SOME RECENT RESULTS ON NON-DETERMINISTIC MODELLING OF BEHAVIOUR SEQUENCES

Ian H. Witten

Man-Machine Systems Laboratory,
Department of Computer Science,
University of Calgary, Canada T2N 1N4

ABSTRACT

This paper surveys several non-deterministic modelling techniques and places them in a uniform framework. Three basically different methods are discussed: enumeration and evaluation of possible models; reduction of a large model by coalescing states; and limited-context methods which analyse and process all strings of a given length which occur in the behaviour.

Three results of recent work are presented. Firstly, the methodology of enumeration and evaluation is extended to the case where the model space permits recursion. Secondly, some new experiments on successive reduction of large models are described, which indicate that the quality of models produced by this technique is rather variable. Thirdly, it is shown that limited-context methods which discard information about transitions cannot identify certain kinds of inputs. However, they can take advantage of assistance from a "teacher" in the form of markers in the input string.

INTRODUCTION

It is simple to form a causal model of a deterministically-generated behaviour sequence if the model is permitted to be as large as the automaton that generated the sequence. However, if there is a limit on the number of states allowed in the model, or if the sequence is noisy or generated non-deterministically, the problem of optimal causal modelling becomes very difficult indeed. Under these circumstances the model must be non-deterministic.

Three distinct methods for non-deterministic structural modelling have been introduced in the last decade:

-- enumeration and evaluation
-- reduction and evaluation
-- limited-context.

These are reviewed in the first part of this paper. To summarize their properties, the first method guarantees to produce an optimal model of any pre-specified complexity, for a given criterion of "optimality". The price paid is extraordinarily long computation time, which grows super-exponentially with the model complexity. Reduction methods are usually based on heuristic criteria, which makes them difficult to evaluate properly. However, this paper presents an economical method of reduction which can be based upon any given optimality criterion. It does not necessarily produce optimal models, because reduction is by successive pairwise merging of states. Nevertheless, it is shown to produce models which are sometimes rather good, in regions where the optimal model is unknown because enumeration cannot reasonably be attempted. Finally, limited-context methods are extremely fast, but can only deal with (non-deterministic) input strings that are "k-testable", and this excludes behaviours which involve counting (modulo an integer) symbols that may be separated by arbitrary strings of other symbols.

The second part of the paper summarizes some recent work on non-deterministic modelling. It is obvious that if state markers are present in the input sequence, any event is trivially k-testable. What is interesting is that the process that generates these markers is itself k-testable and can therefore be learned by the modeller itself. Another area of recent work extends the enumeration and evaluation method to recursive transition networks, providing a subroutining mechanism in the model which brings out explicitly and automatically any structure which appears in the same form at different parts of the model.

ENUMERATION AND EVALUATION METHOD

This method enumerates all possible automata which could have generated a given behaviour sequence, beginning with the simplest. It has been developed into a well-defined methodology by Gaines (1976b, 1977), who defines the subset of admissible models, given a measure of poorness-of-fit of a model to the behaviour sequence and a criterion of model simplicity. The admissible subset contains only models which cannot be improved in simplicity without increasing poorness-of-fit, and cannot be improved in goodness-of-fit without increasing complexity. If a probabilistic model is desired the problem may appear to be insoluble, for the space of possible models is continuous rather than discrete; but this is not so, for an optimal assignment of probabilities to the branches of a non-deterministic model may be made by running the behaviour sequence through the model and making frequency counts (Gaines, 1976b).

Enumeration. Gaines°s ATOM system enumerates models in order of increasing complexity and evaluates their goodness-of-fit, selecting for the admissible subset the best fit at each level of complexity. Enumeration is performed in a data-directed fashion by starting at the beginning of the behaviour sequence and building a model which fits it (in a causal, though non-deterministic, sense). At each point where an arbitrary choice is made, the choice is noted on a stack, together with the position in the behaviour sequence at which the choice occurred. The next new model is constructed from the current one by backing up to the last choice, using the stack; and altering it in a systematic fashion to the next possibility. This method enumerates all observable, reduced, models which fit the behaviour in a causal (although non-deterministic) manner, and only such models. It is an incremental data-driven enumeration, and hence is probably maximally efficient.

Biermann (1972) used the same method to infer Turing machines from traces of their execution. He, however, was concerned only with deterministic machines, with an input which represented the current symbol read from the tape. For example, the trace of a sort of the three-element tape "baa" into "aab" can be written

b bR a bL b bL # -R b aR b bR a bL b bL a aR b aR b bR

5 10 15 20

where single a°s and b°s represent the current symbol read from the tape, and aL means "write an a in the current place and move the scanning head to the Left. The "#" input signifies that the end-marker has been read; nothing should be written over it so a null writing action °-° is specified.

Given that the model should be deterministic, with the single a°s, b°s, and #°s as inputs and the composite symbols as outputs, a data-driven, simplest-first search with recursive backtracking finds it very quickly. Figure 1 shows the models which are generated, expressed in Moore (state-output) form. Model (a) is the initial attempt, and by the time its inadequacy is noticed on symbol 20 (the model predicts "bR" instead of "aR" at this point), choices have been recorded on symbols 6, 12, and 18. Backing up to symbol 18 and making a new state for the "aR", and assuming an unlucky choice for symbol 20, leads to model (b), with choice points at 6, 12, and 20. This is found inadequate on symbol 22; backing up to symbol 20 and making the alternative choice leads to the correct model (c).

The very small number of models that were considered in this example stems from the known determinism of the outputs. ATOM does not make this assumption, and so although it uses the same enumeration method it has to consider a very much larger model space. Outputs and inputs are not distinguished in the enumeration (although they may be during the evaluation). Figure 2 sketches part of the model space that is enumerated, showing the single 1-state Mealy model and a few of the total of 1379 2-state ones, including the admissible one according to the entropy evaluation (Figure 2(c)).

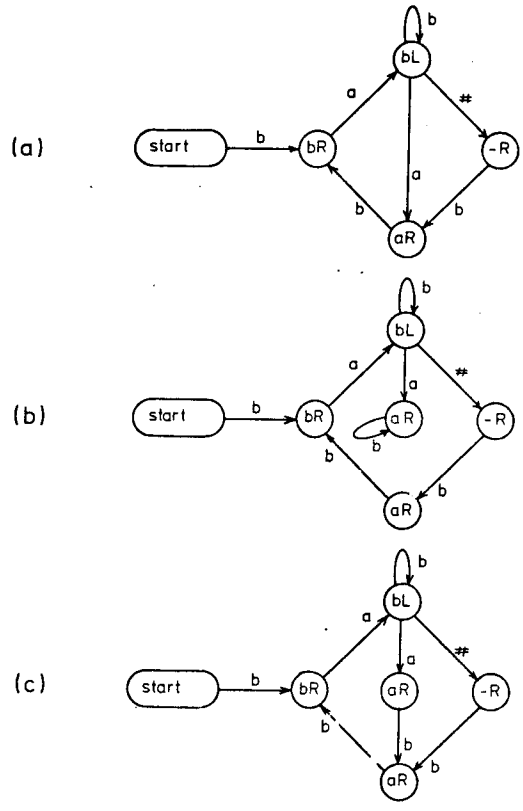

Figure 1 Deterministic Moore models generated during enumeration. Only the last one fits the whole behaviour sequence.

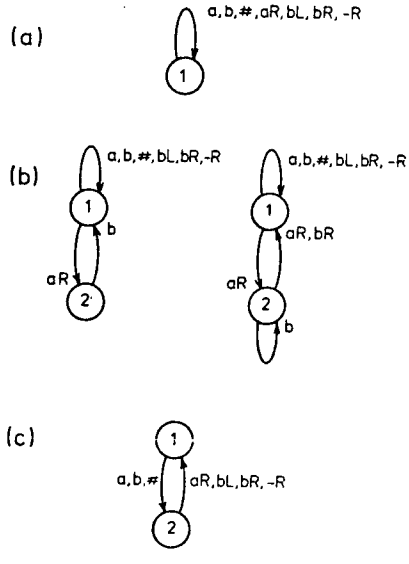

Figure 2 Non-deterministic modelling: a 1-state Mealy model and three 2-state ones.

Note incidentally that the 2-state admissible model has already separated inputs from outputs -- even though they were not distinguished for ATOM.

Evaluation. ATOM evaluates each model as it is enumerated. For Biermann, evaluation is unnecessary because the enumeration only produces a "correct" model of the sequence (if it is not a correct model of the underlying computation, the behaviour sequence needs to be augmented with more example traces). ATOM runs the sequence through the model and notes frequency counts, using these to compute the overall goodness-of-fit through an entropy (or other string-approximation) measure. Note that the frequency counts could be generated incrementally during enumeration, but then they would all have to be stacked at each choice point, giving a massive storage requirement.

The number of models examined by ATOM grows very quickly with the size of model. For example, there are n**nq Mealy models of the form $M : S \times Q \rightarrow S$, where the state set S has n elements and the input set Q has q. Many of these will be ruled out by a particular behaviour sequence because they are unreduced, unreachable, or unobservable; however, the super-exponential behaviour as n increases, for fixed q, remains.

Some advantage can be gained by searching the model space in a different order. Witten (1980) has investigated a special class of "transitive" Moore models which require substantially less computation (bounded by (n+1)**n) in most cases, without sacrificing the goodness-of-fit of the resulting model. It would be interesting to know whether other gains are possible while retaining the full power of this modelling technique.

REDUCTION AND EVALUATION METHOD

One way of avoiding the exponential search required by ATOM is to employ a method of reduction, starting with a large model and successively merging states until the desired model size is reached. Evans (1971) applied this to the inference of picture grammars from examples. Beginning with an initial grammar containing production rules for each of the example objects, he collapsed this by merging states according to some heuristic criteria, amongst which were

-- eliminate multiple occurrences of a rule
-- identify pairs of non-terminals to produce multiple occurrences of a rule
-- add productions of the form
<non-terminal> -> <string of terminals>
to give multiple occurrences of a rule.

It is difficult to assess these heuristic rules according to the quantitative evaluation criterion used by ATOM. However, a modelling procedure based on reduction can easily be defined by taking a large model of a behaviour sequence and repeatedly coalescing pairs of states. The initial model could be a direct representation of the sequence itself (that is, an n-state model of an n-element sequence) if it is short. Other methods of quickly generating large, approximate models have been developed (Witten, 1979) and are discussed below. The most straightforward way to determine which states should be merged is to evaluate separately the models which result from merging each pair of states and select that giving the best evaluation. Suppose we begin with an N-state initial model. Then the number of models which need to be evaluated to produce the (N-1)-state model is C(N,2). To create a spectrum of models with 1, 2, ..., N-1 states takes N(N**2-1)/6 evaluation operations -- a polynomial rather than exponential complexity.

Figure 3 shows the results of some experiments with this reduction method. Several approximate models for a given behaviour sequence were generated by the length-k method (see below), for different values of k. The entropy evaluation of these is plotted against their size (number of states). From each model, except the smallest with only two states, a series of models was calculated by successively merging pairs of states, chosen to give the lowest entropy at each stage. It can be seen that these new models are sometimes better and sometimes worse than the original length-k ones. For example, the line beginning at 100 states passes above the 82-state length-11 model, the 70-state length-10 model, the 64-state length-9 model, and so on. On the other hand, reduction from the length-8, -9, and -10 models all give better 30-state models than the length-7 one! To explain such counter-intuitive results, further investigation of the technique is needed.

Instead of successively merging states in a pairwise fashion, one could generate an n-state model directly from an N-state one (n<N) by identifying the best set of N-n states to coalesce. To do this, C(N,n) models would need to be considered. In fact, to generate the full set of 1-state, 2-state, ..., (N-1)-state models from an N-state one by this method would require evaluation of 2**(N-1) models -- a return to exponential complexity. However, for short behaviour sequences this could lead to an improvement on the ATOM method. If the large, starting model represented the complete behaviour sequence, admissible models would certainly be generated. If one were to examine a sequence of 100 items, the entire set of admissible models could be found by reduction if 2**100-1 models were examined. The complexity of the ATOM method grows as n**nq for n-state models with q different symbols: even for q=2 this approaches 2**100 when n is 13. Hence for non-deterministic models with 14 or more states, this optimal reduction procedure could require less search than ATOM.

LIMITED CONTEXT METHODS

Limited-context methods of forming a structural model of a behaviour sequence assume that the structure can be characterized by the set of consecutive k-tuples of symbols which occur in the sequence, for some limited "context" length k. Different techniques have been developed by Biermann and Feldman (1972), Klir and his colleagues

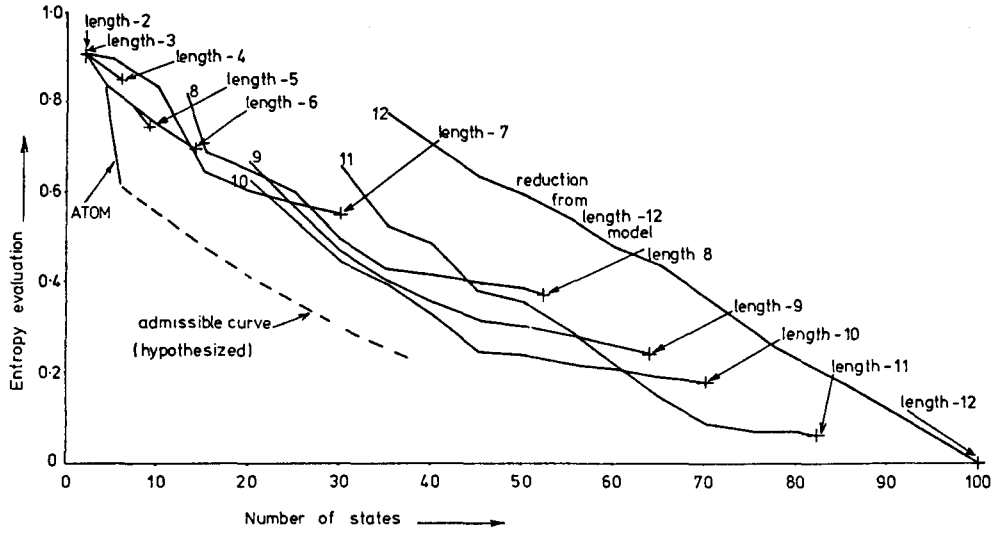

Figure 3 Successive reductions of length-k models by pairwise merging of states

(Klir 1975, 1976; Uyttenhove 1980), Andreae (1977) and Witten (1979).

Approximate modelling of a deterministic sequence. Biermann and Feldman tackled the problem of approximately modelling a deterministically-generated input-output sequence with an automaton which may have fewer states than the generator. They started with Nerode°s (1958) equivalence classes. Suppose the input-output function maps strings of inputs in the set Q to outputs in O: f : Q* -> O. Then the Nerode relation between strings p and r of Q* is

pRr iff f(pw) = f(rw) for all strings w in Q*.

The classical method of forming the equivalence classes is to begin with the relation R(0):

pR(0)r iff f(p) = f(r)

and refine it to R(1), R(2), ..., by

pR(k+1)r iff pR(k)r and pqR(k)rq for all q in Q.

It is easy to show that if the input-output function f is finite-state, there exists k with R(k+m) = R for all positive m; that is, the refinements terminate eventually.

Biermann and Feldman°s modelling method considers a finite context for the equivalence relation R, terminating the refinements prematurely when that context is reached. The resulting relation R(k), which can be defined as

pR(k)r iff f(pw) = f(rw) for all strings w in Q* of length at most k,

is used as the basis of an automaton which realizes the input-output function in a non-deterministic manner. If the function is deterministically-generated, the model becomes deterministic for sufficiently large values of k: as k decreases it becomes smaller and more non-deterministic.

While this technique is an excellent one for the purpose for which it was designed, namely providing a limited-complexity model of a deterministically-generated sequence, it fails to cope with true non-determinism in the spectacular manner described by Gaines (1976a). Faced with non-deterministic input, the models become more and more complex as k increases. Amongst other things, this means that the sequence must be stratified into inputs and outputs for the procedure to work. Consider, for example, a sample sequence generated by the regular expression (A*BA*C)*:

A A B A A A C A A B A A C A A A A B A C A A A ...

The method cannot be used for prediction of such a sequence because it is non-deterministic. However, it can be altered into an input-output sequence where all the non-determinism is accounted for by inputs. Replacing each "A" by "p A", each "B" by "q B", and each "C" by "q A" gives

p A p A q B p A p A p A q A p A p A q B p A p A q A
p A p A p A p A q B p A q A p A p A p A ....

Interpreting A, B as outputs and p, q as inputs, Biermann°s (and Nerode°s) technique will quickly arrive at the correct 2-state deterministic interpretation of the sequence which is shown in Figure 4. Note that although a typographical distinction is made throughout this paper between outputs, written in upper case, and inputs, in lower case, this information is not necessarily available to the modeller. One feature of most non-deterministic modelling methods is that they distinguish outputs from inputs automatically, using entropy considerations.

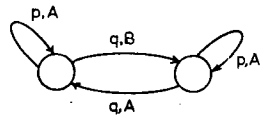

Figure 4 Deterministic automaton with inputs and outputs. If the input symbols are not available to the modeller then it must form a non-deterministic model.

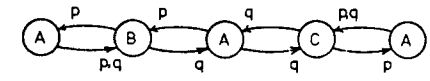

Figure 5 Deterministic automaton with inputs and outputs.

PUSS modelling. Several limited-context modelling methods effectively store information about all k-tuples which appear in the sequence, for some fixed value of k. Andreae°s (1977) PUSS modeller uses a hash table for rapid access. All k-tuples are stored in the table as they occur. When a prediction is required, the k-tuples are retrieved whose first k-1 elements match the last-seen k-1 elements of the behaviour sequence. The k°th elements of these tuples are the various predictions.

A disadvantage of this method is that large amounts of store can be occupied with non-deterministic transitions. In the extreme case, when modelling a random sequence whose elements are drawn from an alphabet of q symbols, q**k events will be stored in the hash table. Thus large areas of store are used to hold events which will be of little use for prediction anyway -- simply because they are non-deterministic.

Length-k modelling. Witten (1979) has investigated how k-tuples can be recorded economically by massaging them into the form of an automaton model. First the "ingenuous" model of k-tuples is constructed, with one state for each tuple and transitions which reflect the succession of tuples in the behaviour sequence. Only the last element of a tuple is recorded as the output of its state. The model is then subjected to a reduction process which coalesces states in a way that does not destroy information about the k-tuples which occurred in the behaviour. For example, if the random sequence mentioned above is long enough, the ingenuous model will have q**k states, but this will become a mere q states (for a Moore model) after reduction. Thus although an increased value of k normally provides a larger, more accurate, model; it does not necessarily do so if structure does not exist in the behaviour sequence -- as in our example. Note the contrast with Biermann and Feldman°s method which will generate ever larger models as k grows, if the input is random.

An important feature of this method, which we call length-k modelling, is that reduction can be done incrementally, each new element of the behaviour sequence being incorporated into an already reduced model to form an updated version of it. This gives an advantage over other methods of non-deterministic modelling which require the complete input sequence to be stored, and re-model it from scratch when new information shows the current model to be inadequate.

Identifying local context in sequences of vectors. Thus far we have taken the behaviour to be a sequence of atomic symbols. Klir (1975, 1976) has examined sequences of vectors, corresponding to several attributes of a behaviour being sampled simultaneously. This provides a richer domain, for correlations may exist between different attributes. To represent these in a model, a 2-dimensional mask is determined which, by its shape, captures the local interactions between the attributes. The sets of values which appear in the mask as it is scanned through the behaviour sequence characterize the structure of the system.

For example, output sequences of the generating automaton of Figure 5 can be represented either in linear form:

A p B p A p B q A q C p A q C p A p C q A p B p ...

which can be modelled by the length-k technique to produce a representation of the generating structure (Witten, 1979), or as a two-dimensional array:

```
A B A B A C A C A C A B
p p p q q p q p p q p p  ....
```

Given that A, B, and C represent outputs and p, q represent inputs, Nerode equivalence will produce the correct model. However, the techniques being discussed here do not need to have that information. Using an appropriate mask shape, namely

```
x x x
x x   ,
```

the mask values are

```
A B A     B A B     B A B     B A C
- -       p -       q p       q q

A C A     C A B     C A C     C A C
- -       q p       q q       p -   .
```

"-" has been used as an abbreviation to indicate either p or q -- in reality, there will be 16 instead of the above 8 different mask values. These masks by themselves have the predictive power of the generating automaton.

Uyttenhove (1980) describes a procedure, called "masksearch", which infers the optimal shape for a mask, given the maximum contextual length that need be considered. Although the example above is a deterministic one as far as the outputs are concerned, in general masksearch deals with

probabilistic matching of masks using an uncertainty measure based upon entropy.

THE WEAKNESS OF LOCAL CONTEXT

The sequence of k-tuples which occur in a behaviour fully characterizes that behaviour. However, if the sequential information is discarded and the set of k-tuples alone is stored, certain kinds of finite-state events cannot be modelled. McNaughton and Papert (1971) define k-testable events as ones which are completely characterized by the k-tuples which they can generate. They show that such events are non-counting in that the operation of counting (modulo some integer) cannot be performed if the elements to be counted can be separated by arbitrarily long sequences of other events. For example, the regular event (A*BA*BA*C)* is a counting event, for two B°s, which may be separated by arbitrary strings of A°s, need to be counted before a C can be predicted. However, (ABAABAAAC)* is non-counting, because the separating strings are of fixed length. Another example is provided by the automaton of Figure 4, which generates the counting event ((pA)*qB(pA)*qA)*, where inputs p and q dictate which non-deterministic branch is to be taken at each step.

Deterministic non-counting events can be identified by Biermann and Feldman°s technique. However, if the event is non-deterministic -- as that of Figure 4 is if the inputs are not available to the modeller -- then structural identification is an extremely difficult problem. As far as I am aware, only exhaustive searching methods like Gaines° ATOM and the optimal reduction technique (both described above) are sufficiently powerful to cope with counting events.

A good example of a non-deterministic, counting event was introduced by Andreae (1977) and discussed by Gaines (1976b; Figure 1) and Witten (1979; Figure 7). It comprises a sequence of heads and tails, so constructed that every third symbol is perfectly predictable. It is an apparently random machine whose peculiar structure is very difficult to guess from the output sequence. Observation leads one to believe that only 50% success can be achieved in predicting each outcome, but in fact 67% success is possible if one knows how. Although the predictable symbols occur at fixed intervals, the fact that the intervening ones are unpredictable makes it a counting event.

Length-k modelling finds the structure of Figure 6(a) in the sequence, provided that k is at least 4. The model is the same for higher values of k if a sufficiently long behaviour sequence is available -- just how long it needs to be is discussed by Witten (1979). Contrary to what was originally thought, the counting nature of the event, which can be expressed as the 6-state Moore model of Figure 6(b), is not completely captured by the length-4 model of Figure 6(a). The latter has considerable predictive power, for there is one state (upper left) which leads unambiguously to H and another (lower right) leading unambiguously to T.

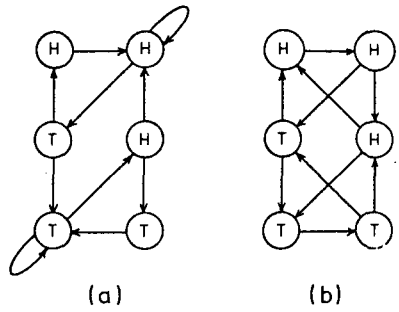

Figure 6 (a) Length-k and (b) ATOM models for a sequence which requires "counting". The ATOM model is the correct one.

However, two of the states have self-loops, corresponding to behaviours

... H H H H H ...

and

... T T T T T ...

When such sequences happen to occur in the input, every third symbol is in fact predictable. One could rewrite the first sequence

... HU HP HU HU HP ...,

where U and P indicate whether the symbol is unpredictable or predictable. Length-k modelling does not have the power to discriminate between this sequence and

... HU HU HU HU HU ...,

while the enumeration method does, because the model of Figure 6(b) has a lower evaluation (better predictive power) than that of Figure 6(a).

Using state markers. It is obvious that the identification problem becomes trivial if the sequence of states passed through by the generator is available to the modeller. For example, if a typical input-output sequence from Figure 4, like

p A p A q B p A p A p A p A q A ...,

is given to a length-k modeller, it will never form a proper structural model of the source -- no matter how great the value of k. However, if the sequence is augmented with state numbers to give

1 p A 1 p A 1 q B 2 p A 2 p A 2 p A 2 p A 2 q A 1

then a length-4 modeller will suffice. The set of length-4 predictions

```
1 p A -> 1
1 q B -> 2
2 p A -> 2
2 q A -> 1
A 1 p -> A
A 1 q -> B
A 2 p -> A
A 2 q -> A
B 2 p -> A
B 2 q -> A
p A 1 -> {p,q}
p A 2 -> {p,q}
q A 1 -> {p,q}
q B 2 -> {p,q}
```

completely characterizes the generator. This set shows immediately that the states and outputs are completely predictable, whilst the inputs are unpredictable. Hence a length-k modeller with state information could eventually interpolate the state markers into the behaviour sequence, as well as predicting the outputs. Such a modeller would accept aid from a higher intelligence to begin with, but, once started on the right track, could generate its own state markers to continue prediction.

It is simple to show formally that if state-information is provided as part of a deterministic behaviour sequence, the behaviour can be identified by length-4 modelling. Assume a Mealy generating automaton with next-state function M : S x Q -> S, and output function L : S x Q -> O, where S, Q, and O are the state, input, and output sets respectively. Then the state-augmented input-output sequence will look like

```
... s    q    L(s,q)    s°    q°    L(s°,q°) ...
                    = M(s,q)
```

where the underlined part shows that substring which depends upon the values of s and q. Since this part is four elements in length, a length-4 model is required. If the behaviour sequence contains all reachable state-transitions of the generator, the set of length-4 strings

```
s  q  L(s,q)  M(s,q)
```

which occur in the behaviour sequence completely characterizes the observable parts of L and M.

If the inputs are not available, so that in the example given earlier we are presented with strings generated by (A*BA*C)*, the state-augmented behaviour sequence will be

```
... s  L(s,q)  M(s,q)  ...
```

Then a length-3 model will suffice to capture all the information in the now non-deterministic output and next-state functions.

Symbiotic modelling: using help from a teacher. Such a symbiotic modelling method has been incorporated into the PURR-PUSS learning system developed by Andreae (1977). It achieves its power by having a simple, uniform storage structure for fixed-length pieces of behaviour sequences (which in itself is sufficient to cope with k-testable events), combined with a method of operator intervention or "teaching" which enables counting events to be learned. Of course, a complete state-characterization of the input does not need to be provided, because k-testable events are assimilated automatically: only at certain points where "counting" needs to be accomplished do state-markers have to be inserted by the teacher. Once taught an event, PURR-PUSS can itself generate the markers which it needs to continue following that event. It is as though, to make a perhaps rather dangerous anthropomorphic analogy, one gives "hints" to a person about how to accomplish some simple task, pointing out, for example, that when the widget is in such-and-such a position, then he should perform such-and-such an action. When these hints have been seen to work and thus assimilated, the task has been learned and the teacher°s role is fulfilled.

It is somewhat awkward to arrange for an identification procedure to interleave state markers which it has generated itself into the correct places in the behaviour sequence. To get around this difficulty, Andreae has introduced the idea of "multiple contexts", which involve a behaviour sequence of vectors like those used in Klir°s methodology. Each context is an input stream. State information can be provided as a stream separate from the behaviour sequence, making it easier for the identification procedure to supply the state markers itself when it has learned how to do so. Because k-testable events are learned automatically, state information needs to be provided by the teacher only occasionally, in most cases.

For the automaton of Figure 3, the two-dimensional input could be

```
p A p A q B p A p A p A p A q A
1   1   1   2   2   2   2   2   1   ....
```

The modeller would begin to "guess" the state inputs, as well as the behaviour sequence itself. Once it began to predict the states correctly, the teacher would let it take over the generation of the state markers. This can be built into a smooth interactive modelling system which at first needs to be given considerable additional information but gradually learns to provide that information for itself.

ENUMERATING RECURSIVE TRANSITION NETWORKS

Let us return now to the method of enumeration and evaluation, as implemented by ATOM. There is no reason why the model class should be restricted to finite automata. Recursive transition networks (Woods, 1970) can be enumerated in the same way. The additional power that recursion (in theory)

bestows on a computing system, namely that it permits reference to, and use of, an unbounded array of storage elements, has no relevance to causal modelling because only finite behaviour strings are of any interest. However, the use of recursion does permit more economical models to be made of certain behaviours by providing a subroutining mechanism for describing common subsequences; and more importantly, it brings out explicitly any structure which appears in the same form at different parts of the model.

An example RTN from Woods (1970) is given as Figure 7. It includes two new kinds of transition, PUSH and POP. The straightforward transition used above we call JUMP. In this RTN, transitions require the following information to be specified:

```
JUMP  current-state  symbol        new-state
PUSH  current-state  PUSH-address  new-state
POP   current-state
```

On a PUSH transition, the new state is pushed on to a stack and the current state becomes that given by the PUSH-address. A POP transition simply pops the new current state from the stack.

Formally, the next-state function can be written as three parts:

```
MJUMP : S x Q  ->  S
MPUSH : S  ->  S x S
MPOP  : S  ->  ø.
```

This increases the bound on the number of models to be considered. For each state, there are n**q possible JUMP transitions, n**2 possible PUSH ones, and one possible POP. Thus for n states, the number of models is bounded by [(n**q)(n**2+1)]**n, or around n**n(q+2). Of course, many of these models will not fit the behaviour sequence, but the space which must be searched can be expected to grow at approximately this rate. Note that the size of the search space for an RTN is the same as if two new behaviour symbols were added in an enumeration of non-recursive models.

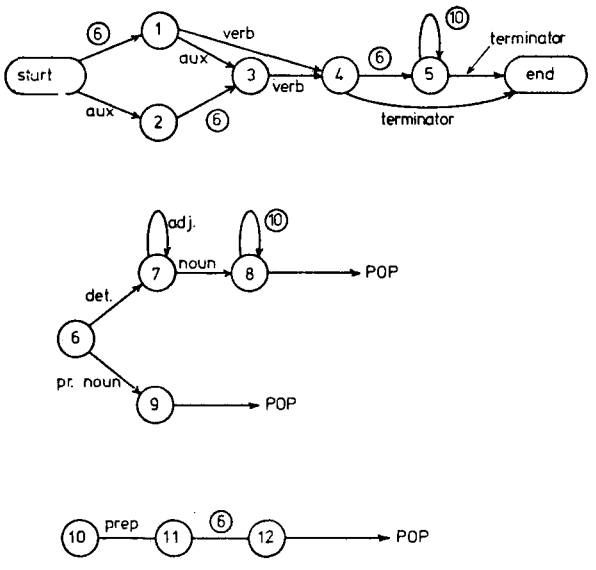

Figure 7 Example RTN, which recognizes a subset of English (after Woods, 1970).

There are additional constraints on the RTN, which will eliminate some of these extra models. Obviously, stack underflow should not occur, for if it does, the next state is undefined. It may be worth insisting that the stack is empty at the end of the behaviour sequence. However, this is overly stringent if the sequence does not represent a complete behaviour but only a part which happened to be observed, and it may be better to constrain the stack to be empty at each delimiter symbol. Then the behaviour can be bracketed by delimiters if it is known to be complete. Although these rules will eliminate some of the candidate models, each one must be generated and the behaviour sequence run through it before it can be discarded.

Extending ATOM°s candidate model space to RTN°s. At first sight, it seems extraordinary that recursive automata can be enumerated finitely. Surely the search would have to include every state of an arbitrarily large stack? In fact it does not, for the stack cannot be arbitrarily large with a finite behaviour sequence. Still, it is worth constructing an existence proof for such a search, to dispell any lingering doubts -- and indeed one has been built, based on ATOM.

First, the structure of RTN°s was modified somewhat from that of Figure 7. Note that the MPUSH and MPOP discussed above do not require an input symbol to be specified. To make the model observable without backtracking, PUSH and POP arcs must never both emanate from one state, and they should be chosen if and only if MJUMP is undefined for the particular input symbol. To make it easier to modify the existing ATOM program, inputs were associated with both the PUSH and POP arcs, so that the three-part transition function is

```
MJUMP : S x Q -> S
MPUSH : S x Q -> S
          by  (current-state,input)  ↦> push-address
MPOP  : S x Q -> ø.
```

Note that in MPUSH no new-state is specified, although a PUSH-address is. Instead, the current state is stacked. This means that some generality is lost, for a subroutine may only return to the state from which it was called. However, an alternative like

```
MPUSH : S x Q -> S x S
```

would involve a truly enormous search of n**2nq. The RTN discussed earlier, exemplified by Figure 7, is really the best choice, but is harder to fit into the data structure for model storage used by the existing ATOM program.

As emphasized earlier, ATOM enumerates incrementally all causal, observable, reduced, non-deterministic models which fit the behaviour sequence. A stack records the point at which each choice is made, and this enables the model space to be searched systematically and economically. Exactly the same procedure will search RTN°s, except that the contents of the RTN°s stack must themselves be stacked by ATOM at each choice point. Thus a stack of stacks is maintained.

Experimental results. A program for RTN-searching has been written, along these lines, and commissioned. Because it is implemented in an experimental way and not yet optimized, it runs rather slowly. For this reason, and because of the rapid growth of the search space with increasing model size, it has been difficult to generate striking examples of its power.

However, Figure 8 shows admissible recursive and non-recursive models, discovered by the program, for the short behaviour sequence

A B C A B B A B A B C A B A B B C B.

The recursive enumeration quickly discovers, at the 3-state level, that A°s and B°s can be paired to give a sequence which comprises alternate B°s and C°s, separated by zero or more such pairs:

(A B) C (A B) B (A B) (A B) C (A B) (A B) B C B.

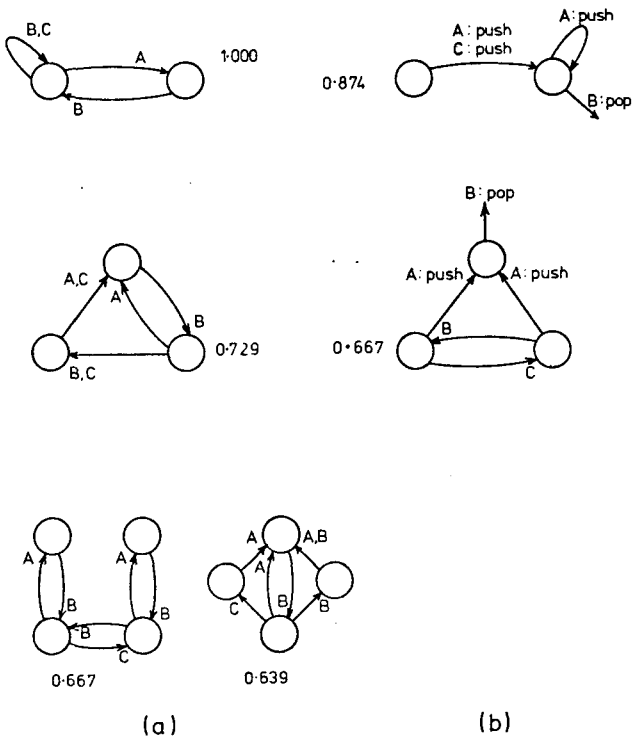

Figure 8 Non-recursive (a) and recursive (b) models, with entropy measures.

This structure is captured by a 3-state RTN (Figure 8(b), entropy 0.667), but needs a 4-state model if recursion is not permitted (Figure 8(a), entropy 0.667). However, there is another 4-state non-recursive model with slightly lower entropy (0.635). It is interesting that the 2-state RTN gives a model which is considerably better (in terms of the entropy evaluation) than a 2-state non-recursive model -- this was not anticipated when the behaviour sequence was conceived.

It is also worth noting that, in this example, the enumeration space is smaller, for a given fit to the behaviour, with recursive than with non-recursive modelling. For example, the 3-state recursive model space contains 13705 models, the best having an entropy measure of 0.667. In the 4-state non-recursive model space there are 48460 models, including one which is functionally equivalent to the 3-state recursive model with entropy 0.667. Thus the recursive technique can give better models for the same amount of search.

CONCLUSIONS

This paper has reviewed some recently-developed methods of producing non-deterministic models of a behaviour sequence. The techniques fall into three categories which, in decreasing order of time complexity, are: enumeration and evaluation, reduction and evaluation, and limited-context methods. Only the first category can provide models of a given complexity which are guaranteed to be optimal according to a global, entropy-like, optimality criterion.

New results have been obtained for each of these classes of techniques. The enumeration and evaluation method has been augmented to include recursive models. This permits more economical models to be made of certain behaviours by providing a subroutining mechanism for describing common subsequences. The amount of extra search which is required to encompass recursive models is surprisingly small: it is approximately the same as increasing by two the number of distinct input symbols. However, the work required to examine each candidate model is, on average, rather greater. For implementation reasons, a form of RTN was used which associates inputs with all transitions (including PUSH and POP ones), and whose PUSH transitions can push only the current state on to a stack (rather than pushing an arbitrary destination state). Although the program is in an early stage of development, results have been obtained which exhibit recursive models with the same predictive power as larger non-recursive ones.

It has been shown how the reduction and evaluation technique can be formalized and applied to give low-entropy models. Results indicate that successive pairwise merging of states produces models of somewhat varying quality. It is not known whether the best ones are optimal, because the enumerative search required to exhibit the optimal model is effectively impossible when the state count exceeds about 10 or 12.

Although reduction has been applied in this paper to minimal length-k models, better results could probably be obtained by starting instead with "ingenuous" length-k models. The procedure which turns length-k models into minimal ones, described by Witten (1979), effectively discards any "counting" information which is implicit in transitions between k-tuples of the behaviour. Thus reduction of minimal length-k models cannot capture such information.

With regard to the third category of model-formation techniques, we have seen how explicit provision of state information in the behaviour sequence can give locality to non-local events, as discussed by Gaines (1977); and how the "multiple-context" technique of Andreae (1977) enables this to be implemented as an interactive, symbiotic, modelling method. This permits events which are not k-testable to be captured by a modeller which deals only with local context. It shows one way in which a machine can capitalize on advice which may be fallible and even inconsistent, to reduce by a huge proportion the amount of computation that would otherwise be required to form a finite-state model by exhaustive search. Whether such techniques can be used to form recursive models is an open question.

ACKNOWLEDGEMENTS

I am indebted to John Andreae and Brian Gaines for suggesting many of the avenues of research which are explored in this paper.

REFERENCES

Andreae, J.H.
1977 Thinking with the teachable machine. London: Academic Press.

Biermann, A.W. and Feldman, J.A.
1972 "On the synthesis of finite-state machines from samples of their behaviour." IEEE Trans Computers C-21: 592-597.

Biermann, A.W.
1972 "On the inference of Turing machines from sample computations." Artificial Intelligence 3: 181-198.

Evans, T.G.
1971 "Grammatical inference techniques in pattern analysis." In J.T.Tou (ed.), Software Engineering Vol 2, pp 183-202. New York: Academic Press.

Gaines, B.R.
1976a "On the complexity of causal models." IEEE Trans Systems, Man and Cybernetics SMC-6: 56-59.

1976b "Behaviour-structure transformations under uncertainty." International J Man-Machine Studies 8: 337-365.

Gaines, B.R.
1977 "System identification, approximation and complexity." International J General Systems 3: 145-174.

Klir, G.J.
1975 "On the representation of activity arrays." International J General Systems 2: 149-168.

1976 "Identification of generative structures in empirical data." International J General Systems 3: 89-104.

McNaughton, R. and Papert, S.
1971 "Counter-free automata." Boston, Mass: MIT Press.

Nerode, A.
1958 "Linear automaton transformations." Proceedings of the American Mathematical Society 9: 541-544.

Uyttenhove, H.J.J.
1980 "On the two-dimensionality in the behavioral system identification problem, Part 1." International J Man-Machine Studies 12: 325-340.

Witten, I.H.
1979 "Approximate, non-deterministic modelling of behaviour sequences." International J General Systems 5: 1-12.

1980 "Probabilistic behaviour/structure transformations using transitive Moore models." International J General Systems 6.

Woods, W.A.
1970 "Transition network grammars for natural language analysis" Communications of the ACM 13: 591-606.

Author:

Ian H. Witten
Man-Machine Systems Laboratory
Department of Computer Science
The University of Calgary
2500 University Drive N.W.
Calgary, Canada T2N 1N4