Recognizing Behaviors and the Internal State of the Participants

Wesley Kerr
Department of Computer Science
University of Arizona
Tucson, AZ 85721–0077
Email: wkerr@cs.arizona.edu

Paul Cohen
Department of Computer Science
University of Arizona
Tucson, AZ 85721–0077
Email: cohen@cs.arizona.edu

***Abstract*—Psychological research has demonstrated that subjects shown animations consisting of nothing more than simple geometric shapes perceive the shapes as being alive, having goals and intentions, and even engaging in social activities such as chasing and evading one another. While the subjects could not directly perceive affective state, motor commands, or the beliefs and intentions of the actors in the animations, they still used intentional language to describe the moving shapes. We present representations and algorithms that enable an artificial agent to correctly recognize other agents' activities by observing their behavior. In addition, we demonstrate that if the artificial agent learns about the activities through participation, where it has access to its own internal affective state, motor commands, etc., it can then infer the unobservable internal state of other agents.**

I. Introduction

Previous research demonstrated that subjects shown animations consisting of nothing more than simple geometric shapes perceive the shapes as alive, having goals and intentions, and even interacting in social relationships such as chasing and evading [1], [2]. For example, subjects in the classic Heider and Simmel study consistently labeled the larger triangle in Figure 1 as a bully who was constantly chasing the smaller triangle and circle [1].

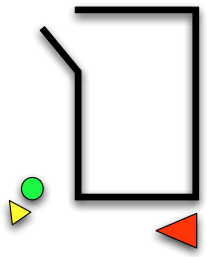

Fig. 1. Single frame from an animations similar to the original Heider and Simmel animation.

When subjects ascribe intentions to geometric primitives like those shown in Heider and Simmel's research (see Figure 1), which information guides the process? Blythe et al. [2] showed that patterns of motion of the actors in animations are sufficient information to classify the activities in the animations. In fact, their classification algorithm outperformed human subjects.

Blythe's algorithm mapped patterns of motion onto class labels for intentional states, which isn't quite the same as knowing anything about intentional states. One of Heider and Simmel's subjects described the larger triangle in Figure 1 as "blinded by rage and frustration." Blythe's algorithm couldn't come up with such a description. An agent that classifies episodes by patterns of motion knows about patterns of motion, not about rage and frustration, even if these words are provided as episode labels. So how might an agent infer affective states?

In both the Heider and Simmel animations and the animations developed by Blythe et al., subjects can only observe a subset of the features that are available, i.e. positions, velocities, sizes, colors, etc. The subjects cannot directly perceive the affective state, motor commands, and the beliefs and intentions of the actors in the animations. Yet they infer affective states and describe them with intentional language.

We think humans infer affective states given non-affective observables such as positions and velocities by calling on their own affective experiences. Observables cue, or cause to be retrieved from memory, schemas that include learned affective components, which are inferred or "filled in" as interpretations of patterns of motion or other non-affective observables.

We designed the CAVE algorithm to classify and visualize episodes. CAVE learns to classify through supervised learning with labeled episodes, such as instances of robot behaviors. Visualization produces a compact image of the sometimes complex interactions between intervals. An agent built with CAVE learns to classify activities by first performing them, itself. It therefore has access to both observable aspects of activities such as motion, and private aspects such as intentions, emotional states, and motor commands. It can use observations of other agents to retrieve activities from memory and project the hidden or private aspects of the activities onto other agents.

We will describe the representations and algorithms for an agent that learns to classify activities in Wubble World. Wubble World is a virtual environment with simulated physics in which softbots, called wubbles, interact with objects [3]. Wubble World is instrumented to collect distances, velocities, locations, colors, sizes, and other sensory information (including motor commands) and represent them with propositions such as $Above(wubble, box)$ (the wubble is above the box) and

978-1-4244-6902-4/10/$26.00 ©2010 IEEE

$PVM(wubble)$ (the wubble is experiencing positive vertical motion). Wubbles perform several kinds of activities: they jump over boxes, jump on boxes, approach boxes, push boxes, and move around boxes.

In outline, Section II describes a representation of activities that supports inference about unobservable states, and Section III describes how these representations are learned. Sections IV and V report an experiment in which hidden aspects of motor control are inferred, and Section VII describes ongoing work in scenarios that resemble very closely those created by Heider and Simmel.

II. Qualitative Sequences

The datasets generated by activities in Wubble World are called *episodes*. Each episode is a collection of *intervals*, and each interval is a tuple containing a proposition and the times at which the proposition becomes true and false. A proposition can become true (and false) multiple times within an episode; each of these instances is represented as a separate interval.

We assume that different examples of one activity share patterns of intervals. More colloquially, the intervals in similar episodes tell the same story with minor variations. Thus, one may classify episodes by their constituent patterns of intervals. This is not the only way to do it: A cleaning agent might classify a cleaning episode by the objects it interacts with, such as pots and pans, rather than what was done with the pots and pans. But our focus here is classifying episodes by patterns of activities, represented by intervals.

Episodes and intervals have different durations, start times, end times, and constituent propositions, so our representation of episodes must accommodate and generalize over these variations. For example, any episode generated by a wubble jumping over a box begins with an interval in which the wubble is in front of a box, followed by an interval in which the wubble moves more or less quickly towards the box. At some point, which will vary between episodes, the wubble will jump. Despite variations, all episodes of jumping over a box contain a common pattern of intervals: the wubble moves towards the box, jumps, moves above the box, and finishes on the ground behind the box.

Relationships between intervals can be described by *Allen relations* [4]. Allen recognized that, after eliminating symmetries, there are only seven possible relationships between two intervals, shown in Figure 2. Allen relations are qualitative in the sense that they represent the temporal order of events, specifically, the beginnings and endings of intervals, but not the durations of intervals.

The intervals in an episode may be sorted to generate a canonical representation of an episode. Ordered intervals, also known as *normalized* intervals in [5], [6], are sorted according to earliest end time. If two intervals finish at the same time, they are further sorted by earliest start time, and if the start and end times are identical, then they are sorted alphabetically by proposition name. A canonical representation of episodes ensures that a classifier only works with one, not many, representations of an episode.

(**x** *before* **y**)	(**x** *b* **y**)	x y
(**x** *meets* **y**)	(**x** *m* **y**)	x y
(**x** *overlaps* **y**)	(**x** *o* **y**)	x y
(**x** *starts-with* **y**)	(**x** *s* **y**)	x y
(**x** *during* **y**)	(**x** *d* **y**)	x y
(**x** *finishes-with* **y**)	(**x** *f* **y**)	x y
(**x** *equals* **y**)	(**x** *e* **y**)	x y

Fig. 2. Allen Relations

Our episode representation, which we call a *qualitative sequence*, is a sequence of Allen relations between intervals in the episode. These intervals are first ordered or normalized according to the rules as we just described. Then we construct the Allen relations between *all* of the pairs of intervals in order, as described in Algorithm 1. An illustrative episode and the resulting qualitative sequence is shown in Table I. The letters **A, B** and **C** denote propositions, and an assertion such as (**C** 1 3) means that proposition **C** was true in the interval [1,3]. The remainder of the paper will use the terms episode and sequence interchangeably.

Algorithm 1 MAKE-SEQUENCE(I)

```
S = ()
for i = 1 to size(I) do
   for j = (i + 1) to size(I) do
      S ← S + allen(I[i], I[j])
   end for
end for
return S
```

A qualitative sequence can be shortened by defining a window that determines how close any two intervals must be to consider one *before* the other. This is known as the *interaction window*. Assume that we have two intervals $(p_1\ s_1\ e_1)$ and $(p_2\ s_2\ e_2)$ such that p_1 and p_2 are proposition names, s_1 and s_2 are start times, e_1 and e_2 are end times, and $e_1 < e_2$. The two intervals are said to interact if $s_2 \leq e_1 + w$ where w is the interaction window. If we set $w = 1$ in Table I, then we would remove the relation (**C** *before* **C**) .

III. Learning

The CAVE learning algorithm is an incremental algorithm designed to discover and highlight patterns in sequences of propositional data. The agent learns to classify episodes given episode and activity label pairs.

Episodes are first converted into qualitative sequences of Allen relations and learning is done with these sequences. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ be a set of qualitative sequences with

978-1-4244-6902-4/10/$26.00 ©2010 IEEE

Intervals	Sequence
	(**C** *meets* **A**)
(**C** 1 3)	(**C** *before* **B**)
(**A** 3 6)	(**C** *before* **C**)
(**B** 4 9)	(**A** *overlaps* **B**)
(**C** 6 10)	(**A** *meets* **C**)
	(**B** *overlaps* **C**)

TABLE I
AN EPISODE COMPRISING FOUR INTERVALS AND THE CORRESPONDING QUALITATIVE SEQUENCE.

the same activity label. We define the *signature* of the activity label, $\mathcal{S}_c$, as an ordered sequence of *weighted* Allen relations. (The only difference between a signature and a qualitative sequence is these weights.) We select a sequence at random from $\mathcal{S}$ to serve as the initial signature, $\mathcal{S}_c$, and initialize all of its weights to 1. After this, $\mathcal{S}_c$ is updated by combining it with the other sequences in $\mathcal{S}$, processed one at a time.

Two problems are solved during the processing of the sequences in $\mathcal{S}$. First, the sequences are not identical, so $\mathcal{S}_c$ must be constructed to represent the most frequent relations in the sequences. The weights in $\mathcal{S}_c$ are used for this purpose. Second, because a relation can appear more than once in a sequence $\mathcal{S}_i$, there can be more than one way to align $\mathcal{S}_i$ with $\mathcal{S}_c$. These problems are related because the frequencies of relations in $\mathcal{S}_c$ depend on how sequences are successively aligned with it.

Relations	Weight
(**A** *finishes-with* **D**)	1
(**A** *overlaps* **B**)	5
(**A** *meets* **C**)	5
(**D** *overlaps* **B**)	1
(**D** *meets* **C**)	1
(**B** *overlaps* **C**)	5

TABLE II
THE SIGNATURE OF AN ILLUSTRATIVE SET OF SEQUENCES.

Updating the signature $\mathcal{S}_c$ with a sequence $\mathcal{S}_i$ occurs in two phases. In the first phase, $\mathcal{S}_i$ is optimally aligned with $\mathcal{S}_c$. The alignment algorithm, described below, penalizes candidate alignments for relations in $\mathcal{S}_c$ that are not matched by relations in $\mathcal{S}_i$, and rewards matches. These penalties and rewards are functions of the weights stored with the signature. Table III shows the signature from Table II aligned with the sequence in Table I. In the second phase, the weights in the signature $\mathcal{S}_c$ are updated. If a relation in $\mathcal{S}_i$ is aligned with one from $\mathcal{S}_c$, then the weight of this relation is incremented by one (e.g., (**A** *overlaps* **B**) in Table III). Otherwise the weight of the relation is initialized to one (e.g., (**C** *meets* **A**) in Table III) and it is inserted into $\mathcal{S}_c$ at the location selected by the alignment algorithm.

Updating the signature relies on the Needleman-Wunsch global sequence alignment algorithm [7]. The algorithm uses dynamic programming to find an optimal alignment between two sequences. Alignments are constructed by selecting operators that modify each sequence to look more like the other. Operators include inserting a symbol from one sequence into the other, deleting a symbol from one of the sequences, substituting a symbol from one sequence for another in the other sequence, or matching a symbol from each of the sequences. However, to ensure that no information from sequences in $\mathcal{S}$ is lost, we allow the sequence alignment only to insert or match, not delete or substitute, relations. Both the cost of insertion and matching are determined by the weights that are stored with the signature as mentioned above.

$\mathcal{S}_c$ Aligned	S_i Aligned	$\mathcal{S}_c$ Updated	
–	(**C** *m* **A**)	(**C** *meets* **A**)	1
–	(**C** *b* **B**)	(**C** *before* **B**)	1
–	(**C** *b* **C**)	(**C** *before* **C**)	1
(**A** *f* **D**)	–	(**A** *finishes-with* **D**)	1
(**A** *o* **B**)	(**A** *o* **B**)	(**A** *overlaps* **B**)	6
(**A** *m* **C**)	(**A** *m* **C**)	(**A** *meets* **C**)	6
(**D** *o* **B**)	–	(**D** *overlaps* **B**)	1
(**D** *m* **C**)	–	(**D** *meets* **C**)	1
(**B** *o* **C**)	(**B** *o* **C**)	(**B** *overlaps* **C**)	6

TABLE III
UPDATING THE SIGNATURE $\mathcal{S}_c$. THE LEFTMOST COLUMN IS THE CURRENT SIGNATURE (FROM TABLE 2). THE SECOND COLUMN IS A SEQUENCE $\mathcal{S}_i$ (FROM TABLE 1). THE THIRD COLUMN IS THE OPTIMAL ALIGNMENT OF $\mathcal{S}_i$ WITH $\mathcal{S}_c$. THE FINAL COLUMN IS THE WEIGHTS OF THE UPDATED SIGNATURE.

Because the process of updating signatures is careful to not remove anything from any of the sequences in $\mathcal{S}$, signatures become stuffed with large numbers of Allen relations that occur very infrequently, and thus have low weights. We use a simple heuristic to clean up the signature: After K training episodes, all of the relations in the signature with weights less than or equal to n are removed. All of our experiments use $K = 10$ and $n = 3$, meaning that the signature is pruned of all relations occurring less than 4 times after a total of 10 training episodes. The signature is again pruned after 20 training episodes, and so forth.

IV. CLASSIFICATION

The signatures learned by the CAVE algorithm function as classifiers as follows. Recall that $\mathcal{S} = \{S_1, \ldots, S_k\}$ is a set of qualitative sequences with the same activity label; for example, all the sequences in $\mathcal{S}$ might be examples of *jump over*. Now suppose we have N sets of qualitative sequences, $\Sigma = \{\mathcal{S}^1, S^2, \ldots, S^N\}$ each of which has a different activity label, and its own signature, derived as described in the Section III. A novel, unlabeled sequence matches each signature to some degree, determined by aligning it with each signature, as described earlier. The novel sequence is given the activity label that corresponds to the signature it matches best.

V. INFERRING HIDDEN STATE

Episodes have observable and unobservable propositions depending on who is observing. When $wubble_1$ is performing the activity it observes all of the propositions, but when $wubble_1$ observes $wubble_2$ performing the same activity,

978-1-4244-6902-4/10/$26.00 ©2010 IEEE

$wubble_1$ cannot perceive the motor commands, emotional state, and intentional states of $wubble_2$.

By *hidden relations* we mean relations that include one or more propositions that are not directly observable in the behavior of other agents, and so must be inferred. Our approach to inferring hidden relations is to have agents learn signatures of their own behaviors, in which these relations are *not* hidden. Then, when an agent observes another's behavior, it matches the observable relations to signatures of its own behavior, and uses these to infer unobservable relations in other's behavior.

To illustrate, assume that the signature in Table II was learned by an agent and the observed behavior of another agent does not include proposition **B**. The first agent would infer that the following hidden relations must also be true: (**A** *overlaps* **B**) , (**D** *overlaps* **B**) , and (**B** *overlaps* **C**) .

In general, sequences can contain many hidden relations. The most frequent of these relations are the most likely to occur when observing other agents. Therefore, our agent selects the α most frequently occurring hidden relations to be the inferred hidden state. In the previous example, if we were to set $\alpha = 2$, then we would remove the relation that occurs the least, in this case (**D** *overlaps* **B**) .

VI. Experiments

Our approach to deal with hidden propositions and relations has two parts: Observables are used to select signatures, then signatures are used to infer hidden relations. Our first experiment tests how accurately observables can select signatures; a test of classification accuracy. Then we test how accurately the selected signatures can fill in or infer hidden relations.

Experiments were conducted in the Wubble World virtual environment described in Section I. Wubbles learned signatures associated with these tasks: *jumping over* a box, *jumping on* a box, *approaching* a box, *pushing* a box, moving around a box to the *left*, moving around a box to the *right*. Episodes were generated by manually controlling a wubble to repeatedly perform one of these tasks. Each episode was unique, in that the wubble would start closer or farther from the box, move more or less quickly, and so on. There were 37 episodes labeled *jump over*, 20 episodes labeled *jump on*, and 25 episodes for each of the remaining tasks.

A. Experiment 1: Classification

The CAVE algorithm was evaluated on its ability to correctly classify episodes in Wubble World. Classifying episodes is not trivial because, even though the tasks that generated them are different, the episodes share propositions and intervals. They all involved the same block, movement, and similar perceptions.

We present results for classification accuracy when all propositions are observable and when some are hidden. In the latter case the hidden propositions are motor commands such as **Forward(wubble)** and **Backward(wubble)**. After we present results for classification accuracy, we will describe how well these hidden propositions and relations that include them are inferred.

The classification performance of the CAVE algorithm is contrasted with a k-nearest neighbor classifier operating on qualitative sequences, that is, sequences of Allen relations. To select a class label for an episode, the k-NN classifier calculates the distance between the episode and all training episodes (across all class labels). Distances are determined by a longest common subsequence algorithm, and as in the CAVE algorithm, more similar sequences have lower distances. The k-NN classifier selects the most frequently occurring class label among the k closest neighbors of an unlabeled episode, and for this paper we've set $k = 10$. We prefer a weighted k-NN classifier because it weights each of the closest neighbors by the distance to the unlabeled episode, so that closer neighbors have more weight. The class label with the most weight attached to it is the label given to an unlabeled episode.

1) Results: The performance of the CAVE algorithm and the k-NN classifier is the average number of correctly classified episodes in a 10-fold cross validation across the 157 episodes, shown in Table IV. Both successfully classify over 90% of the episodes when all propositions are observable. Also shown in Table IV is the performance of the CAVE and k-NN classifiers when motor commands are withheld from the test set. Here we see a dip in performance for each, but both still correctly classify roughly 90% of the episodes.

	Observable		Hidden	
	μ	σ	μ	σ
k-NN	97.8%	3.9	94.3%	5.1
CAVE	94.4%	4.3	89.4%	7.5

TABLE IV
PERFORMANCE ON THE WUBBLE WORLD CLASSIFICATION TASK. μ IS THE AVERAGE PERCENT OF EPISODES CORRECTLY CLASSIFIED AND σ IS THE STANDARD DEVIATION.

Figure 3 shows the learning curve for the CAVE algorithm when tested on episodes lacking propositions corresponding to motor commands. The CAVE algorithm learned signatures for each activity one episode at a time. After presenting a single training example consisting of an activity label and sequence pair, we tested performance on a classification task.

The learning curve was generated by selecting five episodes from each type of activity to function as the test set. The remainder of the episodes were used as the training set to be presented one at a time to the CAVE algorithm. In Figure 3 the *x*-axis is the number of training instances seen by the CAVE algorithm, and the *y*-axis marks the percent correctly classified out of 30 episodes in the test set. The performance is averaged over 20 different randomizations of the test set and training presentation order.

As the graph indicates, the CAVE algorithm learns to classify the episodes in the test set very quickly in both conditions. Episodes in the test set are labeled with one of six possible class labels. After seeing 24 episodes (on average only four episodes per activity) the CAVE agent is able to classify almost 70% of the test set correctly. This demonstrates that the signatures learned by the CAVE algorithm quickly identify

978-1-4244-6902-4/10/$26.00 ©2010 IEEE

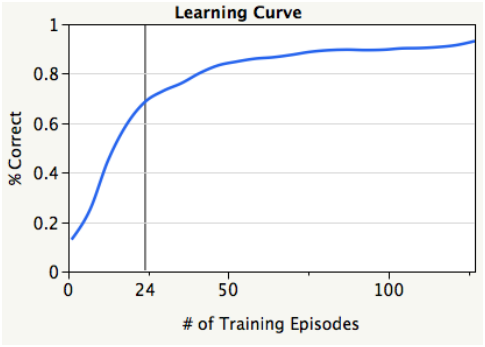

Fig. 3. Learning curve of the CAVE algorithm generated by presenting labeled training instances one at a time to the CAVE algorithm.

relations within the training episodes that allow the algorithm to correctly classify activities.

B. Experiment 2: Inferring hidden relations

The next experiment demonstrates how well our agent can infer unobservable relations, i.e. relations that involve motor commands. The method is to have it learn signatures when it can observe motor commands, then test its ability to infer motor commands when shown sequences in which they are unobservable. This is a precision test.

Five episodes are randomly selected from each type of activity to serve as the test set. The agent builds a signature for each activity from the remaining episodes. From signature we select the $\alpha = 10$ most frequent relations that contain at least one of the motor commands as *inferred* relations. Some subset of these relations occur in each of the test sequences. We measure the overlap between the inferred relations and those in the test sequence.

Table V shows the results. Each cell in the table contains the average amount of overlap between the most frequent relations in the signature and the actual hidden relations that occur in each episode from the test set. On average, 99% of the inferred relations from the *push* signature correspond to hidden relations in unseen *push* episodes. This means that, in expectation, if the agent were to correctly classify a *push* episode, then 99% of its inferred relations would be present in the episode.

What if the agent classifies an episode incorrectly? How will this affect the accuracy of inferred relations? Table V shows the accuracy of inferred relations for all pairs of activities. Along the diagonal are cases where the classification of the activity is accurate, off-diagonal cases are incorrectly classified episodes. For example, if the agent classifies a *push* as an *approach* then only 56% of the inferred hidden relations will actually occur in the episode. (The fact that there is any overlap at all is due to the overlap between the activities: in both cases the agent approaches the box.)

This experiment confirms that if the agent can correctly classify the activity of another agent, it can select the α most frequent relations as inferred hidden relations, and they will be correct with high accuracy.

	Signatures					
	approach	push	jump on	jump over	left	right
approach	0.66	0.12	0.15	0.00	0.31	0.32
push	0.56	0.99	0.14	0.01	0.27	0.25
jump on	0.45	0.22	0.91	0.90	0.25	0.27
jump over	0.40	0.38	0.82	0.99	0.27	0.21
left	0.36	0.49	0.16	0.01	0.99	0.95
right	0.34	0.51	0.15	0.01	0.94	0.99

TABLE V
A MATRIX SHOWING THE PERCENT OVERLAP BETWEEN THE α MOST FREQUENT HIDDEN RELATIONS IN THE SIGNATURE AND THE HIDDEN RELATIONS THAT EXIST IN THE TEST SET, BUT ARE NOT OBSERVABLE.

VII. PREVIOUS WORK

There is a body of literature on extracting patterns of intervals from data. Most of this research focuses on extracting temporal patterns that occur with frequency greater than some threshold, also known as *support*. In [6], support is defined as the frequency of the pattern within a sliding window, ensuring a degree of locality between the internal relationships of the pattern, whereas in [5], [8], the support of a pattern depends on the number of episodes containing the pattern. Other work, such as [9], [10], focuses on extracting patterns that are determined to be statistically significant through hypothesis testing. This is the only work we know of relating support for a pattern to evidence against the null hypothesis that no pattern exists. None of the work cited above focuses on classifying episodes.

Batal et al. [11] proposed a method for classifying multivariate propositional time series. The authors first extract large patterns from the time series with methods similar to those just discussed. The complete set of large patterns are pruned by hypothesis testing to generate a subset of patterns that will serve as a binary feature vector. Each training episode results in a single binary feature vector such that each value is true when the corresponding pattern is found within the episode and false otherwise. The feature vectors and class labels are used to train a traditional classifier (e.g. SVM). One problem with this approach is that the classifier must be completely retrained whenever new training episodes are acquired.

Other research has focused on the problem of inferring intent from motion data alone. In Blythe et al. [2], human subjects controlled simulated insects on networked machines to perform a variety of activities. Activities ranged from chase and flee behaviors to mating rituals that had one insect attempting to court another. The authors trained a three-layer neural network to classify the activities based on observable information, such as relative distance between the insects, relative velocity, absolute velocity, relative heading, etc. The trained neural network outperformed human subjects on a classification task. Intention was not explicitly modeled in this system even though it could detect it through motion cues.

More recently, Crick and Scassellati showed that sequences of belief states constructed from a folk physics-based interpretation of position and motion could correctly classify multiplayer games like tag and keepaway [12]. This work is an

extension of previous work where they could correctly identify which player was "it" in a game of tag [13].

Recent research has proposed the mirror neuron system within humans as a neural mechanism for recognizing and understanding the intentions of other people [14], [15]. Although we do not claim that the CAVE agent understands intentions in the same way that humans do, both lines of research propose a mechanism such that one agents understands the intentionality of another by relating the observed actions with one's own previous memories and internal states.

VIII. Future Work

In the examples presented here, the hidden propositions are motor commands issued by the wubbles, but our ultimate goal is to infer intentional and particularly affective state. This will require wubbles to have intentional and affective state while they learn activities.

To aid in this we have built a 2D physics based simulator wherein the agents are controlled by the SOAR cognitive architecture [16]. We've augmented SOAR with an emotional system based on those described in [17], [18]. A screenshot from the 2D simulator is shown in Figure 4. The blue circle and the red triangle represent two agents and the light green areas in front of them represent the limits of their visual systems. The blue circle has just escaped from the red triangle, and is still experiencing alarm. The agents follow scripts, much like actors in movies, that involve common themes such as "escaping a bully." As agents act out the interactions with other agents, they experience intentional and emotional state, and learn to incorporate them into signatures. We expect these agents to use their signatures to infer the intentional states of other agents in similar interactions, such as the classic Heider and Simmel movie.

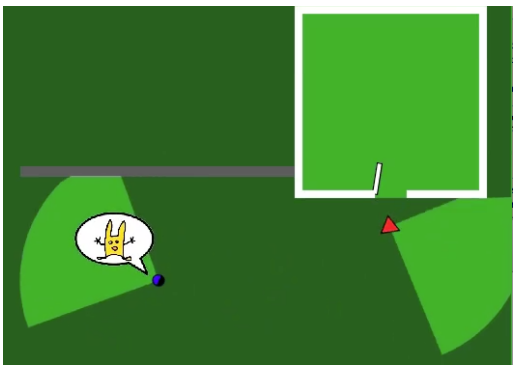

Fig. 4. Still frame from our 2D simulator. The blue circle is expressing an emotion while escaping from the red triangle.

IX. Summary

In this paper we presented a representation of activities that supports inference about unobservable states, and the learning and inference algorithms that go along with it. The agent learns from qualitative sequences of Allen relations that capture relations between intervals. Activities are classified using signatures built up from training episodes with the same activity label. With high accuracy the agent is able to infer hidden relations when it observes other agents performing activities it has already learned about.

In the future, the CAVE agent will work directly with real-valued multivariate time series. Previous research has detailed how to convert real-valued time series into symbolic form [19], [20]. Typically the conversion process introduces new propositional variables that are descriptive of the original time series. We will address how the system scales under much larger numbers of variables and more complex datasets.

References

[1] F. Heider and M. Simmel, "An experimental study of apparent behavior," *The American Journal of Psychology*, vol. 57, no. 2, p. 243, 1944.

[2] P. W. Blythe, P. M. Todd, and G. F. Miller, "How motion reveals intention: Categorizing social interactions," in *Simple heuristics that make us smart*. Oxford University Press, USA, 1999.

[3] W. Kerr, P. Cohen, and Y.-H. Chang, "Learning and playing in wubble world," in *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2008, pp. 66–71.

[4] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[5] E. Winarko and J. F. Roddick, "Armada - an algorithm for discovering richer relative temporal association rules from interval-based data," *Data and Knowledge Engineering*, vol. 63, no. 1, pp. 76–90, 2007.

[6] F. Höppner, "Learning temporal rules from state sequences," in *IJCAI-01 Workshop on Learning from Temporal and Spatial Data*, 2001.

[7] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, March 1970.

[8] P. shan Kam and A. W.-C. Fu, "Discovering temporal patterns for interval-based events," *Lecture Notes in Computer Science*, vol. 1874, pp. 317–326, 2000.

[9] P. R. Cohen, C. Sutton, and B. Burns, "Learning effects of robot actions using temporal associations," in *International Conference on Development and Learning*, 2002.

[10] M. Fleischman and D. Roy, "Grounded language modeling for automatic speech recognition of sports video," in *Proceedings of ACL-08: HLT*, 2008, pp. 121–129.

[11] I. Batal, L. Sacchi, R. Bellazzi, and M. Hauskrecht, "Multivariate time series classification with temporal abstractions," in *Florida Artificial Intelligence Research Society Conference*, 2009.

[12] C. Crick and B. Scassellati, "Inferring narrative and intention from playground games," in *7th IEEE International Conference on Development and Learning*, 2008.

[13] C. Crick, M. Doniek, and B. Scassellati, "Who is it? inferring role and intent from agent motion," in *6th IEEE International Conference on Development and Learning*, 2007.

[14] M. Iacoboni, I. Molnar-Szakacs, V. Gallese, G. Buccino, J. C. Mazziotta, and G. Rizzolatti, "Grasping the intentions of others with one's own mirror neuron system," *PLoS Biol*, vol. 3, no. 3, p. e79, 02 2005.

[15] Z. K. Agnew, K. K. Bhakoo, and B. K. Puri, "The human mirror system: A motor resonance theory of mind reading," *Brain Research Reviews*, vol. 54, no. 2, pp. 286–293, June 2007.

[16] J. F. Lehman, J. Laird, and P. Rosenbloom, "A gentle introduction to soar: An architecture for human cognition: 2006 update," 2006.

[17] C. Breazeal, "Emotion and sociable humanoid robots," *International Journal of Human-Computer Studies*, vol. 59, no. 1-2, pp. 119–155, 2003.

[18] M. Masuch, K. Hartman, and G. Schuster, "Emotional agents for interactive environments," in *The Fourth International Conference on Creating, Connecting and Collaborating through Computing*, 2006, pp. 96–102.

[19] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zaït, "Querying shapes of histories," in *Proceedings of the 21th International Conference on Very Large Data Bases*, 1995, pp. 502–514.

[20] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 107–144, 2007.

978-1-4244-6902-4/10/$26.00 ©2010 IEEE