Fluent Learning: Elucidating the Structure of Episodes

Paul R. Cohen
Department of Computer Science
University of Massachusetts, Amherst
cohen@cs.umass.edu

Abstract

Fluents are logical descriptions of situations that persist, and composite fluents are statistically significant temporal relationships (nearly identical with those in Allen's temporal calculus) between fluents. This paper presents an algorithm for learning composite fluents incrementally. The algorithm is tested with a large dataset of mobile robot episodes. The algorithm is given no knowledge of the episodic structure of the dataset (i.e., it learns without supervision) yet it discovers fluents that correspond well with episodes. More generally, the algorithm elucidates hidden structure in time series of binary vectors.

Introduction

The problem addressed here is unsupervised learning of structures in time series. When we make observations over time, we effortlessly chunk the observations into *episodes*: I am driving to the store, stopping at the light, walking from the parking lot into the store, browsing, purchasing, and so on. Episodes contain other episodes; purchasing involves receiving the bill, writing a check, saying thank you, and so on. What actually happens, of course, is a continuous, extraordinarily dense, multivariate stream of sound, motion, and other sensor data, which we somehow perceive as events and processes that start and end. This paper describes an incremental algorithm with which a robot learns to chunk processes into episodes.[1]

The Problem

Let $\vec{x}_t$ be a vector of sensor values at time t. Suppose we have a long sequence of such vectors $S = \vec{x}_0, \vec{x}_1, \ldots$. *Episodes* are subsequences of S, and they can be nested hierarchically; for example, a robot's *approach-and-push-block* episode might contain an *approach* episode, a *stop-in-contact* episode, a *push* episode, and so on. Suppose one does not know the boundaries of episodes and has only the sequence S: How can S be chunked into episodes? A *model-based* approach assumes we have models of episodes to help us interpret S. For example, most people who see a robot approach a block, make contact, pause, and start to push would interpret the observed sequence by matching it to models of approaching, touching, and so on. Where do these models come from? The problem here is to learn them. We wish to find subsequences of S that correspond to episodes, but we do not wish to say (or do not know) what an episode is, or to provide any other knowledge of the generator of S.

This problem arises in various domains. It is related to the problem of finding changepoints in time series and motifs in genomics (motifs are repeating, meaningful patterns). In our version of the problem, a robot must learn the episodic structure of its activities.

Note that episode is a *semantic* concept, one that refers not to the observed sensor data S but to *what is happening* – to the interpretation of S. Thus, episodes are not merely subsequences of S, they are subsequences that correspond to *different things that happen*. To qualify as an episode in S, a subsequence of S should span or cover one or more things that happen, but it should not span part of one and part of another. Suppose we know the processes P_S that generate S, labelled `a,b,c,f`, and we have two algorithms, X and Y, that somehow induce models, labelled `1, 2,` and `3`, as shown here:

```
Ps  aaaaabbbbbbaaaacccffffffffaaaaaffffffffffaaaa
X   11111122222221122223333333331111133333333333111
Y   22211111111111113333311111112222222222333333333
```

The first five ticks of S are generated by process `a`, the next six by `b`, and so on. Algorithm X does a pretty good job: its models correspond to types of episodes. When it labels a subsequence of S with `1`, the subsequence was generated entirely or mostly by process `a`. When it labels a subsequence with `2`, the subsequence was generated by process `b` or `c`. It's unfortunate that algorithm X doesn't induce the distinction between processes of type `b` and `c`, but even so it does much better than algorithm Y, whose model instances show no correspondence to the processes that generate S.

Fluents and Temporal Relationships

In general, the vector $\vec{x}_t$ contains real-valued sensor readings, such as distances, RGB values, amplitudes, and so on. The algorithm described here works with binary

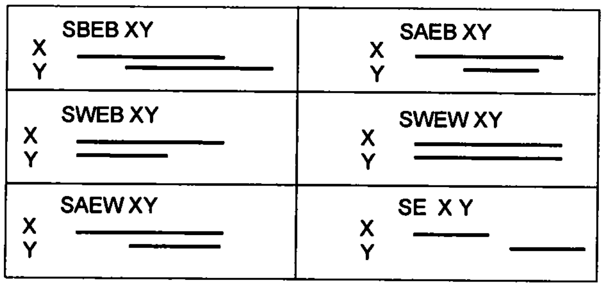
[1] http://www-eksl.cs.umass.edu/papers/AtkinAA97.pdf describes a primitive version of the algorithm tested with simulated data. The current, improved version is tested with several hours of data from a Pioneer 1 robot..

vectors only. In practice, this is not a great limitation if one has a perceptual system of some kind that takes real-valued sensor readings and produces propositions that are true or false. We did this in our experiments with the Pioneer 1 robot. Sensor readings such as translational and rotational velocity, the output of a "blob vision" system, sonar values, and the states of gripper and bump sensors, were inputs to a simple perceptual system that produced the following nine propositions: STOP, ROTATE-RIGHT, ROTATE-LEFT, MOVE-FORWARD, NEAR-OBJECT, PUSH, TOUCH, MOVE-BACKWARD, STALL

Nine propositions permit $2^9 = 512$ world states, but many of these are impossible (e.g., moving forward and backward at the same time) and only 35 unique states were observed in the experiment, below. States are not static: the robot can be in the state of moving forward. Moving forward near an object is a state in the sense that it remains true that the robot is moving forward and near an object.

States with persistence are called *fluents* [4]. They have beginnings and ends. Allen [1] gave a logic for relationships between the beginnings and ends of fluents. We use a nearly identical set of relationships:

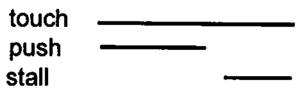
SBEB (X starts before Y, ends before Y; Allen's "overlap");
SWEB (Y starts with X, ends before X; Allen's "starts")
SAEW (Y starts after X, ends with X; Allen's "finishes")
SAEB (Y starts after X, ends before X; Allen's "during")
SWEW (Y starts with X, ends with X; Allen's "equal")
SE (Y starts after X ends; amalgamating Allen's "meets" and "before")



In Allen's calculus, "meets" means the end of X coincides exactly with the beginning of Y, while "before" means the former event precedes the latter by some interval. In our work, the truth of a predicate such as SE or SBEB depends on whether start and end events happen within a window of brief duration; for example, SE XY is true if Y starts within a few ticks of the end of X; these events can coincide, but they needn't. Similarly, SBEB XY is true if Y does *not* start within a few ticks of the start of X; if it did, then the appropriate relationship would be SWEB. Said differently, "starts with" means "starts within a few ticks of..." and "starts before" means "starts more than a few ticks before..." The reason for this window is that on a real robot, it takes time for events to show up in sensor data and be processed perceptually into propositions, so coinciding events will not necessarily produce propositional representations at exactly the same time.

Learning Composite Fluents

As noted, a fluent is a state with persistence. A composite fluent is a statistically significant temporal relationship between fluents. Suppose that every time the robot pushed an object, it eventually stalled. This relationship might look like this:



Three temporal relationships are here: SWEB(touch,push), SBEW(stall,touch) and SE(stall,push). But there are other ways to represent these relationships, too; for example, the relationship SBEW(stall,SWEB(touch,push)) says, "the relationship between touch and push begins before and ends with their relationship with stall." In what follows, we describe how to learn representations like these that correspond well to episodes in the life of a robot.

Let $\rho \in \{SBEB, SWEB, SAEW, SWEW, SAEB, SE\}$, and let f be a proposition (e.g., MOVING-FORWARD). Composite fluents have the form:

$$F \leftarrow f \;\; or \;\; \rho(f,f)$$
$$CF \leftarrow \rho(F,F)$$

That is, a fluent F may be a proposition or a temporal relationship between propositions, and a composite fluent is a temporal relationship between fluents. As noted earlier, a situation has many alternative fluent representations, we want a method for choosing some over others. The method will be statistical: We will only accept $\rho(F,F)$ as a representation if the constituent fluents are statistically associated, if they "go together."

An example will illustrate the idea. Suppose we are considering the composite fluent SE(jitters,coffee), that is, whether the start of the jitters begins after the end of having coffee. Four frequencies are relevant:

	jitters	not jitters
coffee	a	b
not coffee	c	d

Testing the hypothesis SE(jitters,coffee)

Certainly, **a** should be bigger than **b**, that is, I should get the jitters more often than not after drinking coffee.

Suppose this is true, so **a** = **kb**. If the relative frequency of jitters is no different after I drink, say, orange juice, or talk on the phone (e.g., if **c** = **kd**) then clearly there's no special relationship between coffee and jitters. Thus, to accept SE(jitters,coffee), I'd want **a** = **kb** and **c** = **md**, and **k>>m**. The chi-square test (among others) suffices to test the hypothesis that the start of the jitters fluent is independent of the end of the drinking coffee fluent.

It also serves to test hypotheses about the other five temporal relationships between fluents. Consider a composite fluent like SBEB(brake,clutch): When I approach a stop light in my standard transmission car, I start to brake, then depress the clutch to stop the car stalling; later I release the brake to start accelerating, and then I release the clutch. To see whether this fluent – SBEB(brake,clutch) – is statistically significant, we need two contingency tables, one for the relationship "start braking then start to depress the clutch" and one for "end braking and then end depressing the clutch":

	s(clutch)	s(x≠clutch)
s(brake)	a1	b1
s(x≠brake)	c1	d1

	e(clutch)	e(x≠clutch)
e(brake)	a2	b2
e(x≠brake)	c2	d2

Testing the hypothesis SBEB(brake,clutch)

Imagine some representative numbers in these tables: Only rarely do I start something *other* than braking and then depress the clutch, so **c1** is small. Only rarely do I start braking and then start something other than depressing the clutch (otherwise the car would stall), so **b1** is also small. Clearly, **a1** is relatively large, and **d1** bigger, still, so the first table has most of its frequencies on a diagonal, and will produce a significant χ^2 statistic. Similar arguments hold for the second table. When *both* tables are significant, we say SBEB(brake,clutch) is a significant composite fluent.

Fluent learning algorithm

The fluent learning algorithm incrementally processes a time series of binary vectors. At each tick, a bit in the vector $\vec{x}_t$ is in one of four states:

Still off:	$x_{t-1} = 0 \wedge x_t = 0$
Still on:	$x_{t-1} = 1 \wedge x_t = 1$
Just off:	$x_{t-1} = 1 \wedge x_t = 0$
Just on:	$x_{t-1} = 0 \wedge x_t = 1$

The fourth case is called *opening*; the third case *closing.* Recall that the simplest fluents *f* are just propositions, i.e., bits in the vector $\vec{x}_t$, so we say a simple fluent *f* closes or opens when the third or fourth case, above, happens; and denote it open*(f)* or close*(f)*. Things are slightly more complicated for composite fluents such as SBEB(f_1,f_2), because of the ambiguity about which fluent opened. Suppose we see open(f_1) and then open(f_2). It's unclear whether we have just observed open(SBEB(f_1,f_2)), open(SAEB(f_1,f_2)), or open(SAEW(f_1,f_2)). Only when we see whether f_2 closes after, before, or with f_1 will we know which of the three composite fluents *opened* with the opening of f_2.

The fluent learning algorithm maintains contingency tables that count co-occurrences of open and close events. For example, the tables for SBEB(f_1,f_2) are just:

	open(f2,t+m)	open(f≠f2,t+m)
open(f1,t)		
open(f≠f1,t)		

	close(f2,t+m)	close(f≠f2,t+m)
close(f1,t)		
close(f≠f1,t)		

That is, f_2 must open after f_1 and close after it, too. We restrict the number of ticks, m, by which one opening must happen after another: m must be bigger than a few ticks, otherwise we treat the openings as simultaneous; and it must be smaller than the length of a short-term memory. The short term memory has two kinds of justification. First, animals do not learn associations between events that occur far apart in time. Second, if every open event could be paired with every other (and every close event) over a long duration, then the fluent learning system would have to maintain an enormous number of contingency tables.

At each tick, the fluent learning algorithm first decides which simple and composite fluents have closed. With this information, it can disambiguate which composite fluents opened at an earlier time (within the bounds of short term memory). Then, it finds out which simple and composite fluents have just opened, or might have opened (recall, some openings are ambiguous). To do this, it consults a list of accepted fluents, which initially includes just the simple fluents – the bits in the time series of bit vectors – and later includes statistically-significant composite fluents. This done, it can update the open and close contingency tables for all fluents that have just closed. Next, it updates the χ^2 statistic for each table and it adds the newly significant composite fluents to the list of accepted fluents.

The algorithm is incremental because new composite fluents become available for inclusion in other fluents as they become significant.

An Experiment

The dataset is a time series of 22535 binary vectors of length 9, generated by a Pioneer 1 mobile robot as it executed 48 replications of a simple *approach-and-push* plan. In each trial, the robot visually located an object, oriented to it, approached it rapidly for a while, slowed down to make contact, attempted to push the object, and, after a variable period of time, stalled and backed up. In one trial, the robot got wedged in a corner of its playpen.

Data from the robot's sensors were sampled at 10Hz and passed through a simple perceptual system that returned values for nine propositions: STOP, ROTATE-RIGHT, ROTATE-LEFT, MOVING-FORWARD, NEAR-OBSTACLE, PUSHING, MOVING-BACKWARD, STALLED. The robot's sensors are noisy and its perceptual system makes mistakes, so some of the 35 observed states contained semantic anomalies (e.g., 55 instances of states in which the robot is simultaneously stalled and moving backward).

Because the robot collected data vectors at 10Hz and its actions and environment did not change quickly, long runs of identical states are common. In this application, it is an advantage that fluent learning keys on temporal relationships between open and close events and does not attend to the durations of the fluents: A push fluent ends as a stall event begins, and this relationship is significant irrespective of the durations of the push and stall.

Each tick in the time series of 22353 vectors was marked as belonging to exactly one of seven episodes:

A: start a new episode, orientation and finding target
B1: forward movement
B2: forward movement with turning or intruding periods of turning
C1: B1 + an object is detected by sonars
C2: B2 + an object is detected by sonars
D: robot is in contact with object (touching, pushing)
E: robot stalls, moves backwards or otherwise ends D

This markup was based on our knowledge of the robot's controllers (which we wrote. The question is how well do the induced fluents correspond to these episodes.

Results

The composite fluents involving three or more propositions discovered by the fluent learning system are shown in Figure 1. (This is not the complete set of such fluents, but the others in the set are variants of those shown, e.g., versions of fluent 4, involving two and four repetitions of SWEW(push,move-forward) respectively.) In addition, the system learned 23 composite fluents involving two propositions. Eleven of these involved temporal relationships between move-forward, rotate-right and rotate-left. Let's begin with the fluents in Figure 1. The first captures a strong regularity in how the robot approaches an obstacle. Once the robot detects an object visually, it moves toward it quite quickly, until the sonars detect the object. At that point, the robot immediately stops, and then moves forward more slowly. Thus, we expect to see SAEB(near- object,stop), and we expect this fluent to start before move-forward, as shown in the first composite fluent. This fluent represents the bridge between episodes of types B and C.

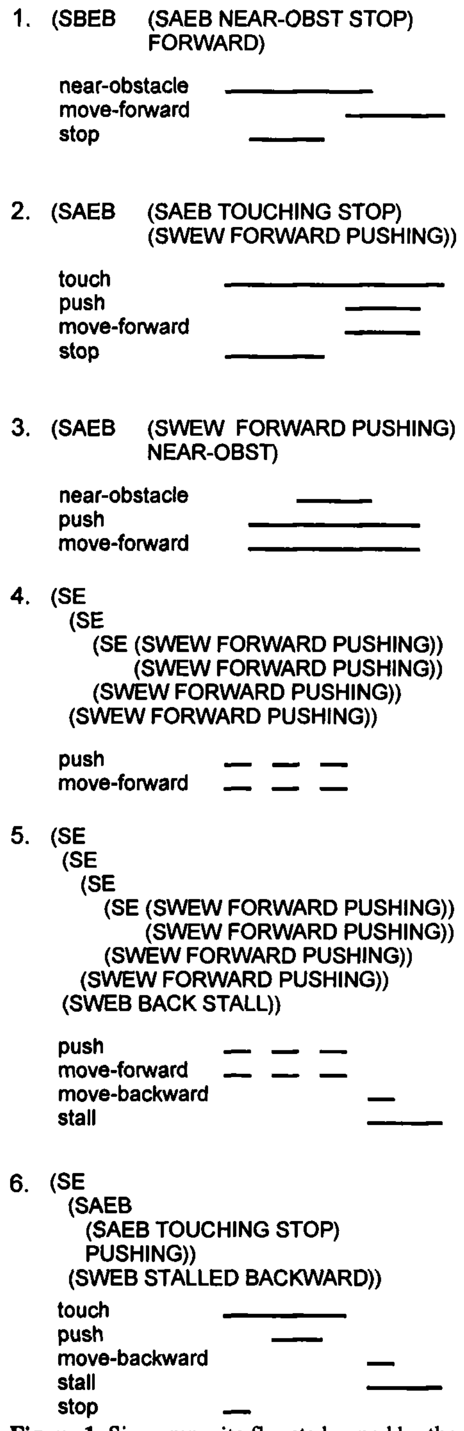
The second fluent shows that the robot stops when it touches an object but remains touching the object after the stop fluent closes (SWEB(touch,stop)) and this composite fluent starts before and ends before another composite fluent in which the robot is simultaneously moving forward and pushing the object. This is an exact description of episodes of type D, above.

The third fluent in Figure 1 is due to the noisiness of the Pioneer 1 sonars. When the sonars lose contact with an object, the near-object fluent closes, and when contact is regained, the fluent reopens. This happens frequently during the pushing phase of each trial because, when the robot is so close to a (relatively small) box, the sonar signal off the box is not so good.

The fourth fluent in Figure 1 represents episodes of type D, pushing the object. The robot often stops and starts during a push activity, hence the SE fluents. The fifth fluent represents the sequence of episodes of type D and E: The robot pushes, then after the pushing composite fluent ends, the move-backward and stall fluent begins. It is unclear why this fluent is SWEW(stall,move-backward), implying that the robot is moving while stalled, but the data do indeed show this anomalous combination, suggesting a bug in the robot's perceptual system.

The last fluent is another representation of the sequence of episodes D and E through E. It shows the robot stopping when it touches the object, then pushing the object, and finally moving backward and stalled.

At first glance, it is disappointing that the fluent learning algorithm did not find higher-order composite fluents – involving two or more temporal relationships between fluents – for episodes of type A and B. During episode A the robot is trying to locate the object visually, which involves rotation; and during episodes B1 and B2 it moves quickly toward the object. Unfortunately, a bug in the robot controller resulted in a little "rotational kick" at the beginning of each forward movement, and this often knocked the robot off its chosen course, and sometimes required it to visually reacquire the object. Consequently, during episodes of type A and B2, we see many runs of combinations of moving forward, rotating left, rotating right, and sometimes backing up. This is why 15 of 23 fluents of two propositions involve these propositions. For example, we have SAEB, SAEW, SWEB, and SBEB fluents relating move-forward and rotate-left.



```
1. (SBEB  (SAEB NEAR-OBST STOP)
           FORWARD)

   near-obstacle
   move-forward
   stop

2. (SAEB  (SAEB TOUCHING STOP)
           (SWEW FORWARD PUSHING))

   touch
   push
   move-forward
   stop

3. (SAEB  (SWEW  FORWARD PUSHING)
           NEAR-OBST)

   near-obstacle
   push
   move-forward

4. (SE
    (SE
      (SE (SWEW FORWARD PUSHING))
          (SWEW FORWARD PUSHING))
      (SWEW FORWARD PUSHING))
    (SWEW FORWARD PUSHING))

   push
   move-forward

5. (SE
    (SE
     (SE
       (SE (SWEW FORWARD PUSHING))
           (SWEW FORWARD PUSHING))
       (SWEW FORWARD PUSHING))
     (SWEW FORWARD PUSHING))
    (SWEB BACK STALL))

   push
   move-forward
   move-backward
   stall

6. (SE
    (SAEB
     (SAEB TOUCHING STOP)
     PUSHING))
    (SWEB STALLED BACKWARD))
   touch
   push
   move-backward
   stall
   stop
```

Figure 1. Six composite fluents learned by the system

None of these fifteen fluents was eventually combined into higher order fluents. Why not? The reason is simply that during episodes of type A and B, it is common to see two things happening simultaneously or sequentially, but it is uncommon to see systematic associations between three or more things.

In sum, the fluents in Figure 1 represent the episodic structure of episodes C, D and E; while episodes of types A and B are represented by composite fluents of two propositions, typically moving forward, rotating left, and rotating right. Qualitatively, then, these fluents are not bad representations of episodes and sequences of episodes in the robot data set. Results of a more quantitative nature follow.

Recall that each of 22535 ticks of data belongs to one of seven episode types, so we can obtain a time series of 22535 episode labels in the set A,B1,B2,C1,C2,D,E. Similarly, we can "tile" the original dataset with a set of fluents. This tiling process uses the code from the fluent that determines, for each tick of data, which fluents have just opened and closed. Each element in this fluent tiling series will contain the labels of zero or more open fluents. Then, we can put the episode-label series next to the fluent tiling series and see which fluents opened and closed near episode boundaries.

A particularly interesting result is that two fluents occurred nowhere near episode boundaries. They are SAEB(SWEW(move-forward,pushing)near-obstacle) and SAEB(pushing, near-obstacle). Is this an error? Shouldn't fluent boundaries correspond to episode boundaries? In general, they should, but recall from the previous section that these fluents are due to sonar errors *during* a pushing episode (i.e., episodes of type D). A related result is that these were the *only* discovered fluents that did not correlate with either the beginning or the end of episodes.

When the fluent tiling series and the episode-label series are lined up, tick-by-tick, one can count how many unique episode labels occur, and with what frequency, during each occurrence of a fluent. Space precludes a detailed description of the results, but they tend to give quantitative support for the earlier qualitative conclusions: The composite fluents in Figure 1 generally span episodes of type D and E. For example, the fifth fluent in Figure 1 spans 1417 ticks labeled D and 456 labeled E (these are not contiguous, of course, but distributed over the 48 trials in the dataset). And the sixth fluent in Figure 1 covers 583 ticks labeled D and 33 ticks labeled E. The third fluent, in which the robot loses and regains sonar contact with the object, spans 402 ticks of episode D.

Not all the higher-order composite fluents are so tightly associated with particular types of episodes. The first fluent in Figure 1, in which the robot stops and then begins to move forward, all while near an object, spans 405 ticks of episodes labeled C1, 157 ticks of episodes labeled D, 128 ticks of episodes labeled C2, and two ticks of episodes labeled B1. Although this fluent is statistically significant, it is not a good predictor of any episode type.

This story is repeated for fluents involving just two propositions from the set moving-forward, moving-backward, rotate-left, rotate-right. Each of these fluents covers a range of episode types, mostly B2, B1, C2 and C1. These fluents evidently do not correspond well with episodes of particular types.

Few fluents cover episodes of type A, during which the robot is searching for an object. And all these fluents cover both episodes of type A and also episodes of type E or of types B2 and C2.

In sum, higher-order composite fluents involving more than one temporal relationship tend to be very strongly predictive of episodes of types D and E (or the sequence D,E). Some low-order composite fluents, involving just two propositions, are also very strongly predictive of episodes (e.g., SWEW(rotate-left,move-forward) occurs almost exclusively in episodes of type B2); but other low-order composite fluents are not strongly associated with a particular episode type. Finally, it appears that the corpus of fluents learned by the algorithm contained none that strongly predict episodes of type A.

Discussion

Fluent learning is one of many approaches to elucidating structure in time series; like all these, it is well-suited to particular kinds of time series. Fluent learning works for multivariate time series in which all the variables are binary. One can easily imagine extending the algorithm to handle variables that take a small number of discrete values (the contingency tables would get bigger, so more training data would be required), but fluent learning is not appropriate to multivariate, continuous time series unless the variable values are binned. Fluent learning does not attend to the durations of fluents, only the temporal relationships between open and close events. This is an advantage in domains where the same episode can take different amounts of time, and a disadvantage in domains where duration matters. Because it is a statistical technique, fluent learning finds common patterns, not all patterns; it is easily biased to find more or fewer patterns by adjusting the threshold value of the χ^2 statistic and varying the size of the fluent short term memory. Fluent learning elucidates the *hierarchical* structure of episodes (i.e., episodes contain episodes) because fluents are themselves nested. We are not aware of any other algorithm that is unsupervised, incremental, multivariate, and elucidates the hierarchical structure of episodes.

Fluent learning is based on the simple idea that random coincidences of events are rare, so the episodic structure of a time series can be discovered by counting these coincidences. Thus, it accords with psychological literature on neonatal abilities to detect coincidences [9], and it has a strong statistical connection to causal induction algorithms [6]; though we do not claim that the algorithm discovers causal patterns. Our principal claim is that the algorithm discovers patterns (a syntactic notion) that correspond with episodes (a semantic notion) without knowledge of the latter. In discovering patterns – the "shape" of episodes – it differs from techniques that elucidate only *probabilistic* structure, such as autoregressive models [3], HMMs [2], and markov-chain methods such as MBCD [7]. Clustering by dynamics and time-warping also discover patterns [5,8], but require the user to first identify episode boundaries in time series.

Acknowledgments

This research is supported by DARPA.under contract(s) No. DARPA/USASMDCDASG60-99-C-0074. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of the DARPA or the U.S. Government.

References

1. James F. Allen. 1981. An interval based representation of temporal knowledge. In IJCAI-81, pages 221--226. IJCAI, Morgan Kaufmann, 1981.
2. Charniak, E. 1993. Statistical Language Learning. MIT Press.
3. Hamilton, J..D. 1994. Time Series Analysis. Princeton University Press.
4. McCarthy, J. 1963. Situations, actions and causal laws. Stanford Artificial Intelligence Project: Memo 2; also, http://wwwformal.stanford.edu/jmc/mcchay69/mcchay69.html
5. Oates, Tim, Matthew D. Schmill and Paul R. Cohen. 2000. A Method for Clustering the Experiences of a Mobile Robot that Accords with Human Judgements. Proceedings of the Seventeenth National Conference on Artificial Intelligence.pp. 846-851. AAAI Press/The MIT Press: Menlo Park/Cambridge.
6. Pearl, J. 2000.Causality: Models, Reasoning and Inference. Cambridge University Press.
7. Ramoni, Marco, Paola Sebastiani and Paul R. Cohen. 2000. Multivariate Clustering by Dynamics. Proceedings of the Seventeenth National Conference on Artificial Intelligence,pp. 633-638. AAAI Press/The MIT Press: Menlo Park/Cambridge.
8. David Sankoff and Joseph B. Kruskal (Eds.) Time Warps, String Edits, and Macromolecules: Theory and Practice of Sequence Comparisons. Addison-Wesley. Reading, MA. 1983
9. Spelke. E. 1987. The Development of Intermodal Perception. The Handbook of Infant Perception. Academic Press