APPENDIX 1 : THEORETICAL CONSIDERATIONS

ARTIFICIAL INTELLIGENCE

Expert systems arose from early research into artificial intelligence and their development was spurred by a combination of circumstances. Firstly from a failure to build a generally intelligent machine because of limited understanding of the nature of intelligence and of limitations of methods of representing or copying it. Secondly from a realization that much expertise stems not from intelligence in the form of general problem solving abilities but from specific factual and procedural knowledge belonging to an experts domain or field of expertise. Acknowledgement of this allowed the first problem to be avoided for a while and Expert Systems developed separately from mainstream Artificial Intelligence. Development has concentrated on techniques for extracting, representing and using expertise to the point where it is now possible to buy 'off the shelf' expert systems for specific fields.

As more effort is expended on making expert systems more realistic, it is being realized that expert knowledge must be placed in the context of 'more' human general knowledge representation and inference strategies to gain improvements, so expert systems research is being forced back towards mainstream development of artificial intelligence. The latest research advances in A.I. are in too much of a state of flux to be used in the properly managed development of an application that may be of use in fisheries management and development.

CONNECTIONISM

Connectionism is a school of thought that proposes parallel distributed processing as the correct metaphor of the mind. Machines based on this theory of design are known as neural networks. These consist of a network of interconnected processors, each of which are capable of carrying out processing functions which are relatively simple compared to those of a typical Central Processing Units. There is no controlling programme, it is inherent in the overall architecture that the processors work in parallel and that they do not usually need to be synchronized in their order of operation. These machines become useful by training them and, there are many different schemes of interconnection and training strategies, some of which can be automated.

Training proceeds by presenting a problem as a set of 'primitives' these can be represented as the input units. However the breaking down of problems into unit representation is a major difficulty. Some classes of problem submit to decomposition easily, e.g. a picture for recognition, but for others it is much more difficult, e.g. a domain such as fisheries management.

The machine is trained to give a desired output by creating patterns of excitation in the network, achieved by varying the weighting attached to each of the connections. Another factor which can be varied is the 'threshold' level of summed input at which the receiving unit will fire and output a signal. Various 'propagation rules' are used to achieve the correct balance of weighting depending on whether the unit thresholds vary deterministically or stochastically. These rules are applied in an iterative fashion until the network stabilises to an equilibrium whilst producing the desired output.

The knowledge of the system as a result of this 'training', is enshrined in the pattern of weighting imposed on the connections between each of the elements. In any future queries these weighting will ensure the correct output when the input problem is presented. This learning process is an attractive feature for proponents of neural networks, as being the true metaphor of the mind.

Unfortunately this metaphor may be too good, the architecture may also enable neural networks to forget or even make mistakes (knowledge degradation) just the same as living intelligent beings. For instance, patterns of weighting will become distorted by additional learning events and this would alter their balance, unless just as with real learning the lesson was repeated often enough for the knowledge to be reinforced. There is also the possibility that poorly learnt, but similar, concepts will merge into one. This may be a feature that causes the natural human tendency to categorize things into generic classes.

The fact that neural networks also exhibit 'graceful degradation' is also cited as further evidence in favour of the hypothesis of them being the best metaphors of the mind. Whereas, the catastrophic degradation of functionalist systems is cited as the crucial evidence to disprove this hypothesis for a neural net, if part of it is damaged or alternatively the input is not complete, the 'correctness' of the output only degenerates in proportion to the degree of damage or missing input information.

Recent psychological and neurological research has produced further evidence in support of the neural net model of the mind. (Shadbolt 1988 in Forsyth ed 1988). The slowness of brain neurones compared to electronic components and the rapidity by which a complex cognitive task is accomplished (e.g. recognition of a face) imposes a real limit on the number of processing steps that can possibly take place (Feldman 1985), for such a constraint to be satisfied, processing functions in the brain must also be organized in a parallel manner.

Furthermore, it has been revealed that, unlike earlier beliefs, the numbers of connections required for such highly parallel architectures do in fact occur in neurones in the brain, and also that the inhibitory as well as excitory effects required in connections in parallel distributed processing networks are also realized between neurone interconnections in the brain.

USES OF CONNECTIONISM IN FISHERIES MANAGEMENT

Fisheries management might well be a very suitable domain to model in neural networks providing a method of 'atomizing' the problem and solutions is found so that these can be suitably represented in input and output.

Why would it be suitable? What are the advantages over conventional management, especially if the later can be enshrined in an expert system. At least such an expert system based as it is on symbolism is <u>scrutable</u>, we can follow its reasoning, and even the very process of formalizing the management process into such a system might help us understand the domain better.

The so called understanding we gain from rationalizing reality into symbolist representation is no way a fundamental truth. Its only real use is as an aid for us to relate cause and effect so that things go the way we wish them to. If connectionism and neural nets can do this better why not use them instead?

This brings us to the critical distinction where a choice needs to be made. Either we choose to :

i) Model the mental processes of management. This would be like instituting an expert system in a neural network; or

ii) Model the fishery sector itself directly as a neural net.

In the second option we could have such things as effort, environmental factors, biological factors, economic factors etc as units of input and then have such factors as catch, profit, employment, government income etc, as units of output. We could thus train the net to form an inherent model with plenty of carefully chosen past examples, adjusting weights until the output conformed to what actually happened given the same input conditions. Once training is complete it could be let loose as a general management model to test the consequences of changing any of the input factors or alternatively divulging which input factors (and in which direction and by which amount), have to be altered to achieve the desired output. Remember it is a two way process. A manager might be perfectly happy with the state of the output factors for his or her fishery (i.e. all objectives are in equilibrium) and have no wish to alter them, but the manager may have to alter one of the input factors for totally extraneous reasons, (outside of the closed world of the model). Therefore he will want to check the effects of this on factors classified as output in the model in order to foresee any deleterious side effects of the change being forced on the input factor in question.

Option (ii) seems extremely attractive **BUT** it is important to realize that a complete conceptual shift has occurred. In effect we are creating a bioeconomic model in a neural net and **NOT** a model of human intelligence and knowledge with a specific problem to deal with. Therefore all the arguments marshalled in favour of the neural net as being the most likely and best model for the mind and its functions are no longer applicable.

Who is to say that the domain of bioeconomics behaves in the same way as the mind, it probably does not. Who is to say that the effect of one factor on another can be accurately represented as a pattern of connections with exitory or inhibitory weighting. A direct neural net model of a bioeconomic domain such as fisheries could feasibly be constructed and may even be useful, but gains nothing from the mind metaphor. It is possible that a direct neural net model of a fishery is no more real a representation, than a symbolist/levels model is of the mind. This however doesn't preclude the use of a fisheries neural net model, on the contrary it would be tempting to interpret and model many dynamic situations fisheries or otherwise in terms of a connectionist structure. In doing this though the principle of 'reality' attached to these structures in relation to the mind is abandoned and comfort of symbolism is embraced. The network itself is only being used as a sophisticated symbol, as is done with graph theory and its associated logic. Currently much work is being done on producing rational models of neural net behaviour.

EXPERT SYSTEMS

There are as many definitions in the literature for Expert Systems as there are working and available systems. One reason for the wide range of definitions is that the emphasis of different expert system applications is dependent on their task. Some operate almost entirely on a heuristic basis, others merely interpret complex results from conventional analyses.

The following would be a reasonable summary :

> **A computerized system for specialist domains that employ analysis and subjective judgement originating from human experience to achieve goals that are otherwise unobtainable because of the complexity of situations that defy conventional algorithmic solutions.**

Expert systems aim to enable less knowledgeable people to do what more knowledgeable and experienced people can do better. It is important to emphasise that they can and do work well in conjunction with conventional software (statistical analysis, model simulation, data processing packages etc.), enabling such software to be used in an effective manner and helping to ensure results are interpreted correctly. In fact there is no hard and fast division between expert and other processing systems. There is an entire spectrum from simple data manipulation systems to full blown artificial intelligence. Expert systems occupy the middle to high ends of this spectrum but also call on and use the results of other processing systems from the lower end of the spectrum.

In undertaking to build an expert system the constraints of symbolism, namely notation and functionality are accepted. These impose design demands over and above what would be required for a neural net implementation. This is because heuristics are formalised into an artificial framework and these have to be satisfied for the resultant system to run. The art of capturing as much of the reality of knowledge as possible has gradually been formalized into various methods that have an agreed order and procedure and is now respectively called 'Knowledge Engineering'.

KNOWLEDGE ENGINEERING

The first problem to be faced is how to represent the knowledge from a particular domain. Depending on the domain, the areas within a domain and the psychological make up of the source expert(s), different types of representation are more suitable than others.

The knowledge engineer then has to decide how to reason with this knowledge and what strategies he should adopt. This is of course influenced by the approach adopted to the first problem of how the knowledge is represented and wether it this reasoning is in accord with that of the expert. Other problems include the choice of software and hardware used to implement the reasoning and how the expertise is to be extracted from the expert and then tested and validated. Does it provide what the user wants? Is the system presented in a way that is practical to use; i.e. fits in with users working practises, availability of hardware etc.

Once the knowledge engineer has accepted the constraints of expert systems he must approach these tasks in as rigorous and formal a manner as possible, while calling on all of the wider methods of general Software Engineering where these are applicable.

As with all software development the principal areas to be considered are :

1. Project selection
2. Feasibility Study
3. Project planning
4. User identification
5. Knowledge elicitation
6. Knowledge Analysis
7. Choice of tools
8. Coding the actual knowledge and de-bugging the programs
9. Validation with experts
10. Maintenance procedure design
11. Evaluation with users
12. Maintenance

KNOWLEDGE REPRESENTATION

Knowledge can be classified into production rules, predicate logic or structured objects (semantic nets or frame based systems). Production rules are said to be a shallow representation of knowledge whilst the other formalisms can represent deeper 'causal knowledge'. These methods are not mutually exclusive. This is often the best way to represent a complex domain because rules alone can be inefficient or worse still insufficient.

All four formalisms share common properties resulting from the fact that normally their implementation occurs in 'pattern-directed inference systems'. The following summary of their properties is based on that of Jackson (1986).

* Programmes consist of a number of relatively independent modules. e.g. rules structures or clauses for production systems, structured objects and predicate logics respectively.

* These modules are matched against incoming data and are activated if any of this data matches their trigger patterns.

* Such activation causes these modules to modify data structure. There is also some form of interpreter that controls the selection and activation of modules on a cyclic basis.

Production Rules

Since domain experts and self taught programmers are usually versed in a background of declarative languages, production rules hold an instinctive appeal and are often chosen as the medium of implementation without further thought. They do however have a number of problems :

* The default 'conflict resolution' strategy of the rule interpreter can have unexpected effects. This is particularly the case if the developer has not programmed the 'inference engine' but is using some form of expert system 'shell'.

* Since native bodies of rules are (often out of necessity) unordered and unstructured, it is difficult to direct sets of rules that perform different functions and to take advantage of the natural structures of the problem domain.

* They are less effective at expressing more subtle forms of knowledge.

* There is poor control of the rule search strategy in that it is 'short sighted' with only the default interpreter supplying the guidelines.

* Many of these problems can be overcome (as in the ProTuna prototype of this study) by the developer explicitly coding various control structures and 'meta-level' rules that control the flow of reasoning, but production rule systems will not implicitly exhibit such behaviour. The other formalisms can exhibit implicit reasoning.

Semantic Nets

These can be used to capture general information. Their nodes represent concepts and the joining arcs represent the relationships between these concepts. Any relationship between two concepts is established by performing an intersection search i.e. expanding ever outwards from each of the nodes until a link is formed. These structures suffer from logical and heuristic inadequacy and this combination can lead to severe problems. Changes have been made to try and improve such nets but little is gained at the expense of their main appeal, simplicity, which is lost.

The logical inadequacy is that the meaning of nodes is often left unclear and it is the meaning that determines the form of inference. Heuristic inadequacy arises because searches are not knowledge based. This in semantic nets causes very inefficient behaviour when trying to deal with negative responses because you have to have a full expansion of the search through the entire net before such negation can be confirmed.

Frame Based Systems

The shortcomings outlined for semantic nets inspired the development of frame based systems. Each frame can be regarded as a complex node in a network. The node is akin to a record containing data for the object it represents. The fields, or 'slots' as they are known in the trade, are filled with the name of the object and values for its common attributes. These can be filled out in advance with typical properties for a class, especially for the higher slots,allowing representations of what are known as prototypical objects for that class. Such structures allow the representation of real world knowledge such as expectations and assumptions. Tree structures or even tangled hierarchies can be created where inheritance of properties from a parent or associated class is allowed. The lower slots cater for the variable properties of a class of object. Again, default values can be inherited, limit conditions can be placed on values for slots or the slots can even call procedures themselves to calculate their own values.

Problems With Structured Objects

The main appeal of structured objects as real world analogues is their simplicity. Unfortunately this is often lost at the implementation stage. They are better at representing the more subtle forms of human reasoning that can't be placed in traditional formal logics such as predicate calculus. The main criticism is that their dedication to default and exception situations tends to undo any formal structure they have. Furthermore, there is great opportunity for inconsistencies especially where there are multiple hierarchies. Rule-based or logic-based systems are not immune to such inconsistencies either. The shortcomings associated with structured objects have led many people to question their epistemological foundations saying that they contribute very little to improving understanding of the structure of knowledge. This leads to the wider debate of truth versus utility. In the final reckoning structured objects are nothing more than higher level programming languages offering a convenience that can be easily misused.

Propositional and Predicate Logic

There is a lot of scope for confusing the role of logic in artificial intelligence. Moore (In Benson 1986) makes the concise observation that the application of logic basically falls into three categories :

(i) as an analytical tool
(ii) as a knowledge representation formalism
(iii) as a programming language.

Usefulness of a form of logic in one of these areas does not guarantee the same in one of the other areas.

Logic as an Analytical Tool

Moore's summary states that any formalisms must embody 'truth conditional semantics', in other words there must be some form of correspondence between an expression and the real world such that it makes sense to ask whether the world is the way the expression claims, a correspondence between form and meaning.

> 'The point of this exercise is that we want to be able to write computer programs whose behaviour is a function of the meaning of the structures they manipulate. However, the behaviour of a program can be directly influenced only by the form of those structures. Unless there is some systematic relationship between form and meaning, our goal can't be realized.'

Knowledge Representation and Reasoning

Though for practical reasons these aspects are separated as much as possible in actual expert systems, (see later), the theory of both is inexorably intertwined and therefor best presented as one for our purposes in this section.

It is difficult when attempting to summarize logic not to keep stepping back to more and more fundamental principles that are needed to justify generalizations, particularly when trying not to assume a readers familiarity with the field. Space does not allow a full exposition of the virtues and vices of logic and thus the reader is directed to textbooks on the fundamentals, Dowsing et al. 1986 is excellent in this respect, particularly for those with an interest in the computing aspects.

Logical systems consist of three components; the syntax, the semantics and the proof theory. The syntax consists of the rules that dictate the correct grammar for any logical expression. Semantics define under what conditions a sentence of the logic is true. Proof theory consists of axioms and sets of inference rules. The simplest case (apart from boolean algebra) of the use of logic in expert systems is the propositional calculus. We can use a truth-functional formulation or a deductive approach. For the purposes of this report the deductive approach is outlined further.

Our knowledge of the real world can be represented as a set of hypotheses or 'theories'. For any one of these theories the syntax rules define the possible constructs of the theory (propositions in this case). There is a set of propositions that is accepted as being self evidently true. These are known as axioms and could be taken as the 'facts' for our particular domain. Within this framework we can then reason with our 'theories' to provide someone with an answer to a question they have posed against the system or even form new theories from existing ones. We use the concept of 'proof' to do this. When such a proof is found we have the 'theorem' of the 'theory'. This theorem is itself a proposition. We can thus represent facts via the notational syntax and have a way of reasoning via proof theory. Any problem can be represented as a proposition to be proved against our knowledge of the world which is represented as a set of axioms, assumptions and theorems. The proving is done using 1st order logic. The solution to the problem can be extracted from the proof. The properties of the 1st order propositional calculus that make it attractive for such tasks are decideability, soundness and completeness.

Decideability :	If it is possible to decide that any given proposition representing our theory of the world is in fact a theorem the theory is said to be decidable. This is a very useful and fundamental property of the propositional calculus.
Soundness :	There is no proposition such that it and its contradiction can be proven to be theorems by the rules of inference.
Completeness :	Any proposition that is a theorem can be proven to be so using only the rules of inference.

We can also reason with assumptions using 'deductions' instead of proofs to arrive at 'valid consequences' instead of theorems.

Despite having these three properties a limitation of the propositional calculus is that it can't represent generalizations over and relations between propositions. (Shadbolt in Forsyth 1986). To do this you need to analyze propositions into predicates and arguments and deal explicitly with quantification (Jackson 1986). This requires extending the syntax to include quantifiers. The extended system is known as first order predicate calculus.

A drawback is that the property of decideability is lost with the predicate calculus. This problem is tolerated because of the added powers of expression and reasoning that are gained e.g. the ability to reason with incomplete knowledge. Completeness and soundness do carry over to the predicate calculus providing quantification is only over individuals and not over predicates. 'Second order' logic allows quantification of predicates but then the completeness property is also lost.

Formal logics have strict requirements which humans do not seem to conform to in their own reasoning. The dilemma is whether to accept the straightjacket of existing formalisms because they are machine implementable or whether to try and find a method of implementing more natural forms of reasoning into a machine, structured objects etc.

The Use of Logic as a Programming Language

There are many aspects to this. Automatic theorem proving, logic programming and knowledge representation. When considering methods of inference in combination with knowledge representation, theorem proving and logic programming are equally important aspects to consider.

A central issue in this particular project was how to have a method of proof that would run on a machine. All of the rules of inference when taken in all of their possible combinations is much to unwieldy a body to use as a control mechanism even if the individual rules can be adequately represented. Robinson in 1965 solved this problem with his simple and logically complete method for proving theorems in first order logic based on the 'resolution principle' (Moore in Benson 1986). This involves converting all axioms and propositions to their simplest forms where only the ~ (not) and V (or) connectives are used. The resultant expressions are said to be in conjunctive normal form (CONF). These are then taken together with a negation of the statement one is trying to prove. If a contradiction is found then the original statement is proven to be true ('proof by contradiction').

Problems Associated with Propositional and Predicate Logic

The use of resolution in programming languages like prolog makes the logical representation of knowledge a much more practical proposition. However there are severe problems to be overcome. The main one is that the search space generated by the resolution method grows exponentially (or worse) with the number of formulae used to describe the problem and with the length of the proof, therefore problems couldn't be solved in a reasonable time. (Moore in Benson 1986) The question of how to control the search strategy remained unsolved and researchers turned away from the use of logic.

However later work revealed that a fundamental problem in A.I., generally was how to reason with incomplete knowledge and that predicate calculus does provide tools for this, since there has been a revival of interest.

Unrestricted use of the resolution principle basically results in using every implication in both a backward and forward chaining manner which causes highly redundant searches. One domain independent way to reduce this redundancy is to impose a uniform search strategy e.g. forward or backward chaining. The problem is that the imposition of a particular search strategy may be inappropriate for particular domains and could still result in redundancy.

Study has shown that choice of search strategy depends on the specific form of an assertion and also the set of assertions within which it is embedded. Hence the way a concept (or problem) is represented logically, can dictate the kind of search control strategy that should be used. Logically equivalent formalization can have radically different behaviour under a given search strategy.

This is further evidence in support of the strong links between knowledge representation and inference. It appears the trick is not so much to choose or control the correct search strategy but being able to formulate your knowledge and problem in an effective manner. If methods cannot be found to do this then predicate calculus and languages based on it will not be able to progress for use on problems where knowledge is incomplete.

METHODS OF INFERENCE

Why is it necessary to consider methods of inference? It has already been shown that knowledge is structured, itself inherently represents a way of reasoning because we have :

* Facts or object knowledge which can be represented as assertions or frames.

* Knowledge of causality which can be represented by chains of statements (rules, propositions, links etc) relating cause to effect.

The problem is that knowledge is more complicated than this. Humans have a range of tactics with which to make more effective use of such bodies of knowledge and expert system builders would like to be able to emulate such tactics. In producing an artificial representation of such reasoning powers, it has been found easiest to try and artificially divide them from the knowledge itself. What we are trying to do is decide which question is it best to answer next? This division is artificial because we know that we have inference in knowledge and knowledge about inference, the boundary is somewhat fuzzy. This is the reason why some discussion on inference has already occurred in the previous section on methods of representing knowledge. However the distinction is a useful one, particularly in constructing conventional computer implementations as opposed to a connectionist implementation.

If a system responds to a query simply by evaluating it against the knowledge representation in a linear manner, it wastes a great deal of time. The more purposeful 'large scale' questions should be asked first to rule out large areas of more detailed questions which do not need to be asked because they are irrelevant. With either method, the raw 'ask all' or the more efficient 'pruning' method we are employing a forward chaining strategy.

This approach has the added advantage that the consultation makes more sense to the user, rather than having the questioning seem to jump around. Another approach might be to start with the aim of proving a given conclusion (or reaching a certain objective). This also has the advantage of appearing to make sense from the users viewpoint, because all questions can be perceived to be related to some aim. It also allows the possibility of asking important 'large scale' questions first. This sense of 'scale' is different from that used for forward chaining. In backward chaining we have the opportunity of asking the more difficult important questions first and if the user can not supply a direct answer (i.e. responds with 'don't know') the system can then split its request up into more and simpler questions and the combination of answers provided by the user, can through other rules, be used to solve the earlier more difficult unanswerable question. This strategy has an obvious appeal for many applications. The main problem is, which conclusion do you try and evaluate first. If you simply try and evaluate them in the order they are coded in the knowledge base it would again reduce user confidence. Perhaps they should be ordered according to likely probability of occurrence.

There are a range of 'conflict resolution strategies' to choose from which fundamentally affect the way a knowledge base is used. Such effects are often not stated, (Furse in Forsyth 1986). The developer must be aware of them and structure the knowledge base accordingly. In an expert shell the inferencing strategy is relatively fixed in the ready made inference engine but can be modified by the use of meta-level rules.

Various heuristics are used to control conflict resolution in inference engines. For example, when processing a rule set and a rule fires, do you stop and return to the start of the set because as a result of that rule firing it might have altered the preconditions for the previous rules that therefore need re-evaluating. Or do you continue to the end of the rule set? Do you keep re-evaluating rule sets on a cyclical basis until the conclusion is reached or until no more alterations are caused in the database of answers? If you are also employing opportunistic forward chaining you would want to use the last method. Other heuristics can be applied, like refractoriness, recency and specificity to refine the inference strategy.

Inference and Purpose

It seems that there has to be a closer relationship between the purpose of a system and its method of inference. This is, of course, more difficult to achieve when using shells with off the shelf inference engines. The majority of examples in the literature seem to get stuck in the very real rut of the symptoms --> diagnosis example and these are nearly always dealt with in production rule symbolism. The conditioning effect of always looking at a problem in this framework is a handicap that the knowledge engineer has to overcome.

The management of objectives in a dynamic system (e.g. fisheries) is a completely different situation. The control of a single objective, lends itself to the backward chaining approach. It is important aspect to bear in mind is the fact that the objectives and all the factors influencing them are in the real world being controlled by management of their trends. Thus objectives and the factors influencing them (management measures) can all be viewed as variables. Thus an objective could be stated as :

(i) Its name as a key variable

(ii) The direction in which its value must move.

In such a case a single objective with bi-directional control of its value, the system would simply backtrack through a tree structure (from root to branches) of related sub-objectives saying at each level in which direction their values have to move. In theory one should end up with a complete set of the finest atomizable management measures and instructions in which direction to move their values in order to achieve the desired shift in the objective key variable.

Of course we rarely have a single objective to control, but several. The complicating factor is that they will share some of their sub-objectives and since the influence between objective and sub-objective is bi-directional, changes in any one of the objectives will also influence the other objectives where there is a link further back up the hierarchy through the sub-objectives. These 'side effects' can be represented these through opportunistic forward chaining. This is a network situation. If the earlier proposal of management as the control of trends of the objectives and the management measures that influence them is accepted, then how is the relationship between these objects represented. Any one object either has a -ve or +ve (and occasionally 0 when in equilibrium) effect on any related object at a given time. The degree of influence will vary and could be ascertained from analysis or judgement based on human experience and intelligence.

Such relations are represented as rule weightings in some production rule systems. As an alternative weight could be attached to an arc in a semantic net. A frame based network where each object had its name, trend direction and perhaps even an absolute value where practical would be more useful. This would allow construction of the necessary tangled hierarchy with relations and weightings represented as additional slots. Trends analysis, optimization theory and graph theory could have useful application here.

Some correspondence with the neural network argument is obvious. It can be argued that developing such complex methods of representing inference particularly if they are implemented through some form of weighted production rules contribute little to rationality, so why not do it much more simply through a physical neural net. There is though, an important difference. In a structured object representation the programmer forms the links and weightings according to a rational model. In a connectionist implementation the connections and weightings are built up in an inscrutable fashion by iterative training with historical situations. This would present practical problems when trying to use models to make predictions on changes in legislation for instance. Such changes might introduce new influencing factors or alter the weighting balance of existing links. Without the historical precedent to train with, it is difficult to see how such effects could be implemented in a neural net.

Because connections and weightings can be made explicit in a structured object it is easier to experiment with both introducing new influencing variables and altering patterns of links and weightings. By this it is possible to build up a realistic representation of what goes on in a real system. The relative importance of different links becomes apparent and a clearer idea of how to manage the 'balance' of the system emerges. Using a neural net, even if it provided a more accurate predictive model, would not because of its 'black box' nature give any insights into the functioning of the system. Insights, even if flawed in some respects, can aid us in controlling a real world system.

A structured object model has the important 'two way' property of not only being able to predict changes required in the influencing variables to control the objectives but can also estimate the effect of changes imposed on the influencing variables (from outside of the closed world model) that are being controlled. This property is similar to that described earlier for the neural net model.

The following diagram illustrates a proposed structured object to represent a fisheries example (Figure 1). For ease of understanding it has been simplified to illustrate the key points. Everything is represented as objects with a variable state that can increase or decrease. There is no difference between the objectives and all the management measures that influence them. The only thing that distinguishes an objective from the other objects is that we wish to control its trend. The other object trends are not important in their own right, we don't care in which direction they go in, providing that the movement helps control the objective objects. In this case there are three objective objects, namely employment, catch, and catch per unit effort (CPUE). By tracing through the network we can see the effect that changing the values of any of the management measure objects would have on the objectives. Alternatively we can estimate the changes required in the management measure objects to achieve a given shift in one of the objectives (employment). The side effects of this on the other two objectives would also be made apparent.

Note that 'exploitation' is a 'fact' and controls the way catch influences CPUE.

Conclusion

This discussion of some aspects of inference has revealed a way of atomizing the fisheries situation into objects with variable states (see Figure 1). It also shows how links could be formed and the notion of weighting these be used to control trends. Consideration of inference has shown that structured objects are a potentially powerful method of combining inference and knowledge representation. It has tempered the earlier enthusiasm for a neural net implementations by showing that rationality has important and practical uses in management. A neural network could be viewed as one huge multiple regression. Many statistical models used for management are often applied in no less an intractable manner than neural nets would be. Compared to such 'cookery book' applications it appears that the more clearly causal representation offered by a structured object may be preferable.

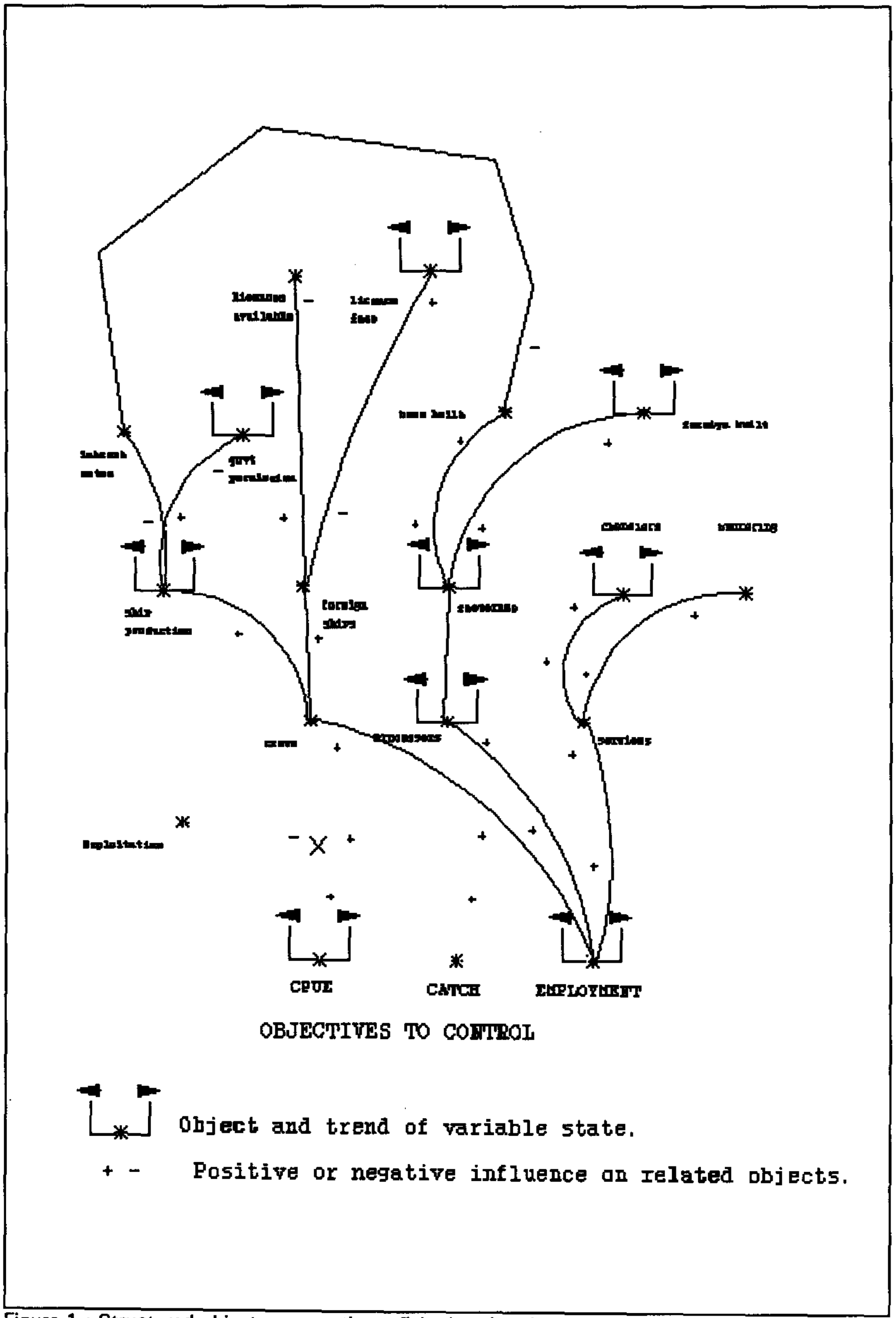

Figure 1 : Structured object representing a fisheries situation.

KNOWLEDGE ELICITATION

This is the other main plank of knowledge engineering. Three general problems can arise here (Goodall in Forsyth 1986) :

- Deliberate resistance
- Inarticulacy
- Cognitive mismatch between brains and programs

Deliberate resistance

This can occur where an expert sees himself as being done out of a job. It pays to chose the expert carefully. The expert must take the project seriously and be willing to allocate enough time to the work.

Inarticulacy

When the knowledge engineer is also the domain expert then the problem of inarticulacy is alleviated to some extent. However if the expert isn't an knowledge engineer there is a problem then of how to make the expert aware of and proficient in methods of expressing expertise i.e. all the representational and inferential formalisms that have been described.

The classic situation is where the domain expert and knowledge engineer are mutually ignorant of each others fields. A whole battery of standard techniques can then be applied to create an expert system. The theory is that for a good system to be produced it must mimic the unique methods of the expert therefore allowing their knowledge representation and inferencing methods to be reproduced as accurately as possible. Experts are supposed to be notoriously bad at making their knowledge and problem solving strategies explicit. The expert may be able to prevent the engineer going to far off beam with his expression of facts, but if the domain expert is ignorant of the methods of representing knowledge and inference and can not conceptualize his or her structures and strategies, then there is no person to stop the knowledge engineer putting forward their own interpretation. This interpretation may be considerably different from the experts actual way of thinking. Many of these problems can be avoided if one person can function both as domain expert and knowledge engineer. Rarely does such a situation exist.

A limited understanding of domain and engineering issues can be used very effectively because the combined knowledge gives it added value. Any spurious inspiration from an engineering technique can be kept in check by the understanding of the domain and vice versa. Maintaining a dialogue with more experienced people from both fields helps ensure progress of a project in a balanced manner, it is however a slow process.

Cognitive Mismatch

Gammuck and Young (1985) have identified misrepresentation of the domain structure as one of the reasons for the so called 'knowledge acquisition bottleneck'. Eliciting knowledge is difficult if it is forced into an inappropriate formalism (more often than not a rule structure) especially when there is poor understanding of the domain. This problem has even arisen with acclaimed systems such as MYCIN and DENDRAL (molecular modelling) where knowledge engineers have complained that their experts don't have or can't make explicit any form of codifying their knowledge. (Forsyth 1986).

Induction

One way to improve this situation is through the automatic creation of rules through a process of induction. These systems assume that the domain can be adequately represented via rules. Experts provide the system with example situations and the machine generates the rules relating cause and effect. The expert tests these rules against new situations and corrects or expands the rules where there are problems. Induction is liked because it seems that this method is a more natural way of transferring knowledge with the expert taking more of an active part.

Genetic Algorithms

This is another approach aimed at creating more representational rule sets based on the work of Holland (1979). The biological principles of selection and mutation are employed to evolve ever more efficient and successful rule sets. It is a rare example of classical biology offering impetus to computational science. Genetic algorithms are not only used for rule generation but a good example is Forsyth's 'BEAGLE', which stands for Biological Evolutionary Algorithm Generating Logical Expressions. It is a complete package that is intended to be a tool kit for the knowledge engineer. It works in the following manner :

* When presented with a data set it splits it into a training and test set.
* It then calculates simple statistical parameters for fields from the training set (averages, limits etc).
* It then creates a set of random rules which have these parameters as conditions.
* These rules are then tested against the training set.
* Rules that don't hold against relations in the data score low, more successful rules score higher.
* High scoring rules are retained, low scoring rules are discarded.
* New rules are created my mutating and splicing existing ones.
* The last five steps are applied on an iterative basis until a highly successful rule set is assembled and saved.
* The rules can then be verified against the other half of the data used as a test set. In tests that compare the relations BEAGLE forms between variables and relations formed by conventional statistical analysis, those created by BEAGLE have been shown to be the better representation, (Rowley 1990).

When using inductive and genetically derived rules great care has to be taken that example data covers the whole spectrum of possible situations, often the rules developed though successful are often complex and difficult to understand.

Structured methods

Of all the tools and techniques suggested for knowledge elicitation, Swaffield and Knight (1990) point out that there are few accepted methods of how to use these tools in a structured way similar to that used for 'structured systems analysis and design methodologies' (SSADM) which is used for formally resolving designs in conventional computing. They argue that often there is a lack of domain analysis in order to find a suitable structuring method. This is attributed to the lack of a formal design approach. Formal designs should have representations like data flow diagrams, functional decomposition and data modelling which can be shown to the users for 'validation' before progress is moved to the implementation stage. They also criticise the lack of user involvement in the design process. Little thought is often given to how the user interacts with a finished expert system. All the attention is instead paid to the expert. This seems to be a critical oversight. For example MYCIN the oft quoted classic medical expert system is not widely used because it does not fit in with a doctors working environment.

One of the reservations about the design of ProTuna is the ad-hoc use of knowledge elicitation tools. For this reason a more structured SSADM approach as recommended by Swafield and Knight is recommended for future work. These authors direct the reader to an existing structured method for knowledge engineering called KADS which has a defined set of stages and steps.

Hickman (1989) describes this methodology in detail along with a few methods that already exist for producing conceptual models of an expert domain, such as 'Extended Relational Analysis' (ERA), Addis (1985).

Standard Knowledge Elicitation Tools

These are divided into direct and indirect methods :

DIRECT METHODS

INTERVIEWS :

* Ask which objects are thought about?
* Ask how they are related or organized?
* Ascertain processes gone through in making a judgement, solving a problem or designing a solution.
* Questions should be free form narrow specificity, e.g How do you go about your task?
* Do not impose your own understanding.
* Limit sessions to coherent tasks.
* Recognize fatigue and attention limits.

QUESTIONNAIRES :

* Whereas interviews can illicit unforseen information, with a questionnaire the expert is much more in control.
* It is efficient time wise.
* Expert can fill them out at leisure.
* Good for eliciting objects, relationships and uncertainty (use bar or menu for this)

OBSERVATION OF TASK PERFORMANCE :

* Done in real time. Can suffer from pressure of time available and subjective interpretation by the engineer.
* Video performance. Expert can explain his actions but there can also be instances of actions they can justify in real time for which they have no recall subsequently.

INTERRUPTION ANALYSIS :

* Process cannot be resumed.
* Used when comparing expert to first prototype

INFERENTIAL FLOW ANALYSIS :

* Question about relations used to build up causal networks among concepts or objects.
* Identify key concepts.
* Ask what relationship between them is.
* Identify intervening variables.
* Form network links, give them a weighting of 0.5 and increase it for each mention.

INDIRECT METHODS

A variety of these were assessed in the development of these expert systems. For details the reader is referred the paper by Olson and Renter (1987) or a good expert systems textbook.

METHODS BASED ON SIMILARITY MATRICES

* Multidimensional scaling
* Johnson hierarchical scaling
* General weighted network

ORDERED TREES FROM RECALL

REPARATORY GRID ANALYSIS