Learning Elements of Representations for Redescribing Robot Experiences

Laura Firoiu and Paul Cohen

Computer Science Department
University of Massachusetts at Amherst,
Amherst, MA 01003-4610
{lfiroiu, cohen}@cs.umass.edu

Abstract. This paper presents our first efforts toward learning simple logical representations from robot sensory data and thus toward a solution for the perceptual grounding problem [2]. The elements of representations learned by our method are states that correspond to stages during the robot's experiences, and atomic propositions that describe the states. The states are found by an incremental hidden Markov model induction algorithm; the atomic propositions are immediate generalizations of the probability distributions that characterize the states. The state induction algorithm is guided by the minimum description length criterion: the time series of the robot's sensor values for several experiences are redescribed in terms of states and atomic propositions and the model that yields the shortest description (of both model and time series) is selected.

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 [4], 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 [7]. 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 [8].
- 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 first step in the process of learning logical representations is to redescribe the episodes as state sequences. Our intuition is 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 stages: accelerating, approaching, being near the object. We want to identify the states that correspond to these stages and ground them in patterns of sensor values.

A technique that allows identification of states is that of 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. The states are thus characterized by stable probability distributions over the output alphabet. We identify the episode stages with the states of an HMM induced from all the data collected during a batch of episodes. Since they form a single vocabulary for all episodes, 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 becomes a sequence of states described by these atomic propositions.

These 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 choose the principle of minimum description length to guide the learning process. This principle is implemented with the help of a cost function that measures both the size of the representations (atomic propositions, states, episodes as state sequences) and how well they describe the raw data. Our algorithm identifies the states and the atomic propositions that heuristically minimize the cost of these descriptions.

2 Identifying Experience Stages with Hidden Markov Models

2.1 Hidden Markov Models

A discrete hidden Markov model [6] 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 according to the transition distribution of its current state. The activity of the source is observed indirectly, through the sequence of output symbols. 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).

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 simplifying assumption that allows this immediate extension is conditional independence of variables given the state.

2.2 Input Preprocessing

We collected time series of sensor values from a Pioneer 1 robot. The robot has about forty sensors, and almost all of them return continuous values. While a continuous HMM appears more appropriate for this domain we chose discrete HMMs because our simple method of inducing atomic propositions works readily for probability distributions but not for probability densities. The sensor variables are discretized independently with unidimensional Kohonen maps [3]. Each continuous input value is mapped to one unit and the resulting symbols are the map units.

Not all of the robot's sensors are relevant to our experiments. Besides slowing down considerably the HMM induction algorithm, the irrelevant sensors introduce noise that leads the algorithm into creating meaningless states. We selected the sensors that we considered important and discarded the others from the sensor vector.

The sensor values are “jittery” and can bias the state induction algorithm toward 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. To introduce this bias, we create new variables by calculating the slopes (derivatives) of selected sensor variables and adding them to the sensor vector. The slopes are calculated by fitting lines piecewise to the time series. The algorithm has two steps:

1. Initialization: create a graph 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);
2. Find the shortest path in the above graph between the nodes corresponding to the first and last time step (Dijkstra’s alg.).

The path calculated at step 2 defines a piecewise linear fit with the property that the sum over the individual fragments of the mean square error is minimized.

2.3 State Splitting Algorithm for HMM Induction

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[1] by a cost function. We designed the cost function according to the minimum message length (MML) principle [5], 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, the robot must store two pieces of information. The first is its model, that is the collection of atomic propositions and states. The second is the encoding of each episode’s time series by taking advantage of the model.

The cost function is a sum of two components: the model cost and the data cost. The cost of the model is a measure of the length of the model description. The data cost is a measure of the size of all the episode encodings. The two cost components are presented in section 2.5.

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

[1] 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 $log(n)$.

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

State splitting stops because for the data cost to decrease, the model cost must increase , so the total cost cannot decrease indefinitely. 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 the 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.

2.4 State Characterization with Atomic Propositions

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 the most likely sensor values, that is the values that have a probability higher than a certain threshold. An example of a proposition based on the distribution of the translational velocity (trans-vel) sensor is:

distribution:	0 0 0 0 0 0.14 0.33 0.54 0
atomic proposition:	$trans\_vel\_5\_6\_7$

In the example above, the proposition definition covers values 5 through 7. 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.

Propositions are thus simple facts of the form "sensor S takes values x or y". Given a sensor model that describes the kind of information a sensor returns, we can transform these propositions into predicates. For example, if the sensor model specifies that the translational velocity sensor returns the translational velocity property of the constant *robot*, then the proposition $trans - vel\_5\_6\_7$ becomes the predicate $trans - vel\_5\_6\_7(robot)$. We can assume that for a simple experience, a sensor returns information about the same object throughout the experience. Transforming propositions into predicates and then composing them into more complex representation is the focus of our future work.

2.5 The Model and Data Encoding Costs

The model is a set of atomic propositions and states. We encode it by concatenating the descriptions of states and atomic propositions. An atomic proposition is described by enumerating the values it covers and its cost is:
$\#\ covered\_values * log(\#\ all\_sensor\_values)$.

The description of a state s_i has two parts. The first part specifies the codes for next states, according to its transition probability distribution. These codes are used in the encoding of time series [2] as follows: if the current state is s_i and the next state is s_j, then the optimal [3] code for s_j, given s_i, has $-log(prob(s_j|s_i))$ bits. The cost for all next state codes is $-\sum_{j=1}^{\#states} log(prob(s_j|s_i))$. If a transition probability $prob(s_k|s_i)$ is 0, then we replace the k-th term in the sum with $log(\#states)$.
The second part of a state encoding is its characterization with atomic propositions. This cost is defined as: $\#\ propositions\_in\_state * log(\#\ all\_propositions)$. The model cost is the sum of the costs of the descriptions of propositions and states.

The time series of experiences are encoded as the most likely state sequences in the induced HMM. A state specifies the set of atomic propositions that hold in the state and these propositions carry information about the sensor values. The 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. For example, if a propositional characterization of an HMM state says "translational velocity is 2, 3, or 4." and the robot's translational velocity in the state is actually 5, then the proposition is inaccurate. If 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 $s(t-1)$; as discussed above, this cost is either $-log(prob(s(t)|s(t-1)))$ or $log(\#\ states)$.
2. the length of encoding the sensor vector at the current time step, given the current state; for each sensor this component of the cost is either $log(\#\ covered\_values)$ if the proposition is imprecise, or $log(\#\ all\_sensor\_values)$ if the proposition is inaccurate

[2] Although these codes appear in the redescription of experiences, they must be specified in the model description because otherwise the encoded experiences cannot be decoded.

[3] We do not have to specify what this optimal code. For our cost function, we need to know only its length.

3 Experiment

3.1 Experiment Setting

Sensor value time series were collected from twelve simple experiences of the Pioneer 1 robot. The experiences fall into four categories: pass object on right, pass object on left, push object and approach object. There are three experiences of each kind. For all experiences, the object is perceived by the visual channel A, which was calibrated to detect blue objects. The perceived object will be referred from now on as "object A". The data were collected [4] in a less noisy environment, with the robot executing forward motions along an almost empty corridor. The noise reduction proved to be beneficial: no spurious objects – that usually mislead the state splitting algorithm – were detected in the visual field.

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. These are:

- "trans-vel" is the robot's translational velocity
- "vis-A-area" is the area occupied by the object in the channel A visual field
- "vis-A-x" and "vis-A-y" are the coordinates in the visual field of object A
- "grip-front-beam" and "grip-rear-beam" return 1 when an object is between the two gripper arms and 0 otherwise

The slopes (derivatives) of the first four sensors were also added, yielding four more variables: "trans-acc" is the derivative of translational velocity and "diff-vis-A-xxx" are the derivatives of the visual sensors.

The sensor values were discretized with Kohonen maps, one unidimensional map for each sensor variable. Figure 1 shows the resulting discretization for the visual A area sensor. As it can be seen in this figure, the map is topologically ordered, that is $value(map\ unit\ 0) < value(map\ unit\ 1) < \ldots < value(map\ unit\ 8)$. Topological ordering is a property of unidimensional Kohonen maps, so the maps of all sensors are ordered. Due to this property, we can easily interpret the atomic propositions. For example, the atomic proposition "vis-A-area.0-1" tells us that a small object is seen in visual channel A, while "vis-A-area.7-8" signals the presence of a large object.

3.2 Results

The results of the state splitting algorithm for two of the twelve experiences, and the corresponding partitioning into stages are shown in fig. 2. For the states that occur during these two experiences, table 2 lists their probability distributions over sensor values and the induced atomic propositions. The most likely HMM state sequences for all experiences are shown in table 1.

[4] We thank Zack Rubinstein for providing the data and for collecting them in a less noisy environment.

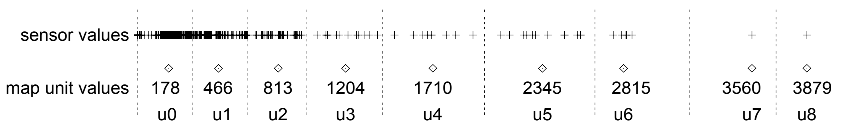

Fig. 1. Discretization of the "vis-A-area" sensor with a linear Kohonen map. The map has 9 units, u0 through u8, thus yielding 9 discrete symbols. The plot shows the values of the map units and the approximative intervals of sensor values allocated to each unit. It can be noticed that most of the sensor values are mapped to the first three units, while the last two units get only one value each.

Table 1. The middle column shows the most likely HMM state sequences for the twelve experiences and the right column shows their corresponding compressed stage sequences. In the compressed stage sequences, c_i stands for a composite stage and s_i stands for a simple stage.

experience	state sequence	compressed state sequence
pass-right-A	$s_1^+\ s_5^+\ s_4^+\ s_8^+\ s_0^+$	$c_{10}\ c_{13}$
pass-right-A	$s_1^+\ s_5^+\ s_4^+\ s_8^+\ s_0^+$	$c_{10}\ c_{13}$
pass-right-A	$s_2^+\ s_1^+\ s_5^+\ s_4^+\ s_8^+\ s_0^+$	$s_2\ c_{10}\ c_{13}$
pass-left-A	$s_2^+\ s_1^+\ s_5^+\ s_4^+\ s_8^+\ s_0^+$	$s_2\ c_{10}\ c_{13}$
pass-left-A	$s_2^+\ s_1^+\ s_5^+\ s_4^+\ s_8^+$	$s_2\ c_{10}\ s_8$
pass-left-A	$s_2^+\ s_1^+\ s_5^+\ s_8^+\ s_0^+$	$s_2\ c_9\ c_{13}$
push-A	$s_3^+\ s_1^+\ s_5^+\ s_4^+\ s_6^+\ s_7^+$	$c_{12}\ s_7$
push-A	$s_3^+\ s_1^+\ s_5^+\ s_4^+\ s_6^+\ s_7^+$	$c_{12}\ s_7$
push-A	$s_3^+\ s_1^+\ s_5^+\ s_4^+\ s_6^+\ s_7^+$	$c_{12}\ s_7$
approach-A	$s_3^+\ s_1^+\ s_5^+\ s_4^+\ s_6^+$	c_{12}
approach-A	$s_3^+\ s_1^+\ s_5^+\ s_4^+\ s_6^+$	c_{12}
approach-A	$s_3^+\ s_1^+\ s_5^+\ s_4^+\ s_6^+$	c_{12}

We can see from figure 2 and from table 1 that we can indeed identify a contiguous run of one HMM state, s_i^+, with an experience stage – call it s_i. Furthermore, some pairs of stages, for example $\langle s_1\ s_5\rangle$ appear quite frequently. Such frequent pairs can be merged into composite stages. By replacing subsequences of simple stages with composite stages, even more simplified redescriptions of experiences are obtained. In order to explore this possibility, we implemented a simple compression algorithm that creates composite stages, guided again by the minimum description length principle. The description that must be minimized has two parts: the description of composite stages in terms of simple stages and the redescription of each individual experience with both composite and simple stages. The cost of each part is a measure of its description length. Creating a new composite stage has the effect that the cost of the first part increases, while that of the second part decreases. Therefore, the total cost, which is the sum of

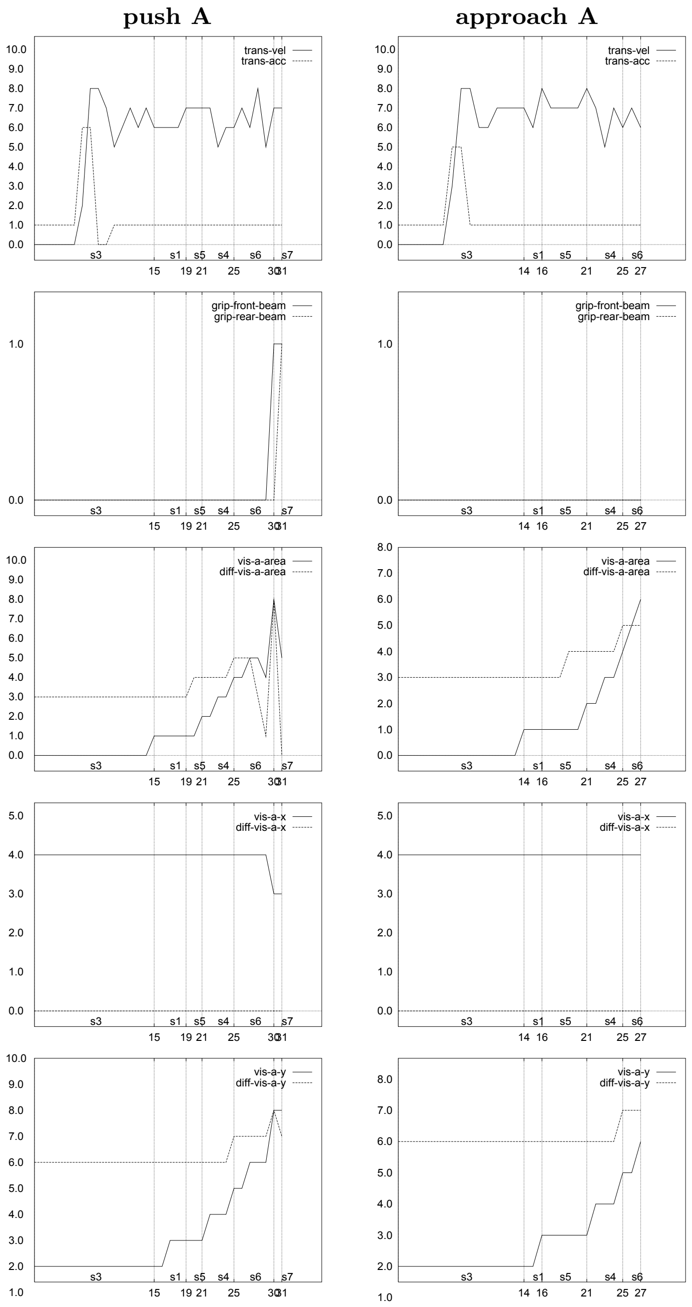

Fig. 2. HMM state fragmentation for two experiences: the left column contains the plots from a "push A" move and the right column from an "approach A" move. The units on the x-axis are time steps and the units on the y-axis are discretized sensor values.

Table 2. The states and atomic propositions that occur in the two experiences in figure 2. The states are listed in their order of appearance: $\langle s_3\ s_1\ s_5\ s_4\ s_6 \rangle$ for "approach A", and $\langle s_3\ s_1\ s_5\ s_4\ s_6\ s_7 \rangle$ for "push A". An atomic proposition like vis-A-x.3.5.6 means that the "vis-A-x" sensor mostly takes values from the set $\{3, 5, 6\}$, while "vis-A-area.5-8" means that the "vis-A-x" sensor takes values in the range $5 \ldots 8$.

state	atomic proposition	probability distribution	interpretation
s_3	trans-vel.0-8	.38 .01 .01 .03 .01 .01 .18 .28 .09	either accelerated or
	trans-acc.0.1.5.6	.06 .81 .00 .00 .00 .06 .06 .00 .00	constant move
	grip-front-beam.0	1.0 .00	no object within
	grip-rear-beam.0	1.0 .00	gripper arms
	vis-A-area.0	1.0 .00 .00 .00 .00 .00 .00 .00 .00	very small object
	vis-A-x.4	.00 .00 .00 .00 1.0 .00 .00 .00 .00	in the lower central
	vis-A-y.2	.00 .00 1.0 .00 .00 .00 .00 .00 .00	region of the visual field
s_1	trans-vel.0.1.6.7	.24 .05 .00 .00 .00 .00 .46 .24 .00	mostly constant move
	trans-acc.1-2	.00 .90 .10 .00 .00 .00 .00 .00 .00	at high speed
	grip-front-beam.0	1.0 .00	no object within
	grip-rear-beam.0	1.0 .00	gripper arms
	vis-A-area.0-1	.10 .90 .00 .00 .00 .00 .00 .00 .00	small object
	vis-A-x.2-6	.00 .00 .39 .02 .34 .17 .07 .00 .00	in the lower central
	vis-A-y.2-3	.00 .00 .24 .76 .00 .00 .00 .00 .00	region of the visual field
s_5	trans-vel.5-8	.00 .00 .00 .00 .00 .05 .50 .35 .10	constant move
	trans-acc.1.8	.00 1.0 .00 .00 .00 .00 .00 .00 .00	at high speed
	grip-front-beam.0	1.0 .00	no object within
	grip-rear-beam.0	1.0 .00	gripper arms
	vis-A-area.1	.00 1.0 .00 .00 .00 .00 .00 .00 .00	small object
	vis-A-x.1.2.4.5.6.7	.00 .02 .30 .00 .40 .04 .20 .04 .00	somewhere in the lower
	vis-A-y.3-4	.00 .00 .00 .78 .22 .00 .00 .00 .00	region of the visual field
s_4	trans-vel.5-8	.00 .00 .00 .00 .00 .16 .09 .63 .12	constant move
	trans-acc.1	.00 1.0 .00 .00 .00 .00 .00 .00 .00	at high speed
	grip-front-beam.0	1.0 .00	no object within
	grip-rear-beam.0	1.0 .00	gripper arms
	vis-A-area.2-3	.00 .00 .70 .30 .00 .00 .00 .00 .00	small object
	vis-A-x.1.4.6	.00 .23 .00 .00 .58 .00 .19 .00 .00	in the central
	vis-A-y.3-5	.00 .00 .00 .14 .70 .16 .00 .00 .00	region of the visual field
s_6	trans-vel.5-8	.00 .00 .00 .00 .00 .05 .50 .35 .10	constant move
	trans-acc.1	.00 1.0 .00 .00 .00 .00 .00 .00 .00	at high speed
	grip-front-beam.0	1.0 .00	no object within
	grip-rear-beam.0	1.0 .00	gripper arms
	vis-A-area.4-6	.00 .00 .00 .00 .45 .45 .10 .00 .00	medium sized object
	vis-A-x.4-5	.00 .00 .00 .00 .65 .35 .00 .00 .00	in the upper central
	vis-A-y.5-6	.00 .00 .00 .00 .00 .60 .40 .00 .00	region of the visual field
s_7	trans-vel.5-7	.00 .00 .00 .00 .00 .12 .25 .62 .00	constant move
	trans-acc.1	.00 1.0 .00 .00 .00 .00 .00 .00 .00	at high speed
	grip-front-beam.1	.00 1.0	object present within
	grip-rear-beam.0-1	.38 .62	gripper arms
	vis-A-area.5-8	.00 .00 .00 .00 .00 .38 .38 .12 .12	large object
	vis-A-x.3.5.6	.00 .00 .00 .25 .00 .13 .62 .00 .00	in the upper central
	vis-A-y.7-8	.00 .00 .00 .00 .00 .00 .00 .25 .75	region of the visual field

the costs of the two parts, may decrease as the result of creating a new composite stage. The algorithm continues to merge stages greedily, until the total cost stops decreasing. The results are shown in table 1 in the column "compressed stage sequence".

It can be noticed in table 1 that every "approach-A" experience is described by one composite stage, c_{12}, and every "push-A" experience is described by the same c_{12}, followed by the simple stage s_7. It can be seen in table 2 that state s_7, which defines stage s_7, is the only one to be characterized by the atomic propositions "grip-front-beam.1" and "grip-front-beam.0-1". These two propositions tell us that the robot is in "contact" with an object (the object is within the gripper arms). While it is obvious to us that "contact" is the difference between an "approach" and a "push", the algorithm does not get explicit information about the differences between experiences, and does not have the explicit goal of finding them. It is interesting then, that the minimum description length principle led to a re-representation of experiences that makes this distinction apparent.

It can be also noticed that the stage sequences allow a good clustering of experiences: the first two "pass-right-A" experiences share the same stage sequence and so do all the "push-A" and respectively, "approach-A", experiences.

While the above remarks are encouraging for the validity of our approach - applying the minimum description length criterion for inducing meaningful elements of representation - we can see in figure 2 that our algorithm fails to identify the acceleration stage for the two presented experiences. Although the first state in the sequence, s_3, is the only one that assigns nonzero probabilities to high acceleration values, the transition to the next state, s_1, is not triggered by the change in the acceleration regime, but by the change in the "vis-A-area" sensor from value 0 to 1. As a matter of fact, it can be noticed that for both experiences, there are other state changes triggered by this sensor as well: $s_5 \to s_4$ occurs when "vis-A-area" becomes 2 and $s_5 \to s_4$ when "vis-A-area" becomes 4. This indicates that the partitioning of these experiences into stages is mostly determined by the visual area of the object, and that the stages are identified with different degrees of closeness to the object. While this partitioning is not meaningless, it does not distinguish the important acceleration stage. The main reason is that the algorithm has no measure of the relative importance of the sensor variables, other than the reduction in description length obtained by distinguishing states based on their values. Another reason is that, as discussed in section 2.3, the cost of the description cannot be minimized globally.

4 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 *re-description* [4]). 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 simple (short) descriptions 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 fragmentation of time series into states and their corresponding propositional characterizations often appear to agree with our interpretation of the evolution of experiences. But this fragmentation is not perfect: for example, as discussed in the previous section, there is no distinct "acceleration" stage, because the algorithm has no information that the acceleration sensor is more "important" than others. Meaningful representations must not be only simple, but also useful ([1]). We consider useful the elements of representations that predict the outcome of an experience, predict when a state change occurs, explain the differences between experiences or explain reward. Our next goal is to define the utility criterion for representation elements and redesign the learning algorithm to incorporate both the utility and the minimum description length criteria.

Acknowledgments

This research is supported by DARPA/AFOSRF and DARPA under contracts No. F49620-97-1-0485 and No. N66001-96-C-8504. 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] 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. To be presented at the *Sixteenth National Conference on Artificial Intelligence*, 1999.
[2] S. Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.
[3] Teuvo Kohonen. *Self-Organizing Maps*. Springer, 1995.
[4] Jean M. Mandler. How to build a baby: II. Conceptual primitives. *Psychological Review*, 99(4):587–604, 1992.
[5] 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.
[6] Lawrence R. Rabiner. A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.
[7] 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.
[8] 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.