delta = 0.0$, and $M = 5$.

References

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

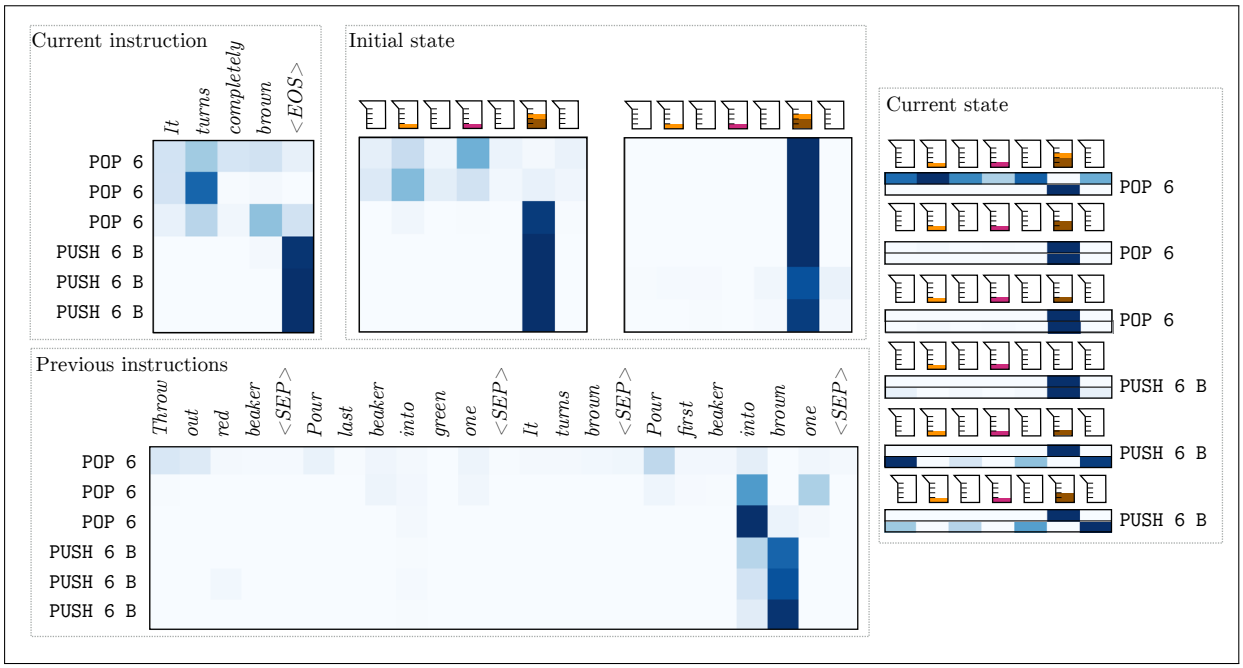

Figure 1: Example of the attention distributions for executing the instruction *It turns completely brown* in ALCHEMY. This is the fifth instruction in the interaction. The correct action sequence mixes the chemicals in the sixth beaker by removing the three units and re-adding three brown units. Our model correctly predicts this sequence. We show the different attention distributions when generating this sequence of actions. Clockwise starting from the top left: (a) attention over the current instruction; (b) two attention heads over the initial state; (c) two attention heads over the current world state, which changes following each action; and (d) the attention over the previous instructions in the interaction.

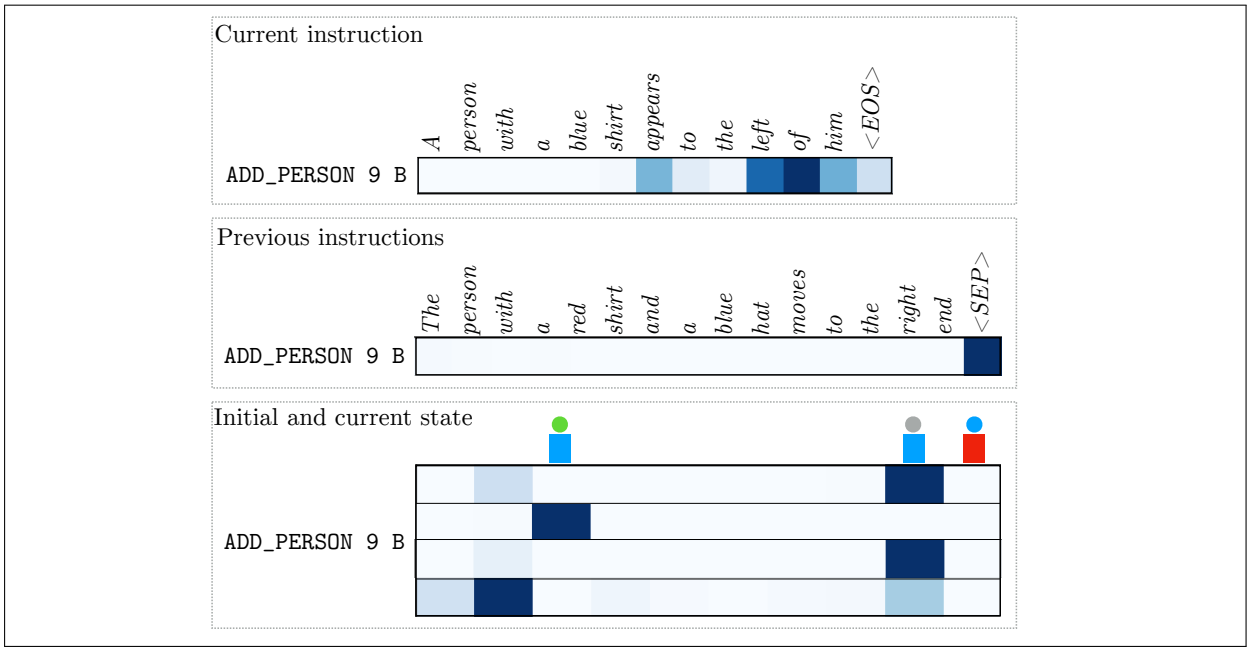

Figure 2: Example of attention for a randomly selected instruction from the development set for SCENE. The instruction *A person with a blue shirt appears to the left of him* is the second in the interaction, following the instruction *The person with a red shirt and a blue hat moves to the right end*. The correct action sequence consists of a single action, `ADD_PERSON 9 B`, where a person wearing a blue shirt appears in position 9, to the left of the person in the red shirt. Our model predicts this action correctly. We show the different attention distributions when generating this sequence of a single action. From top to bottom: (a) attention over the current instruction; (b) attention over the previous instruction; and (c) attention over the world state. As the sequence contains a single action only, the current and initial world states are the same, and their distributions are shown together. There are two attention heads over both the initial (top two rows) and current (bottom two rows) world states.

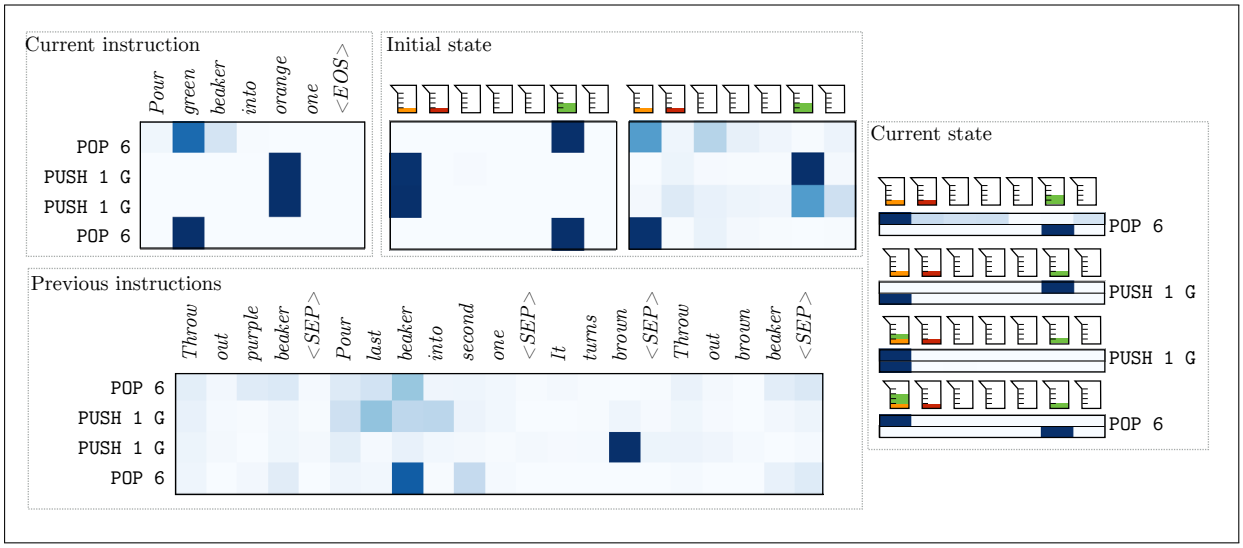

Figure 3: Example of attention for a randomly selected instruction from the development set for ALCHEMY. The instruction executed is *Pour green beaker into orange one*, the fifth instruction in the sequence. We show the different attention distributions when generating the correct action sequence, which removes green items from the sixth beaker and adds the same number of green items to the beaker containing orange. Clockwise starting from the top left: (a) attention on the current instruction; (b) the two attention heads over the initial state; (c) the two attention heads over the current state as it changes during execution; and (d) attention over previous instructions.

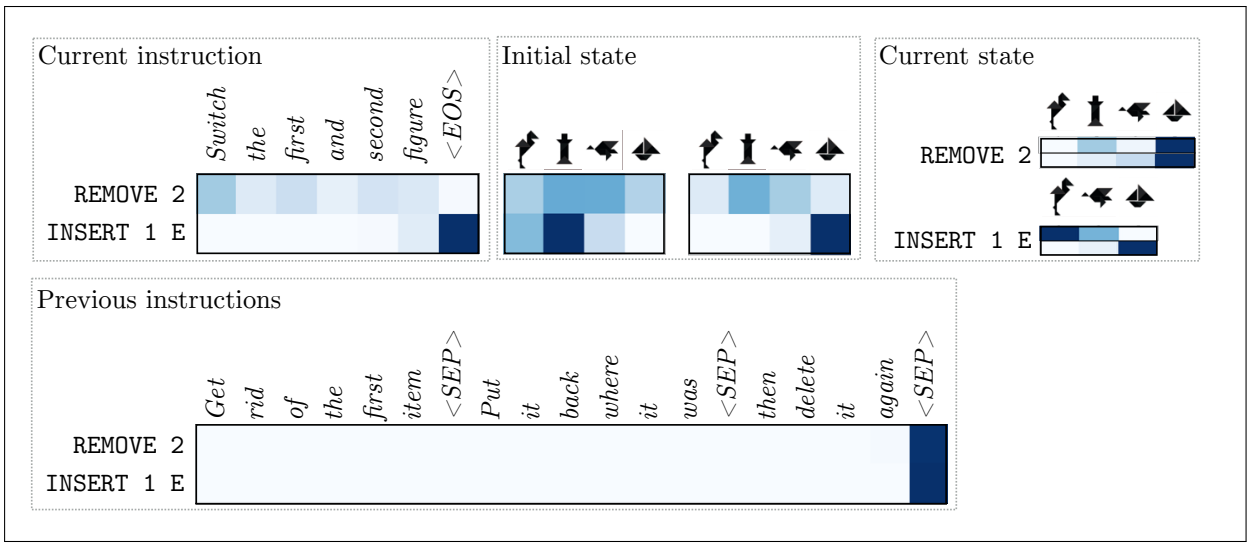

Figure 4: Example of attention for a randomly selected instruction from the development set for TANGRAMS. The instruction executed is *Switch the first and second figure*, the fourth instruction in the sequence. We show the different attention distributions when generating the correct action sequence, which removes the figure in position two and adds it in position one, thereby swapping the first two items. Clockwise starting from the top left: (a) attention on the current instruction; (b) the two attention heads over the initial state; (c) the two attention heads over the current state as it changes during execution; and (d) attention over previous instructions.