A New Approach to Unsupervised Learning in Deterministic Environments

RONALD L. RIVEST (rivest@theory.lcs.mit.edu)
ROBERT E. SCHAPIRE (rs@theory.lcs.mit.edu)
MIT Laboratory for Computer Science, Cambridge, Mass. 02139

Abstract

We present a new approach to the problem of inferring the structure of a deterministic finite-state environment by experimentation. The learner is presumed to have no *a priori* knowledge of the environment other than knowing how to perform the set of basic actions and knowing what elementary sensations are possible. The actions affect the state of the environment and the sensations of the learner according to deterministic rules that are to be learned. The goal of the learner is to construct a perfect model of his environment — one that enables him to predict perfectly the result of any proposed sequence of actions.

Our approach is based on the notion of a "test": a sequence of actions followed by a predicted sensation. The value (true or false) of a test at the current state can be easily determined by executing it. We define two tests to be "equivalent" if they have the same value at any global state. Our procedure uses systematic experimentation to discover the equivalence relation on tests determined by the environment, and produces a set of "canonical" tests.

The equivalence classes produced correspond in many cases to a natural decomposition of the structure of the environment; one may say that our procedure discovers the appropriate set of "hidden state variables" useful for describing the environment.

Our procedure has been implemented, and appears to be remarkably effective in practice. For example, it has successfully inferred in a few minutes each the structure of Rubik's Cube (over 10^{19} global states) and a simple "grid world" environment (over 10^{11} global states); these examples are many orders of magnitude larger than what was possible with previous techniques.

1. Introduction

1.1 Unsupervised Learning of Environments

We address the learning problem faced by a robot in an unknown environment. The robot and the environment form an interacting system (see Figure 1): the actions chosen by the robot are inputs to the environment, causing it to change state, and the state of the environment determines the sensations experienced by the robot.

A newly created robot may have little built-in knowledge of its environment. (To coin a term, we use "newbot" to denote a newly created robot with little or no experience and knowledge about its environment; the term robot is generic.) In our case we assume that a newbot knows how to perform basic actions and can experience elementary sensations, but no more.

At each point in time the environment is in some (global) state; we denote the set of all possible such states by Q. In this paper we assume that Q is finite.

We also assume that the set B of basic actions is a finite, relatively small set.

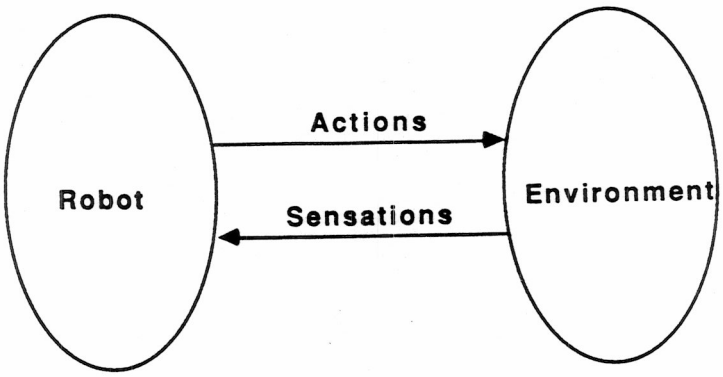

Figure 1: The Robot-Environment System

If the robot performs a basic action $a \in B$ the environment makes a transition from its current state q to a new state $q' = \delta(q, a)$; here δ is the (deterministic) transition function for the environment.

For convenience, we define the set A to be the set of all *actions*; where an action is defined to be a sequence of zero or more basic actions. We similarly extend the interpretation of δ so that $\delta(q, a)$ is the environmental state resulting from executing the action a (i.e., executing the basic actions of a in sequence).

We assume that the robot has a finite set S of *senses*. Think of a *sense* as a sense modality, such as vision, hearing, touch, etc. Each sense has a corresponding sensor which "reads" or "senses" the value of that sense. We call the values returned by the sensor "sensations". We let V denote the set of all possible sensations, and assume that different sensors can not return the same sensation. As an example, one sensation might be "a tone of 1000 hertz"; another might be "a temperature of 68 degrees". These would be returned by audio and temperature sensors, respectively. (Note that if we wish to model the perception of several tones of different frequencies simultaneously, we would call each such tone a "sense", which returned a sensation of "present" or "absent".) In what follows we will regard each sensation as a binary predicate of the current environment, although in practice (as in some of our implementation examples) the more general notion of a sense may be useful.

Abstractly, the situation a newbot faces is exemplified in Figure 2; the newbot has one button to push for each basic action $a \in B$, and one light for each sensation $v \in V$. At each moment a subset of the lights will be on, indicating which sensations obtain in the current global state.

The goal of the newbot is to learn enough about his environment to build a perfect model of it. We say that a robot has a perfect model of its environment if it can predict perfectly what sensations would result from any desired sequence of basic actions. Having a perfect model is clearly useful for planning, although we do not address planning in this paper.

We would like to point out one realistic feature of our model: there is typically no

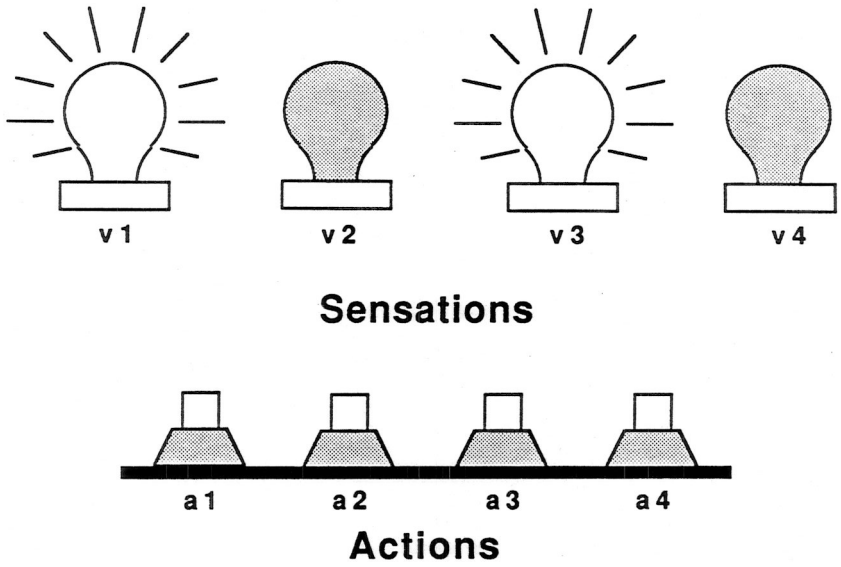

Figure 2: The Buttons and Lights Model

"reset" or "undo" buttons available to the robot — the actions of the robot have a cumulative effect on the global state.

Also, there is no teacher. The robot is engaged in unsupervised learning.

Finally, we note that while the robot may indeed converge on a perfect model of its environment, it might never be certain that its model is perfect — the fact that the model has been predicting perfectly for a while is no guarantee that it will continue to do so. Of course, this is the standard problem with induction.

1.2 An Example: The Little Prince's Planet

As a concrete example, consider a newbot just delivered to the "Little Prince" (Saint-Exupèry, 1943) on his home planet (an asteroid, really). This planet has a rose and a volcano, which the newbot can see when he is next to them; the available sense values are v = "See Volcano" and r = "See Rose". The planet is very small – it takes only four steps to go all the way around it. The basic actions available to the newbot are `f` = "Step Forward", `b` = "Step Backward", and `t` = "Turn Around". See Figure 3. In the state shown, the robot has no sensations, but he will see the volcano if he takes a step forward, and will see the rose if he takes a step backwards (or turns around and takes a step forwards). There are eight (global) states: the robot can be facing clockwise or counterclockwise, and can be in one of four positions. By symmetry of the North and South Poles, however, there are only four states in a reduced sense.

1.3 Previous work

Our learning problem can be viewed as the problem of inferring a finite automaton

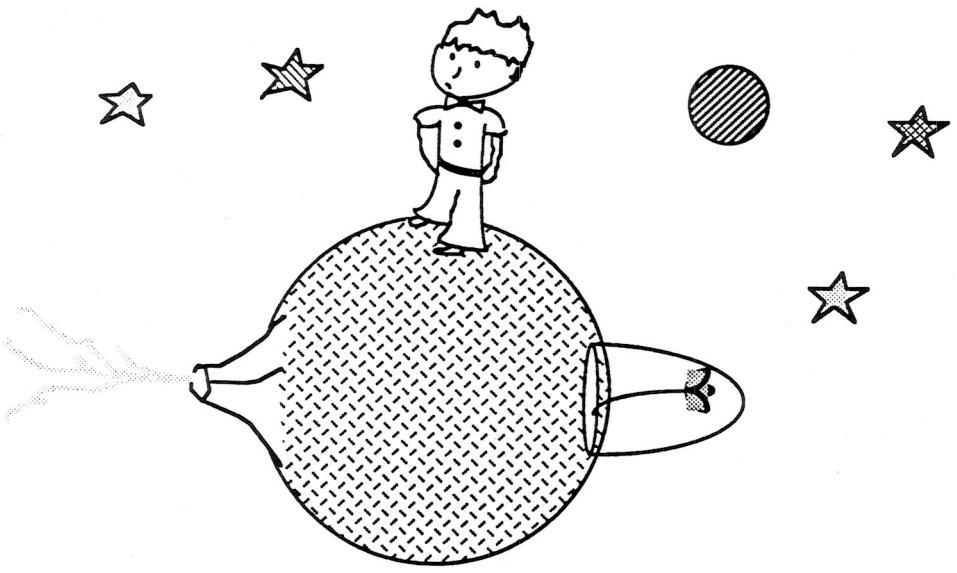

Figure 3: The Little Prince's Planet

from its input/output behavior. Angluin and Smith (1983) give an excellent overview of the field of inductive inference, including a discussion of this problem. The following previous works were particularly inspirational to us.

- Gold (1972) gives an algorithm for inferring a machine from a sample of its input/ouput behavior. Angluin (1986) elaborates this algorithm to show how to infer an automaton using active experimentation with the automaton, and a "teacher" who can provide counterexamples to incorrect conjectures.
- Angluin (1978) and Gold (1978) prove that finding an automaton of n states or less agreeing with a given sample of input/output pairs is NP-complete. (We note that the concern here is with the number of states in the inferred automaton, and that the data is *given* – the inference algorithm does not have access to the automaton).
- Angluin (1982) shows how to infer a special-class of finite-state automata, called "k-reversible" automata, from a sample of its input/output behavior.

2. Our Inference Procedures

2.1 Tests

The notion of a *test* is essential to the development of our work: a *test* is an element of $T = AV = B^*V$; that is, a (possibly empty) sequence of actions $a \in A$ followed by a (predicted) sensation $v \in V$. We say that a test $t = av$ *holds* or *is true* at a given state q if sensation v obtains in the state qa reached by executing action (sequence)

$a \in A$ starting in state q. We denote the value (**true** or **false**) of test t in state q as qt.

For example, for the state shown in Figure 3, the test `fv` is true: if the robot steps forward he will see the volcano. Similarly, the test `btffv` is true, while the tests `r`, `bbtv`, and `bv` are false.

(In some of our procedures, we generalize the notion of a "test" to include sequences of the form AS: a sequence $a \in A$ of actions followed by a sense $s \in S$. In this case the value of a test $t = as$ at a given state q (which we denote qt) is not **true** or **false**, but rather the element of V that the selected sensor s yields in the state qa. This "extended test" variation does not affect our theory very much, but does provide some practical benefits in certain cases.)

There are an infinity of tests in $T = AV$; to say that the robot has a perfect model of its environment is to say that it knows the value of each test in T — that is, it knows what sensations would result from any sequence of actions. How can this knowledge be obtained and represented?

2.2 Equivalence of Tests

We say that tests t and t' are *equivalent*, denoted $t \equiv t'$, if they yield the same result from any state: $(\forall q \in Q)(qt = qt')$.

We observe that if the number $|Q|$ of global states is finite, then the number of equivalence classes of tests is also finite. This number, which we call the *diversity* of the environment and denote by D, is at most $2^{|Q|}$, and may be as small as $\log_2(|Q|)$. (Proofs omitted; see Rivest and Schapire, 1987.) We have discovered that for many "natural" environments the diversity is actually much smaller than the number of states. Our aim is thus to design representations and procedures whose cost is polynomial in the diversity, rather than the size of the environment.

For the environment of asteriod B-612 (ref. Figure 3) there are four equivalence classes of tests:

- $v \equiv \mathtt{tv} \equiv \mathtt{ffr} \equiv \mathtt{fbv} \equiv \mathtt{ftfv} \equiv \ldots$
- $r \equiv \mathtt{tr} \equiv \mathtt{ffv} \equiv \mathtt{fbr} \equiv \mathtt{ftfr} \equiv \ldots$
- $\mathtt{fv} \equiv \mathtt{tfr} \equiv \mathtt{br} \equiv \mathtt{tbv} \equiv \ldots$
- $\mathtt{fr} \equiv \mathtt{tfv} \equiv \mathtt{bv} \equiv \mathtt{tbr} \equiv \ldots.$

We can thus select as "canonical" the tests

1. v: do you see a volcano?,
2. r: do you see a rose?,
3. `fv`: if you step forward will you see a volcano?, and
4. `fr`: if you step forward will you see a rose?.

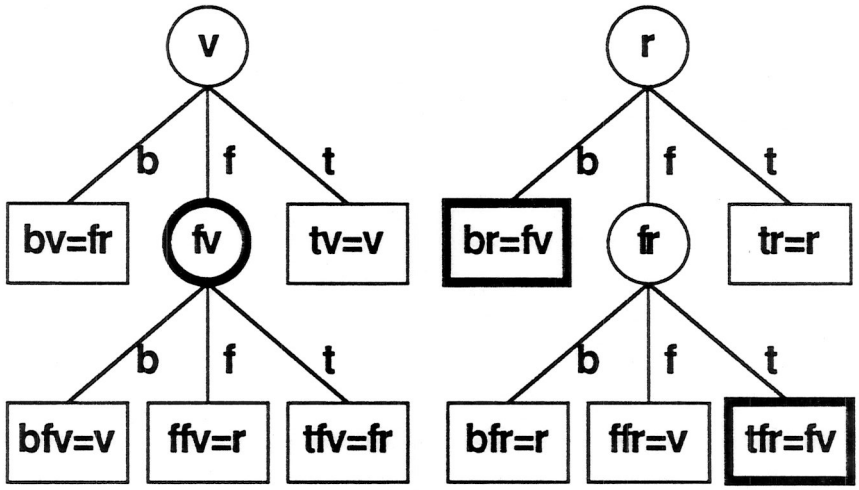

Figure 4: The Forest of Tests

Any other test will be equivalent to one of these; thus, knowing the values of these four tests is sufficient to predict the value of any other test. Knowing the values of these tests determines what the global state is, as far as possible (in this world none of the available tests allow you to tell if you are at the North or the South Pole).

Although this example is very simple, we see that the equivalence classes correspond to the natural state decomposition; it is important to know if you are at the rose or at the volcano, and (if not) whether you are facing the rose or the volcano.

A nice feature of this approach is that it allows the robot to create a model which consists of *independent* parts, which are *independently testable.* (This is reminiscent of Shapiro's (1981) work.) In our case, the independent parts are elementary statements of equivalence between two specified tests.

Let $\ll$ denote the ordering relation between tests defined as follows: we say $t \ll t'$ if t is shorter than t' or if they are of the same length but t precedes t' in the usual alphabetic ordering (using some standard ordering of the symbols in B and V).

Let $\langle t \rangle$ denote the least (in the sense of $\ll$) test in the equivalence class containing test t; we say that $\langle t \rangle$ is the "canonical" test in its class. For example, $\langle \texttt{ffr} \rangle = v$.

Using this ordering of tests and definition of canonical, it is easy to prove that if t is canonical, then so is any non-empty suffix of t. Putting it another way, if t is *non*-canonical then so is xt, for any sequence of actions $x \in A$. (Note that if $t \equiv \langle t \rangle$ then $xt \equiv x\langle t \rangle$.)

If $t = at'$ is non-canonical (where $a \in B$), but t' is canonical, we say that t is *square*; square tests are the minimal non-canonical tests. Each square test will be equivalent to some canonical test; discovering the set of canonical tests and the square-canonical equivalences is the goal of our inference procedure.

It is convenient to draw the set of canonical and square tests in the form of a forest, as illustrated in Figure 4. There are as many trees as elementary sensations

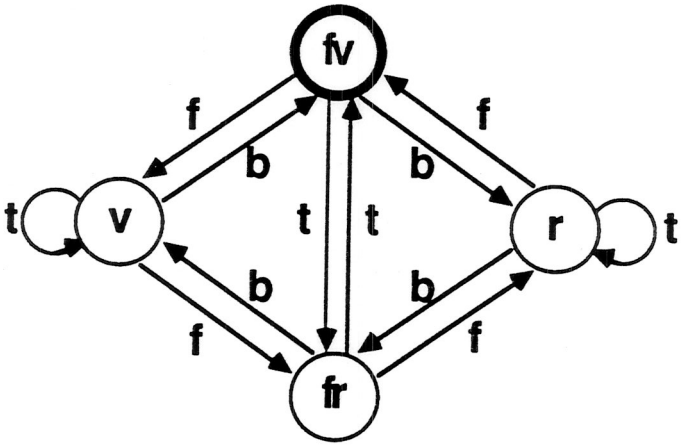

Figure 5: The Update Graph

(each sensation is the root of a tree), and the children of a test t are tests of the form at for $a \in B$. The square tests form the leaves of the tree, and arrows (omitted from the figure) may connect a leaf t with its canonical form $\langle t \rangle$.

As another means of representing the test classes, we may build a graph, as in Figure 5, in which each vertex is a canonical test, and an edge labeled $a \in B$ is directed from $\langle t \rangle$ to $\langle t' \rangle$ if $t \equiv at'$. We call this the *update graph* of the environment. The update graph may alternatively be derived from the forest of tests described above by directing each tree edge up, and merging each square test with its equivalent canonical test.

We associate with each vertex $\langle t \rangle$ its value in the current state q. When action a is executed, the value of each vertex is passed along each outgoing edge labeled a yielding the new value of each test in state qa.

Similarly, in the forest of tests, we again associate with each test its current value. Under action a, the new value of each canonical test is received from below along branch a, while each square is assigned the new value of its equivalent canonical.

In both Figures 4 and 5, the borders of the square and canonical tests which are **true** in the current state depicted in Figure 3 have been darkened.

2.3 Simulation and Prediction

The following theorem states that to have a "perfect" model (in the sense of being able to correctly predict the results of any sequence of actions), it suffices to know the canonical tests, their current values, and the relationship between the square tests and the canonical tests.

Theorem 1 *To simulate an environment it suffices to know:*

1. *The set of canonical tests.*

2. *The value qt of each canonical test t at the current state q.*
3. *For each square test s in S, the test $\langle s \rangle$.*

Proof: Suppose a transition is made from state q to state qa, for some $a \in B$. We need to compute $(qa)t = q(at)$ for each canonical test t. However, the test at is either canonical or square, so that in either case the canonical test $u = \langle at \rangle$ is known. By assumption, we know qu; this is the desired value of $(qa)t$. ■

2.4 A Simple Inference Procedure

The simplest version of our inference procedure can be described as follows, using a subroutine (not described here) for determining if two tests are equivalent:

1. Begin with W (the set of tests to be examined) equal to V (the set of elementary sensations), and C (the set of tests known to be canonical) and SQ (the set of tests known to be square) equal to the empty set.
2. If W is empty, stop. Otherwise remove from W the least test t (under the ordering $\ll$).
3. If t is equivalent to any test $u \in C$, place t in SQ, record that $\langle t \rangle = u$, and return to step 2.
4. Otherwise, add t to C, for each $a \in B$ place the test at in W, and return to step 2.

When this program terminates, it has gathered all of the information needed for the simulation theorem; it has a "perfect" model.

We claim that the number of equivalence tests performed by this algorithm in step 3 is polynomial in the diversity D of the environment. More specifically, it is at most $D \cdot (D + D \cdot |B|)$. (Note that C can contain at most D tests, and no more than D canonical and $D \cdot |B|$ square tests can be added and extracted from W; as each test is removed from W, it may be compared in step 3 to all the tests in C.)

2.5 Determining Equivalence of Tests

Now for the final step: how can we tell if two tests are equivalent? The key difficulty here is that we have no "undo" operator, so that we can't check whether $qt = qt'$ by executing test t, backing up to state q, and then executing test t'. We need to know whether qt is **true** or **false** *without actually executing it*, so we can execute t'.

To do this, we repeatedly execute the test t until the results become periodic and predictable. Then we can execute t' from a state q where we know qt without executing test t. (It is not too difficult to prove using the update graph that the period of the results of test t is at most the diversity D of the environment.)

If we discover that $qt \not\equiv qt'$, we conclude that $t \not\equiv t'$; otherwise we repeat the procedure until we are confident that $t \equiv t'$. The end result is a conclusion that either $t \equiv t'$ or $t \not\equiv t'$. (The repetitions make use of randomization to increase our confidence — for example, a random walk may be taken between each execution of the procedure, hopefully to bring the environment into a random starting state from which the procedure will have a chance of distinguishing t and t', if inequivalent. Additionally, a random sequence x may be chosen, and the tests xt and xt' compared for equivalence in the manner described above; with a lucky choice of x, it may be relatively easy to show $xt \not\equiv xt'$, which implies $t \not\equiv t'$.)

(A fine point: in order to be sure that the algorithm has discovered the correct period for t, it must either be given an upper bound on the diversity D of the environment, or it must repeatedly try out increasingly larger upper bounds, say $2, 4, 8, \ldots$.)

2.6 Extensions and Variations

By exploiting a few simple observations, the efficiency of this basic algorithm can be improved by two orders of magnitude.

The first observation is that the values of some pairs of tests can be compared directly without any need of repeating one test until its values become periodic. We call such pairs *compatible* and say that two tests $t = av$ and $t' = a'v'$ are compatible if a is a prefix of a' or a' is a prefix of a. In the Little Prince example, the test `fv` is compatible with `fbr` because we can in a single sequence of actions and senses determine the values of both. In particular, we can move forward, test if we are at the volcano (this gives us the value of `fv`), and then move back and test if we are seeing the rose (this gives the value of `fbr`).

Some experiments comparing two incompatible tests can be transformed into ones in which the tests are compatible, using knowledge already attained. Suppose the Little Prince has learned that `ffffr` $\equiv r$ and that he now wishes to compare `fr` to `tv`. Their equivalence would imply that `ffftv` $\equiv$ `ffffr` $\equiv r$. Thus, if the easy experiment of the compatibles r and `ffftv` turns out negative, the Little Prince may conclude that `fr` $\not\equiv$ `tv`.

For the special class of *permutation environments*, the equivalence of the derived compatible pair of tests is not only necessary but sufficient to the equivalence of the original incompatible pair. Conceptually, these are environments in which no information is ever lost: every action can be reversed by some other fixed sequence of actions. The Little Prince's planet is an example of a permutation environment, as is the Rubik's Cube example described below. Thus, it is never necessary to test for periodicity for this class of environments.

We have been able to prove that our algorithm is correct with high probability for this special case, and have determined the number of experiments and the sort of randomization needed to conclude the equivalence of two tests with arbitrarily strong confidence. More precisely, we have proved that, given an error tolerance parameter

$\epsilon > 0$, and a permutation environment with diversity D and basic actions B, our algorithm correctly infers a perfect model of the environment with probability at least $1 - \epsilon$ in time polynomial in $D, |B|$ and $\log(\frac{1}{\epsilon})$ (Rivest and Schapire, 1987).

In many instances, whole sets of tests can be compared against each other in a single experiment. Suppose as above that the Little Prince has learned that `fr` has period four and that the test `bv` does as well. He knows that the tests `r`, `fr`, `ffr` and `fffr` are distinct, as are `v`, `bv`, `bbv` and `bbbv`. To compare each test in the first set against each in the other would require sixteen experiments. To compare one such pair $\texttt{f}^n\texttt{r}$ and $\texttt{b}^m\texttt{v}$, the Little Prince could derive and test the compatible pair `r` and $\texttt{f}^{4-n}\texttt{b}^m\texttt{v}$ as described above. The important observation is that all of these pairs can be tested with a single experiment, namely the sequence of actions and senses `rfrfrfrfvbvbvbv`. The reader can verify that all sixteen experiments are contained in this one, a tremendous savings for the newbot. (This technique generalizes to tests involving action sequences of length greater than one.)

Unlike the original algorithm, for these variations the conclusions drawn from an experiment depend on knowledge derived from other tests. Thus, our assertion that two tests can be compared in an experiment independent of all others has been compromised, in return for enhanced efficiency. The price we pay is a greater vulnerablility to error: one mistake can snowball into an avalanche of wrong conclusions.

3. Experimental Results

We present in this section three of the other toy environments we have used to test our algorithm, and how our implementations performed in each instance.

In the Grid World (Figure 6), the newbot finds himself on a 5×5 grid with wrap-around, each tile of which is colored either red, green or blue. The newbot can see only the color of the tile he is facing, so he has one sensation for each of the colors. His basic actions allow him to paint the square he is facing any of the three colors, to turn left or right, or to step ahead one tile. Stepping ahead causes the color of the tile he moves to to replace that of the one he previously occupied. Because there is no way of recovering the color overwritten by one of the painting actions, the Grid World is not a permutation environment.

In the second micro-world, the newbot can fiddle with the controls of a car radio (see Figure 7) and can detect what kind of music is being played. There are three distinctive stations which define V: rock, classical, and news. The newbot can use the auto-tune to dial the next station to the left or right (with wrap-around), or can select one of the two programmed stations, or can set one of these two program buttons to the current station.

The last environment is based on "Rubik's Cube" (see Figure 8). The newbot is allowed to see only three of the fifty-four tiles: a corner tile, an edge tile and a center tile, all on the front face. Each of these three senses can indicate any one of six colors.

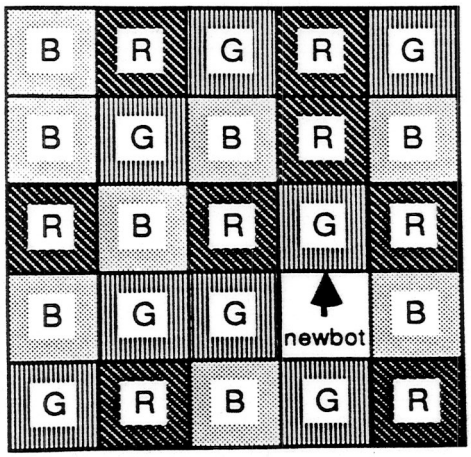

Figure 6: The 5×5 Grid World

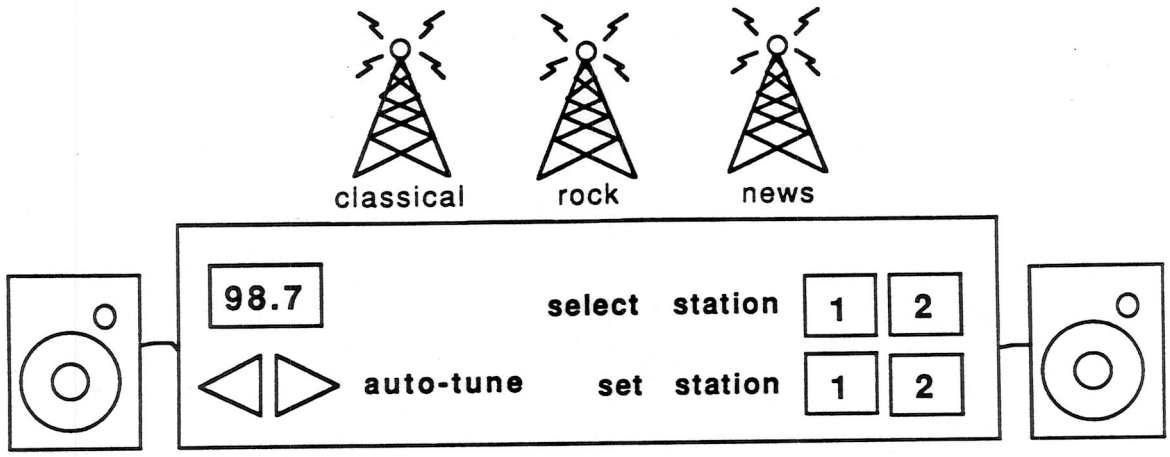

Figure 7: The Car Radio World

The newbot may rotate the front face, and may turn the whole cube about the x and y axes. (By reorienting the cube he can thus turn and view any of the six faces.)

Table 1 summarizes how our procedures handled these environments. The most complicated environment (Rubik's Cube) took approximately two minutes of CPU time to master — we consider this very encouraging.

Rubik's Cube and the Little Prince worlds were explored with an implementation (version "P") which exploits the special properties of permutation environments, but which only compares one pair of tests at a time. All worlds were explored as well by version "M", which tries to compare many tests against many other tests in a single experiment. The run times given are in seconds. The last three columns give the number of basic actions taken by the robot, the number of sense values asked for, and the number of experiments performed. (An experiment is defined loosely as a sequence of actions and senses from which the robot deduces a conclusion about

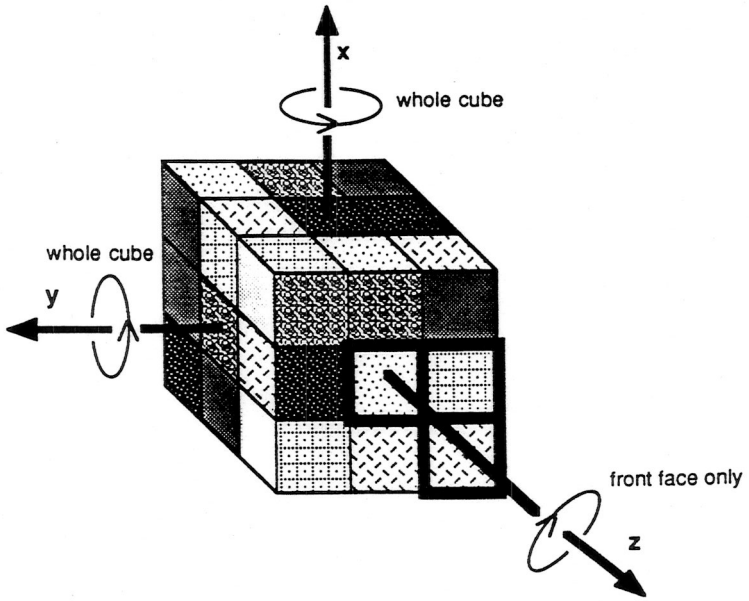

Figure 8: The Rubik's Cube World

Environment	Diversity	Global States	$\|B\|$	$\|V\|$	Version	Time	Moves	Senses	Experiments
Little Prince	4	4	3	2	P	0.1	303	102	51
					M	0.2	900	622	50
Car Radio	9	27	6	3	M	3.7	27,695	9,557	1,146
Grid World	27	$\approx 10^{11}$	6	3	M	90.4	583,195	123,371	9,403
Rubik's Cube	54	$\approx 10^{19}$	3	18	P	126.3	58,311	4,592	2,296
					M	401.3	188,405	79,008	2,874

Table 1: Experimental Results

equivalence between tests. Information about several tests may be obtained in a single experiment, and the same sequence of actions and senses may be repeated several times, each repetition counting as one experiment.) Except for the Little Prince world, we take a test to be of the form AS as described in Section 2.1. These implementations were done in C on a DEC MicroVax II workstation.

4. Conclusions and Open Problems

We have presented a new technique for inferring a deterministic finite-state environment by experimentation. This technique can be highly effective when the environment is very regular.

One outstanding and rather unique characteristic of our technique is its ability to

create "hidden state variables" in order to describe the environment. Each equivalence class of tests forms such a "hidden state variable", and our method in many cases will identify just the right set of such variables for the most economical description of the environment.

However, the work presented here is merely a first step, and it has many significant limitations. The following ones are probably most significant; further research is needed to see how best to overcome these difficulties.

- *Anisotropy:* Consider a 5×6 Grid World instead of the 5×5 Grid world. The hidden state variable indicating which axis the robot is oriented along is *not* an equivalence class of tests, and this world is not well handled with our technique.
- *Actions with probabilistic effects:* Consider a "spin" action in the Grid World, which leaves the robot facing in a random direction.
- *Actions with conditional effects:* Consider a Grid World with boundaries, so that the "step ahead" action has no effect *if* the robot is facing and up against the boundary.
- *Dependence on global state variables:* An "on-off" switch in the Car Radio World is not handled very well. (This is essentially the same as the "anisotropy" problem.)
- *Difficult to reach states:* If a long complicated sequence of actions is required to get the environment into a state needed to learn something, then our method of "random walks" is unlikely to work efficiently, and the careful planning of experiments is required. (We have some work in progress in this regard.)

Acknowledgements

This paper prepared with support from NSF grant DCR-8607494, ARO Grant DAAL03-86-K-0171, and a grant from the Siemens Corporation.

References

Angluin, D. (1978). On the complexity of minimum inference of regular sets. *Information and Control,* 39, 337-350.

Angluin, D. (1982). Inference of reversible languages. *Journal of the ACM,* 29(3), 741-765.

Angluin, D. (1986). *Learning regular sets from queries and counter-examples* (Technical Report YALEU/DCS/TR-464). New Haven, CT: Yale University, Department of Computer Science.

Angluin, D., & Smith, C. H. (1983). Inductive inference: theory and methods. *Computing Surveys,* 15(3), 237-269.

Gold, E. M. (1972). System identification via state characterization. *Automatica,* 8, 621-636.

Gold, E. M. (1978). Complexity of automaton identification from given data. *Information and Control,* 37, 302-320.

Rivest, R. L., and R. E. Schapire (1987). Diversity-based inference of finite automata. *Proceedings 28-th IEEE Foundations of Computer Science Conference.* To appear.

de Saint-Exupèry, A. (1943). *The Little Prince.* New York, NY: Harcourt, Brace & World.

Shapiro, E. Y. (1981). *Inductive inference of theories from facts* (Research Report 192). New Haven, CT: Yale University, Department of Computer Science.