Observation and Generalisation in a Simulated Robot World

CLAUDE SAMMUT (claude@cheops.oz)
DAVID HUME (daveh@cheops.oz)

Department of Computer Science, University of New South Wales
P.O. Box 1, Kensington NSW, Australia 2033

Abstract

This paper describes a program which observes the behaviour of actors in a simulated world and uses these observations as guides to conducting experiments. An experiment is a sequence of actions carried out by an actor in order to support or weaken the case for a generalisation of a concept. A generalisation is attempted when the program observes a state of the world which is similar to a some previous state. A partial matching algorithm is used to find substitutions which enable the two states to be unified. The generalisation of the two states is their unifier.

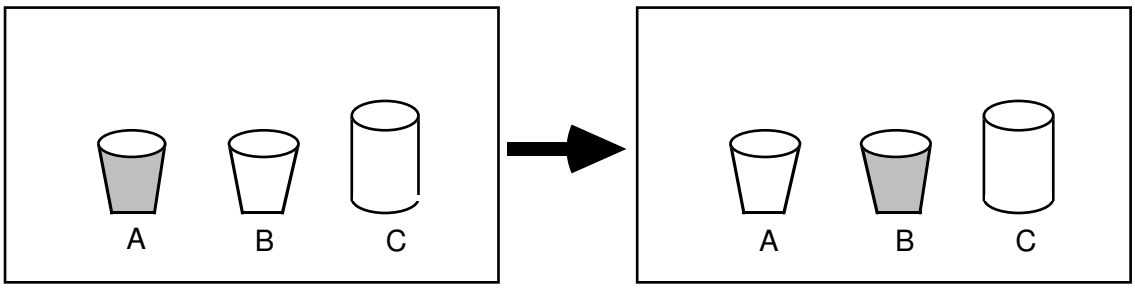
1. Introduction

Consider a simulated world that contains two robots. One is under the control of a learning program which has little knowledge of the world. This is referred to as the *child*. The second robot already "knows" about the world and can perform a variety of tasks. This is referred to as the *parent*. The child learns about the world by observing the parent performing some task and then using the observation to guide it in exploring its environment. For example, children often learn by trying to imitate the actions of adults. That is, when a situation arises which is similar to one where the parent has previously performed some action, the child may attempt the same action. One problem with an imitation is that it can never be a perfect copy of the parent's actions since exactly the same state of the world very rarely repeats itself. In practice, imitation requires a generalisation. In order for two states to be considered similar, their differences must be considered irrelevant under some generalisation. We will see that this can lead to some interesting learning behaviour.

There are several advantages in using a rich, simulated world as a domain for learning. If a learning system is given the goal of discovering the concepts which represent legal behaviour in a world, it encourages the idea that learning is incremental. That is, concepts accumulate in memory over a period time. Sometimes concepts must be revised in the light of new experience. Often learning one concept helps to learn others. The designer is also required to make the program as autonomous as possible, even to the extent of choosing its own tasks and objectives. A major difficulty with this kind of learning is getting the system to restrain itself to a search for reasonably useful concepts. There are many ways of limiting the search (Carbonell & Hood, 1986; Langley, Kibler & Granger, 1986; Rappaport, 1986). In the work described here guidance is provided by observing the behaviour of other actors in the world.

2. The CAP System

CAP is a program which learns by observation and experimentation. To demonstrate how it works, suppose the world consists of a solid cylinder and two cups, one with some liquid in it, the other empty. A "parent" robot's task is to pick up the full cup and pour the liquid contents into the empty one. At the completion of this task, the liquid is no longer in the original container, so it is not possible to exactly duplicate the same set of actions. If a child robot wishes to imitate the parent then it must be satisfied with a partial match of the starting conditions with some later state of the world. By partial match we mean that two states can be considered *similar* if some simple transformation can be applied to one state to turn it into the other. Figure 1 shows the before and after states of the contents of cup, *A*, being poured into cup *B*. Suppose the child wishes to imitate the action immediately after the parent has finished. *A* no longer contains the liquid, however, by comparing the descriptions of the original state and the final state we see that by substituting *B* for *A* we can use *B* as the source of the liquid. Similarly, substituting *A* for *B* allows *A* to be the destination.



cup(A)
cup(B)
cylinder(C)
contains-liquid(A)

cup(A)
cup(B)
cylinder(C)
contains-liquid(B)

Figure 1: Finding a partial match between states.

Imitation based on a partial match is a useful way of learning. In this case, because *A* and *B* can be used interchangeably, the child will have learned the generalisation that liquid can be poured into objects which are cups. Imitation can be viewed as an experiment whose purpose is to confirm or deny a generalisation. For example, after swapping *A* and *B* it can be predicted that the result will be that after pouring the liquid it remains in *A*. If the prediction is proved to be true then the generalisation is confirmed. Let us now see how another experiment will fail to produce a predicted result but still yield useful information.

The partial match described above is obvious since one cup simply maps onto the other. However, there are more subtle similarities present in the scene. Assume that the concept:

$$\text{circular-cross-section}(X) \leftarrow \text{cup}(X).$$
$$\text{circular-cross-section}(X) \leftarrow \text{cylinder}(X).$$

is known to the system. That is, an object has a circular cross-section if it is a cup or a

cylinder. This tells us that *A*, *B* and *C* are all similar according to at least one criterion. Therefore, another possible substitution would allow *C* to be the destination of the pouring action. The previous experiment tested the effects of pouring liquid into another cup, thus permitting the generalisation that any cup can contain liquid. Another experiment, this time with the cylinder, tests the generalisation that objects other than cups can also contain liquids. Of course, this time the liquid does not stay in the cylinder. Thus the generalisation is shown to be incorrect.

A complete description of CAP's operation will be given later, but before we can do that we must define a few terms.

3. Partial Matching, Experiments and Predictions

The child observes and records changes in the world as a sequence of states, where each state is represented by a description (in first order logic) of the configuration of objects in the world. In the implementation, a more compact representation is used, but the principle is the same. Suppose there is a sequence,

$$S_0 ; S_1 ; ... ; S_N$$

and a current state, *S*. Although each S_i is a conjunction of atomic predicates, it is also useful to think of it as a set of predicates. Thus, a partial match can exists if there is some state, S_m: $0 \leq m \leq N$ such that

$$S \cap \sigma S_m \neq \varnothing$$

That is, under some substitution σ states *S* and S_m share common terms in the state description. For example, if S_m consists of a full cup and an empty one and *S* consists of a full cup and a cylinder then a partial match exists with a substitution of the cylinder for the empty cup. However, in order for this substitution to work, it must have been recognised that cylinders and cups can be equated in a some way. So before looking for a match, the system must first elaborate on the state description by using concepts, such as circular-cross-section. This is done by treating the concept description as a set of forward chaining rules as described in Sammut and Banerji (1986).

The partial match permits CAP to propose the following task: Since cups and cylinders are similar in at least one respect (they both have circular cross-sections) they may also be similar in their ability to contain liquids. Therefore, it should be possible to perform actions which will result in a liquid being poured into a cylinder, just as had been done with the cup (for which the cylinder has been substituted). This is *prediction*, namely, that it should be possible to create a sequence of states in the world which corresponds to the sequence obtained through the matching process. The attempt to achieve the sequence in the modelled world is called an *experiment.*

Until now, we have described the world in terms of predicates describing static configurations. Of course, these configurations only change after the performance of some action. So why are actions not part of the description of the world? One reason is that we wish to have different actors achieve similar effects. Thus it is the result, not the actions which are important. Also, were we to pose CAP's task as: have the child carry out actions similar to those of the parent, it would be difficult for the child to "put itself in the parents place", transforming all the movements, *etc*. Instead, it is easier to record snapshots of the

world and then infer the necessary actions required to transform one state into the next. Note that this does not require any involved planning procedure since each state only represents a small change from the previous one.

A plan P consists of a sequence of actions $A_0 ; ... ; A_{N-1}$ inferred from a sequence of states $S_0 ; ... ; S_N$. Action A_i is performed after state S_i with the expectation that state S_{i+1} will result. If the desired state is not produced then the experiment failed to produce the predicted result. The results of the experiment determine whether a generalisation is recorded as a new concept or not.

4. Generalisations and Learning

If an experiment has been concluded successfully, that is, the results match the prediction, then the child has grounds to propose a generalisation. When the attempt to pour a liquid into another cup succeeds then it may be proposed that liquids can be poured into any cup. The pouring action, A_i, transforms a state S_i into another state S_{i+1}, written as,

$$A_i : S_i \rightarrow S_{i+1}$$

Thus, S_i contains the preconditions for the action A_i. By generalising S_i the applicability of the action is broadened. The next experiment, trying to pour the liquid into the cylinder, generalises S_i even further. This generalisation is incorrect, however, it can be recorded as an exception condition for action A_i. As other exceptions are encountered for A_i they may be combined with previously recorded cases. Thus, it is possible to build up knowledge about when an action can be used and when it cannot.

Analysing why a particular substitution worked or did not work can lead to further experiments. If a particular generalisation was successful, then it is worth looking for other objects which are covered by that generalisation. If an attempt is made to broaden the generalisation by substituting another object, and this fails, the difference between the object causing the failure and the previously successful objects helps to define the generalisation more precisely. This refinement of the description can be achieved by trying to make the unsuccessful generalisation more specific and performing another experiment.

An interesting side effect of this learning problem is that it provides a simple criterion for clustering objects into new, unnamed concepts. Objects form a cluster if they can be used in the same roles. Containers made of glass or plastic can both hold water, a brick and a table can both be used to support other objects. As an object is added to a cluster, a generalisation may be performed in order to arrive at a concise description of the cluster.

5. CAP and Magrathea

The CAP system works within the the simulated world called *Magrathea* (Hume, 1985). It is worth noting some of the compromises required in order work with reasonable efficiency in this domain.

Magrathea produces a large number of predicates for each state of the world. In order to deal with this overload of information some heuristics are used to eliminate unwanted data. The first is that only predicates which change from one state to another are recorded. Obviously, the complete state information can be derived from the successive changes. In

addition to compactness, concentrating on change provides a focus for the partial matching procedure. Once the knowledge base of the system becomes very large, there will be a large number of irrelevant partial matches. However, by looking only concepts which include those predicates which have changed, the amount of searching is greatly reduced without much loss of generality.

Throughout a sequence, those predicates involving movements of the robot are discarded. These are relational predicates giving the positions of various parts of the robot relative to other objects in the world. Whole states may be discarded because they only represent small changes in the robot's position. This information can be discarded for the reason given earlier, that is, the result of an action is important to CAP, not the action itself.

For an indication of the compression of information this process produces some results involving a from a particular "stacking" sequence are presented. The description consisted of three properties: shape, size, and colour; and four relations: X, Y, Z positions, and touching; and five objects: two blocks, a sphere, an arm, and the ground. Seven states were involved in stacking a red block on a blue block. Each of these averaged 40 predicates each. The total number of predicates was 280. All of these would have been involved in the partial matching process however, keeping only the changes resulted in a sequence with an average state size of 6 predicates with a total of 42 predicates. Removal of robot actions resulted in a sequence with an average state size of 3 predicates and 4 states became redundant: total 9 predicates. This simple compression on an original sequence involving 280 predicates produced a sequence involving only 9 predicates thus facilitating partial matching.

Even after removing irrelevant predicates, there is still enough information to allow many partial matches between the current state of the world and stored knowledge. The present system uses a simple rule to rank matches. The match which results in the largest intersection is said to be the *nearest* match. This is not entirely satisfactory since some properties of objects will be more important than others in different circumstances. For example, colour is not relevant to the conditions required for containing liquids, but if there is a match involving many similar colours, it would be considered nearer than one in which the shape is the same but the colours are different. The only solution to this problem is to allow the system to learn to weight predicates according to the current context. This is yet to be implemented.

6. CAP's Learning Strategy

We are now, finally, in a position to give a summary of CAP's learning strategy:

1. The parent carries out a plan which results in a sequence of world states being created.
2. The sequence is compressed and stored.
3. CAP attempts a partial match between the current state and a stored state. The changes which resulted from the parent's actions are used as a focus for the search for a partial match.

   Note that matching need not be restricted to the most recently recorded actions of the parent. A sequence which was the result of previous learning may also be recorded for later matching. If this is the case then a sequence can be successively generalised through a number of experiments (i.e attempts at imitation).

4. Since a stored state belongs to a sequence, the nearest partial match is used to generalise the sequence starting from the matched state. The sequence thus generated attempts to predict the result of the experiment to follow.
5. A plan of action is inferred from the prediction sequence.
6. This plan is executed.
7. If the result of the experiment was as predicated then the preconditions for the actions in the plan are generalised otherwise, the exceptions conditions are generalised.
8. As long as there is nothing else to do (i.e. the parent is not doing anything which should be observed) the increasingly distant matches are used to create generalisations and experiments.

7. Conclusion

The work described here attempts to provide a solution to the problem of guiding and constraining a learning system which does not have a trainer. The solution requires the learner to observe the world, passively at first, and later use the observations as seeds for experiments. It should be noted that the parent robot is not a trainer. Any changes in the world are observed by the child. The parent is merely an agent of change.

An extension of this work, not considered here, involves partially matching sequences with other sequences, not merely states. This will allow the system greater flexibility in generalising plans.

References

Carbonell, J.G., & Hood, G. (1986). The world modeler's project: Objectives and Simulator Architecture. In T.M. Mitchell, J.G. Carbonell & R.S. Michalski (Eds.), *Machine Learning: A Guide to Current Research* (pp 29-34). Boston, MA: Kluwer.

Hume, D. (1985). *Magrathea: A 3-D robot world simulation*. Honour's thesis, Department of Computer Science, University of New South Wales, Sydney, Australia.

Langley, P., Kibler, D. & Granger, R. (1986). Components of Learning in a reactive environment. In T.M. Mitchell, J.G. Carbonell & R.S. Michalski (Eds.), *Machine Learning: A Guide to Current Research* (pp 167-172). Boston, MA: Kluwer.

Rappaport, A. (1986). Components of Learning in a reactive environment. In T.M. Mitchell, J.G. Carbonell & R.S. Michalski (Eds.), *Machine Learning: A Guide to Current Research* (pp 261-268). Boston, MA: Kluwer.

Sammut, C.A. & Banerji, R.B (1986). Learning Concepts by Asking Questions. In R.S. Michalski, T.M Mitchell & J.G. Carbonell (Eds.), *Machine learning: An artificial intelligence approach (Vol. 2)*. Los Altos, CA: Morgan Kaufmann.