Abstracting from Robot Sensor Data using Hidden Markov Models

Laura Firoiu, Paul Cohen
Computer Science Department, LGRC
University of Massachusetts at Amherst, Box 34610
Amherst, MA 01003-4610

February 1, 1999

Abstract

This work is the first step of a larger effort aimed at learning logical descriptions from robot sensory data. These representations are more compact than sensory traces and will support logical reasoning. We view the robot's experiences as trajectories through an unknown state space. The robot receives information about the state of the world through its sensors. We present a technique to automatically extract atomic propositions from these sensors. Our assumption is that a state means that something is invariant in the world, and that this invariance is reflected in some constant sensor values, or constant functions of sensor values. Our task is then to find the states and their invariant characterization. We employ a hidden Markov model to find the states and their distributional characterization. From the probability distributions of the sensor values in states we then create atomic propositions that describe the states. The transformation of atomic propositions into predicates should be straightforward given sensor models that specify the arguments of these predicates, but this has yet to be implemented.

Keywords: Abstraction learning, Hidden Markov Model, Minimum Message Length

Also submitted to IDA-99, the third symposium on Intelligent Data Analysis

1 Introduction

We are interested in learning without supervision elements of logical representations of episodes. The episodes in question are generated by robots interacting with their environments. Just as human infants bootstrap their sensorimotor experiences into a conceptual structure and language [3], so we want our robot to learn ontologies and language through interaction. Previous work has focused on learning *sensory prototypes*, which represent robot interactions in terms of how the interactions appear to the sensors [5, 8]. For example, driving toward a wall and bumping into it is represented as a decreasing series of sonar values followed by the bump sensor going high. While sensory prototypes support some kinds of reasoning (e.g., predicting that the bump sensor will go high) they do not contain explicit elements that represent the robot, the wall, and the act of driving; and so do not support reasoning about the roles of entities in episodes [1]. This work takes the first step from sensory prototypes to logical representations. Logical representations have two advantages:

- Because they contain terms that denote the entities in a scene and the relationships between them, logical representations such as "push(robot, object)" are compact, and easily support planning and other reasoning. The sensory prototype of pushing objects does not support these easily [9].

- Abstraction can be over predicates and properties of entities, rather than over patterns in sensory traces. For instance, the extensional category of pushable objects is the set of elements i such that the robot has experienced "push(robot,i)" in the past. Given the extensional category, one can imagine learning the intensional concept of pushable object, the properties that make objects pushable. Neither kind of categorization is feasible given only sensory prototypes.

If logical representations are so advantageous, why not build them into our robots, that is, make them part of the robots' innate endowment? The reason is that we want to explain how sensorimotor activity produces thought—classification, abstraction, planning, language—as it does in every human infant. So we start with sensors and actions, and in this paper we explain how elements of a logical representation might be learned from these sensorimotor beginnings.

The representations are "passive" in the sense that, currently, they are not used by any problem solving system. These representations do not specify what to do in a certain situation or predict what will happen if an action is taken. In the absence of supervision and a problem solving task, we need a criterion for good representations to guide the learning process. We chose the principle of minimum description length. Experiences are represented as sequences of states, where each state is characterized by atomic propositions that denote true facts about that state. Our algorithm identifies the states and the atomic propositions that minimize these descriptions.

The representation of experiences as sequences of states corresponds with our intuition that experiences unfold through several relatively static stages. At least for simple robot activities, the robot's world tends to remain in the same state over some periods of time, so we expect the state sequences to be simple. For example, the experience of moving toward an object has some well defined parts: accelerating, approaching, being near the object. Ideally our learning algorithm would learn a state description for each part.

Dividing an experience into states is the first step in the process of learning logical representations of episodes. We want to identify these states and ground them in patterns of sensor values. A technique that fits our task very well is hidden Markov model (HMM) induction. The assumption behind the HMM is that the data sequence is produced by a source that evolves in a state space and at each time step outputs a symbol according to the probability distribution of the current state.

We identify the episode stages with the states of an HMM induced from all the data collected during a batch of episodes. These states are characterized by, or grounded in, stable probability distributions over sensor values. They form a single vocabulary for all episodes, so similar stages can be identified across experiences.

The second step in the process of learning logical representations is to find atomic propositions that denote facts in the current state. Since the states found by the HMM are characterized by probability distributions, the atomic propositions must be derived from them. We define the atomic propositions simply as disjunctions of the most likely sensor values according to these distributions. For example, the characterization of "accelerate" is given by some positive values of the acceleration sensor and by the velocity sensor varying within a range of values. A representation of an episode is a sequence of states described by these atomic propositions.

Each representation has a cost that reflects both the size of all its atomic propositions and states, and how well they describe the raw data. The latter criterion has two components, precision ($a \vee b$ is less precise than a) and accuracy. We minimize this cost heuristically by applying a state splitting technique during the HMM induction process. States are split and the resulting model re-trained as long as the description cost decreases. This technique is somewhat similar with McCallum's "utile distinction memory" [4]. The main difference is that in McCallum's work, state splitting is driven by an "active" criterion, that of predicting utility, unlike our "passive" minimum description cost criterion.

2 The Sensor Vector

The robot receives data in the format of time-series of sensor values. From the forty or so sensors of the Pioneer 1 robot, we selected six that we consider relevant for describing the twelve experiences in our experiment. The experiences were all similar: the robot went towards two cups, one blue and one red, and then either hit or passed the cups. The sensors we selected are: translational velocity, contact, and for each color, visual area and visual distance. An example of an episode sensor time series can be seen in figure 1.

While we can easily see in figure 1 the different stages of the episode, the "jittery" sensor values can bias the state induction algorithm towards frequent state changes. We correct this bias with one of our own: in a stable world, sensor values remain constant or change in a regular, not jittery, way. One way to introduce this bias is to smooth the sensor time series. We go one step further, fitting lines piecewise to the time series. These lines fragment the episode into stages in which the derivatives of sensor values are constant. The fitting algorithm works as follows:

1. A graph is constructed such that:
    - there is a node for each known point on the curve (time stamp);
    - there is one arc between any two distinct nodes; the arc points to the node with higher time stamp; the weight of the arc is the mean square error of the regression line fitted to the curve fragment defined by the two nodes (time stamps);

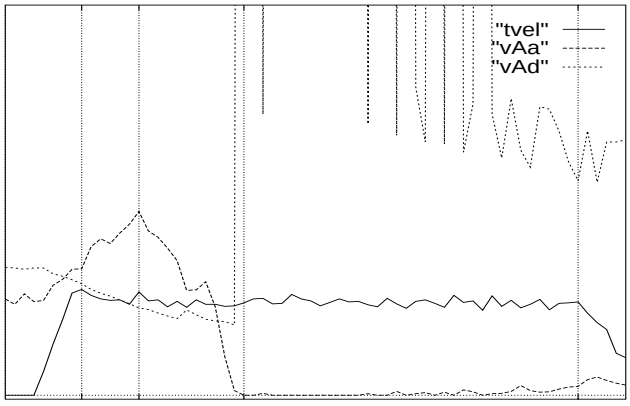

Figure 1: Sequence of sensor values for pushing an object, where the sensors are: translational velocity (tvel), blue visual area (vAa) and blue visual distance (vAd). We can identify five stages of the experience: accelerate towards blue object, approach object with approximately constant velocity, pass object, constant move with no object in view, decelerate.

2. Dijkstra's shortest path algorithm is applied to the graph thus constructed; the resulting path defines a piecewise linear fit with the property that the sum over the individual fragments of the mean square error is minimized.

An example of such a fit is shown in figure 2.

A more refined and realistic definition of state is that arbitrary functions of sensors—not just linear fits—are constant in states. We plan to learn such functions in the future.

3 Hidden Markov Models

A discrete hidden Markov model [7] is defined by a set of states and an alphabet of output symbols. Each state is characterized by two probability distributions: the transition distribution over states and the emission distribution over the output symbols. A random source described by such a model generates a sequence of output symbols as follows: at each time step the source is in one state; after emitting an output symbol according to the emission distribution of the current state, the source "jumps" to a next state

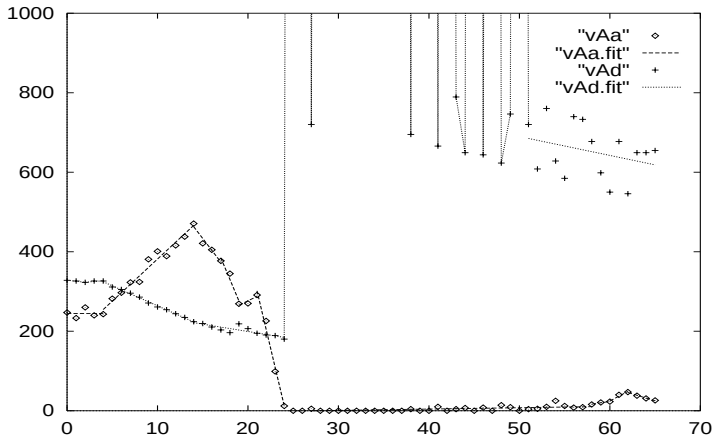

Figure 2: Piecewise fit of the sensors visual A area (vAa) and visual A distance (vAd) for one of the experiences.

according to the transition distribution of its current state. Since the activity of the source is observed indirectly, through the sequence of output symbols, and the sequence of states is not directly observable, the states are said to be "hidden". A continuous HMM emits symbols from a continuous space, according to probability densities instead of probability distributions. For either discrete or continuous HMMs, efficient dynamic programming algorithms exist that:

- induce the HMM that maximizes (locally) the probability of emitting the given sequence (the Baum-Welch algorithm)
- find the state sequence that maximizes the probability of the given sequence, when the model is known (the Viterbi algorithm).

While a continuous HMM appears more appropriate for our domain (the robot's sensors return continuous values) we chose discrete HMMs because our simple method of inducing atomic propositions works readily for probability distributions but not for probability densities. Since the sensors return continuous values, we must discretize them. Each sensor variable is discretized individually with a unidimensional Kohonen map [2]. Each continuous input value is mapped to one unit and the resulting symbols are the map units.

The HMM model definition can be readily extended to the multidimensional case, where a vector of symbols is emitted at each step, instead of a

single symbol. The assumption that allows this immediate extension is conditional independence of variables given the state. Thus we can learn state descriptions based on vectors of sensor values.

4 State characterization with atomic propositions

A single HMM is learned for a batch of robot episodes, and each episode is represented by its most likely state sequence (Viterbi path) through the HMM. By representing its experiences with an HMM, the robot adds to its knowledge about the world. Specifically, the states in the model support recognition of stages of episodes and of similar episodes.

But these states do not explicitly represent the *scene elements*—physical objects, activities, spatial relations and so on—that correspond to what the robot is doing at the times the states occur. States are just stable probability distributions over sensory inputs, thus they contain only implicit information about scene elements. To represent scene elements explicitly, the robot must create representation elements that denote them. The first step toward such representations is to create atomic propositions, which are logical representation elements denoting simple facts about the scenes experienced by the robot.

To characterize an HMM state by a set of logical propositions, we replace for each sensor the probability distributions over its values with logical descriptions of the distributions. These descriptions are disjunctions of possible sensor values. As such, propositions are simple facts of the form "sensor S takes values x or y". These propositions enable the construction of more complex representations as we describe later.

When inducing a proposition from a probability distribution over a sensor's values, we assume that if the distribution has a low entropy, then some constant process affects the sensor. In this case the atomic proposition is defined only for the most likely sensor values. While we would like to ignore as noisy the high-entropy distributions and do not create propositions for them, this is not a clear-cut decision for two reasons. First, a steady state can be characterized by a high-entropy but stable distribution. Second, our goal is to minimize the length of descriptions of experiences. If we ignore high-entropy distributions, then the shortest description of any experience might

be just an uninformative sequence of the one "noise" state. So we chose not to ignore the high-entropy distributions and defined atomic propositions for them, as well.

An example of a proposition based on the distribution of the translational velocity (t_vel) sensor is:

distribution:	.0	.0	.0	.0	.0	0.14	0.33	0.44	0.08	0.01
atomic proposition:						$t\_vel\_5\_6\_7\_8$				

We consider that all the values in the proposition definition are equally likely to occur in a state in which the proposition holds. Thus, the proposition is defined as a generalization of the distribution from which it was derived to the uniform distribution over the covered values. This crude generalization reduces the proliferation of propositions and allows identification of common propositions across states. In the example above, the covered values are 5 through 8. Because the velocity varies over a contiguous range of values [1], we interpret this proposition to mean that the robot either accelerates or decelerates. The interpretation is clarified when the acceleration sensor [2] is considered as well. To the robot, this proposition "means" only a pattern of its sensor values.

Given a sensor model that describes the kind of information a sensor returns, we can transform these propositions into predicates. For example, if the translational velocity sensor returns the translational velocity property of the constant *robot*, then the proposition $t\_vel\_5\_6\_7\_8$ becomes the predicate $t\_vel\_5\_6\_7\_8(robot)$. The implicit assumption in our sensor model is that during an experience, a sensor returns information about the same object. This means that the constant *object* in the predicate $vis\_A\_2\_3(object)$ that holds at one time step (state) during the experience, is the same as the constant *object* in the predicate $vis\_A\_1\_2(object)$ that holds at the next time step of the same experience, although $vis\_A\_2\_3(\cdot)$ and $vis\_A\_1\_2(\cdot)$ are different predicates.

This is the current extent of our work on transforming propositions into predicates, although it is the focus of our future work. The remainder of this section describes a criterion for inducing atomic propositions.

[1] By discretizing with a unidimensional Kohonen map, the topology of the original real-valued space is preserved.

[2] Positive values of acceleration (corresponding in this experiment to discrete values greater than 2) indicate acceleration.

4.1 HMM state splitting

A limitation of HMM induction algorithms is that the number of states must be known in advance. Often, there are either too few states and the resulting propositions are too vague (for example a sensor can take any value) or there are too many states and propositions, such that the representation of experiences becomes long and not intelligible. Since we consider good representations to be "short" representations, our algorithm splits states as required to minimize the size of these representations, as measured[3] by a cost function. We designed the cost function according to the minimum message length (MML) principle [6], as a measure of the information needed to re-generate the original data (the time series of the experienced episodes). As in the MML paradigm, there are two pieces of information the robot must store:

1. its model; in our case the model is the collection of states, atomic propositions and state[4] characterizations with atomic propositions;

2. the encoding of each episode within the model;

The cost function is a sum of two components, one for each item above. The first is the cost of the model, which is a measure of the length of the model description. The second is the data cost, which is a measure of the size of all the episode encodings. The two cost components are presented in section 4.2.

The state induction algorithm proceeds by recursively splitting states and re-training the resulting HMM until the cost cannot be improved:

1. initialization: the HMM has only one state

2. iterate while cost is decreasing:

   - for each state, compute the cost resulting from splitting the state
   - select the state that yields the largest cost reduction and split it

[3]The cost function is not the exact length of the encoded information, but a measure of it. For example we ignore string delimiters or the exact number of bits when defining the cost of encoding a number n as $log(n)$.

[4]The information contained in the state transition probabilities is not considered here because it is implicit in the encoding of an episode as a state sequence.

By choosing to split the state that yields the largest cost reduction at the current iteration, the cost is minimized heuristically. We cannot attempt to minimize cost globally, because an exhaustive search of all the splitting possibilities is exponential in the final number of states, and the HMM fitting algorithm is guaranteed to find only a local maximum, anyway.

State splitting stops because the two components of the cost functions are inversely proportional and the cost reaches a minimum.

4.2 The cost function

The model component of the cost function measures the size of the description of atomic propositions and states. An atomic proposition is described by enumerating the values it covers. We define its cost as:
$\#\, covered\_values * log(\#\, all\_sensor\_values)$.
The description of a state and its characterization is simply an enumeration of the propositions that are true in the state. Its cost is defined as:
$\#\, propositions\_in\_state * log(\#\, all\_propositions)$.
The cost of the model is the sum of the costs of propositions and states.

The data component of the cost function is the sum of the costs of the individual episodes. Recall that an episode is encoded as the most likely state sequence in the HMM induced from a batch of episodes, and further that the probability distributions over sensor values within each state are replaced by sets of logical propositions. These propositions generalize over the distributions from which they were derived and lose information that was present in the distributions. Consequently, the propositions may be inaccurate, meaning they specify incorrect sensor values, or imprecise, meaning they specify a range of sensor values. Suppose, for example, a propositional characterization of an HMM state says "translational velocity is 2, 3, or 4." If the robot's translational velocity in the state is actually 5, then the proposition is inaccurate, and translational velocity is 3, then the proposition is imprecise.

To re-generate a time series of sensor values from logical state descriptions, one would have to store additional information, either for specifying one of the covered values when the proposition is not precise, or for correcting errors when the proposition does not hold at that time step.

The cost of an individual experience is defined to include both the size of its encoding as a state sequence within the model and the additional information required for correcting the description, if necessary. Specifically,

the cost is a sum over all time steps of:

1. the length of encoding with the optimal code the current state $s(t)$, given the previous state $s(t-1)$; if we consider $s(t-1)$ the symbol emitted by the source $s(t)$ according to its transition probability distribution, then the optimal code, as defined by Shannon [10], is $-log(prob(s(t)|s(t-1)))$; this code has the property that the expected length of the encoding of one symbol equals the entropy of the source, $\sum_{s_i} -prob_i * log(p_i)$, where p_i is the transition probability $prob(s_i|s(t-1))$; a theorem of Shannon states that no code can achieve a lower expected symbol encoding length then the entropy;

2. the length of encoding the sensor values at the current time step, given the current state; for each sensor this component of the cost is one of the following:

   - the precision cost: if the proposition that characterizes the state for this sensor covers the current value, then the cost is the entropy of the proposition; for example, for $t\_vel\_5\_6\_7\_8$ the cost is $log(4)$ because the exact value must be specified out of four equally likely possibilities; this cost reflects the precision of the proposition definition: a proposition over only one value is extremely precise and has zero cost
   - the error cost: if the current value is not covered, for example the value is 9 and the proposition is $t\_vel\_5\_6\_7\_8$, then the cost is that of specifying the exact sensor value, namely $log(\#\ all\_sensor\_values)$

5 Experiment

In twelve similar episodes, the robot was oriented toward a group of two cups, then it received the command to go for roughly 80 time steps (approximately 8 seconds). From each episode, sequences of sensor values were collected for six sensors. The contact sensor takes value 1 when either the robot gripper bumpers or beams sense an object. The translational velocity sensor takes values between 0 and 250. The visual area and visual distance sensors for channels A and C detect blue and red objects, respectively. The visual sensors return information only about the largest object (of the corresponding color)

in the visual field. These are six of roughly forty sensors on board the Pioneer 1 robot. We selected these by hand because we do not know yet how select them algorithmically.

The result of the state splitting process for one experience is shown in figure 3. The final number of HMM states for this experiment is 6. Different experiments were run by varying the duration of HMM training or the regime. The training regime refers to the way the HMM is re-trained after a state is split: either the emission probabilities of all states are re-adjusted, or training is restricted to only the two states that resulted from the split. The results shown in figure 3 are from an "all states" training experiment. This training regime seems to give better results, because the search for a model is not constrained to the space of probability distributions of two states only, and thus a better HMM may be found.

The sequence of states and the defined propositions that describe the experience in figure 3 are:

- state 3 can be interpreted as "accelerate with object A and object C in view":

contact.0	*t_vel*.0.1.2.3.4.6.7.8.9	*t_acc*.2_4
vis_A_area.0_2	*diff_vis_A_area*.2	
vis_A_dist.3_5	*diff_vis_A_dist*.2	
vis_C_area.1	*diff_vis_C_area*.2	
vis_C_dist.4	*diff_vis_C_dist*.2	

  The "accelerate" predicate can be mapped to the two propositions "*t_vel*.0.1.2.3.4.6.7.8.9" and "*t_acc*.2_4"; the two objects are inferred from the visual sensor propositions.

- State 5 can be interpreted (albeit with a slight difference discussed below) as "steady-go with two objects in view":

contact.0	*t_vel*.5_8	*t_acc*.2
vis_A_area.1_4	*diff_vis_A_area*.2_3	
vis_A_dist.2_3	*diff_vis_A_dist*.2	
vis_C_area.0_3	*diff_vis_C_area*.2	
vis_C_dist.1_4	*diff_vis_C_dist*.2	

  The "steady-go" predicate comes from "*t_vel*.5_8" and "*t_acc*.2". There is a slight difference between what is happening during this stage of

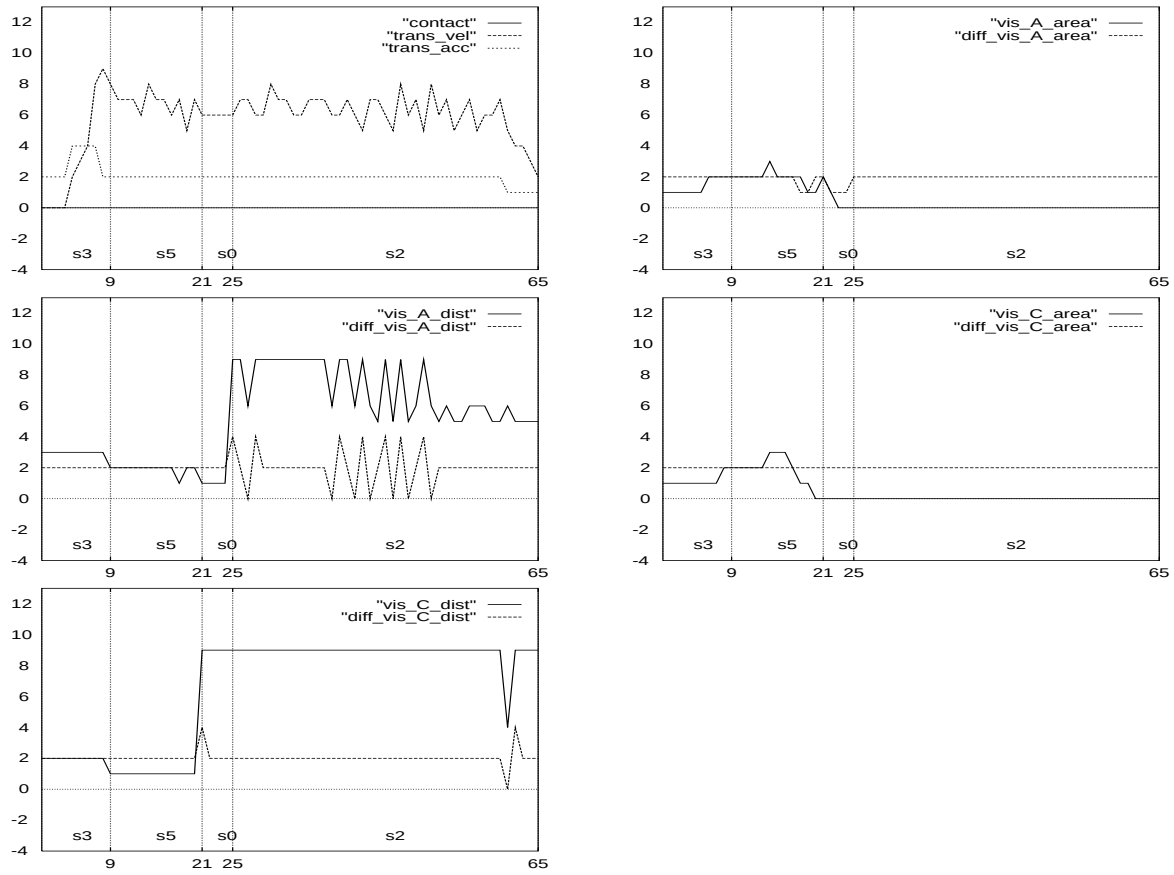

Figure 3: Partitioning of an episode into stages: a contiguous sequence of a state defines a stage. The plots show the discrete values.

the experience and the propositional definition of s_5. In the plot of *vis_C_area* in figure 3 we notice that the object C disappears, as the *vis_C_area* sensor goes from 2-3 to 0 and *vis_C_dist* goes from 1 to 9 (the highest value). While the propositions *vis_C_area*.0_3 and *vis_C_dist*.1_4 capture a change in the perception of object C, their counterparts (the slopes of their linear fits) *diff_vis_C_area*.2 and *diff_vis_C_dist*.2[5] imply no change of perception. Due to the discretization process (*diff_vis_C_dist* and *diff_vis_C_area* have constant value 2 throughout s_5), the proposition is slightly inaccurate.

[5]The discrete value 2 corresponds to a unit in the Kohonen map with value 0.

- State 0 cannot be so easily interpreted. In the context of the episode in figure 3 it means "pass object A", but the proposition description makes it a transition state (it also lasts only for 3 time steps in this episode):

contact.0_1	*t_vel*.0.5.6.7.8	*t_acc*.2
vis_A_area.0_2	*diff_vis_A_area*.0_2	
vis_A_dist.0.1.6.9	*diff_vis_A_dist*.0.2.4	
vis_C_area.0.2.3.4.5.6.7.8	*diff_vis_C_area*.1_4	
vis_C_dist.0.1.9	*diff_vis_C_dist*.2	

  The propositions *vis_A_area*.0_2 and *diff_vis_A_area*.0_2 support the "pass object A" interpretation, but the propositions *vis_C_area*.0.2.3.4.5.6.7.8 and *vis_C_dist*.0.1.9 introduce an "uncertain object C in view". In reality, object C was passed during the previous stage. While not erroneous, the proposition is imprecise.

- State 2 is easily interpreted as "go or decelerate with no objects in view":

contact.0	*t_vel*.0_8	*t_acc*.0_2
vis_A_area.0	*diff_vis_A_area*.2	
vis_A_dist.5.6.9	*diff_vis_A_dist*.0.2.4	
vis_C_area.0	*diff_vis_C_area*.2	
vis_C_dist.9	*diff_vis_C_dist*.2	

While the fragmentations of experiences seem meaningful in most cases, the state characterization by propositions does not always correspond to what is happening in during the episodes. For example, as discussed above, our interpretation of state s_0 is slightly different from its propositional characterizations. The costs of the inaccuracies and imprecisions in the definition of this state seem to be out-weighted by the reductions in the description length and it is not split.

Transforming the state sequences of the 12 experiences into regular expressions we obtain:

$s_3^+ s_5^+ s_4^+ s_0^+ s_2^+$	for two traces	$s_3^+ s_5^+ s_4^+ s_1^+ s_2^+$	for one trace
$s_3^+ s_5^+ s_4^+ s_0^+$	for one trace	$s_3^+ s_5^+ s_0^+ s_2^+$	for one trace
$s_3^+ s_5^+ s_4^+ s_1^+$	for two traces	$s_3^+ s_5^+ s_4^+ s_0^+ s_2^+ s_0^+$	for two traces
$s_3^+ s_5^+ s_4^+ s_0^+ s_1^+$	for one trace	$s_3^+ s_5^+ s_4^+ s_0^+ s_1^+ s_0^+$	for one trace
$s_3^+ s_5^+ s_4^+ s_0^+ s_1^+ s_0^+ s_1^+ s_0^+$	for one trace		

We notice that almost all sequences start with the pattern $s_3^+ s_5^+ s_4^+$. This shows that an extension of the learning process toward generalizations over state sequences will decrease the cost of the descriptions.

6 Conclusions and Future Work

During the first year of an infant's life, she apparently develops increasingly rich and efficient representations of her environment (Mandler calls this process *redescription* [3]). We have shown how to re-describe multivariate time series of sensor values as rudimentary logical descriptions, by creating new objects that are associated with parts of the world at different abstraction levels. The objects at one level are grounded in, or mapped to, objects at the previous level. Because both memory and time are finite resources, the criterion of minimum description must govern the process. In this work we tried to apply these ideas at the lowest levels of abstractions, by creating atomic propositions grounded in probability distributions over raw sensor values (physical level). The results are generally good: the fragmentation of episodes into states and their corresponding propositional characterizations appear most times to conform with our interpretation of the episode stages. But sometimes they do not conform, as in the case of state s_0 in the example in the previous section. The main reason for the mixed results is that the criterion that drove the learning process is purely descriptional: only "passive" descriptions are minimized. There is no special reinforcement signal to differentiate the states and there is no reasoning involved for explaining the differences between experiences or for predicting the evolution of an experience. If the agent also reasons based on its internal representation then the minimization principle will promote representation elements that simplify not only the "passive" descriptions but the "active" descriptions (computations) as well, that is the derivation of explanations and predictions. Our next goal is to create, starting from the obtained propositions, elements of representations (predicates and formulas) that explain the difference between episodes like "push object" and "pass object", or explicitly state that the robot will hit the object if it continues to move on its path.

References

[1] Paul R. Cohen and Mary Litch. What are contentful mental states? Dretske's theory of mental content viewed in the light of robot learning and planning algorithms. Submitted to the *Sixteenth National Conference on Artificial Intelligence*, 1999.

[2] Teuvo Kohonen. *Self-Organizing Maps*. Springer, 1995.

[3] Jean M. Mandler. How to build a baby: II. Conceptual primitives. *Psychological Review*, 99(4):587–604, 1992.

[4] R. A. McCallum. First results with utile distinction memory for reinforcement learning. Technical Report 446, Computer Science Department, University of Rochester, NY, 1992.

[5] Tim Oates, Matthew D. Schmill, and Paul R. Cohen. Identifying qualitatively different experiences: Experiments with a mobile robot. Submitted to the *Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

[6] J. J. Oliver and D. Hand. Introduction to minimum encoding inference. Technical Report 4-94, Statistics Dept., Open University, September 1994. TR 95/205 Computer Science, Monash University.

[7] Lawrence R. Rabiner. A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[8] Michael Rosenstein and Paul R. Cohen. Concepts from time series. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 739–745. AAAI Press, 1998.

[9] Matthew D. Schmill, Tim Oates, and Paul R. Cohen. Learned models for continuous planning. In *Proceedings of Uncertainty 99: The Seventh International Workshop on Artificial Intelligence and Statistics*, pages 278–282, 1999.

[10] Jan C. A. van der Lubbe. *Information Theory*. Cambridge University Press, 1997.