

A CYBERNETICAL APPROACH TO A GENERAL MODEL
IN ARTIFICIAL INTELLIGENCE

BY

ANDREI SORIN

Thesis submitted for the degree of Doctor of Philosophy
University of London

June 1975

ABSTRACT

A general approach is proposed for simulating intelligent behaviour, after showing why a direct simulation of the natural intelligent system should be more advantageous than the attempt to simulate separately each one of its aspects. These aspects are viewed as the interaction between an organism and its environment in general, and intelligence is defined as the ability of the organism to adjust efficiently to the environment.

The basic models for the organism and the environment are then developed, and it is shown that when they are joined together as an integrated system their interaction can be represented as a succession of states and intelligent behaviour can be interpreted as the capability of the organism to predict the sequence of states. This system can be viewed as a feedback control system and then intelligence is equivalent to the tendency towards stability.

Different types and levels of complexity are defined for the environment to simulate the repetitiveness and the uncertainty found in real processes. The extensions from one level to the next are based on the increase in the complexity of the time patterns of states. Environment generators are implemented for producing these sequences and several parameters are provided for changing their characteristics. A method for adjusting to the environment sequences, i.e. learning to predict them, is developed and organism models are implemented to match the various levels

of environments, having their own parameters for obtaining the best possible adjustment.

Several experiments are carried out with these systems and their response is analysed. Their behaviour is level-independent, as a result of the original approach of equating intelligence to the capability of adjustment regardless of the level of complexity of the system.

ACKNOWLEDGEMENTS

I wish to thank my supervisor Prof. D. Howarth for his constant help and encouragement throughout the research period; and Mr. E. Nixon for his helpful suggestions.

This work was supported in part by the scholarship fund of the Bnai Brith Leo Baeck London Lodge.

CONTENTS

	<u>Page</u>
ABSTRACT	2
CHAPTER 1. INTRODUCTION	6
1.1 Background	6
1.2 The conventional approach	10
CHAPTER 2. PRELIMINARIES	22
2.1 The definition of intelligence	22
2.2 The organism and the environment	31
2.3 The system	40
CHAPTER 3. THE ENVIRONMENT	54
3.1 Simulation	55
3.1.1 The problems	55
3.1.2 Definition of levels and types	65
3.2 Implementation	75
3.2.1 The environment parameters	76
3.2.2 The programs	81
CHAPTER 4. THE ORGANISM	95
4.1 Simulation	95
4.1.1 The problems	95
4.1.2 The strategy	103
4.2 Implementation	109
CHAPTER 5. THE INTEGRATED SYSTEM	131
5.1 Introduction	131
5.2 The type 1 system	137
5.3 The type 2 system	140
CHAPTER 6. CONCLUSIONS	160
CHAPTER 7. RELATED WORK	171
APPENDIX. AN EXAMPLE	192
REFERENCES	198

CHAPTER 1

INTRODUCTION

1.1 Background.

The first paper to deal seriously with artificial intelligence was probably Turing's "Intelligent Machinery".¹ He argued that a general purpose digital computer can be programmed to display intelligent behaviour comparable to that of man, pointing out that the knowledge must be input to the machine in a "teaching" process. This process is applied to an initially unorganized system which in time should be able to cope with any tasks, of any complexity, this method being best suited to specific intellectual applications.

Twenty-five years later we are still a long way from attaining the goal of an artificial intelligence comparable to the human one; but before trying to find possible reasons for this we will have a look at what has been achieved.

In spite of the concise definition of the research goals in artificial intelligence - the development of a systematic theory of intelligent processes² - or probably because of it, this research covers a large variety of problems. The obvious explanation is that in the description "intelligent processes" are normally included all the higher mental activities of man.

One of the first fields of investigation was that of pattern recognition; today it is still the most important

subarea and its implications have reached into many branches of science. The problem in pattern recognition can be described as that of a many to one mapping, from a set of experiences to a much smaller set of internal meanings, or names.³ The pattern to be recognized can be either simulated or actually input by any of a variety of ways - a TV camera for example for visual patterns. Almost always this pattern is represented internally as a two-dimensional array of discrete symbols. In real-life problems the number of such patterns is too large for a direct mapping to be practicable (by using a table for instance). The task of the program is to simplify the pattern and decide to what class it belongs, within given constraints of time and accuracy. This is usually done by a process of characterization which includes preprocessing of the pattern and various measurements. The preprocessing stage uses a set of transformations in order to regularize the original pattern: normalization for size, sharpening of edges, rotation, elimination of "noises", etc. The measurement phase varies significantly from program to program, but its task is always to analyse in the most efficient way the preprocessed pattern either in one or in several stages, by looking for certain characteristics, evaluating and comparing them. These measurements are best done in a combined sequential and parallel mode, although the choice may depend on the particular type of problem.

The crucial problem in pattern recognition programs is the choice of the best set of measurements to be made

and the order in which to make them, i.e. the decision between sequential and parallel search at each stage of the processing. These factors greatly affect the speed and indeed the feasibility of the program.

Another important area of research is that of game playing. It had been suggested that computers are an excellent medium for this research even before they came properly into existence. Turing, Shannon, von Neuman and others described ways to solve the time and storage constraints by look-ahead, evaluation and minimaxing, used as far as it is practicable. The games of chess and checkers have been the favourites in this research, which was extended to game learning as well: the program changes some of its parameters according to previous failures and successes, in order to improve its future performance.

The reason game playing programs have received so much attention is that games provide complex and yet well defined environments in which to test general problem solving methods. Thus, the heuristic search and learning methods used in these programs are similar to those in other areas of artificial intelligence.

Closely linked to game playing is the area of theorem and hypothesis proving; the common factor in all these problems is that they are solved in steps, and the process at each step is that of a decision making. The program follows a certain preset strategy and uses a heuristic free search at each step. Many of the aspects considered earlier for pattern recognition are also encountered in

these programs and in fact in most fields of artificial intelligence.

Another area in which much research is being done is that of robotics or, as it is usually called, integrated cognitive systems. These systems consist of complex receptors which act as input devices and send their signals to a computer; various programs then analyse these signals and send their decisions to an output device - a mechanical arm for example⁴. Such a system can be used for many tasks simulating intelligent behaviour in manipulating objects.

Other areas normally classified in the field of artificial intelligence and which have received considerable attention are: the interpretation and translation of natural languages, pattern creation, simulation of cognitive processes, etc.

*

All the problems mentioned above have been the subject of considerable research effort. In a few cases the achievements are significant. Some pattern recognition programs for example have found wide applications in science (although they still have serious restrictions on the input patterns). And some game playing programs, notably Samuel's checkers program⁵, can perform satisfactorily against better than average human players.

But these are the exceptions. Against them one can

mention the failure of the language translation projects², the chess playing programs which are still at amateur level and many other problems. In particular, the problem of recognizing unconstrained hand printed characters has probably received more attention than any other single problem in the field of artificial intelligence, and still has no satisfactory solution⁶.

In the next section we will try to find possible explanations for this situation and we will propose an alternative way for dealing with these problems, which may bring us closer to achieving a higher level of artificial intelligence.

1.2 The conventional approach.

Most artificial intelligence programs have traditionally been specific-task programs. These programs are designed for solving one particular type of problem. Such a problem could represent a certain aspect of intelligent behaviour, but is usually even more restricted - for example the recognition of a specific class of patterns.

This state was reached mainly as a result of the failure of the early general purpose programs. The ideal system in artificial intelligence has always been one that can simulate intelligent behaviour in general, by translating the various problems into a common internal representation and then solving them by using the same mechanism. This approach proved to be far too difficult because no such

representation could be found, whose implementation would be practicable. The generality of these systems was thus gradually relaxed, in order to bring them within the range of attainable goals.

While such a step may seem to be quite logical, it was probably the main reason for the slow advance we are witnessing in artificial intelligence. The appeal of the specific-task research cannot be denied: less effort is needed and the chances of finding a quick solution are better for simplified, individual problems. It may be sometimes useful to adopt a method which is likely to bring a short-term, partial solution to a problem, but these cases should only be the exceptions. Avoiding the general approach just because it is too difficult cannot be considered as a serious scientific argument. In fact the adepts of the specific-task approach tend to invoke the plausible argument that the complex problems are composed from simpler ones and it is easier to solve them by tackling first the lower level problems. But this clearly is only an assumption and there is no evidence that by combining the solutions of partial problems we will solve a complex one. The results obtained so far confirm this view - apparently we are still at the stage of subdividing the problems into increasingly simpler ones; the stage of putting the solutions together seems to be as remote as ever.

The systems normally classified as general are not actually general in the original sense. The problems they

can solve belong to well-defined, restricted sets and thus they are in fact specific-task programs, only their scope is wider. The term "general" is only relative to the specific approach, in that these systems do use a common representation for solving different problems. Unfortunately, little research has been done even for this limited type of general purpose programs. It is interesting to see that, in spite of the fact that this general approach solves directly more complex problems of intelligent behaviour, the performance achieved for the individual subproblems is comparable with the performance of the specific-task programs. The GPS⁷ and its various improvements and derivations discredit to some extent the common view that the efficiency of a problem-solving program is inversely related to its generality. This makes one wonder what would have been the state of the art today if all the research effort spent in the pursuit of short-term partial solutions had been spent on the development of a general theory of intelligence.

Artificial intelligence is a field in which piecemeal research is useless. We are trying to simulate the intelligent behaviour of man and this behaviour is governed in its entirety by the central nervous system. What may appear as different aspects of intelligence from an external point of view, are a result of the same internal mechanism - clearly we do not use different parts of the brain for each kind of problem we have to solve. The first mistake was made by trying to simulate general intelligent behaviour

without first understanding this mechanism. It was thought that computers are powerful enough to overcome this ignorance and the whole problem is only one of programming. We still come occasionally across those tables which compare the features of the brain and the computer, with the conclusion that apart from the fact that the computer is not a parallel processor, it is many orders of magnitude more efficient than the brain.

Then it was realized that the problem we face is not mere programming and that we might have to understand more about the natural system before creating an artificial one. But since intelligent behaviour in general is too complex, most research concentrated on the study of its separate aspects and this was the second mistake. The separate aspects of intelligence - when viewed as the whole set of problems the natural system has to solve - are far from being unrelated, and important common factors only become apparent when studying the system as a whole. By choosing individual aspects the problem is indeed reduced to one of programming, but little knowledge about the natural system can be used. The results are well known: even at this low level, no matter how sophisticated the programs are, their performance is still far from the human capabilities. Furthermore, the research in any one of the separate aspects cannot enhance the understanding of the mechanism of the natural intelligent system as a whole, hindering thus its own advance.

This is the way the current situation was reached - research in artificial intelligence is mainly research in programming techniques. This is unavoidable, because by disregarding the integrated natural system one also eliminates the clues to its working procedure. One has then to find ways for an efficient simulation of a system one does not understand, and what should only be technical details in the implementation become the main subject of research.

*

The area of pattern recognition is typical of the research done in the development of specific-task programs. There is a proliferation of effort and a variety of approaches and motivations, but little concrete realization of practical working systems⁸. Much research is put into the development of sophisticated mathematical techniques but while the simple problems are easily and successfully solved, no significant advance has been made in the solution of the more complex, real-world problems.

The way this situation was produced is also typical of the conventional research in artificial intelligence. It was quickly realized that the recognition of patterns by using the simple and obvious method of template matching is impracticable for major applications. According to this method, the patterns are mapped into matrices and the

corresponding elements are compared in order to determine how close two patterns are. While it is very reliable for fixed patterns, simple matching cannot cope with such common aspects as size invariance, rotation, etc. In other words, patterns which are instantly recognized by humans as similar may produce widely different templates.

The possibility that the natural system uses simple template matching for pattern recognition was thus regarded as very unlikely. The research turned then towards finding ways of preprocessing the patterns and analyzing their features in various ways, as part of the recognition procedure. Little effort was devoted however to understanding the way in which the natural system processes these patterns; even when such research was done, the criterion for the correctness of an assumption on the natural mechanism was usually only the success of its simulation. The computer became thus a tool for testing various hypotheses and the research in pattern recognition consists now largely of the development of mathematical techniques needed for implementing these hypotheses.

Meanwhile we still do not know how the natural system works when recognizing patterns and those who do research by following this approach still hope to discover a theory which can simulate the natural mechanism, without actually understanding it. As we have pointed out, all aspects of intelligent behaviour are interconnected and it is wrong to study on their own problems in pattern recognition; but

the specific-task approach goes even further - specific problems within this field are studied separately. Pattern recognition is a good example of the negative results of the conventional, specific-task approach. It is inherent to all the activities of the central nervous system, as we will see in Chapter 2; and yet, by following this approach the results from research in pattern recognition are generally irrelevant to the other fields of artificial intelligence, except maybe for obvious technical aspects of implementation.

The natural system may or may not use some complex transformation and analysis processes before recognizing patterns. But why devote such a big research effort to finding ways for simulating these processes before understanding them, or even establishing their existence. Maybe we recognize different patterns as similar not because of some obscure process but simply because we have learned that. In other words we recognize them as different patterns, but having under certain circumstances the same meaning - there are thus several levels of recognition and this is clearly a problem which has to be studied together with all other aspects of intelligent behaviour. We will see in Chapter 6 (and in the Appendix) that the simple matching process, when used in the context of our proposed general approach, is not as naïve as it may seem.

*

The scattered research which has resulted from the conventional approach has little chance of producing a breakthrough and there is no wonder that artificial intelligence is looked upon as the "no win" area of computer science⁴. Ironically, this research in the wrong direction has produced results which are useful in computer science in general. The heuristic methods, the various optimization and search methods and the list processing languages are mainly a result of the overemphasis placed on programming techniques while trying to simulate intelligent behaviour. They are recognized now as major software developments in their own right and oddly enough, they are invoked as evidence for the importance of research in artificial intelligence, with the implication that this misdirected research is going to continue.

The fact that research in artificial intelligence has become involved mainly with implementation details also means unfortunately that its advance, like generally in computer science, is tied to advances in software and hardware technology. But unlike in other fields, in artificial intelligence we have the natural system we can study and it is up to us to take full advantage of this. If we try to simulate some of its aspects without considering first its whole mechanism, then we have a challenging implementation problem, but little hope of finding the right solution.

We are considering here a different approach, which avoids the mistakes mentioned earlier: we will attempt to find a satisfactory explanation for the concept of intelligent

behaviour; and we will do this by studying the natural system as a whole. The ultimate goal is to design a system which can simulate the entire set of higher mental activities without restrictions. Clearly such a project can only have a long-term solution and the first attempts can only show the way for future research. Also, it is impossible to predict at an early stage whether such a system will be technically feasible when in an advanced form. Therefore implementation considerations must be viewed only as secondary problems in the initial phase of the research.

The logic followed is as follows. Any problem we would like an artificial intelligent system to solve is a problem that can be solved by a natural intelligent system (by "natural intelligent system" we mean the human species, although some of the problems can also be solved by lower species). Any program in this field effectively tries to simulate one or several human capabilities. But all these capabilities are made possible in the natural intelligent system by the same organ - the brain. Then why not simulate directly the activity of the brain, and then all the other problems will be solved automatically.

The simple answer is of course that the activity of the brain is too complicated to simulate, and we do not completely understand it anyway. These arguments however are not as strong as they may seem. We do not have to simulate the low level actions that take place in the brain, for simulating the higher activities; it would not be practicable to do this even if we understood them. We

only need to know those actions which can be observed from "outside", by considering the brain a "block box" and watching its input and output signals. This method has long been followed in psychology and the relevant results are well established.

But we can go even further. Let us regard the brain not as a certain system whose activity we are trying to simulate, but as what it in fact is - the product of a long process of evolution and natural selection during which it has achieved its present set of capabilities. Rather than study its activity, we can therefore study the circumstances under which its capabilities have evolved. When we understand this aspect we can design a system which will respond to similar situations in the most efficient way. This will ensure that the artificial system works in the same way the brain does and will eventually have all its capabilities. Another advantage of simulating directly the natural intelligent system becomes now apparent: this is a perfectly general system that not only is known to work but also, being the result of a continuous evolution process, is the best possible system for the problems it has to solve. Why bother to discover mathematical and optimization techniques for one problem at a time, when we have at our disposal the final product and the proof that it can solve any of these problems in the most efficient way.

The idea of simulating the activity of the brain is not new. Many of the specific-task and general purpose

programs discussed earlier model their working procedures on the natural system. Our approach however is an attempt towards complete generality - a direct simulation of the natural system without considering any application in particular. Those programs only simulate some aspects of the natural intelligent behaviour, as relevant to their applications and always departing from the original idea because of practical implementation constraints. Since we are not looking for an immediate solution to a particular problem, we can afford to follow with more fidelity our theory. We believe that in the long run this will pay off by providing answers that the solution of partial problems cannot provide.

The results of the conventional approach have been scattered research effort and a departure from the practical aims through too much involvement with the means of achieving them. The hardware constraints can always be blamed for the limited performance but each new generation of computers has seen its capabilities (which were orders of magnitude higher than the previous ones) stretched to the limit within a short period, without a significant improvement being achieved in the "intelligence" of the programs. We ought to be sceptical therefore of ever achieving performances of the human level by these methods. It has been remarked in connection with the chess programs that for making a significant breakthrough what is needed is a radical advance in the theory of machine intelligence and not just an improvement in programming and hardware technology⁹.

It is still generally believed that by trying to solve

specific problems we will eventually achieve performances comparable to the human ones; and that if we can solve elementary problems then we can solve problems of any complexity, simply by breaking them down into elementary processes. The evidence accumulated so far contradicts these beliefs. A frontal attack on the problem of intelligence as a whole seems to be the only way left. This is a long-term project and we cannot expect any spectacular results in the near future. But then, the specific-task programs were thought to be simple projects with guaranteed short-term results, and two decades later we are still struggling with the original problems.

CHAPTER 2

PRELIMINARIES

2.1 The definition of intelligence.

In this section we want to find a definition for the intelligence we will try later to simulate. It may seem unnecessary to look for a definition, in view of the fact that we have decided to simulate the natural intelligent system as it is, and take it for granted that it is the best such system, as a result of the evolutionary process. However, we need a definition and there are two reasons for this.

First, we have to remember that although we want to simulate the natural system, we do not know how it works; we have decided to regard it as a black box and study its behaviour as seen from "outside" only. This means that we will not be able to simulate the basic mechanism and then expect the behaviour of the resulting system to resemble the natural one. We must simulate directly this behaviour, as we know it and by means best suitable to a digital computer, whether the basic process resembles or not the natural one. Therefore, before we attempt to simulate it, we have to find out precisely what this behaviour consists of and since all the higher mental activities are based on it, it will be our definition for intelligence.

The second reason is technical. When we simulate the intelligent system we will need a criterion on which to

judge its performance. Also, we will try to improve it, by changing the values of parameters for instance, and again we will need a criterion to decide which is the best version. A simple and unambiguous definition for intelligence is thus essential.

The literature on artificial intelligence offers many definitions for the intelligence of a system. They range from the general ones, e.g. the ability to display a behaviour similar to a human one under the same circumstances, to very precise definitions limited in scope to particular tasks. As one would expect, the general definitions tend to belong to the general purpose programs while the more precise ones to the specific-task programs. The best known definition, which is also the most general of all, is the Turing test¹⁰: if a man communicating via a teleprinter for a sufficient period of time cannot decide whether at the other end is a computer or another man, then we can conclude that the behaviour of the program under which the computer works is similar to a human behaviour. This test can in theory cover all the aspects of human intelligence but, being only a test, it cannot help us in actually designing the program.

The specific definitions concentrate on the particular aspects of the human behaviour the specific-task program tries to simulate. For the pattern recognition programs for example, intelligence means the ability to classify a pattern, normally visual, according to certain characteristics. Usually the definition is even more restricted, to specific

types of patterns: geometric forms, letters, etc. For game playing programs intelligence means the ability to choose the best winning strategy, to vary it in the course of a game as necessary and perhaps to learn from its own experience, and so on.

The general purpose programs also limit their definition of intelligence to specific mental tasks, although these tasks include more aspects of human behaviour. The task is usually problem solving, as in all integrated cognitive systems (robots) and in the General Problem Solver mentioned earlier. It is argued that any human intelligent behaviour involves problem solving of some sort and that if a problem can be adequately represented internally the program can solve it by using a fixed, basic procedure. Unfortunately, as we pointed out in Chapter 1, there is no proof that any complex problem can be solved by breaking it down hierarchically into subproblems, and therefore we cannot take the ability to solve problems as our definition of intelligence. Apart from this, the internal representation of the problems is a very difficult question, as yet virtually unsolved.

Psychology is a much older science than artificial intelligence and we could expect to find in this field suggestions for the definition of human intelligence. It is interesting to see however that in psychology the research has been following the same lines: there is much literature on specific aspects of behaviour and practically none on an integrated theory of intelligence. It is common to

find texts dealing separately with subjects like learning, memory, problem solving and concept identification, without making any attempt to put them together and find their interrelations - something that could be taken as the basic definition of intelligence. More recently, when psychologists started using computers for simulating their theories, this trend has continued and the theories simulated are invariably models of specific aspects of intelligence^{11, 12}. These models are in fact closely related to the specific-task programs in artificial intelligence. It is suggested that the heavy use of computers for modelling specific intelligence tasks has produced a decline in the interest psychologists show for a broad, general approach to intelligence in their experiments with human subjects¹³.

*

It is quite easy to define intelligence for a particular task - it means solving that task in the most efficient way; and the more restricted the task is, the easier it is to test reliably the intelligence of the system built to simulate it. It is meaningless to try to define intelligence as an abstract term, without considering the particular context in which it has to be applied. The definitions discussed above have one thing in common: their economy. Clearly, there is no need to use a broad definition and build a model to satisfy it if we are only interested in a subset of its specifications.

There are several ways we can try to find a definition which would encompass the whole range of intelligent behaviour. One is to put together all the definitions for particular tasks and build a model which can cope with each one of these tasks; but this - if at all feasible - is little more than using a separate model for each task. Another way is to find what are the common requirements for the solution of all individual tasks. It must be reemphasized that our basic assumption is that there is a set of features present in all aspects of mental behaviour, and that the natural intelligent system does not have subsystems specialized for the individual tasks it can cope with. In other words, these tasks only look different when seen in the context of the organism as a whole, but they must all be processed in an identical way by the central nervous system.

If we can find these common requirements then we can build the ideal general intelligent model. But finding them is a difficult problem, since it means using a deductive method and analysing all the mental tasks one can think of. We will follow a different and easier approach: we will define the intelligence needed for solving a specific task, but this specific task will be precisely the set of common requirements needed for all aspects of mental behaviour. We are thus defining intelligence not at the level of the organism as a whole, but at the level of the brain. All we have to do then is to find out what is the task of this

organ.

We have no intention to go into any physiological details, because it is sufficient for our purposes to regard the brain as the result of a long process of evolution, being now in the higher species the main instrument of survival. The problem of survival is a problem of being able to adapt to the environment and it has two aspects. One is the survival of the species as a whole, a slow process of changes in the anatomical configuration of the animal in the course of many generations. The ability to produce such changes is essential in a changing environment and the mechanism of natural selection ensures that only those species which can change together with their environment will survive.

The second aspect is the survival of each member of the species. Clearly no anatomical changes are possible for the individual and besides, the changes in the environment that the individual member has to cope with in its limited life span are of a different nature. These are much faster changes and most of them are only temporary. The organism not only has to adapt to the changes in its environment as quickly as possible, but may also need the ability to discard new habits as quickly as it had acquired them. It is for this crucial task that the brain has specialized in the course of evolution.

All aspects of mental behaviour are therefore aspects of the brain's ability to adapt to the organism's environment.

We can formulate then our definition of intelligence as the ability to cope with the environment in the most efficient way. We have reached this conclusion following a logical argument, but obviously it cannot be taken as a formal proof that any mental task can be reduced to the task of adapting to the environment. We take it as a definition - the most general definition of intelligence - and we will use it in the future for defining the system which should exhibit this quality, implementing, improving and studying its performance. Also we must remember that the definition as formulated above applies to the brain only, not to the organism; thus the other organs, in particular those which come into contact with the environment, can be considered as the tools by which this general definition of intelligence is translated into intelligence for solving specific problems in the sense of the specific-task models described earlier. We only intend to simulate the behaviour of the brain, to which this definition applies. Therefore, whenever we use in the future the term "organism" it must be understood that what we mean is the "owner" of the brain - the organism as such is not the subject of our research. The brain by itself is clearly only a theoretical concept: it could not exist and it could not have evolved to what it is anyway. The inclusion of the organism in the discussion is essential for using our definition for intelligence, the organism being the means of communication between the brain and the environment, and also the reason for its existence.

Our approach to the definition of intelligence is very

much cybernetical¹⁴. This science, being an integration of concepts from many fields - psychology and artificial intelligence among them - has a very broad interpretation for intelligent behaviour¹⁵. In fact, it does not even refer to behaviour as "intelligent", because of its implication of an animal system. Cybernetics does not differentiate between natural and artificial systems, disregarding technical details and studying them from a general point of view by emphasizing the behaviour of the system as a whole rather than its individual parts. The problem cybernetics is concerned mostly with in a system is its stability. All closed systems, artificial or natural, possess a feedback mechanism which tends to bring them to a state of equilibrium. The organism and the environment are regarded in cybernetics as a closed natural system¹⁶, the organism being able to adjust its relationship with the environment so that the system is always in equilibrium. It is this quality we have interpreted as intelligence in our definition, but we argued that it has developed as a necessity for the survival of the organism¹⁷ rather than involving the more general and abstract cybernetical criterion of stability.

*

We will consider now the implications of the definition we have adopted for intelligence. The organism is aware of its environment through its sense organs and can act

on it through its motor organs. The brain must communicate with the environment via these organs and by means of specially coded signals. Coping efficiently with the environment has to be interpreted as producing an adequate response in each particular situation (we will see later what the terms "response" and "situation" must mean). Therefore the action of the brain must be to recognize and to produce specific combinations of signals. If we replace the term "combination of signals" with "signals pattern" then this becomes a problem of pattern recognition. But there is more to it than a simple pattern recognition process. The brain must be able to find the correct responses by itself from the continuous stream of patterns, and also to change them according to the changing environment (we disregard the innate behaviour, which only amounts to a small proportion of the total intelligent behaviour in the higher species, especially man). In addition, the time dimension has to be taken into consideration in the pattern recognition process. The organism would adjust better to its environment if it could remember not only the correct responses, but also how long before and how often had each situation occurred; and it would be even better off if it could also remember the degree of success (in the sense of coping with the environment) achieved with the various responses.

As already pointed out, we do not intend to try to find out how the brain does this; we only know that in order to cope efficiently with the environment it has to do it, and what we intend to do is try to simulate this process.

The model for simulating the intelligent behaviour of the brain cannot be a simple pattern recognition program. It has to learn and modify its knowledge by itself through its contact with the environment only, and it has to cope with complex combinations of patterns. Still, it may seem surprising that such a model should be able to simulate all aspects of intelligence, just by this process. It is impossible to prove it at this stage. We can only say that, according to the logic we have followed, it ought to - but we must remember that we are only simulating the brain and many tasks of intelligence rely heavily on specific organs.

Our problem is then to simulate an organism which can adjust to its environment as efficiently as possible. Next, we will define more formally this system, so that we can later implement it and study its performance.

2.2 The organism and the environment.

We are still viewing the organism and the environment models as black boxes, since at present we are only interested in their external behaviour. What we need at this stage is to find the minimum requirements which would enable them to act as the two components of the system described in section 2.1.

We will consider first the organism model. Its first requirement is obviously the sense and motor organs, these being its only means of communication with the environment

and thus essential for fulfilling our definition of intelligence. A natural organism has a limited number of such organs. Furthermore, each organ can only act in one direction and is capable of transmitting from or to the environment messages only from a specific and limited range of values. All these restrictions are due to physical limitations and since we want to simulate the natural organism as it is and we will eventually have to implement this model, there is no reason to relax any of them. The organs are connected to the brain by nerves which act as direct communication links. The brain receives and sends signals which correspond to the signals exchanged between the organs and the environment. Since we are only interested in simulating the brain, we will only consider the signals between the brain and the organs. These signals also are from a specific range for each organ but unlike the organ-environment signals they are organ-independent, similar to one another in their nature. This means that in simulating them we can use the same type of values for all of them.

We will use the term "input and output channels" in our model for designating the connections of the brain with the sense and motor organs respectively. We are simulating an organism in general, therefore the number of channels and the range of signals are not important. We can state then that our model must have m input channels and n output channels, each one transmitting signals of the same type, in possibly (but not necessarily) different ranges of values.

The other requirement for the organism model is a

processor P which acts on these signals so that the net result, as seen from outside, is an intelligent behaviour in the sense of our definition. Clearly, this is the central component in the model, whose simulation is the most difficult. Its mechanism is not relevant at this stage and will be discussed in Chapter 4. The basic organism model and its relation to the environment is represented in Figure 2.1 (i and o stand for the input and output channels respectively). It must be remembered that we have eliminated the organs from their role of

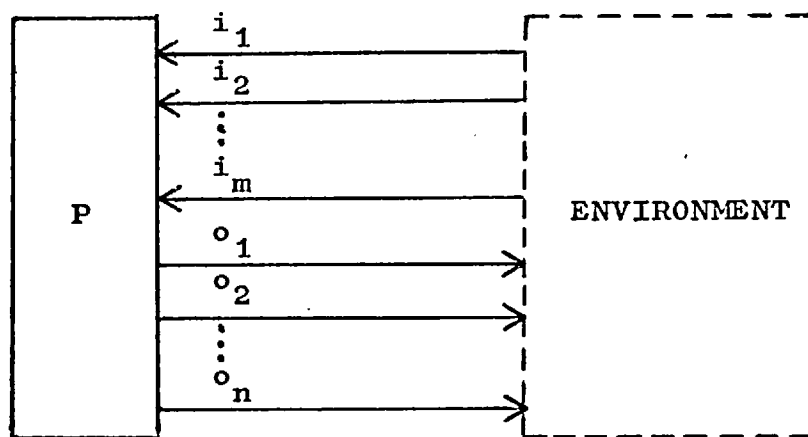


Figure 2.1 The basic organism model.

intermediaries between the channels and the environment, since we consider them as irrelevant for a general model. It may be convenient in fact to regard them as part of the environment: the organism is as much restricted by their capabilities as it is by the characteristics of the environment.

In the natural organism the signals between the organs and the environment are of an analog nature, while those

between the organs and the brain (the ones simulated by the channels in our model) are more complex, their values having both an analog and digital significance. We are not interested in their exact nature since we have to consider our own implementation problems. The model will eventually be simulated on a general purpose digital computer, therefore the simulation will have to be digital for both the values of the signals and the time dimension. In other words, the signals themselves will be converted from analog to digital and their values will be observed at discrete time intervals. The action of the processor must also be geared to this digital sampling method, so that the behaviour of the model as a whole is only defined at the sampling time steps t_n . This will not pose any problems because at this stage we will not have to make any analog to digital conversions. The model is purely theoretical, all processes being simulated. When the actual time dimension of a real process has to be simulated, the time intervals can be made arbitrarily small so that no information is lost.

For convenience we will decide that the subscript n of the time steps can only be a positive integer, the simulation of a process starting at time step t_1 and continuing as long as necessary. The origin of the time axis is thus zero, but the process itself would be the same if translated anywhere on this axis. In other words, it is not a function of the numbers n , but only of the relative position of the time steps t_n from the origin.

There will be no time units as such and we can conveniently

define the time interval as one unit. But these units have no meaning outside the context of the particular process simulated; the interval is a function of this process, specifically it is the longest possible interval which still produces a correct simulation. In fact these intervals do not even have to be of constant length throughout the process: they will be matched to it so that only the signals at significant steps are taken into consideration. Since the system organism - environment is a closed, fully simulated system, this is clearly the most efficient way of representing the time dimension. As we will see later, the number of time steps needed for simulating a process is strongly correlated to both the time and storage required. For a real-world process it is usually easy to define the significant steps and this will not be a problem in the later stages of development. In the following we will not mention it any more and when referring to a certain time step t_n , previous step t_{n-1} or next step t_{n+1} , we will mean the logical steps, as relevant to the process simulated.

We are now going to make some simplifications to our model. Rather than having m input channels and n output channels, each transmitting signals from a specific range, the model can have only one input and one output channel with signals from wider ranges. These ranges will be a function of, respectively, the ranges of all the original input and output channels. It can be easily shown that this does not result in any loss of generality. The signals are all defined at every time step and have values of the

same type, which can be represented by integers, all positive and within a fixed range for each channel (due to their digital representation). Each such combination of values can be coded into a unique representation, for example by using a number system with the radix larger than the largest possible value of the signals and using these values as digits, in a fixed order. In this way the signals can be unambiguously coded or decoded, according to whether they are from the input or output channels. In practice of course the method used for this reduction will depend on the implementation used. The simplified version for the organism model is shown in Figure 2.2.

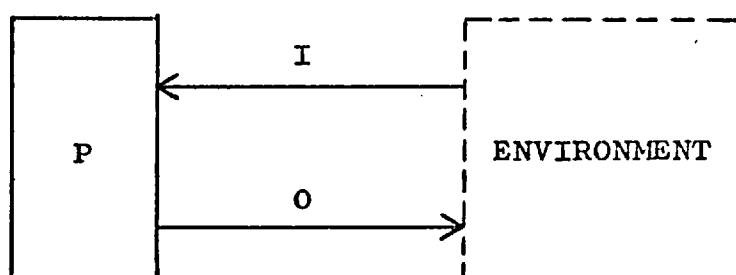


Figure 2.2. The simplified organism model.

The next simplification is another step in the same direction. We can combine the values of the new input and output channels into a single value, in a way similar to that used for the combination of the several original channels into one. We will call the new value "state" since, at each time step, it completely identifies the state of the relationship between the organism and the environment (strictly speaking, between the brain and the organs). The process simulated can then be represented, as far as

the signals are concerned, by a succession of states S_n^0 defined at time steps t_n (the superscript 0 means that these states are viewed from the organism side). The diagram for this final version is given in Figure 2.3.

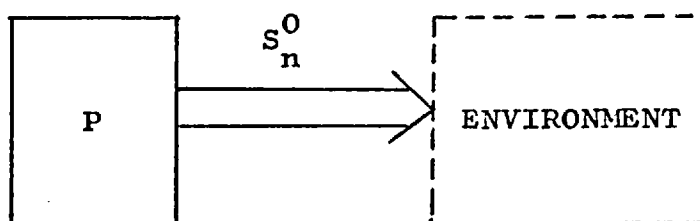


Figure 2.3. The final organism model.

Since the value of S_n^0 is a combination of two values which act in opposite directions, the direction of the arrow in Figure 2.3 is a matter of convention. We decide to point it towards the environment and the state S_n^0 should then be regarded as the action of the organism on the environment. But we must keep in mind that its value has two parts - the former input and output values - and we will have at times to consider them separately.

There is clearly a finite number of values a state may have, because these values are formed from the finite number of the original signals, each of which could only have a finite number of values. But for any actual applications this number must be very large, being in fact a compact coding of all possible relations between the organism and the environment.

The reductions described above have no effect on the processor P. In fact we cannot make any changes in the

processor at this stage, because all we know about it is its role, having left all the details for the implementation stage. The one implied simplification is that it will have to handle now only states, rather than several input/output channels. Its only contact with the environment is therefore in the form of a sequence of states and this surely will be reflected in its internal mechanism. Also, the range for the values of these states, as well as the number of different states that may exist, will have to be restricted according to the time and storage requirements they will pose on the processor, when implemented.

*

We will examine now the model for the environment. We are only considering here the environment from an external viewpoint, as we did for the organism. The model therefore will be a black box, with input and output channels, these being its only contact with the organism. The environment model must also have a processor P whose role is similar to that of the organism: to act on the signals transmitted by the channels so that the net result, when seen from the side of the organism, will be a simulation of the process to which it has to adapt, i.e. to exhibit intelligent behaviour as defined earlier. The functions of the two processors are different, but the difference will only be reflected in their implementation and it is not apparent

at this stage. The diagram for the environment model will be similar to that of the organism and is shown in Figure 2.4.

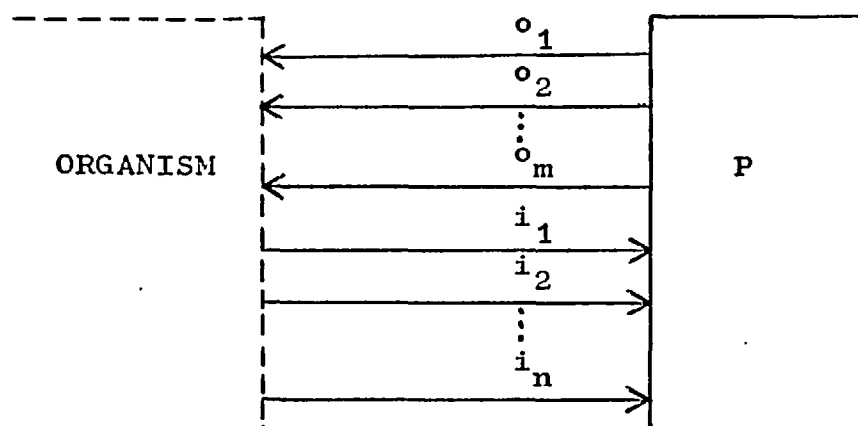


Figure 2.4. The basic environment model.

There are m output and n input channels, which correspond on a one to one basis to the m input and n output channels of the organism model. It is important to realize that, although these channels are in a one to one correspondence, we have to define the two models without the requirement that at each time step the corresponding signals are identical. This may seem strange and the explanation is as follows. If all the corresponding signals are equal by definition, it means that the organism fits perfectly in its environment by definition. But we require a process of adaptation and this is incompatible with permanent equality. Apart from this, if the signals were equal it would be superfluous to discuss both models, or to have channels of communication between them.

The definitions for the input and output signals have to be in accordance with their functions in the natural system. For the organism it is obvious and has already been stated as the signals the organism receives from and

sends to the environment. For the environment the definition for the output and input signals will be respectively, as those sent to the organism and those expected from the organism if a perfect adjustment is to exist. This will be discussed in more detail in the next section.

The model of the environment can be reduced in a similar way to that of the organism, but with the channels considered now from the side of the environment. The final version is given in Figure 2.5.

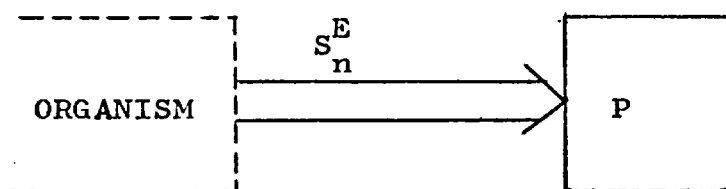


Figure 2.5. The final environment model.

As a matter of convention the arrow points towards the environment as it does in Figure 2.3, but clearly the states S_n^E (the superscript E means that these are environment states) may not be identical to the corresponding states S_n^O , since they are a coding of the environment signals (which may not be equal to the organism signals, as explained above). The way the coding from the original signals to the final states is done must however be identical, so that the states correspond like the original channels did.

2.3 The system.

In this section we will combine the definition of intelligence and the two models we have developed. In

other words, we will find out what must be the behaviour of these models, when joined together, so that the definition of intelligence is fulfilled.

The natural organism and environment cannot exist separately. They are the result of an evolution process which they have undergone while constantly interacting. Our way of studying them separately was purely theoretical. We have assumed the minimum requirements for fulfilling their respective roles and now, if we combine them the resulting system must display features similar to those of the natural integrated system. The feature we are interested in is the tendency towards stability, intrinsic to all closed systems. This stability is achieved in the natural system by the organism's capability to adapt to its environment. If the organism cannot adapt, then it will not survive and the unstable system will cease to exist. Perfect adjustment is only essential to survival in the long term; in the short term there are bound to be transient periods of improper adjustment when sudden changes occur in the environment. If intelligence is the ability to cope with the environment in the most efficient way, this means that the organism should be able to make these situations as few and as short as possible. This is another way of saying that it should be able to keep the system as stable as possible.

Similarly, in our artificial system formed by joining the two models, the requirement for stability must be a requirement for the organism to adjust to the environment.

Recalling the definitions of the signals to and from the separate models, adjustment is achieved by the equality of the organism and environment states: the organism is perfectly adjusted at time step t_n if $S_n^O = S_n^E$. Clearly, for the adjustment concept to have any meaning, we have to assume that the two states occur simultaneously, i.e. the organism cannot "see" first the environment state and "copy" it for achieving equality - it has to predict it. The interpretation of intelligence in the artificial integrated system must therefore be: the ability of the organism to predict the environment states as accurately as possible. The qualification "accurately" must be considered in relation to the way the coding from the original signals to states is done. What is meant is that we require the signals to be as nearly equal as possible in as many corresponding channels as possible. The result of the signals comparison is not a yes-or-no answer but one of relative closeness and that for the states comparison must be the same. The signals can vary over wide ranges and we cannot require perfect identity, especially after their coding to states. The results of intelligence tests can therefore be only relative: an organism model is more intelligent than other if its states are closer (as explained above) to the environment states. Although the actual degree of closeness is relevant, we should not ask for identity in a real-world application. It must be emphasized that this is only a problem in practical applications of the system and, as we will see in Chapter 4, for our present purposes we can ignore it.

We have thus reduced the definition of intelligence to the ability of the organism to predict the environment states, and we make no restrictions on the way this prediction is to be done. In particular, it is not necessary that only the previous environment state S_{n-1}^E is used for determining S_n^E . The organism may well have to use the information gained several time steps before or the results of some internal processes. The only restriction is of course that it decides on the value of S_n^O before the time step t_n .

Also, we will only be interested in the kind of intelligence that uses only the information in the environment sequence (from time step t_1) for predicting its future states. Although not restricted by the definition of intelligence, we will not consider the case of innate behaviour, when the organism has built-in knowledge about the environment. It is possible to display perfectly intelligent behaviour by using only this kind of knowledge, which can contain whole sequences of environment states. This knowledge cannot be modified and clearly such a system can only be successful for simple processes, i.e. a limited number and combinations of states and no changes in the environment. In the lower species the innate knowledge forms a larger proportion than the acquired one and the environments they have to cope with are accordingly relatively simple and constant. They can be perfectly adjusted, displaying intelligent behaviour in the sense of our definition. But we will only consider the organism which displays intelligent behaviour through adjustment, starting with no knowledge at all at time step t_1 , and all its acquired knowledge being

modifiable. Such an organism is a simulation of the higher forms of life, particularly man, for which the innate behaviour is only a small proportion of the total intelligent behaviour they are displaying after being in contact with their environment. Our organism model must therefore be able to predict the environment states only by using the previous sequence - its "past experience".

How can we interpret the necessity for the organism to predict the environment states? For answering this question we have to consider the states as formed from an input part and an output part: the two values from which the value of the state is derived and which in their turn are a coding of the original input and output signals. Also we have to remember that when comparing the organism and environment states we are effectively comparing the corresponding one's input channels with the other's output channels, and vice versa, due to the agreed coding. As a result of this coding we are also comparing the combined organism input part with the combined environment output part, and vice versa.

In the previous section we have defined the input and output parts of the organism states as the signals it receives from, and the signals it sends to the environment, respectively; and the input and output parts of the environment states as the signals the environment expects from the organism for a perfect adjustment, and the signals it sends to it, respectively. The prediction of an environment state has therefore a twofold effect. Firstly, if the organism predicts accurately the environment's output

part (and makes its own state's input part equal to it), we can interpret this as an ability to foresee the next event likely to affect it and thus be prepared for this event and be affected in the most advantageous way. This is a passive action for the organism and does not include of course possible responses it may produce in the following time steps. Secondly, if the organism predicts accurately the environment input part (and makes its own state's output part equal to it), we can interpret this as an ability to foresee the next action which, if produced by it, will cause the best possible adjustment at that time step. This is the action which requires the least effort, and which affects the environment in the most advantageous way for the organism.

These interpretations have no practical value and are only given to show the similarity with the natural system we are simulating. In practice, after the value of the state is formed, no more reference needs to be made to the original signals, or to the input and output parts. The organism's processor sees the environment as states, and can only produce states.

There is one more thing worth mentioning in this context. We have assumed an unlimited number of input and output channels, and that all these signals can vary from one time step to the next, forming any possible combination. But in practice these changes are much more restricted and only in complex processes will the change from one state to the next be due to changes in more than

a few of the signals. In most processes a change is likely to occur only in the input or output part, or indeed only in one channel. In particular, the commonest case is when in one state the relevant change is in the input part and in the next state in the output part, this being the simulation of a stimulus-response situation. Both changes in this case are usually due to changes in one channel only, all the other signals being constant. Indeed, as we know from experiments with human subjects, it is difficult to cope with as few as two or three signals at the same time. But simulating the whole states and considering the resulting changes in the value of the states does not detract from the simplicity of the simulation or from the efficiency of its implementation: we can always reduce the number of different states allowed, and this is effectively a reduction in the number of channels and/or in the range of their signals.

*

We have decided that the organism and the environment must be regarded as one system whose main feature is the tendency towards stability. We also said that it is the responsibility of the organism to achieve this stability. But this statement needs a justification. For clearly, as far as the system as a whole is concerned, stability can be achieved by an adequate response of either the organism or the environment. In practical terms this

means that it does not matter which one is trying to predict the other one's states and equate its own states to them, as long as equality is achieved. It could be the organism, or the environment, or both at the same time.

We cannot dismiss the possibility of the environment displaying intelligent behaviour and adjusting itself to the organism as illogical and unworthy of consideration. For example, in the natural system the environment can be at times another organism which, in its turn, regards the first organism as its environment and tries to adapt to it.

We will assume however that only the organism can contribute to the stability of the system. The reason is that, although the environment does adjust to the organism in the natural system, this happens so much slower that we can safely ignore it. While the adjustment of the organism to the environment is a short-term process requiring fast changes in behaviour, the adjustment of the environment to the organism is unlikely to be noticed except over long periods of time and cannot affect the stability of the system. In fact these changes are usually so small that they can only affect the species as a whole and are insignificant during the life span of an individual organism. For example, if we consider the language as part of the human environment, it is the individual that has to learn it as it is, i.e. to adapt to the environment. Although the language itself tends to change and develop according to the needs of the people using it at various periods,

these changes are so slow that they are hardly noticed by the individual (the addition of some technical words to the vocabulary for instance).

And even when in the environment there are other organisms, we have to remember that the organism we are studying normally interacts with several others. Although at each such interaction the organism in the environment may try to adjust to the first one as much as this one tries to adjust to it, the same happens between all other pairs, so that the net result is a very slow mutual adaptation (for example the habits which are considered traditions in a human society).

Since we will use our system for simulating relatively short processes, we can decide that it is only the organism that has to adjust its behaviour to that of the environment and we can safely ignore the slow adjustment of the environment to the organism. When we simulate the sequences of environment states we will assume that, irrespective of its complexity, the environment behaviour is constant throughout the duration of the process to which the organism tries to adapt. The environment states can then be preset for a whole process, because they cannot be affected by the organism. It is important to see that this assumption does not imply that the environment cannot be acted upon by the organism. What does not change is the best such action at each time step, the one that the organism should make for perfect adjustment.

The usual cybernetical approach is to view the system organism-environment as a closed system which achieves its stability by means of a negative feedback process. It is then similar to any other control system, natural or artificial, not only in its external behaviour, i.e. tendency towards a stable state, but also in the way it achieves it. As we have already stated, this is not a coincidence: firstly, equilibrium is a basic requirement for all systems and one that could not achieve it would simply cease to exist; secondly, the only way equilibrium can be achieved in a closed system is by negative feedback. But from the previous discussion it is obvious that, although the system organism-environment can be described as a feedback control system, it does not achieve its stability with a simple feedback loop. In fact we can distinguish between three levels of behaviour in this system.

At the lowest level are those biological processes commonly mentioned in cybernetics as examples of the similarity with the man-made control systems ^{16,18}. These are the mechanisms which regulate the body temperature and hydration, the concentration of various chemicals, etc. - all these being known as homeostatic mechanisms. Homeostasis is the term used for stability in this context and these are the elementary processes needed for the organism's adjustment to its environment. The typical action is as follows: The value of a certain variable in the organism has to be kept within fixed limits. If some changes in the environment produce a dangerous deviation in this variable, the difference

between the safe and the actual value is sensed by the organism and used for changing other variables until the value of the first one is again within its safe limits. Several of these mechanisms may interact, and some may only act inside the organism (without contact with the environment), but homeostosis as a whole can be viewed as a typical feedback control system. Its main feature is a relatively fast response: a change in the environment produces a virtually immediate adjustment of the organism. We should also include in this category the more complex, slower mechanisms like the basic instincts and drives. To conclude, the first level of stability can be described as adaptation through built-in knowledge, and must exist in any species.

At the next level comes the adaptation through acquired knowledge. The stability achieved by these means is less important for the short-term survival problems, which are generally solved by the first level mechanisms. It is needed for an efficient and successful integration of the organism in its environment, i.e. the long-term survival problem, and is thus essential for the survival of the species to which the organism belongs. This mechanism - interpreted in our simulated system as the prediction of the environment sequence - has already been described, but here we must point out that its feedback control is different from the simple biological mechanisms. Apart from the fact that its response is very much slower (a necessary feature that helps it to ignore the temporary, trivial changes in the environment dealt with by the first level

mechanisms), the difference between the correct and the actual response is not fed back for its immediate use in correcting the organism's response. Instead, it is used for changing some internal variables towards a more accurate response next time the same situation is encountered. The negative feedback process is applied to each situation separately, so that the correct response to a particular one needs not be a function of the response to the immediately preceding ones.

This kind of adaptation is clearly superior to the first one, since it imposes less restrictions on the number of changes in the environment and on their extent. In fact, for the higher species - man in particular - there is virtually no limit. Such a level of adaptation cannot be reached directly by innate mechanisms which are always biologically limited; what is necessary then is an innate capability to achieve this kind of adaptation, like a general purpose system that is not designed for any particular application. A system having this quality is called an ultra-stable system, for emphasizing the fact that the stability it can achieve is of a higher level than that of the simple feedback control systems.

The third level of stability is that achieved by the adjustment of the environment to the organism. As we have explained, this process is so slow that it is insignificant in the study of the individual organism. It is never taken into account, except in studies on the evolution of a species.

For our simulation model we are only considering the second level of stability, that achieved through constant interaction between the organism and the environment. Although we have defined intelligence as the general ability of the organism to adjust, we ignore the contribution of the trivial processes of the first level. Since we are simulating the environment as well, we can do this simply by eliminating from the process that part to which the homeostatic mechanism would respond in the natural system.

Another point already mentioned but worth reemphasizing is that we are only simulating the brain. We are using the term "organism" throughout the discussion to designate that part of the system which interacts with the environment, but it must be understood that its intelligent behaviour is achieved by the brain only - the rest of the organism is simply viewed as the means by which the brain comes into contact with the environment. This is a purely theoretical approach, needed for isolating the organ whose behaviour we want to simulate; clearly the brain cannot exist without the organism and it would not be needed in the first place. This approach is also required if we intend to simulate a general system, and not a particular organism and environment. We can only consider those features which must exist in any intelligent organism and ignore the ones which would only help a particular organism adjust to a particular environment. The sense and motor organs - which we have called input and output channels in our model - are obviously specific to each type of organism and have evolved to help that organism

fit best into its environment. But if we only consider the signals transmitted by these channels, in the way they are seen by the brain (and reduce them to states, doing a similar operation on the environment side), then all the details specific to the interaction in a particular system are eliminated - without detracting from the requirements for an intelligent behaviour.

This approach is different from that of robotics where the problem of simulating specific tasks demands a specific system whose input/output channels cannot be ignored. An integrated cognitive system has to be able to observe its environment and to act on it, both in specific situations (requiring for example a TV camera and a mechanical arm). Such a system poses many technical problems. We have chosen to ignore these problems at the present stage and concentrate on the problem of intelligent behaviour, assuming that in a practical application the real-world environment will be simulated or easily converted to an internal representation.

CHAPTER 3

THE ENVIRONMENT

In Chapter 2 we started by defining the basic models for the organism and the environment and then we studied them as an integrated system. We saw that intelligence - the capability of the organism to adapt to its environment - when viewed as a feature of the system as a whole, is a capability to achieve equilibrium.

Our aim is to develop a model of the organism, capable of displaying intelligent behaviour. But since intelligence is defined in the context of the environment, it is impossible to develop the organism model by itself - we need a simulation of the environment as well. Although we are mainly interested in the organism, we will have to simulate and study the integrated system. We can then regard our goal as either making the organism adjust more efficiently to the environment, or increasing the stability of the system; in both cases we have to reach it by providing an adequate behaviour for the organism. As we have explained, the result must be the correct prediction of the environment states.

We will develop in this chapter the environment model, and in Chapter 4 the organism model. For each of them we will discuss first the simulation problems and then their implementation. The models were implemented on an IBM 360/65, using the F1/1 Optimizing Compiler. In Chapter 5 we will see the results obtained with the two models working as an integrated system.

3.1 Simulation

3.1.1 The problems.

The simulation of the environment is easier than that of the organism. Although we have shown that, when viewed from outside, the two models are similar, we will simulate their performance in different ways; it is easier to produce the environment than to learn to predict it. We concluded in Section 2.3 that the environment states cannot be influenced by the organism, therefore we can preset the whole sequence of states for which the intelligence of the organism model has to be tested. We will not consider at this stage the actual real-world situations that might produce this sequence but only the sequence itself. We can choose then the most convenient method for the simulation.

The fact that the organism cannot be studied separately has important implications. If intelligence means an ability to adjust to the environment, then an organism model may display intelligence towards one type of environment and yet be inadequate for another. In the definition of intelligence there is no requirement for an ability to cope with several different types of environment. This is not surprising of course when we remember that intelligence must be equivalent to the stability of the integrated system; it must be a feature of the organism and of the system at the same time and this can only be achieved by requiring the organism and the environment to match each other. An organism cannot be simply intelligent - it must be intelligent

for a particular environment.

How can we reconcile this with our original decision of simulating a general model of intelligence? We have assumed that all aspects of intelligence must have a common mechanism and that it is possible to design a model that can display intelligent behaviour regardless of the kind of environment it is presented with. The answer is that there are not different kinds of environment, but only different levels of complexity. Therefore we must think of an organism model as capable of displaying general intelligence as long as it can adapt efficiently to a particular environment, i.e. if the requirement for the stability of the system is fulfilled. The more complex the environment is, the more complex will have to be the organism model that can cope with it and therefore there must be different levels of complexity of the system. At each level the organism can be considered as perfectly intelligent according to our definition, and the system as a whole as perfectly stable.

This situation can be observed in nature, where the various species - organisms at various levels of complexity - live each in its particular environment. The complexity of the environment matches of course the complexity of the species, being usually a subset of the environment of a higher species. Since all these species have survived through the process of natural selection, we can assume that the system organism-environment in each case has

achieved the required stability, regardless of its level of complexity. Clearly intelligence is not a matter of complexity but only of efficient adjustment, although of course if a species is particularly successful it can evolve into a more complex one.

Our aim is to achieve a highly complex system - in the sense of environment complexity - because we will eventually require the organism model to display intelligence comparable to that of man. The best way is to start from a low level of complexity for the environment and the simplest model for the organism, and improve this model until we get the simplest version that can match the environment. On going to the next level of complexity we should expect this model to fail; we will then further improve it, or replace it, so that we get the simplest version for the new level. We can continue this process until we are satisfied with the complexity of the system, but we may have to stop before that because of time and storage restrictions imposed by the simulation.

We can also think of this development process as follows. Our aim is to increase the level of complexity of the system organism-environment, while keeping its basic feature - stability. Since this feature is a result of the interaction between the organism and the environment, we must change one of them at a time. We are mainly interested in the model of the organism, therefore we will consider the level of complexity of the environment as a parameter and try, for different values of this parameter,

to achieve a stable system by changing the level of complexity of the organism.

*

We have already decided to simulate the environment states directly, without considering any particular configuration of input and output channels that might produce them in a real-world situation. Bearing in mind that the simulation will be implemented on a general purpose computer, we decide that the most convenient way to represent the states is simply by decimal numbers. The reasons are: they are convenient for both internal (in a binary form) and external representation and handling, they are easy to compare (for the stage of the simulation of the whole system) and there is practically no limit to the number of different states that can be easily represented even in a high level language.

It is easy to see that the decimal number representation does not limit the generality of the situations simulated by these states. We have shown that the states are a coded representation of the signals in the original channels and that we can in fact regard them as combinations of digits of a large radix. It is obviously possible then to represent these numbers as combinations of decimal digits, again by a one-to-one, unambiguous mapping. It is important to see that the resulting decimal numbers may not be sequentially arranged in a certain range, because they are the final result of several stages of coding starting from

the values and identities of all the input/output channels. It is better then to imagine the states simply as patterns formed with a certain number of decimal digits and probably using much less than all the possible combinations. For any practical application, the number of states for a high level of complexity must be very large, since they must represent all the possible situations between the environment and the organism. In the examples below we will use letters to represent the states and it must be remembered that each letter stands for a certain pattern of decimal digits.

In the real world there is not a continuous interaction process between the organism and the environment. There are periods of interaction, of varying length, interspaced with periods in which there are no signals in any of the channels. By this we do not mean that the value of the signals is necessarily zero, but rather that they have no relevance to the organism at that instance. During these periods the organism may not be inactive - there may be changes in its internal structure - but since there is no interaction with the environment we will call these periods "zero-states".

It has already been mentioned that the time steps at which the changes from one state to another occur do not have a fixed value in the real time scale. They are geared to the specific process being simulated so that only significant changes appear in the sequence of states. These states represent therefore the division of the process into its smallest logical sequential time units. Similarly,

the zero-states will represent in this sequence the logical periods of lack of interaction with the organism. The length of these periods varies and their meaning depends on this length. They have a special significance in the environment sequence and the organism model has to interpret them correctly. The zero-states are a convenient simulation of the real-world situation, where the organism has developed special ways to interpret the meaningful periods of lack of interaction (for example by the decrease in time of the excitation caused by a previous interaction). Since we are only simulating the effects of the natural process and not its mechanisms, we are justified in choosing the zero-states method for these periods and later designing the organism model to recognize their meaning. It will be a coding "understood" by both the organism and the environment models but internal to their system - like the real process which is internal to the natural system.

Rather than having different states for the different possible lengths of zero-states, we prefer to have only one - the number 0 (equivalent to the all-zero pattern of digits) - and to use it in strings of one or more according to the length of time being simulated (this will be explained when the levels of environment are defined). These 0's will appear then in the sequence and from the environment point of view they will be states like all the others. The organism of course will interpret them in a special way. In the examples below, for clarity, we will use hyphens (-) for the zero-states.

It can be argued that, since these zero-states do not mean lack of signals in the environment but only the fact that they are ignored by the organism, if we omit these signals in the simulation then we will not obtain the true complexity of the environment. The answer is that since these signals are ignored as irrelevant by the real organism, they cannot be considered as part of the environment we have to simulate. The organism and the environment must match each other, therefore if we include these signals in our simulation then the simulated organism will have to cope with them for achieving perfect adjustment; then we will not be simulating the real organism, but a more complex one (there is obviously no point in including these signals and then designing the organism to ignore them). In other words, if the real organism can only cope with a somewhat simplified version of the real environment it simply means that it is not complex enough to cope efficiently with all of it. The whole environment is too complex for it in the same sense that higher level environments would be too complex.

Why then define as the environment of the organism at each level of complexity, that environment which is slightly too complex for it and not that subset with which the organism actually can cope? There would be no zero-states then and there will be a continuous interaction, with significant signals in all channels, at all times. The reason is that the environment of each organism in the natural systems is always at a somewhat higher level of

complexity than the organism itself. The organism tries to adapt to it by its reduced means, but the adjustment can never attain absolute perfection. The system is always slightly unstable and the necessity to close this gap is the drive that leads to evolution. We will see later the close relationship between the level of complexity of the system and the zero-state strings in the environment sequence.

*

There are two features basic to any environment in the real world. They can be described as repetitiveness and a tendency to oppose this repetitiveness. Together they decide the level of complexity of the environment and they are the main factors we have to consider in our simulation. The recurrence of past situations in a process makes the environment predictable for the organism, whereas its irregularity and uncertainty compel the organism to constantly update its knowledge. We may be tempted to see this in one of two ways: either the environment is basically highly regular and predictable and any uncertainty there is should be considered as an exception; or it is basically random and the regular patterns in it are the exceptions. Clearly neither view is correct and both features are necessary for increasing the flexibility and efficiency of the system organism-environment. While some repetitiveness

is essential if the organism is to be able to learn from past experience, a constant variation in these patterns is essential for the evolution of the system towards more complex levels.

Without some repetitiveness in the environment sequence of states, it would be impossible for the organism to predict these states with a better than random accuracy. We have decided to ignore the innate knowledge since it only has a small influence in the higher organisms we want to simulate. We require that all the information the organism has on the environment comes from its previous interaction with it and the prediction of each state must therefore be a function of this information. If there is no repetitiveness in the sequence of states then any previous knowledge the organism had acquired, no matter how accurate and complete, is useless.

In terms of information theory, there must be redundancy in the environment. The information contained in a sequence of states must be less than maximal or, in other words, the uncertainty for the occurrence of a certain state at each time step must be as small as possible to enable the organism to predict it. The more information there is in the environment, the more complex must the organism be for adjusting to it.

Any simulated process in the environment is a sequence of states, including strings of zero-states of various lengths. At first glance it would seem that the level of complexity of this environment can be judged only from

the lack of repetitiveness that can be observed - the more repetitious, the less complex it must be. While this is true in a superficial sense, we must remember that it is not only the degree of repetitiveness that is important but also its level of complexity.

We have to clarify this point. The states represent the smallest significant unit in the environment sequence. The lowest level of repetitiveness can therefore be at the level of states - a certain state always being followed by another particular state, whenever it occurs in the sequence. But if we analyse a real environment we can see that repetitions occur at several levels. There are not only states following specific states but also certain groups of states which always follow other groups, and groups of groups of states behaving in a similar manner. In fact, for any process this repetitiveness must exist at all levels up to groups of states whose length is significant in relation to the length of the process itself. The reason is that if there are no situations in a latter part of the process, that resemble some situations in an earlier part, then the process can be viewed simply as several sequential processes which are either unrelated or trivial duplications of the same process (depending on the degree of repetitiveness at the lower levels).

The level of complexity of an environment is therefore related not only to its irregularity and lack of repetitiveness in general, but also to the level of this repetitiveness.

The complexity of the environment increases with the increase in this level and/or the changes in the pattern of the repetitions. In order to cope in full with its environment, the organism must identify its repetitiveness and also any changes in this repetitiveness.

It is important to see that our approach to the simulation of the organism satisfies both these requirements at the same time. Since the organism adjusts its behaviour, i.e. its prediction of the environment states, solely from its interaction with the environment (by comparing its predictions with the actual states), it makes no difference to it whether the inaccuracy in a certain prediction is caused because it is the first time that situation occurred or because of a change from a previous occurrence. The action it has to take in both cases is the same: to update its internal structure so that next time the same situation occurs the prediction will be more accurate. In cybernetical terms, the organism must ensure the stability of the system regardless of the source of disturbance.

3.1.2 Definition of levels and types.

Proceeding from the two basic features we have described we will define now the levels of complexity of the environment. The levels are numbered from 1 upwards and there are three types at each level.

The level 1 type 0 environment is the random sequence of states. This is not the simplest environment according

to the earlier arguments, since none of its states is predictable and it contains maximum information. We prefer however to list it first because this is its logical place, as it will become apparent later. An example of such a sequence is:

A E - B M A - F - N B F D -

We are using letters in these examples to represent states, and the hyphens stand for zero-states.

In all level 1 types of environment there are only strings of one zero-state and their significance is simply that there is no relation between the groups of states they separate. The length of time they stand for (and which needs not be the same for all zero-states) is always enough for avoiding any influence between the adjoining states (this is irrelevant for type 0 of course, where the sequence of states is random).

The level 1 type 1 environment is different from type 0 in that each particular state is always followed by the same state. For example:

C D E - B C - L M N O P - A B C - M -

In the first position in each group all states are equally likely to occur, but the rest must follow one another according to fixed rules. The relationship between states is a one-to-one connection: a certain state can only be followed by one certain other state and conversely, two different states cannot be followed by the same state.

The number of states in each group is random. The zero-states are relevant here, since they signify an

interruption in the chain of related states.

This is an easily predictable environment, because there is never a departure from the succession rules. The number of times each pair of succeeding states must occur and similar other problems depend on the mechanism of the organism model, but it is clear that each pair must occur at least once in the sequence for the rule to be identified, because the relationship between states cannot be inferred from any other connections. In other words, the succession rules apply purely at the level of individual states.

Because of the random factors, the organism can only predict with random accuracy the first state in each group and the time step when the zero-state occurs at the end of the group. These permanent uncertainties emphasize the fact that the repetitiveness at this level of environment is only for states and the groups themselves have no meaning.

The third type in level 1 is the type 2 environment. Type 2 is similar to type 1 but in addition there may be changes in the succession rules. If a state is followed by a certain other state in a particular instance, it does not mean any more that it must be followed by the same state in all other instances. Clearly the extreme case of this uncertainty produces the type 0 environment, but we are not interested in this case. Rather, we will consider the possible changes in their practical, real-world sense.

In the real environment the changes in repetitiveness are due either to momentary disturbances or to modifications in its structure. In the simulated environment the disturbances will appear as random departures from the succession rules, in any of their occurrences. This can be simulated by having several versions for each rule and choosing one of them at random when that rule is required. Since these changes must appear as disturbances only, one of these versions will be considered as the main one and in order to enable the organism to identify it, it must be used more frequently than the others. The second kind of changes - in the form of modifications in the structure of the environment - can be simulated by changing from time to time the versions for some of the rules. There must be of course several parameters for controlling the degree of uncertainty introduced by all these changes.

These are the three types for the level 1 environment. Their common feature is that the relevant connections (if any) can only be between succeeding individual states. This is also true for the changes in type 2, which only apply to the relations between states. In none of these types is there any relation between adjoining groups of states and the groups themselves have no significance. We will refer to these features as level 1 repetitiveness and level 1 groups.

At the next level of complexity is the level 2 environment. This environment contains information also at the level of groups of states, in addition to the information at the level of individual states. Specifically, at this level the groups of states behave in a way similar to the states at level 1, i.e. according to succession rules (except type 0), while the states themselves form random chains inside the groups. The level 1 information in this environment is thus maximal (no level 1 repetitiveness), and the level 2 information depends on the type of environment (0, 1 or 2 - corresponding to the types discussed earlier for level 1).

In level 2 the individual states serve as building units for the level 1 groups and this is their only meaning. They are only used as combinations for forming different patterns which identify the groups, similarly to the way the decimal digits are used for forming patterns for identifying the states themselves. The level 2 environment is thus a logical extension from level 1 and we must view the groups as patterns of information (the units being individual states), in the same way that the states are patterns of information (the units being in fact the original signals in the input/output channels, coded as decimal digits). The difference between them is that the states are instant space patterns (in the space formed by the environment's channels), and the groups are time patterns (or rather time and space patterns, if we regard them as formed from the basic, original signals).

In level 2 there are two kinds of groups and therefore

we will need two kinds of zero-state strings to separate them. We will use one zero-state to separate the level 1 groups (groups of states), for signifying that there is a relation between adjoining groups (except for type 0). And we will use two zero-states for separating the level 2 groups (groups of groups of states), for signifying that there is no relation between adjoining groups - similarly to the use of one zero-state between level 1 groups in a level 1 environment. In a real environment two zero-states would correspond to a longer period of lack of interaction with the organism, which results in the extinction of the influence from the previous group of interactions. It is worth reemphasizing here that the zero-state strings do not stand for certain fixed periods of time, but (like all the time steps) for logical periods, in the context of the process simulated.

Type 0 in level 2 is a sequence of random groups of states separated by both kinds of zero-state strings. Except for the strings of two zero-states, it is identical to type 0 at level 1, but we must view it as a level 2 construction, namely as a sequence of groups of states, rather than of states. The randomness of states within the groups is a feature common to all types at level 2, whereas in level 1 it is specific to type 0 and it is the randomness of the groups that is common to all types.

The type 1 environment corresponds to type 1 at level 1. It is a sequence of groups of groups of states; the

groups of states follow one another according to succession rules, and the groups of groups of states (level 2 groups) have no meaning (like the level 1 groups had no meaning in level 1). We refer to the groups of states as level 1 groups, irrespective of the level of the environment; to the groups of groups of states as level 2 groups, etc.

A simple level 2 type 1 sequence is the following:

KM - PR - X - N - PST -- KM - PR -- N - PST -- N --

The group KM is always followed by the group PR, and the (single state) group N is always followed by PST.

Inside these level 1 groups however, there are no rules: the state P is followed by the state R in the group PR, and by the state S in the group PST. The number of level 1 groups in each level 2 group is random, and so is the choice of the first level 1 group in each case. The others however must follow according to the fixed succession rules.

It is easier to follow the logic of the correspondence between the two levels of environment if we imagine replacing each particular level 1 group in the level 2 sequence with a unique identifying symbol. Then the type 1 example above would look like a level 1 type 1 sequence, with the new symbols having the role of states.

It is more difficult for an organism to adjust to a level 2 than to a level 1 environment. It is obviously useless now to try to find connections between adjoining states because there are none, and therefore the prediction of states must be based on a higher level understanding of the environment. Specifically, for predicting the state

at a particular time step the organism has to be aware of the time pattern itself, i.e. of the level 1 group that contains the required state. It has to be able to recognize from the sequence of states the succession rules between the level 1 groups. Once it had identified the group it should also be able to predict the correct sequence of states in the group, this sequence being unique. But even for a perfect organism there will still be the uncertainties due to the random factors: the number of level 1 groups and the first level 1 group in each level 2 group.

The type 2 environment must correspond to the type 2 environment in level 1, where the succession rules are not fixed. For level 2 the changes will apply at the level of groups of states, since only at this level there are succession rules. But the way these changes are simulated is identical to level 1. The reason is that the nature of these changes in the real environment is similar for all levels: random disturbances or modifications in the environment structure.

Like in level 1, it is this type of environment that is the most demanding for the organism's capability of adjustment. But here the organism's problems are at a higher level since for each change it has to identify not a new rule applying just between two states, but between two groups of states. This means that the change in the amount of information contained in that sequence is larger, and so is the uncertainty.

It is important to see that no such changes occur at the level of states in the level 2 environment. While it is true that changing a whole level 1 group implies a change in the states of this group, there are no succession rules for the states before the change and it is irrelevant to consider the new states as derived from the previous ones. In a level 2 environment the states are only units for forming the group patterns and the relevance of their sequence in a particular group is limited to the identification of that group.

*

The extension of this logic to higher levels of the environment is straightforward. Each level has three types: type 0 is a random sequence of patterns, in type 1 they follow fixed succession rules and in type 2 there are changeable succession rules. These rules always apply only to the patterns which are characteristic for the particular level: in level 1 the individual states, in level 2 the groups of states, in level 3 the groups of groups of states, and so on. The zero-state strings between the various kinds of groups must be adequate for an organism to be able to identify the hierarchy of patterns: it should be able to distinguish whether there is a connection between two adjoining groups of states, and what is the level of this connection. From each level, the next higher one is obtained by using its characteristic patterns as

units for building the patterns of the new level.

It is interesting to see how the growth in complexity when going from one level to another ensures a growth in the amount of information contained in the environment. What effectively happens is that the level at which repetitiveness occurs is pushed higher in each new level. The groups which at a certain level are meaningless will be the characteristic groups for the next level, the ones whose sequence is controlled by succession rules. At the same time the characteristic groups of the previous level lose their repetitiveness and their sequence becomes random. Since repetitiveness means predictability, i.e. a small amount of information, each jump to a new level liberates a new group level from this constraint and turns it into an information - containing unit by making its occurrence less predictable (or rather its predictability more complex). In practice this is achieved by allowing it to occur in more contexts, not only following another specific unit, and thus increasing the number of possible combinations for the units which form the characteristic patterns of the new level.

A specific process in a real-world environment must correspond to a specific level of complexity in the simulated environment. Since the adjustment to any environment is based on its repetitiveness, and since the level which contains this repetitiveness increases with the level of complexity of the environment, it is obvious that a complex real-world process will eventually require a complex organism

model for coping with it.

3.2 Implementation

In this section we will discuss the implementation of a generator for producing the simulated environment we have defined. The requirements from such a generator are to produce an environment with the precise features of a specified level and type, and also to be flexible so that many sequences of states with different characteristics (but of the same type and level) can be easily produced. The first requirement is needed because we intend to develop the system in stages, with an organism model to match each level of complexity of the environment. The second requirement is needed because one of the ways for developing the organism model at each level is by testing it against several possible environment sequences of that level. Flexible environment generators for the various levels and types will also be needed at a later stage, when exact simulations of real processes will be required.

At each level type 1 can be considered as a special case of type 2 (zero-valued changes in the succession rules). Type 0 (random sequence) is useless since no organism can adapt to it anyway. Therefore it would be enough to implement the type 2 generator, but we found it helpful to implement first the others because each type is derived from the previous one. We will follow the same logic in the discussion.

3.2.1 The environment parameters.

The sequence of time steps is composed of alternate strings of states and zero-states. As explained earlier, we use combinations of decimal digits for the states and the number 0 (the all-zero pattern) is reserved for the zero-state. The decimal digits are thus the basic element of the time pattern for level 1, and the characteristic pattern is the state. At level 2 the basic element is the state and the characteristic pattern is the group of states, and so on for higher levels. The characteristic patterns at each level are those patterns to which succession rules apply and the basic elements are those patterns from which the characteristic patterns are directly formed (the next lower level patterns). The basic patterns at each level are in their turn formed in a similar way from lower level patterns (down to digits) but no succession rules apply to any of these and therefore they are not relevant for that level of environment merely serving as building blocks for the characteristic patterns.

For development purposes all these patterns can be chosen by the program itself, by using a random generating process to ensure dissimilarity between different sequences. But we must impose on the program some restrictions for the production of the patterns and this is done by specifying two parameters for each level, from states up to characteristic patterns. At each level (of patterns) L the first parameter is for the maximum number of patterns from the previous

level that can be used for forming each pattern at level L ; the second is for the number of different patterns at level L that have to be produced (and which will be used for forming the patterns at the next higher level - up to characteristic patterns).

For the level 1 environment there is only one pair of parameters, because the states are also the characteristic patterns: N_1 will be the maximum number of decimal digits in each state and S_1 the number of different states in the environment. For a maximum of 3 digits for example, there could be up to 999 states but in practice S_1 is usually significantly less (the states are just patterns formed with the digits and have no meaning as decimal numbers). For a level 2 environment N_1 and S_1 will be as for level 1, and in addition N_2 will be the maximum number of states in each group and S_2 the number of different such groups. There can be groups of 1, 2, ... N_2 states each (and there can be, like in level 1, states formed from 1, 2, ... N_1 digits); and the number of different such groups (S_2) will usually be a small proportion of the total number possible. Similarly, for level 3 there will be 3 pairs: $N_1, S_1, N_2, S_2, N_3, S_3$; and so on for higher levels. By using these parameters we can easily change the complexity of the environment while keeping its level and type constant.

As explained in Section 3.1. the structure of the environment is as follows: a group containing a random number of characteristic patterns followed by an adequate zero-state string (to signify lack of connection to the

next group) - and this cycle repeated. The characteristic patterns themselves are separated by shorter zero-state strings, which signify a relation between these patterns but are long enough to signify lack of connection between the adjoining patterns of the next lower level; and so on down to the zero-states between groups of states. In practice we set a limit to the number of cycles to be produced (each cycle contains a major group - formed from characteristic patterns - and the corresponding zero-state string). This is done by the parameter T.

For a level 1 environment there is only one kind of zero-state strings, since the states are also the characteristic patterns. At level 2 there are two kinds - those between groups of states (which are related through succession rules) and those between groups of groups of states (which are unrelated, each one being a random-length group of characteristic patterns). As an example, the structure of a level 3 sequence (ignoring details) would be like this:

--- [(XX-XXX-X)--(XX-XX)--(XXX-X)] --- [(XX)--(XX-XXX)] ---

The brackets are obviously for illustration purposes only. The individual states are designated by X and there are three kinds of zero-state strings: one zero-state between groups of states (level 1 groups), two zero-states between level 2 groups, and three between level 3 groups. The succession rules (if any) would apply between level 2 groups (which are the characteristic patterns), and there is a random number of such groups in each level 3 group.

The states themselves are only used as elements for building the level 1 groups, which in turn are only relevant as elements of the level 2 groups.

Although the number of characteristic patterns in a major group is random, we have to specify an upper bound for it because it influences the complexity of the environment produced; this is done by the parameter M . The program chooses in each cycle a random number X ($0 < X < M$), and produces then X characteristic patterns for that cycle: the first one at random and the others according to succession rules (if any) and the changes in these rules (if any).

There are four other parameters which are needed for controlling the changes in the succession rules; therefore they only apply to type 2 environment generators. The first one (V) is the number of different versions for each rule. The second (P) is the proportion of usage of the main version out of all versions of a rule. The third (W) is the number of cycles elapsed between consecutive modifications in the general structure of the rules. The fourth (C) is the number of rules to be changed each time. These parameters will be discussed later, when the implementation of the type 2 environment generator is described.

*

There are several cases when the environment generator has to make a random choice. They belong to two categories:

either this is required in the definition of the environment, i.e. for our purposes the corresponding real-world environment is also virtually random in relation to that feature; or a specific value would be irrelevant for the uses intended and it is easier to choose a random one (with certain constraints).

To the first category belong (apart from the random choices of type 0) the following: the choice of the first characteristic pattern and the number of such patterns in each major group, and the control (in type 2) of the changes. To the second category belong the choice of the S_i patterns from the total possible with up to N_i patterns of the lower level (for all pairs needed, according to the level of environment), and also the choice of the succession rules. Clearly all the patterns from the characteristic ones down to states are meaningless combinations for our purposes (they are only relevant in a practical application) and a random choice is sufficient as long as the program treats them after they are chosen as a fixed, unchangeable set. We let the program choose them because it is simpler than specifying them for each case. We only need to specify sets of two parameters and let the program do the rest - this also has the advantage that we only need to change the parameters for obtaining a different complexity of patterns. This procedure is used in all versions, for the patterns at all levels. Also, the succession rules are only relevant when the environment has a specific

meaning, and for our purposes we can have any rules.

In the implementation we have used the power residue method¹⁹ for generating random decimal numbers which were subsequently transformed as necessary for each one of the random processes in the program. The advantages of this method are its simplicity (only two operations are needed for each random number generated, including the truncation to the number of digits required), and the good randomness produced. From the 15 digit numbers only a few of the most significant digits are used, according to each case.

3.2.2 The programs.

We will describe now the implementation of the various versions of the environment generator and show how each is derived from the previous one in ascending order of type and level. As we have mentioned earlier, although the type 0 programs are never needed (no organism model can be meaningfully matched with a random environment), and although the type 1 environment can be produced with the type 2 programs (by specifying no changes), we found it useful to develop them in their logical sequence since each one is effectively an extension of its predecessor.

Level 1 type 0: First the parameters N_1 and S_1 are used for choosing S_1 random different states from all the possible combinations of up to N_1 decimal digits (except for the all-zero combination of course, reserved for the zero-state). The states chosen are stored in a table (in

a numeric representation, this being the most convenient way). The program then starts producing the environment sequence, working in cycles; each cycle consists of a random number (from 1 to M) of time steps containing a state (chosen at random from the table) this group being followed by a zero-state. The sequence is ended after T cycles. The flowchart of this program is given in Figure 3.1.

Level 1 type 1: This program is derived from the type 0 program by providing succession rules for the states. All the parameters are used in the same way, but it is more convenient now to store the states in a circular list rather than a table (this is a list in which each entry points to the next one and the last one points to the first, so that they are linked as an endless chain). The order of the states in this list is chosen at random, and the succession rules are thus random - but they are kept unchanged throughout the production of the sequence. The first state in each cycle is chosen at random from the list and the others are found by following the pointers, the total number in cycle being as before random between 1 and M. There is no difference in the production of the zero-states. The flowchart of this program is given in Figure 3.2.

Level 1 type 2: This program has to produce changes in the succession rules, as specified by the definition of type 2 environment. The program is derived from the type 1 program as follows.

At the beginning the program creates a circular list of states similar to the type 1 list, but here each entry

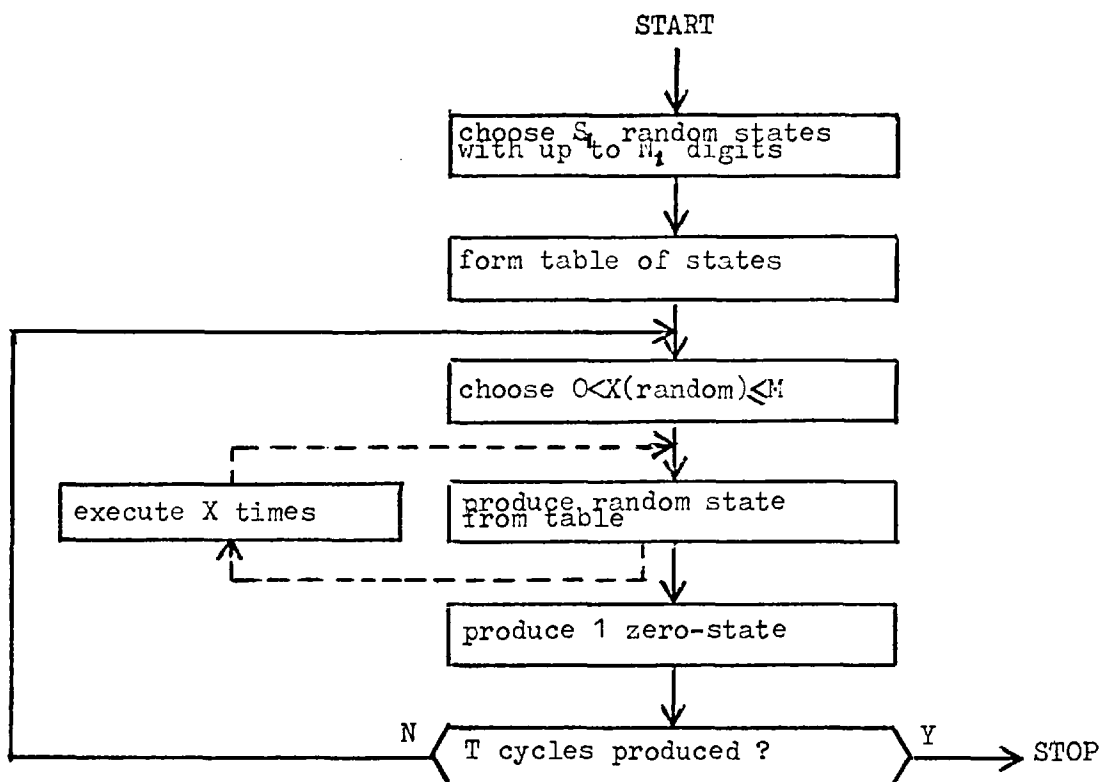


Figure 3.1. Level 1 type 0 environment generator.

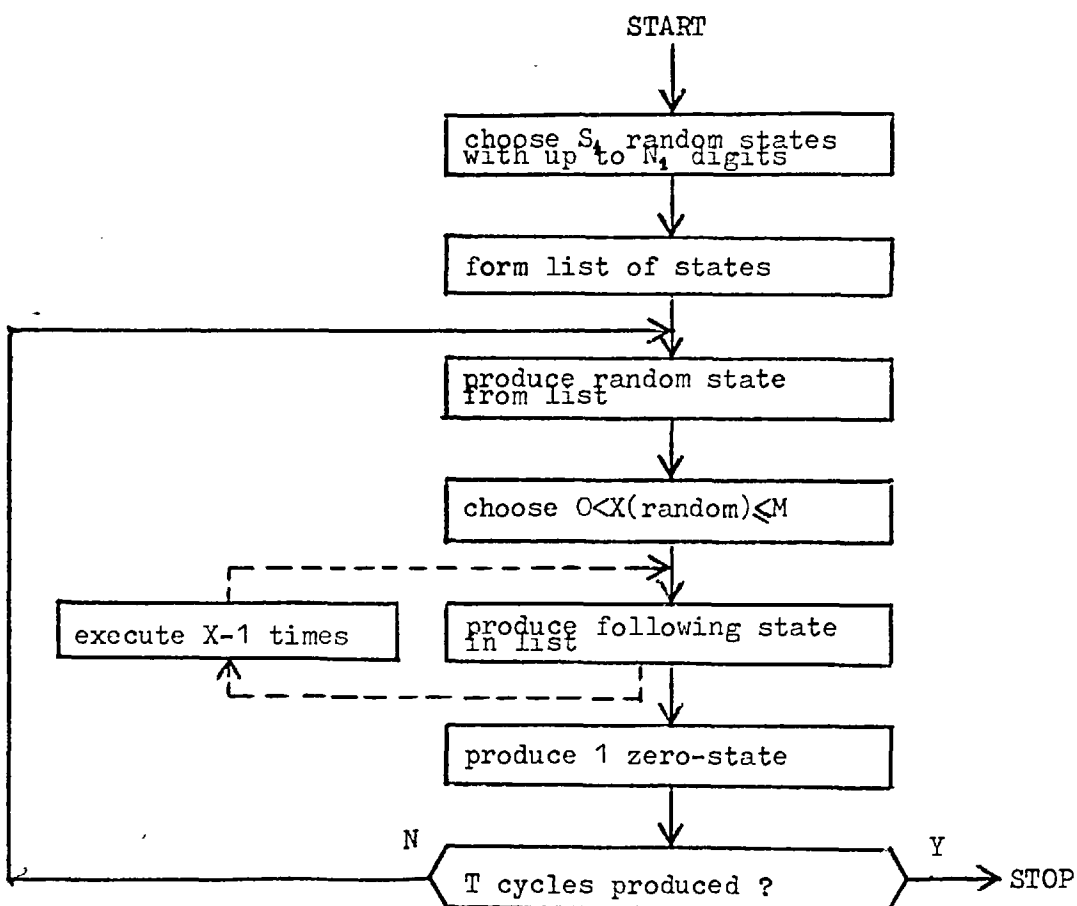


Figure 3.2. Level 1 type 1 environment generator.

has V pointers. The program uses these pointers for linking the states in V different (random) ways, effectively creating V different versions for the succession rules. (It may happen that for some of the rules some of the versions will be identical, but this does not matter, since V can be considered as an upper bound). The program then associates with the first pointer of each entry the proportion P , and divides the remainder to 1.0 randomly among the other pointers (the second pointer will have the same proportion for all entries, and so will the third, etc.). The pointers in the first position will form thus the list containing the main versions, but they are not used in any special way. The program chooses which pointer to use at each time step at random - but according to their proportions. It produces then the state to which that pointer is linked; there it chooses the next pointer, and so on. Since some of the versions may be identical (more than one pointer from a state pointing to the same state), the proportion for such a version will be larger than intended. This is not a problem because for the main version the proportion P can be considered as a lower bound, and for the others the proportions are normally much smaller than P , so that there is no danger that they will surpass the proportion of the main version.

This is the simulation of the first kind of changes - the random disturbances. The simulation of the modifications in the structure of the environment is done as follows.

The structure described above is effective for W cycles, after which the program rearranges the pointers of C entries in a random way. The proportions associated with the V pointers remain unchanged. The rearrangement produces V new lists, each of which has C rules (the same in all lists) different from the previous one (all the versions being different). This new structure is effective for another interval of W cycles. The process is repeated until T cycles had been completed.

If we choose $V = 1$ (and $P = 1.0$) and $C = 0$ (or $W > T$), then obviously no changes will be produced: the sequence created will be a type 1 sequence. The type 2 program can thus be considered as the general version for the level 1 environment generator. Its flowchart is given in Figure 3.3.

*

Level 2 type 0: This environment can be superficially viewed as similar to level 1 type 0 (apart from the two kinds of zero-strings). We prefer to produce this environment in the context of level 2 because it is the simplest sequence at this level and its program is useful for solving the implementation problems characteristic of level 2 before the problems of the more complex types 1 and 2.

The level 2 programs first use the parameters N_1 and S_1 as in level 1 for choosing S_1 random states with up to

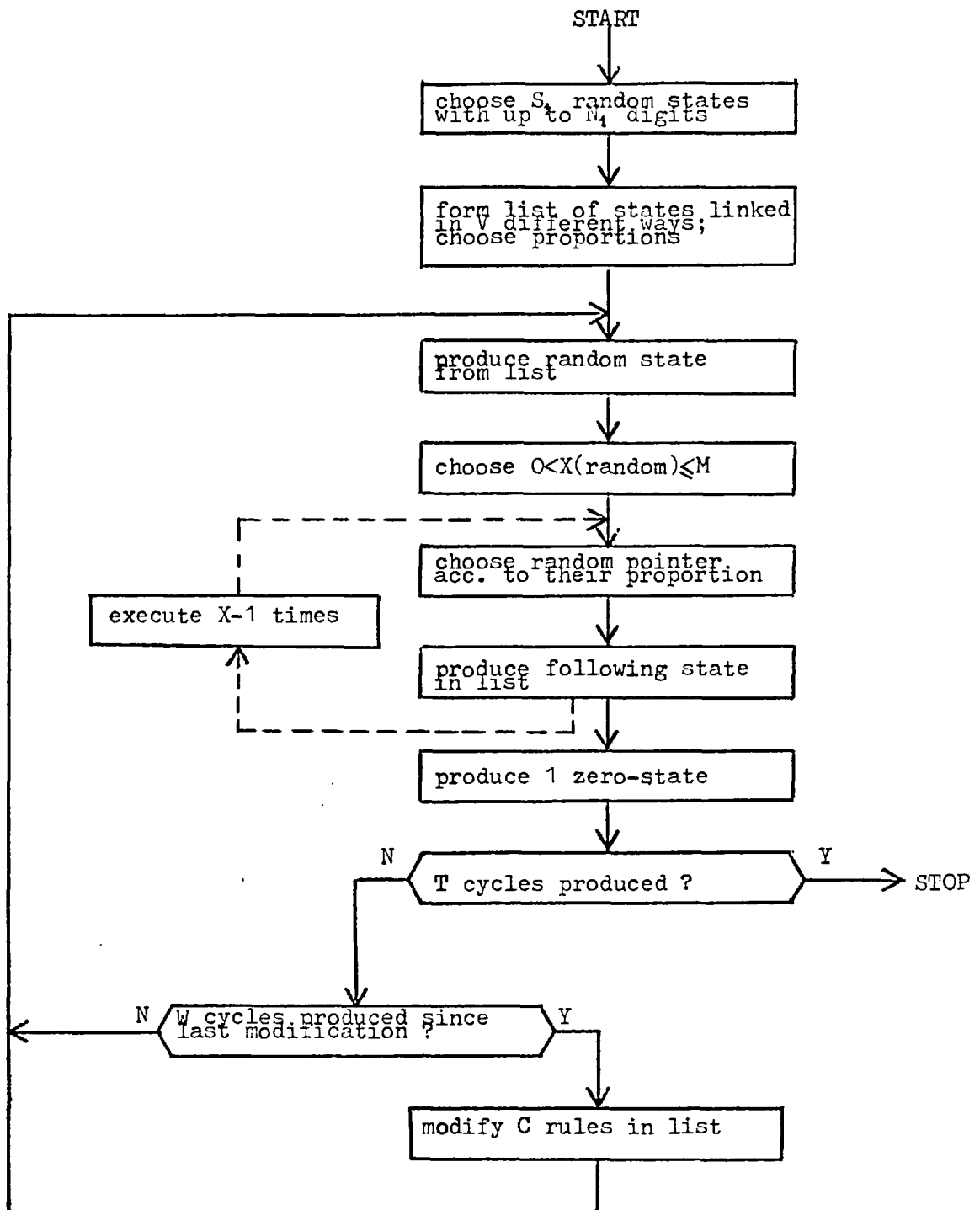


Figure 3.3. Level 1 type 2 environment generator.

N_1 digits each. But since there is another level of patterns, the programs need a second set of parameters - N_2 and S_2 - for choosing S_2 random, different groups of states, each one containing up to N_2 states (from the S_1 states chosen before). The process for the second choice is similar to the first, the states being used as building units for the groups as the digits were for the states. The original states are not needed any more for the production of the sequence, their task being finished: the production of the characteristic patterns of this level. The chosen groups of states are stored in a table like the states are in level 1 type 0.

We found that the most convenient internal representation for the groups of states is by converting the decimal numbers which represent the states to character strings and joining these strings together (in the order of the states in the group). The resulting string is used as an entry in the table. The same method was also used for higher levels (in several stages). In the character string representation the patterns at all levels are converted into the same length (N_1 for states, N_2 for groups of states, etc.) by padding them with blanks on the left, for facilitating the unpacking process when the environment sequence is produced.

The parameter M is used as in level 1, but here of course it is for the number of level 2 patterns, these being the patterns which form the major groups.

For type 0, after forming the table of groups of states

the program produces the environment sequence in cycles. The main difference from level 1 is that it produces now groups of states rather than states; each group is chosen at random from the table and unpacked into its states, these being output in their proper order. Between the groups of states (still inside the major group) the program produces one zero-state, and at the end of the cycle two zero-states. The process is ended at the end of T cycles. The flowchart for this program is given in Figure 3.4.

Level 2 type 1: This program is derived from type 0 in the same way it was derived in level 1, i.e. by replacing the table with a circular list. The entries in this list are now the character string representation of the groups of states, these being the patterns to which the succession rules apply. All the parameters are used in the same way as in type 0. The flowchart of this program is given in Figure 3.5.

Level 2 type 2: The logic of this program is similar to that of the type 2 program in level 1 and the program is derived from the type 1 program in the same way. The changes now apply to the list of groups of states, but apart from the representation by character strings rather than numbers, this list is similar to level 1. This is the general environment generator for level 2, and it can produce a type 1 sequence if no changes are specified. Its flowchart is given in Figure 3.6.

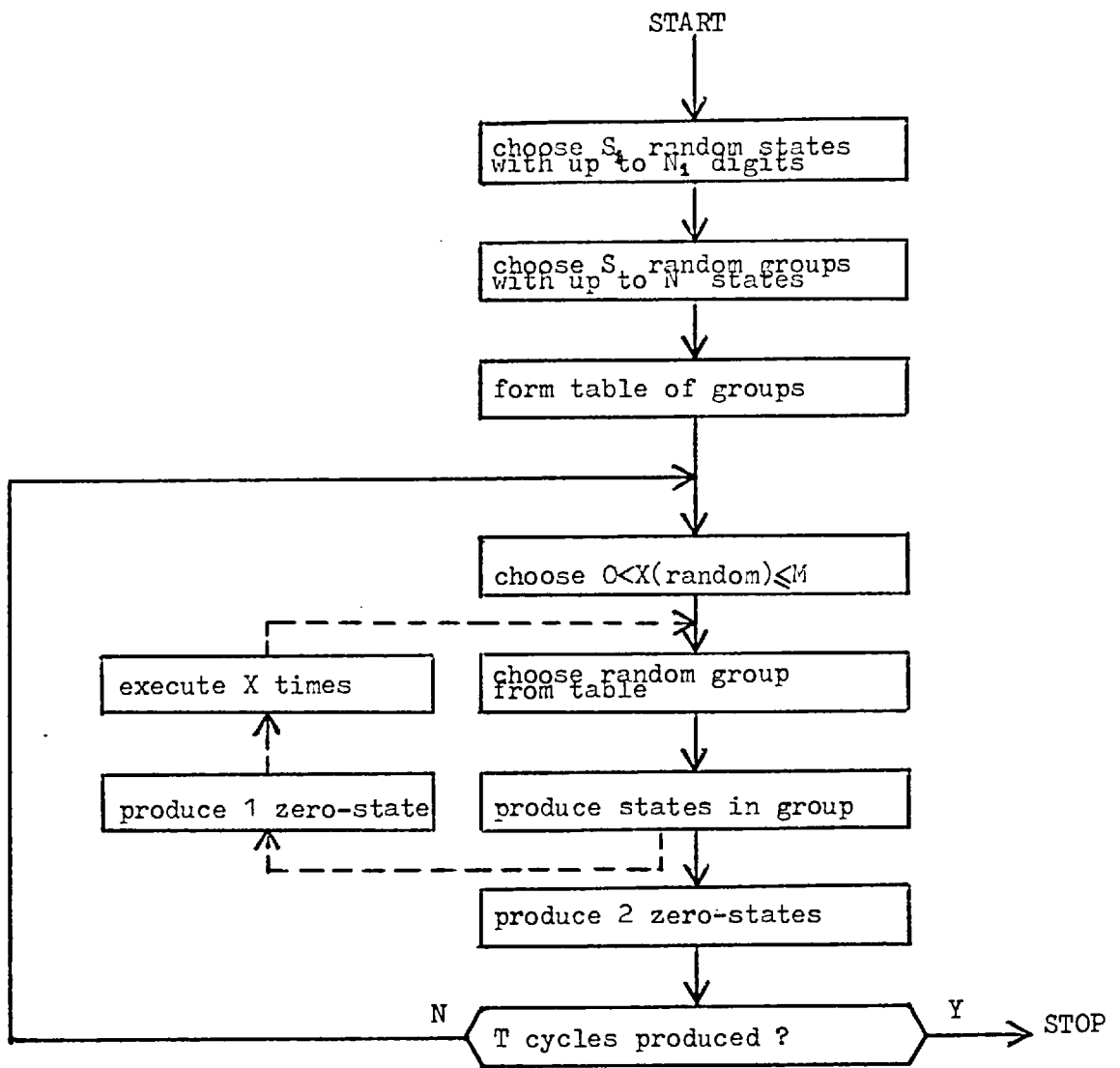


Figure 3.4. Level 2 type 0 environment generator.

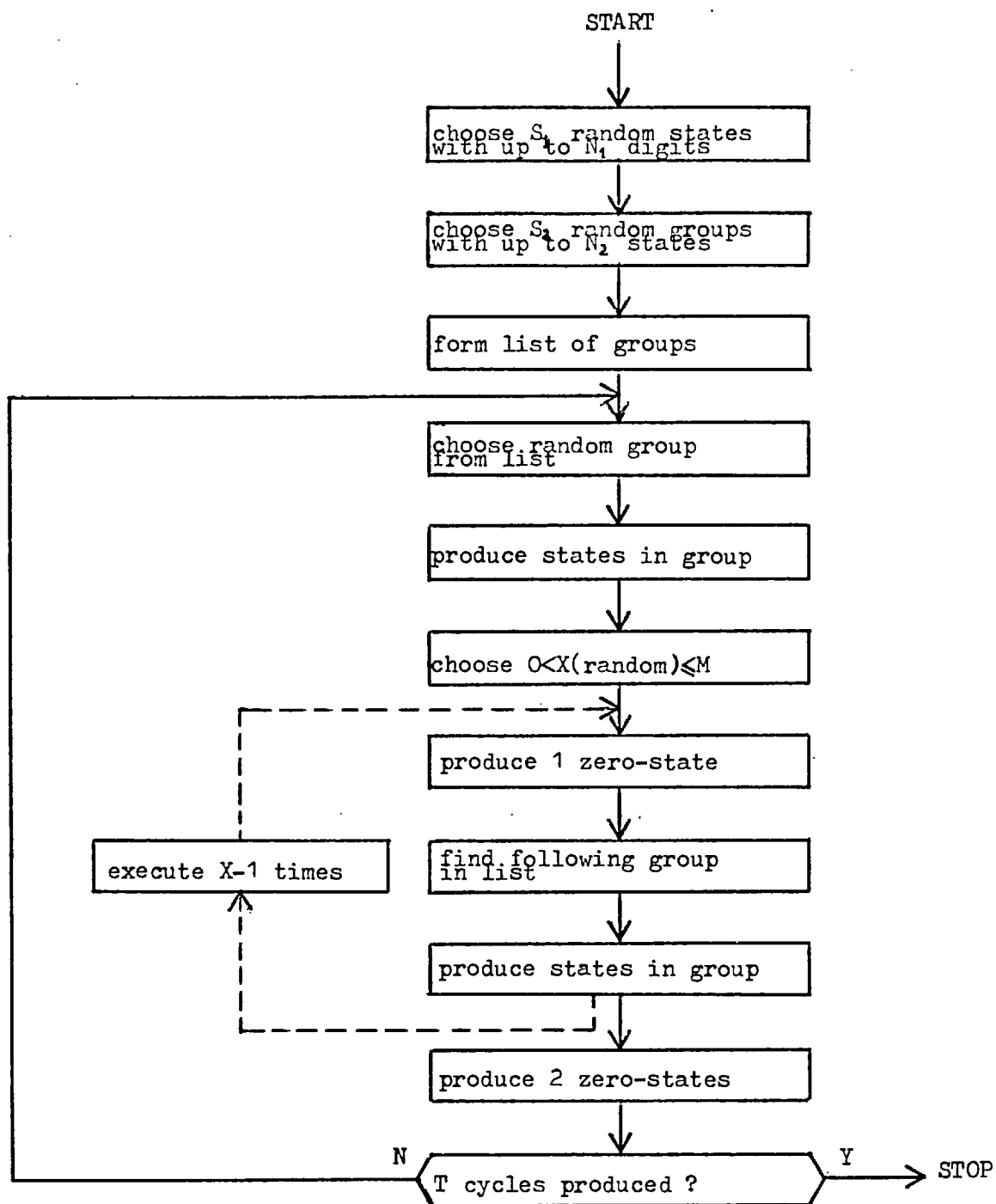


Figure 3.5. Level 2 type 1 environment generator.

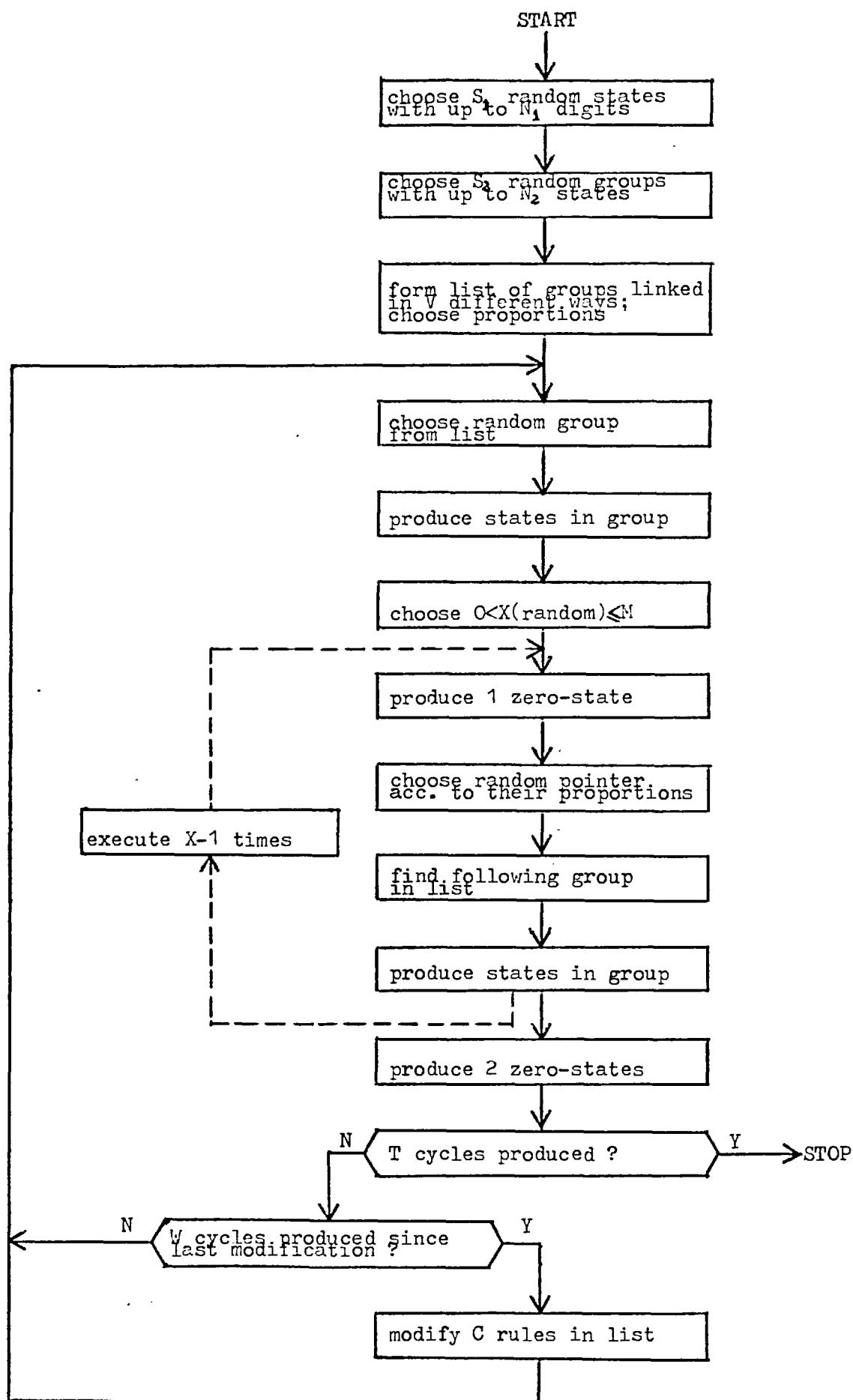


Figure 3.6. Level 2 type 2 environment generator.

The extension to environment generators of higher level is straightforward. The representation of the characteristic patterns is always as a character string, using the parameter sets S/N for choosing the patterns from states up to characteristic patterns; these are finally stored in a list (or table for type 0) and all the intermediate ones are discarded. When an entry from this list has to be used it is unpacked in stages into its lower level patterns, down to states. Between the various groups zero-state strings are inserted according to the level of the groups. The parameter M always applies to the same level of groups to which the succession rules apply - the characteristic patterns.

The different levels were only implemented as needed, after the organism model for the lower levels had been developed. We only reached level 3 before encountering serious restrictions in the simulation of the integrated system. The level 3 type 2 flowchart is given in Figure 3.7.

It is possible of course to have a general program that can produce environments of any level. The level required would be specified as a parameter and the program would decide, according to this level, the number of levels of patterns to be produced and all the other processes which are basically similar but repeated at several levels as a function of the environment level. We decided however against such a program because we never needed more than three levels and the complexity of the task was not justified. It would have been also against our approach, which is the

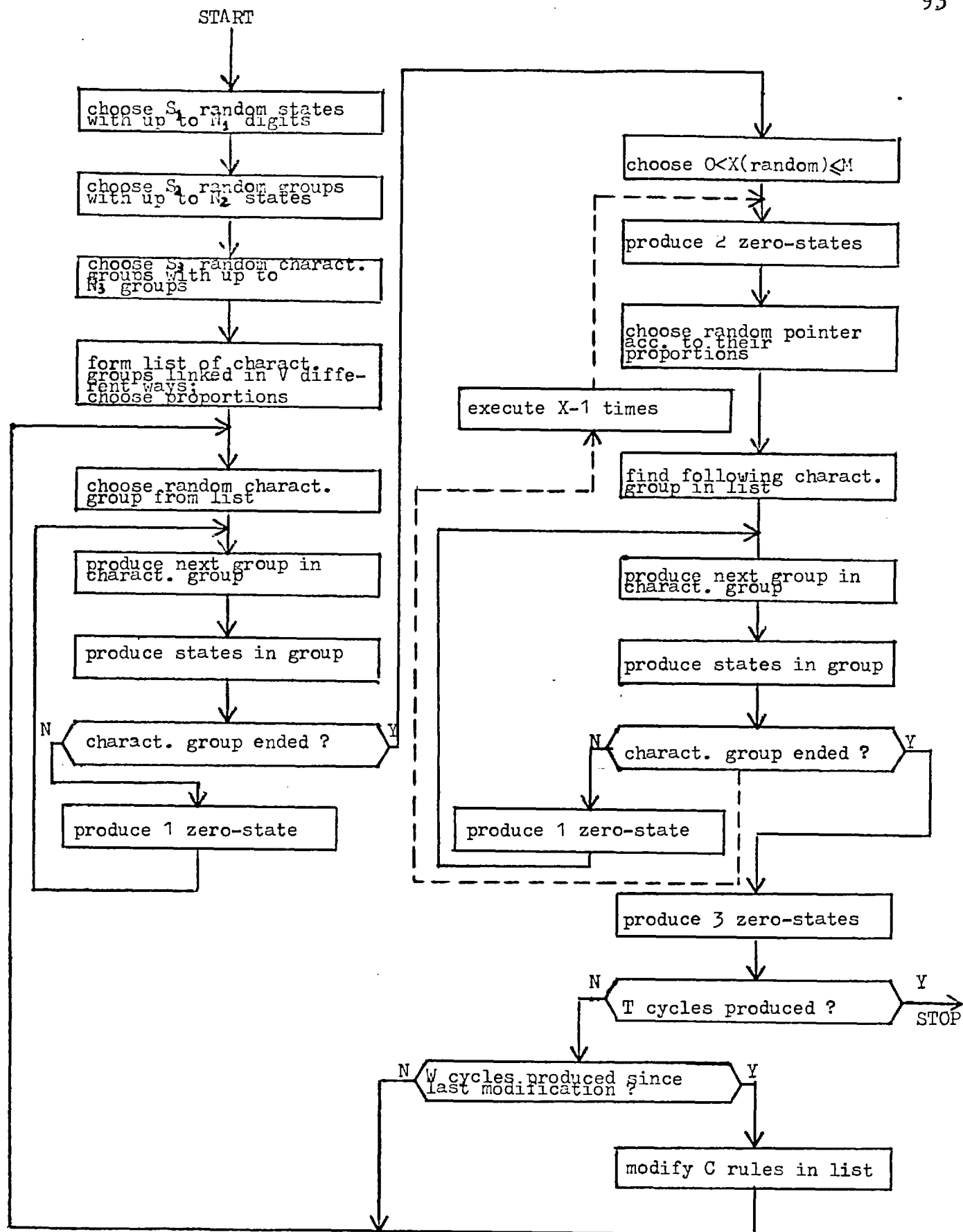


Figure 3.7. Level 3 type 2 environment generator.

development of this system in stages, and always using at each stage the simplest models that can efficiently simulate it. This approach was also followed for the organism model, which was developed to parallel the different levels of the environment generator.

CHAPTER 4

THE ORGANISM

4.1 Simulation.

In this section we will discuss the problems posed by the simulation of an organism model that has to adjust to the environment sequences produced by the generators described in Chapter 3. We will not consider the environment as a sequence of states about which nothing is known in advance. We have explained earlier that we regard the environment and the organism as an integrated system, and therefore they have to be developed together. It is impossible to design a model for the organism without taking into consideration the characteristics of the sequence of states that it must eventually learn to predict. We decided to define first the various environments, and now we can turn to the other part of the system. We will find the simplest models for the organism to match the levels and types of environment we have defined, in ascending order of complexity. As in the case of the environment models we will find this helpful, since each version is logically derived from the less complex ones.

4.1.1 The problems

In Chapter 2 we reduced the organism to a processor whose only connection with the environment are the states. This processor, represented there by a "black box", is the

only part of the whole system that has remained to be defined and we will see now what must be its features. As already explained, there are two kinds of states in the system - the environment states and the organism states. The former are produced by the environment and in the present context we can view them as an input to the organism processor, while the latter, being produced by this processor, can be viewed as its output. We must keep in mind however that the real input and output signals between the environment and the organism were those from which the states are formed and the two kinds of states sequences are only a convenient way we have adopted for our simulation purposes. It has enabled us to split the system - which is in fact permanently closed - into its two parts. This in turn enables us to design separate models for simulating the two parts and we have decided to have the environment states as the frame of reference and to make the organism responsible for the equality of the two kinds of states, at all time steps.

For fulfilling this task the organism must be able to "observe" the environment states, and also the difference between these and its own states. In the actual simulation it will be necessary to regard the two sequences as the input and the output of the organism processor, but they should not be confounded with the original input and output notions. They are only a result of the way we have decided to simulate the system - the last part in the chain must have input and output in order to be connected to those parts already defined.

The way the states are compared is another problem that has to be considered. Ideally the organism should be able to measure the degree of inaccuracy in its predictions and take corrective actions directly proportional to the difference observed. This is the process in a self-regulated system, when the error is amplified in the negative feedback loop and used in such a way that the output is corrected. But it is impossible to do this in our simulation. The reason is that we have decided to work with states and ignore the actual environment that might produce these states. The states themselves are only a coding of the signals in the original input and output channels of communication between the organism and the environment. This means that if a certain combination of such signals produces a state (whose value lies somewhere in the range allowed for that environment), then a slight change in one of the signals produces another combination, and thus another state, whose value may or may not be close to that of the first state. The values of these states, although represented by decimal numbers, are only patterns of digits and therefore it is impossible to perform a comparison operation with them. They can be either identical or not, and the idea of degree of accuracy or difference between states is thus meaningless.

At the level of individual channels there is of course meaning in the idea of accuracy, but the only way "close" combinations of signals can be detected at the level of states is by having all the information about the

channels themselves and the coding process. The patterns of decimal digits can then be compared and their "difference" ascertained, since we would know what the digit in each position stands for with respect to the original channels. But the comparison would not be based on the trivial format of the patterns, and in particular not on their arithmetical value as decimal numbers.

In our simulated system we have bypassed the channels and coding stages as mere technical details only necessary in a practical application, and we are therefore unable to define "closeness" between states. The comparison between the organism and the environment states has to be then on a yes-or-no basis, the states either being identical patterns or not. In terms of their representation as decimal numbers the states are compared arithmetically, but the only meaning of the result is whether the states are equal or not. No problems can arise from this restriction, because the states predicted by the organism are based on the states observed in the environment and there is no distortion of these numbers in the system. Therefore if the organism can make a correct prediction it can also produce the exact number and the comparison will be successful.

Our requirement for the organism was to be able to predict the environment states only "as accurately as possible", for deciding to what extent it displays intelligent behaviour. Although the yes-or-no comparison may seem to be a severe departure from this approach, it must be realized that it is only a simplification of the real-world

situation, necessary for simulation purposes. What effectively happens is the usual restriction imposed by the digital representation of analog values. We can view the states as a limited number of values that represent many more possible combinations of values of the original signals. These combinations are divided into groups based on their similarity (closeness of their respective values), each group being mapped into a state. There is no need then to consider closeness of states, since this is done at the level of the original signals. This simplification can be done in that part of the system we decided to ignore, being a function of each particular application.

Since the system only uses a small proportion of the total number of states possible with the given number of digits, we can conveniently imagine the absent states as the results of those combinations of signals that are close to the ones producing the states actually used. We are using then one state for representing a whole group of close states and the requirement for equality of states is thus equivalent to the requirement of closeness of the original combinations of signals.

*

If there were no implementation restrictions, the safest way the organism model could be simulated would be by storing the whole sequence of environment states. At each time step the processor would analyse this sequence

and decide what is the state most likely to occur next.

None of the facts that might help it in this analysis would be lost - it could scan the sequence as many times as necessary for every prediction it has to make.

What makes this method impractical is not the storage but the time restriction. There is no limit to the length of the environment sequences, and even reasonable values for the parameters with a level 3 sequence would result in a large number of states, because there are three levels of groups of states. But since the states in the sequence are represented as numbers (2 or 4 bytes per state) this would not pose a storage problem. The real problem would obviously be the time needed at each time step for analysing the whole past sequence.

But there is no need to do this for every time step - clearly all the results obtained from such an analysis at time step t_n would be the same as the results obtained by scanning the segment from t_1 to t_n at a later time step. At each time step the processor would get the same information it had at the previous time step, plus the latest state. There is no reason it should do the analysis of the environment sequence at every time step - it is enough to analyse the latest state only, in the context of the information it already has on the preceding sequence.

By using this method, the processor is effectively spreading the analysis of the environment sequence over the whole length of this sequence. The information it is

constantly acquiring must however be stored somehow and, since this is the same information the processor would have acquired from its analysis at each time step by the first method, what it does in fact is trade the time needed for this analysis with the storage space needed for saving it from the previous time steps. The information stored at a certain time step must then replace all the information contained in the environment sequence up to that time step. The sequence itself will not be available any more for inspection and therefore the way this information is stored is crucial for the correct prediction of the environment states.

There are two requirements for this storage system: it must contain all the relevant information on the environment and it must be efficient. The efficiency aspect is manifold. Firstly, there must be efficiency in storage space, i.e. the usage of the smallest space possible for a certain amount of information. Secondly, the storage method must be such that the relevance of the occurrence of a particular state can be readily ascertained at every time step, as a function of the information already stored. And thirdly, the prediction of the next state must be as fast as possible. We have not stated the obvious requirement that the prediction made by using this storage system should be correct, because it is implied by the first requirement, that of containing all the relevant information on the environment sequence. The prediction made should

then be the best possible for that environment.

The problems in designing the storage system are thus what to store and how to store it. The second problem is purely an implementation one and its solution has to be based of course on the solution of the first problem. We will concentrate here on the first problem - what to store. When we defined the features of the environment we saw that it is the repetitiveness in the sequence of states that can make this sequence understandable to the organism. The randomness hinders this process by detracting from the knowledge gained from the repetitiveness, reducing in effect the probability of those events repeating. The information stored and used in predicting the state at time step t_n must be based on the repetitiveness found in the environment sequence between time steps t_1 and t_{n-1} . As little as possible should be included from the irrelevant randomness in this sequence, and its influence on the prediction of the next state should be minimal.

The processor must identify the repetitiveness and eliminate the randomness. These features are strongly related to the types of environment, and in addition the level of the repetitiveness depends on the level of the environment. The storage system has therefore to be developed as a function of the environment. This is not surprising - we have already seen that the organism model cannot be designed independently and has to match the environment. Since the types and levels of the environment

are in fact extensions of one another in order of increasing complexity, we should expect that the storage systems needed for analysing them can also be derived from one another in the same order.

4.1.2 The strategy.

It is impossible to implement an organism model capable to adjust to a type 0 environment, at any level. There is no repetitiveness in a type 0 sequence and therefore no model can predict its states with a better than random accuracy, no matter how long it is observing such a sequence. We are not interested in such a random system and we have only studied the random environment for each level because this was the natural way to develop the other types.

In a level 1 environment the repetitiveness for types 1 and 2 is represented by the succession rules between states. There is little randomness in type 1 and we will discuss this type first. The only randomness allowed by the definition is the first state and the number of states in each group. The organism processor can easily identify these two factors, both being related to the zero-states between groups. Therefore, although it can never learn to predict them it can at least completely eliminate their influence on the repetitive factors. Since the succession rules are fixed, the processor can assume that if a certain state occurred once after another state, then this relation will also hold in the future. It is enough then to store

the relations between the states, which indeed represent all the relevant information in the environment. Each succession rule must appear at least once in the sequence, after which the probability of a correct prediction for that rule should increase in a perfect model from random to certainty.

The process of adjusting to a type 2 environment is much more complex. The repetitiveness is like in type 1 - in the form of succession rules. But, in addition to the type 1 randomness with which the processor can easily cope, there is the uncertainty introduced by the changes in the rules themselves. Each rule now has several versions and each of them may occur with a certain probability. One of the versions has the highest probability and is called the main version. Thus, for a particular state, the next state can be any of several others and each such combination will occur as a certain proportion of the total number of occurrences of the first state. In addition to this uncertainty (which simulates disturbances in the environment) there are the periodic modifications of all the versions for some of the rules. These modifications are random and although the new structure of the sequence is similar to the previous one, some of the knowledge the organism model might have gained in the previous interval is useless now.

The degree of uncertainty of the sequence is decided by the parameters V , P , C and W in the environment generator. Since the values of these parameters do not affect the

basic way in which the changes are made, we can design the type 2 model by considering their influence only. We do not wish the mechanism of the model to depend on these values.

We will consider now the general strategy for adjusting to a type 2 sequence. We must view the sequence from the side of the organism, i.e. preset and available one state at a time, but we have to take into consideration all the features we know that such an environment has. We are designing an integrated system and the task of the organism model is not to find out these features but to adjust to them. We should therefore use in full all the knowledge we have on the way the sequence is produced.

The actual version of a rule used, at each time step, is chosen at random and there is no way the organism model can ever learn to predict with certainty any state in the sequence. The best it can do is to identify for each rule the main version - the one that occurs with the highest probability. It can do this by observing the various rules over a sufficiently long period and counting the number of times each version occurred for each rule. Then it can decide that the version with the highest count is the main one, and predict it whenever its rule has to be used. If it does this for all rules (and assuming that it had identified correctly the main version for all of them), it can achieve a rate of successful predictions equal to the proportion P of the main versions - this proportion being common to the main versions for all the rules. (Slightly less in fact, due to the permanent type 1 randomness, also present

here.)

The fallacy of this approach is of course that the various versions of the rules may be modified throughout the sequence. The organism model cannot afford to observe the environment for a large number of cycles because by the time it reaches its conclusions on the main versions it may not get the opportunity to use them before a modification occurs. Or even worse - it may not reach any conclusions in one interval and carry on with the analysis of the sequence into the next interval without being aware of the modifications. No meaningful identification can then be made and clearly the whole sequence can be wasted without any conclusions being reached.

The interval each modification lasts is set by the parameter N and its power is now obvious: no matter how well designed the organism model is, if the value of N is below a certain number of cycles it is impossible to identify the main versions simply because there are not enough occurrences of the rules for making a statistically meaningful decision. Such values for N are of no practical interest and for the adjustment process to have any meaning we require not only that the main versions can be identified for most rules in most intervals, but also that the number of cycles in which these conclusions can be used is a significant proportion of the number of cycles that structure lasts.

In practice the proportion of occurrences of the main version is much higher than any of the others; in fact

the parameter P will always be at least 0.5. We must remember that the other versions are only a simulation of disturbances and in a practical application the main version should be easily identifiable. One of the effects of the parameter V (the number of versions for each rule) can also be seen in this context; for a relatively low value of P it will be easier to identify the main version if there are many other versions (a large V), than if there is only one other, because the ratio of the number of its occurrences to any of the others will be higher. The worst case is when $V = 2$, when the organism will need the longest period to identify the main version (and it is obviously impossible to identify it if P is as low as 0.5). If $V = 1$ then we must set $P = 1.0$ and then the identification is trivial - we have in fact a type 1 sequence (except for the periodical modifications).

We have seen that the organism model has to identify the main versions as quickly as possible, but this must be done at the beginning of each interval, after the modifications had taken place. Since it has no knowledge on the cycles on which the modifications occur, or on the rules that will be modified, it has to compare constantly its decision on the main versions with the actual proportions in the sequence. The appraisal of these proportions is based on a continuous observance of the occurrence of the versions over a sufficient number of cycles. The problem is that a momentary change in the predominance of the main version for one or several rules can be observed even inside

an interval, without a modification having taken place. This may happen because the choice of the version to be used at each time step is made at random in the environment generator. The organism model cannot decide to change its main version every time such a change is observed in the sequence and it must wait long enough to make sure that the change is indeed due to a modification. On the one hand then the response of the organism model has to be fast enough to minimize the length of the periods of adjustment when the new main version of a rule must be identified; on the other hand this response has to be slow enough to minimize the number of times the model will be wrong in its decision of changing the main version. The right speed of response to the changes in the environment is very important for an efficient adjustment and it must be controlled by parameters in the organism model, in order to achieve the best match to a particular set of environment parameters.

*

We have discussed the problems involved in the simulation of an organism model that has to adjust to a level 1 environment. Most of the problems were encountered for a type 2 environment, because the adjustment to a type 1 sequence is trivial. It is obvious that an organism model that can adjust to a type 2 sequence can also adjust to a type 1 sequence, but its capabilities will not be all

used with this simpler environment. We can therefore consider the type 2 model to be the general model of the level 1 system, as we considered the type 2 environment generator to be the general model for this level.

The models for level 2 can be derived from the level 1 models in a straightforward way. The difference between the level 2 and the level 1 environments is that the succession rules apply to groups of states rather than to individual states (we ignore type 0 at all levels, as far as the organism model is concerned). The mechanism of identifying and using these rules, as well as coping with the changes, can be the same in level 2 as in level 1, since the ways these environments are constructed are identical - except for the difference in the level at which the repetitiveness and changes apply. The stages from the input of the environment sequence to the adjustment mechanism are only an implementation aspect.

The extension to higher levels is similar. There must be a process of separating the characteristic patterns for each level (recognizing them in the environment sequence) and then the adjustment mechanism is applied to these patterns. The type 2 model is the general one at each level. In the next section we will discuss the implementation of these models.

4.2 Implementation

Although we have to develop the models of the organism and the environment separately, their separate simulation

is meaningless. We are only interested in them when working together, as an integrated system. For the environment model this means that the only usefulness of its output sequences is for being input to the organism model. For the organism model itself it means even more - it cannot be simulated at all without having an environment generator to produce the sequences it has to adjust to.

We are now at the stage where we can produce any environment sequence we need, and we can change its characteristics with the help of several parameters. We also know the construction of the resulting sequences in all its details. We have decided on the general method for adjusting to these sequences and what is left is to find the best implementation of this method.

The organism model sees the environment sequence as a continuous stream of states which is in fact a stream of decimal numbers (there is no need to regard them at the lower level of characters). The construction of the different sequences can be identified by looking for zero-states, these being represented by the number 0. Several consecutive zero-states may be used in a zero-state string in levels 2 and higher. In level 1 there are only strings of one zero-state, between groups of states; in level 2 there are strings of one zero-state between groups of states (the characteristic patterns) and of two zero-states between groups of groups of states; and so on for higher levels. In terms of cycles, there are L zero states between the major groups in a level L environment, and each such group

together with the following zero-states string form a cycle.

Since we have different models for the different levels, the program knows what kinds of zero-state strings to expect. The input of the environment sequence is a scanning process for dividing it into cycles and further into the various levels of groups of states. The most important actions must take place at the levels of cycles and of characteristic groups. The characteristic groups (or patterns) are that level of groups to which the succession rules apply. They are therefore the main units on which the identification of the repetitiveness and its changes is based. In fact the groups at levels lower than the characteristic ones (if there are any) are only needed for identifying them, like they were used when producing the sequence - as building units for the characteristic patterns.

The cycles are the main time units in the sequence, both for the environment generator and for the organism model. The cycles themselves convey no particular information (there is a random number of characteristic groups in a cycle, and there is no relation between adjoining cycles), but being the largest logical unit into which the sequence can be divided, all the parameters for time characteristics apply to cycles. These include the parameters T, W and M for the environment generator and also the organism parameters, as we will see later. This means that their influence is geared to the particular

process being simulated. This is the logical way to use them and we have adopted this approach at all stages. The time steps themselves are not absolute time units and they represent the smallest significant unit in the sequence, being therefore subject to each particular process themselves. The zero-states also have only a symbolic meaning: the number of zero-states in a string is needed for identifying the various levels of groups, but the actual length of time each such string stands for depends on the process being simulated. Their significance is the level of connection they provide for the adjoining groups and this clearly can only be defined in conjunction with the actual process. In practice the zero-state strings are likely to represent periods of time significantly longer than those of the corresponding level of groups they separate.

As long as these facts are considered in parallel in both the environment generator and the organism model, any process can be correctly simulated, since the significance of all its logical parts is understood in the same way by both sides of the system. We will come again to this later, in the explanation of various decisions taken in the implementation.

*

There is no organism model that can match the random, type 0 environments. As for the type 1 models, we implemented them although their task can be accomplished

by a type 2 model for the same level. The reason is that they are useful for the development of the more complex type 2 programs, which are derived from the type 1 models by adding the features necessary for coping with the changes in succession rules.

The level 1 type 1 model only has to look for strings of one zero-state, one string in each cycle of the sequence. Since the rules are fixed, no provision has to be made for more than one version of each rule and the predictions are certain to be correct after the first occurrence of that rule. The organism model does not know at the beginning of the simulation the number of different states in the sequence, i.e. the value of the parameter S_1 in the environment generator. Therefore the succession rules (there is one rule for each state) must be stored in such a way that new rules can always be added. The most efficient method is to use a list, in which new entries are added as new rules are being found, so that only the amount of storage actually needed is used. The program initially has no knowledge on the maximum number of digits in a state (set by the parameter N_1), but this is of no consequence since it treats the states as decimal numbers and in practice N_1 is always well below the hardware restriction. Each entry in the list contains two states, in the order in which they appeared in the sequence. When the first state in a rule occurs again in the sequence, the program predicts that the next state will be the second

state of that rule. (A circular list would increase the efficiency by reducing the number of times the rules are searched for in each cycle. But such a list can only be formed after all, or most of the rules had occurred in the sequence, and we are not interested in the trivial response of the model after this point anyway.)

The prediction will always be wrong at the end of a group of states (when the actual state is a zero-state) because of the random number of states in a group. But there are two cases when the program knows that its prediction is likely to be wrong: for the first state in a group (which is random) and when a state occurs, for which the rule had not occurred before. In these cases, rather than predict a random state, the organism makes no prediction at all and only reads the next state in the sequence (and adds the new rule to the list in the second case). There is no danger for the simulation because the organism knows that the "missing" time steps are the logical time units for the unknown states (we could design the program to predict a special "don't know" state, with the same affect on the simulation). As we have explained, as long as the same logic is used in both the actual and the predicted sequences, we can safely simplify the models as much as necessary. The unpredicted states are obviously recorded as wrong predictions. (It is true that designing the program to predict a random state - from those already in the list - could reduce the overall number of wrong predictions, since sometimes the random ones may be correct.

But we are not designing here models for practical applications and we are only interested in their comparative performance for various parameters. Our policy is therefore to keep such trivial optimization techniques, which would apply to all models, at a minimum.)

The organism model reads the environment states, predicts the next state whenever it can and continuously adds new rules to its list, until it detects the end of the sequence (after T cycles). Clearly the performance of this model will improve after the first few cycles, when the proportion of new rules in the sequence decreases. For a large number of different states and a low number of cycles, it may happen that the simulation ends before all the possible rules had occurred. It is mainly in such cases that this model is of interest, since after all the rules had occurred its response is trivial. But even before this its performance is highly predictable, due to the simplicity of the environment it has to cope with. The flowchart of the level 1 type 1 model is given in Figure 4.1.

*

The model for the level 1 type 2 environment is derived from the type 1 model. The way the sequence is scanned is identical, since they are at the same level. There are only strings of one zero-state, as in type 1, and the groups

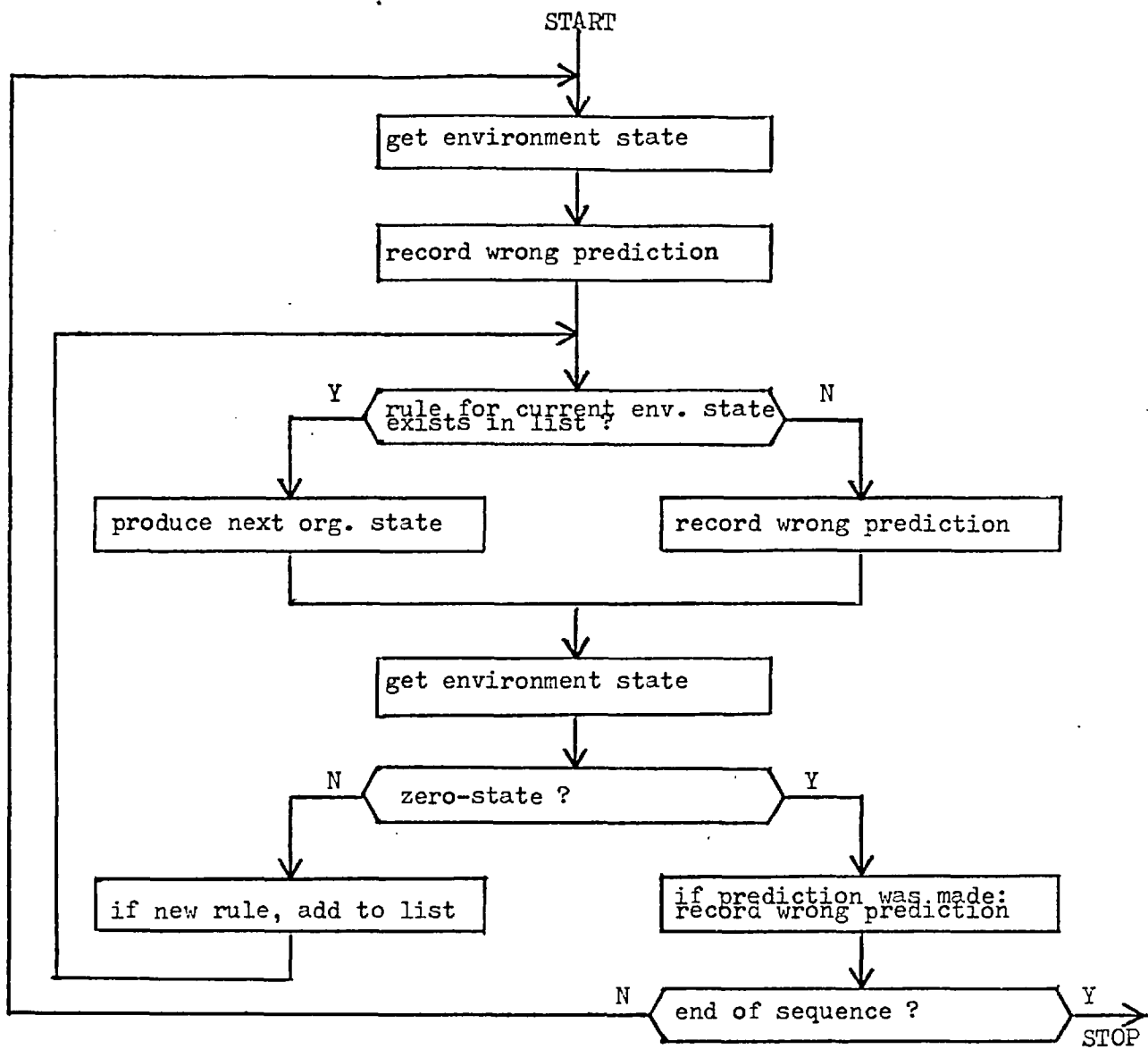


Figure 4.1. Level 1 type 1 organism model.

are similar. Therefore there will be wrong predictions at the end of the groups and there will be no prediction for the first state in each group and for those states whose rule had not occurred before. But the list used in the type 1 model for storing the succession rules is inadequate here, because the program must take into consideration several possible versions for each rule.

We must find now the best way of implementing the mechanism for adjusting to the type 2 sequence according to the strategy discussed earlier. Since the organism model has to identify the main version of each rule at a particular time by its superior proportion of usage, the obvious way to implement this would be by counting all the occurrences of the various versions and choosing the version with the highest score. The number of occurrences must be observed over a sufficiently large number of cycles in order to obtain significant results. But since the state that implies a certain rule is used in the sequence in a random way (due to the random first state in each cycle), the usage of the rule varies and therefore if we want a minimal number of occurrences before making a decision the waiting period will have to vary. In other words, we cannot solve this problem without considering at the same time the other problem - the speed of response to a change in the main versions - and it is best to solve them together.

A simple way to simulate a counting process and provide

at the same time the facility of controlling the speed of adjustment to a change in proportions, is by using an exponential function. Each version of a rule will have a value associated with it, which increases exponentially every time that version is used and decreases exponentially when it is not used. The speed of increase and decrease can be controlled by two (different) parameters and thus we can decide how fast the influence of each version is to grow or diminish. We need an exponential change rather than a linear one because the response is most important in the first few cycles of a change (for both the increasing and the decreasing influences), and the power of the parameters is much stronger in that period with an exponential function.

We could adjust the values of these parameters for the optimal response - fast enough to identify immediately a change in the main version of a rule, and slow enough to cope with the randomness of the sequence and avoid wrong, premature decisions. But we can achieve better results and increase the flexibility of the model by adding another pair of similar parameters and changing the values associated with the rules versions in two stages. The first pair of parameters will provide a relatively fast response and will control an intermediate value, according to the occurrence of the particular version in the sequence. The second pair will provide a slower response and will control the actual value associated with that version, according to the intermediate value. The first pair provides changes which are faster than our requirements, and the second pair

slower. By adjusting the four of them we can have a better control on the final response, than with one pair only. With a combined exponential influence we have a wider variation in the speed of response and over a period longer than can be achieved with one function only.

*

Like in type 1, the organism does not know the number of different states in the sequence and therefore a list is used for the first state of each rule. But since to each one of these states may correspond several second states for the several versions that may concurrently exist, the list has to be extended in two ways, as seen in Figure 4.2.

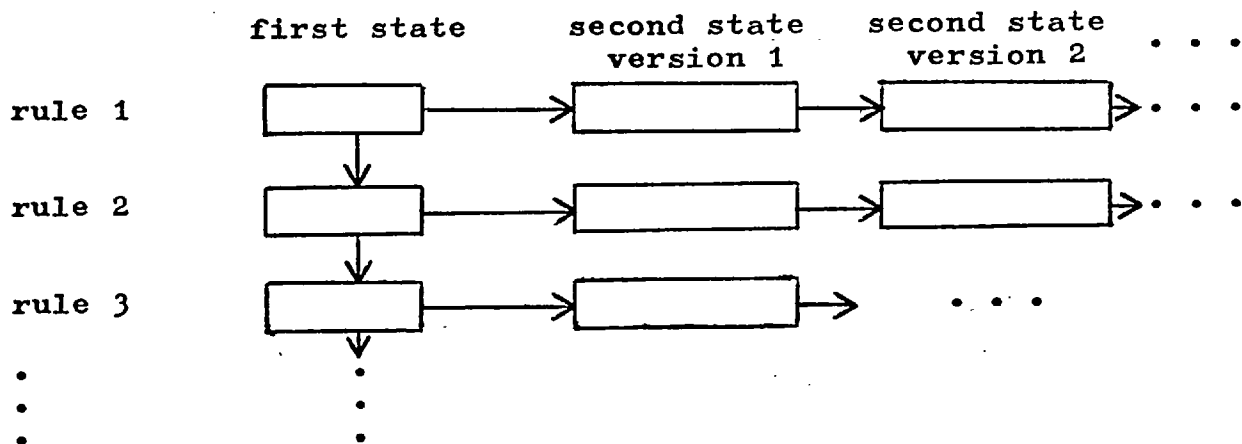


Figure 4.2. The storage system for the type 2 model.

The first entry in the list for each rule is the first state. The other entries for each rule are linked together and each one contains the second state for one of the

versions that are in effect at that time. In addition to the state itself, each such entry contains two values, called the primary and the secondary values, needed for evaluating the influence of that version, as described earlier. We will return to these values shortly.

The mechanism of the list is similar to the type 1 model: when a new rule occurs the program adds the first state to the list and attaches one version-entry for this rule, initializing its values. When a new version of an existing rule occurs, the program only adds a new entry to the list of versions of that rule. In this way only the minimum necessary amount of storage is used for an environment with a certain number of rules and changes.

The amount of usage of a version influences the two values associated with it. In order to keep track of this usage the program forms in each cycle a temporary list with those versions that occurred in the environment sequence at least once during that cycle. These versions are included in the temporary list regardless of whether they are new versions or not, or whether the prediction had been correct or not. This temporary list is identical to a type 1 list. At the end of each cycle the primary values of the various versions are modified according to this list.

We have found experimentally that the best way to form the double exponential response with the two values is as follows. The primary value is initially 0 and it is always less than 1. It changes between these bounds

exponentially, the cycle being used as the time unit. The speed of its increase and decrease are set by two parameters called respectively A_I and A_D . The values of these parameters can be between 0 and 1. At the end of a cycle the program scans all the versions for all the rules. For those which had been used in that cycle (according to the temporary list) it increases their primary value exponentially towards 1 with the parameter A_I , i.e. if the current value is V_P , then the new value will be:

$$V_P' = 1 - (1 - V_P) \times A_I$$

For all the others the primary value is decreased exponentially towards 0, with the parameter A_D :

$$V_P' = V_P \times A_D$$

After having modified the primary value, the program also modifies the secondary value of that version. The secondary values also lie between 0 and 1 and are initially 0. They increase and decrease exponentially but with different (and much slower) speeds than the primary values. The parameters for those changes are respectively B_I and B_D . Another difference is that the secondary values do not tend towards 0 or 1 in their decrease and increase, but towards the corresponding primary values. Before the modification of the secondary value the program compares the two values and if the primary one (after its own modification) is higher, then it increases the secondary one according to the following formula (V_S is the current secondary value and V_P' is the new primary value):

$$V_S' = V_P' - (V_P' - V_S) \times B_I$$

If the primary value is lower, then the program decreases the secondary value as follows:

$$V'_S = V_S \times B_D$$

Before the beginning of the next cycle the temporary list is discarded and this process is repeated until the end of the sequence. For deciding which version of a particular rule to use in its predictions, the program compares the secondary values and uses the version with the highest one. When there is only one version the decision is trivial, and when there are no versions at all (either because the rule had not occurred before in the sequence, or because all its versions had been discarded - as explained below), the program makes no prediction and adds the version to the list. The flowchart for the level 1 type 2 model is given in Figure 4.5.

There is obviously no need to keep all the versions for all the rules in the list indefinitely. Due to the modifications made in some of the rules from time to time, some of the versions will never be used again (unless they are later selected again by the random process in the environment generator). The indication for the lack of usage of a version is the decreasing secondary value and therefore we can decide that when it gets below a certain limit the entry for that version is deleted from the list. The reason for this deletion is only for efficiency purposes - to free the storage space which is not needed any more, and to save the time taken by comparing values which are insignificantly small anyway. The value chosen for this

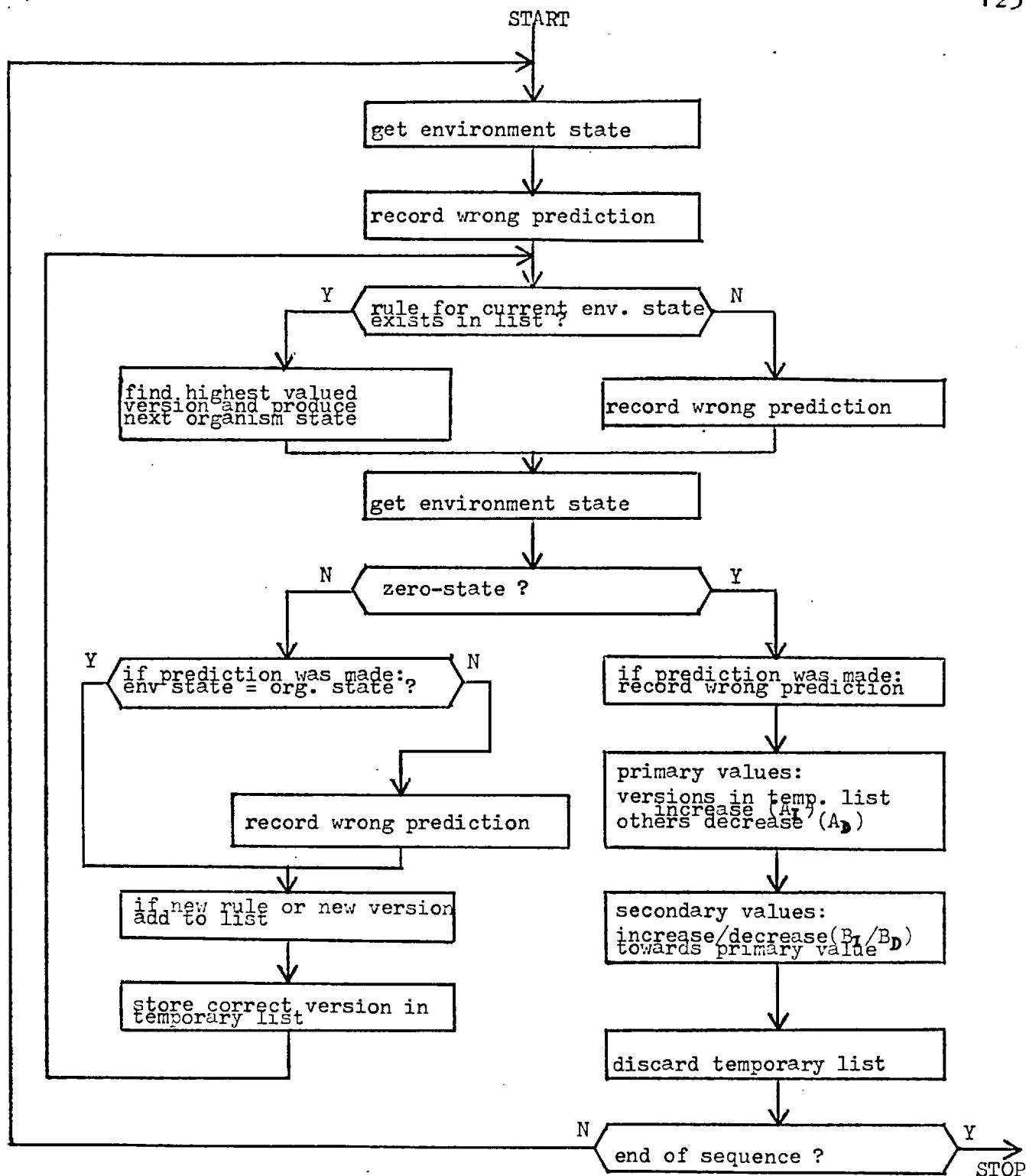


Figure 4.3. Level 1 type 2 organism model.

lower limit is only an implementation parameter and does not effect the response of the system in the sense of the other parameters. If it is chosen too high and versions which are still active are deleted, then the model will regard them as new versions when they recur in the sequence and will have to recreate their entries. The decision whether to delete an entry or not is considered on each cycle, after the modification of the secondary value for that entry.

*

The models for level 2 are derived from the level 1 models. The succession rules (and their changes in type 2) apply now to groups of states rather than individual states. This means that although we can keep the same basic mechanism for storing these rules (in both type 1 and 2 models) the entries in the lists must be groups of states. Also, the scanning process must be changed so that both levels of groups existent in a level 2 sequence can be identified.

For the internal representation of the groups of states we can use the same method we used in the environment generators, namely character strings. This is a compact form for storing several decimal numbers, and the states are easy to pack into and unpack from this representation when a fixed number of digits is used for all of them (by padding with blanks if necessary).

The program identifies the cycles in the sequence by the strings of two zero-states which separate the major groups. The groups of states are separated by strings of one zero-state. As it reads the states, the program packs them into a character string until a zero-state is encountered. The resulting string will be the internal representation of that particular group of states. After this point the states themselves inside the group are irrelevant and the group is treated as a pattern, in the same way the states are treated in the level 1 models. These patterns are stored in the list of rules, compared with one another and predicted as necessary. This method is also extended for the characteristic patterns at higher levels. This means that each entry in the list for the type 1 model will be a pair of such strings, and in type 2 just one string. For the versions in type 2 the primary and secondary values are identical to those in level 1, and so is their use.

When a group is predicted from the list as the next group to occur, it is unpacked into its composing states which are then produced in their proper order. Both for input and for output then, the states have at level 2 the role the digits have at level 1 - that of forming the characteristic patterns. The extension to level 2 presents therefore only problems of implementation: in the scanning stage when the sequence is divided into cycles and groups and the internal representation of the groups of states is formed; in the production of states from this internal representation; and in the storage system itself, where

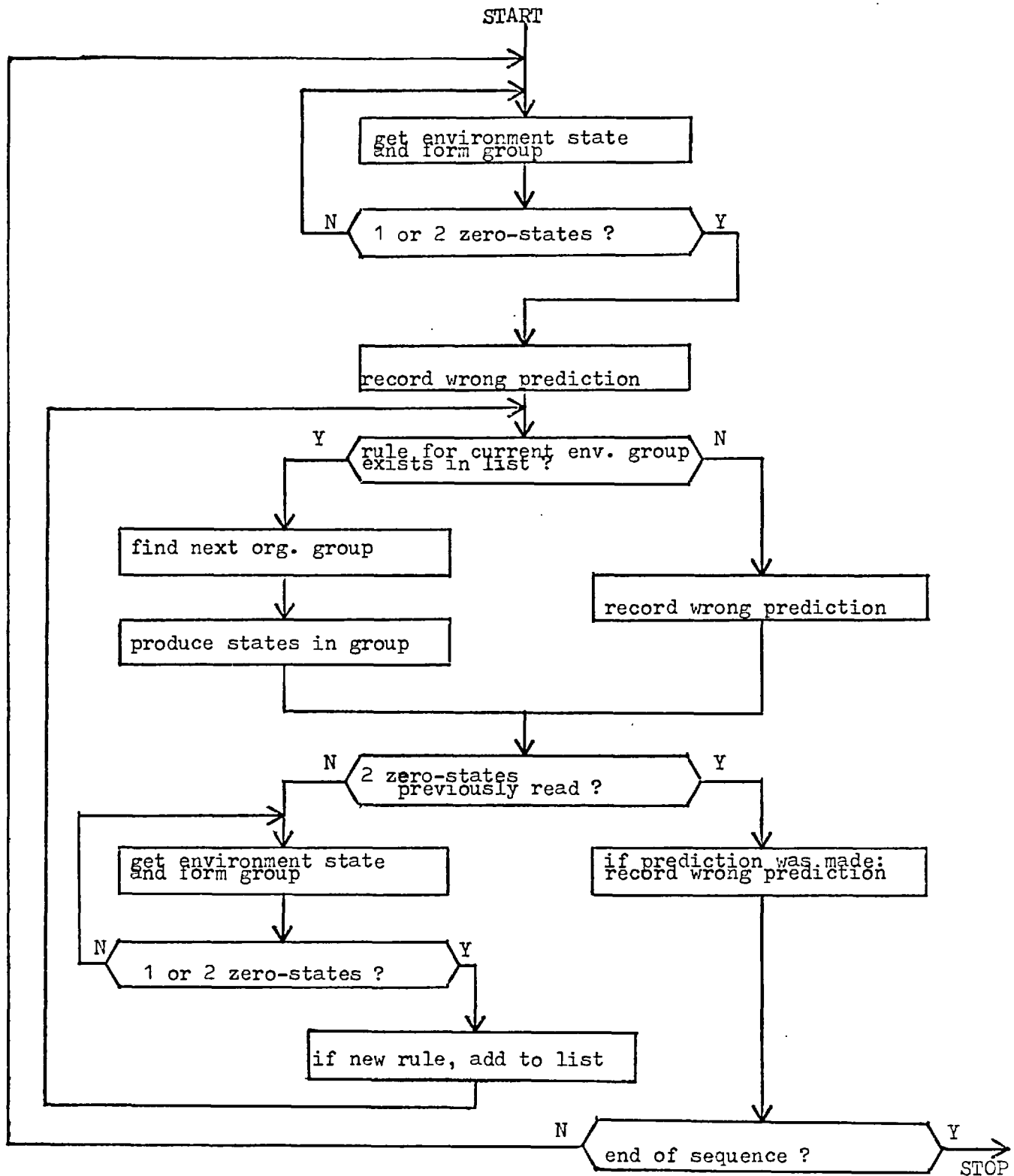


Figure 4.4. Level 2 type 1 organism model.

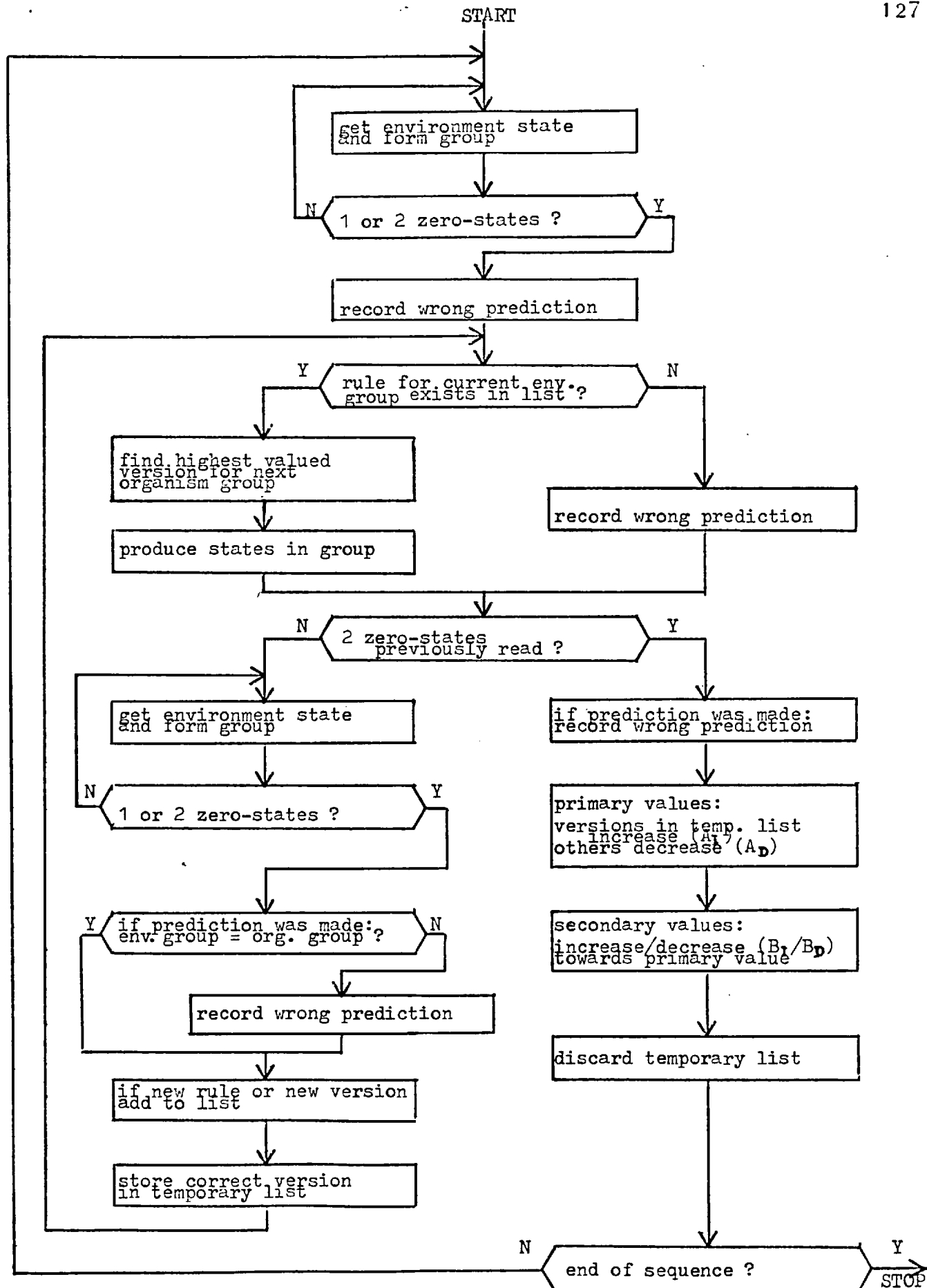


Figure 4.5. Level 2 type 2 organism model.

the entries are character strings rather than numbers. The mechanism for adding rules or versions to the list and predicting the next characteristic pattern is unchanged, although of course the comparisons are done with strings rather than numbers (since the comparison is on a yes-or-no basis, the results are the same in both cases). The flowcharts for the level 2 models are given in Figures 4.4 and 4.5.

The level 3 models are derived from level 2 by a similar logic. The characteristic groups are now the level 2 groups, i.e. the groups of groups of states. There are strings of up to three zero-states, therefore the scanning process has to be one level higher. The internal representation is as character strings, but here there are two levels of packing from states to characteristic patterns.

Because of the large amount of storage needed for the list (in particular in type 2), and also the time needed for handling it due to the complexity of its entries (which affects mainly the comparison processes), some restrictions have to be imposed on the values of the parameters in the environment generator for this level. We did not attempt to simulate higher levels. The rate of growth from level i to the next level $i+1$, for the amount of storage needed for the list is approximately equal to the parameter N_{i+1} . This is due to the internal representation in the form of character strings, and it is assumed that all other relevant

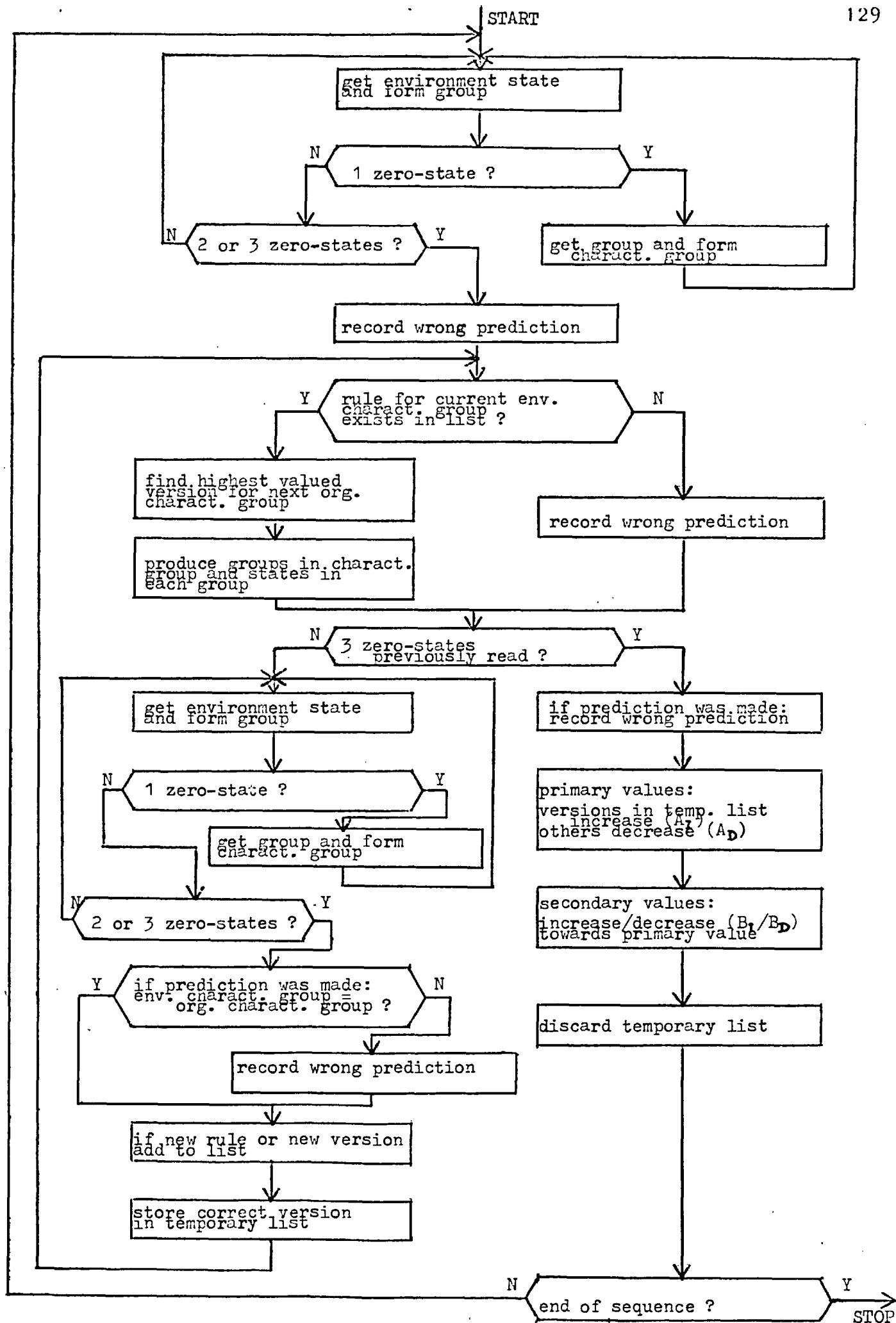


Figure 4.6. Level 3 type 2 organism model.

parameters remain unchanged and that $S_{i+1} = S_i$. Any further increase in the level of the system would necessitate therefore severe constraints on the various parameters of the environment generator, resulting in much too simple sequences. For the simulation of a practical, real-world process, level 3 is probably the highest possible. Its level of complexity is probably enough for many applications. The flowchart for the type 2 of level 3 is given in Figure 4.6 (like in the lower levels, type 2 can be considered as the general model since it can cope with type 2 sequences as well - there would only be one version for each rule).

CHAPTER 5

THE INTEGRATED SYSTEM

5.1 Introduction.

In the previous chapters we have designed the models for the simulation of the environment and the organism. When joined together, these models form an integrated system and we have to study its behaviour as a function of the various parameters. There are parameters in the environment generator for changing the characteristics of the sequence of states (for each level and type), and in the organism model for changing its ability to adapt to these sequences (for type 2). In the implementation of the system it is possible to produce the states and simultaneously let the organism read them, one at a time; but we preferred to keep the two programs separate and to produce a whole sequence before presenting it to the organism model. Only the first three levels were implemented; as already mentioned, the implementation of higher levels would have imposed serious restrictions on the values of the parameters, for keeping the simulation within reasonable limits of time and storage.

The parameters S/N are common to both type 1 and type 2 systems. These parameters set the way in which the various levels of patterns are formed from the lower level patterns, from digits up to characteristic patterns. There

are i such pairs for a level i system. As defined in Chapter 3, in level 1 there are S_1 different states, each one formed with up to N_1 digits; in level 2 there are S_2 different groups of states, each one formed with up to N_2 states, and S_1/N_1 are similar to level 1; in level 3 there are S_3 different groups of groups of states, each one formed with up to N_3 groups of states, and S_2/N_2 , S_1/N_1 are similar to level 2. The parameter S_i at each level i is the number of characteristic patterns for that level (which are time patterns, except for level 1 where they are just states). The other parameters N_i , N_{i-1} , ... N_1 and S_{i-1} , ... S_1 (if any) only decide the way the characteristic patterns are ultimately built (they also decide the maximum number possible for these patterns but in practice the actual number of patterns at each level is very much smaller than the maximum possible).

From the viewpoint of the system behaviour at level i , from all the S/N parameters only S_i is important. Being the number of characteristic patterns, it is also the number of succession rules. N_i only decides the format of these patterns, S_{i-1} (for levels 2 and higher) is the number of patterns from which the characteristic ones can be built, and so on down to digits. The decimal digits themselves are patterns for building the states and of course there are always ten such patterns. The parameters N have a strong influence on the storage and time needed for the implementation, their effect being multiplicative

(for level 3 for instance we need $N_1 \times N_2 \times N_3$ bytes for representing one characteristic pattern, this also affecting in a similar way the time needed for handling them). But they have no influence on the response of the system, nor do the S parameters - apart from S_i . Furthermore, we should expect the parameter S_i to have the same influence on the system irrespective of its actual level i . The mechanism (of both the environment generator and the organism model) is the same for all levels at the stage of characteristic patterns.

Apart from storage and time differences the systems at all levels must display identical prediction capabilities for sequences that are similar in their characteristic patterns and cycle format. This point has to be explained. Our criterion for a good performance of the organism model is the proportion of correct predictions of characteristic patterns out of the total number of such patterns in the sequence. Since two characteristic patterns are identical only if all the lower level patterns from which they are built (down to states) are identical, this means that for a certain proportion of correct predictions of characteristic patterns there will in fact be the same proportion of correct states predicted, irrespective of the level of the system. The performance of an organism model for any level must ultimately be judged from the prediction of the states, these being the smallest logical time units. But since the result is the same, we can limit ourselves to the characteristic patterns and this makes it possible

to disregard the level of the system when only its behaviour is of interest. (For a type 2 system it would be possible to improve the predictions at levels 2 and higher by checking the correctness of the states themselves - inside the characteristic patterns - as they are produced and, in case of error, continuing with the version which has the second highest value. If still in error, the process can be repeated until the correct version is found - if it is in the list. Such a method would increase considerably the proportion of correct predictions but, as already mentioned, we are not interested in this kind of optimization. While they may be useful in a practical application where only the final results count, it is against the logic of our approach and we prefer to make the predictions level - independent.)

Although there are no limits to the values of the parameters S/N (except for the hardware restriction on the number of digits in a state), we chose low values for N_1 , N_2 and N_3 (never more than 5) so that we can freely increase the value of S_i (the values of the parameters S_{i-1} , ..., S_1 do not affect the time or storage needed). This eliminates the restrictions at level 3 and allows us to use the same values of S_i for all three levels - this being the only parameter from all the S/N ones we are interested in. In practical applications of course each one of the S/N parameters may have widely different values, to match the particular environment being simulated.

The best way for representing the behaviour of the system and the influence of the various parameters is by showing the correctness of the organism's predictions as a function of time. The unit for time is the cycle - because it is the largest logical unit, it is level-independent and all the parameters related to time are for cycles. For both types 1 and 2 the system performance is always in connection with cycles: the number of cycles required for adjusting to the sequence at the beginning of the process and, for type 2, after a modification. The choice of the number of cycles elapsed from the beginning of the simulation as the independent variable is quite obvious. For the dependent variable however there are several possibilities. It must represent the behaviour of the system at a particular cycle, but we measure this behaviour as the proportion of correct predictions of characteristic patterns from the totality of such patterns. For a relevant result as many occurrences as possible must be used, certainly more than there may be in a cycle. But if we extend the count over many cycles for each instant in time, we lose the accuracy necessary in the description of the momentary behaviour of the system - particularly relevant in type 2. For this reason we decided to use an averaging method, as follows.

The experiment for each set of parameters was done several (at least 20) times, each time changing the random order in which the first characteristic pattern in each major group (cycle) is chosen in the environment generator. This can be easily done by changing the starting value for

the random process which controls this choice. This results in a different sequence of characteristic patterns throughout the cycle, the succession rules being applied after the first pattern. Also, the random process for the choice of the length of a cycle was similarly modified in each experiment, but keeping the parameter M (for its upper bound) unchanged. For each cycle, the ratio of the total number (in all experiments) of wrong predictions, to the total number (in all experiments) of characteristic patterns in that cycle (the zero-state string at the end of the cycle counting as one pattern), was then calculated. This ratio effectively represents the probability of a wrong prediction in that cycle. These results apply to systems of any level (the experiments were made with the first three levels). The probability can be viewed as a result for the characteristic patterns in each cycle, or for any of the states themselves - the proportions are the same. The graphs will represent therefore the probability of a wrong prediction at any time step within a particular cycle, as a function of the distance of that cycle from the beginning of the simulation.

The method of taking the average of many experiments for each set of parameters can also be used for eliminating all the uneven features introduced by the random factors when generating the environment. For this reason, in each of the experiments the other random processes (in addition to the two mentioned above) were modified as well,

but always keeping the parameters themselves unchanged. The various patterns (including the characteristic patterns), as well as the succession rules and for type 2 the various changes in these rules, are thus different in each experiment. The resulting graphs do not represent then an actual response, which would be much more irregular. They only represent the probability for such a response. (There were still irregularities even in the average graphs, but they were ignored in the drawings - a lot more than 20 experiments are needed for producing a smooth average response.) Since no general conclusions can be obtained from separate experiments, in the following discussion whenever we mention the system response we will mean the average response.

5.2 The type 1 system.

The only parameters which affect a type 1 system are S_i for the number of characteristic patterns (for a level i system) and M for the upper bound of the number of such patterns in a cycle, which we will call the "length" of the cycle; this length is random, between 1 and M . The Behaviour of the type 1 system stabilizes after all succession rules had occurred at least once in the sequence, because after that they are unchanged until the end of the sequence. The system displays then the smallest probability for a wrong prediction and this probability is a function of M .

When the organism model has all the succession rules in its list the only wrong predictions are for the first characteristic pattern in each cycle (which is random) and for the zero-state string at the end of the cycle (because the length of the cycle is random). Thus, there are two wrong predictions per cycle in the steady-state situation, irrespective of the length of the cycle. Larger values for M will increase then the average length of the cycles and decrease the proportion of wrong predictions.

This effect of the parameter M is also seen in the earlier part of the response, before the steady-state is reached. At the very beginning there must be a probability of 1.0 for a wrong prediction and the value of M makes no difference. But as soon as some rules recur in the sequence and the probability is less than unity, the two permanently wrong predictions per cycle start influencing the probability, this influence growing relatively to the influence of the new rules until it is the only source of errors in the steady-state.

In addition to this influence, the parameter M also affects indirectly the response of the system in relation to the occurrence of new rules. Since to each characteristic pattern corresponds one succession rule, an increase in the average length of the cycle results in an increase in the probability that after a certain number of cycles, a certain number of rules had occurred at least once. A large value for M improves therefore the performance of the system before

the steady-state not only by decreasing the influence of the two permanently wrong predictions per cycle, but more strongly by providing more characteristic patterns per cycle and hence reducing the number of cycles needed for the rules to occur.

The number of characteristic patterns in a type 1 system is also the number of succession rules. There are therefore S_i succession rules (for a level i system) and clearly an increase in S_i must result in an increase in the number of characteristic patterns in the sequence necessary for a certain number of rules to occur at least once. Since the first rule used in each cycle is random (due to the randomness of the first characteristic pattern), the distribution of all the rules in the sequence is random and many rules may occur several times before others had occurred only once. The probability that the adjustment period will end after only S_i characteristic patterns is extremely low and the occurrence of new rules may extend over a much longer period. However, the proportion of new rules, and hence of wrong predictions, decreases continuously.

The response is measured as a function of cycles and therefore the influence of S_i must be seen against a fixed value of M , i.e. a fixed average number of rules per cycle. In Figures 5.1 and 5.2 the response of the system is shown for $M = 10$ and $M = 20$ respectively. In each case there are three values for S_i - 50, 200 and 1000. The results

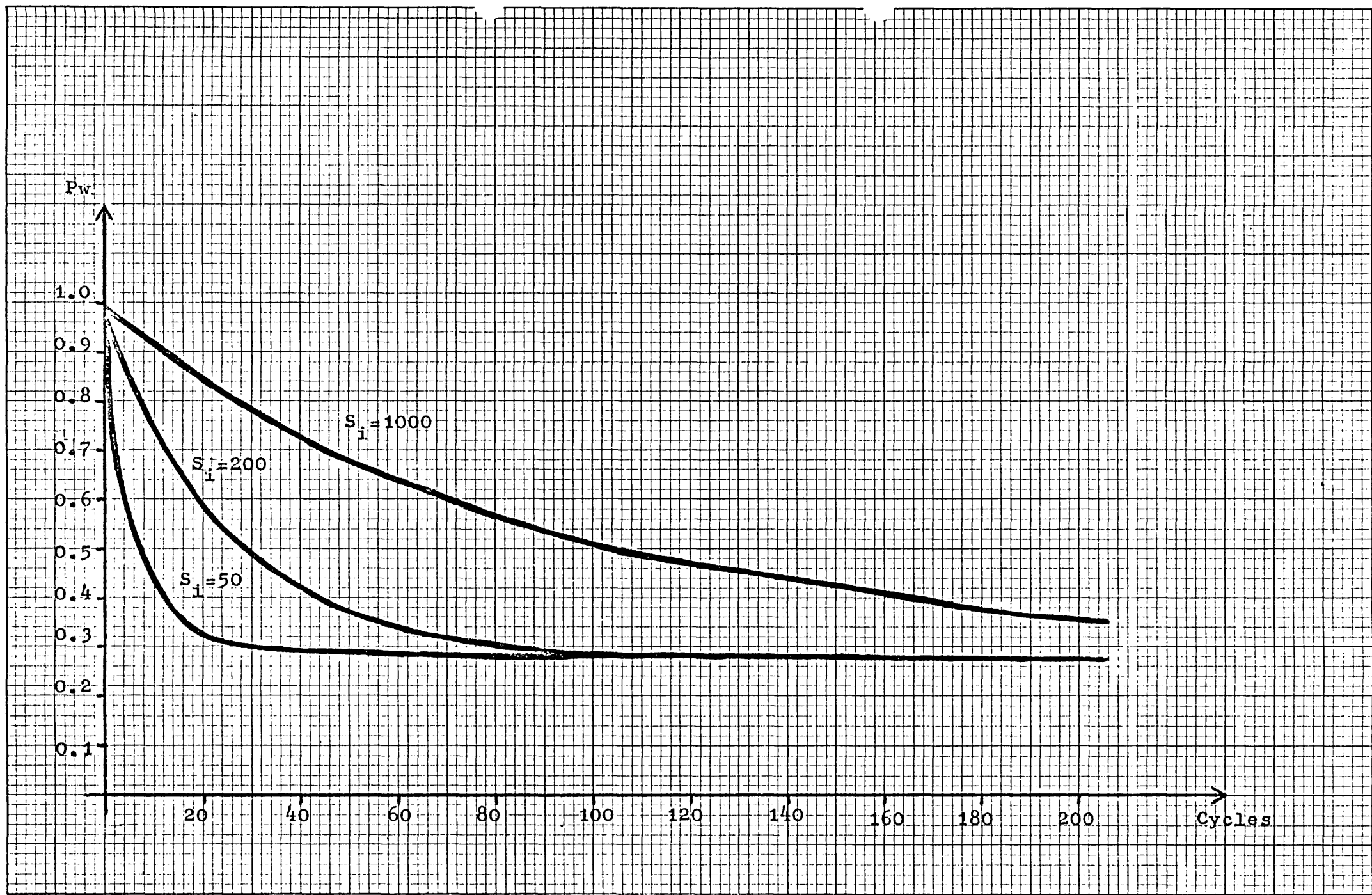


Figure 5.1 Influence of S_i on the probability for a wrong prediction P_w for $M=10$ (type 1 system).

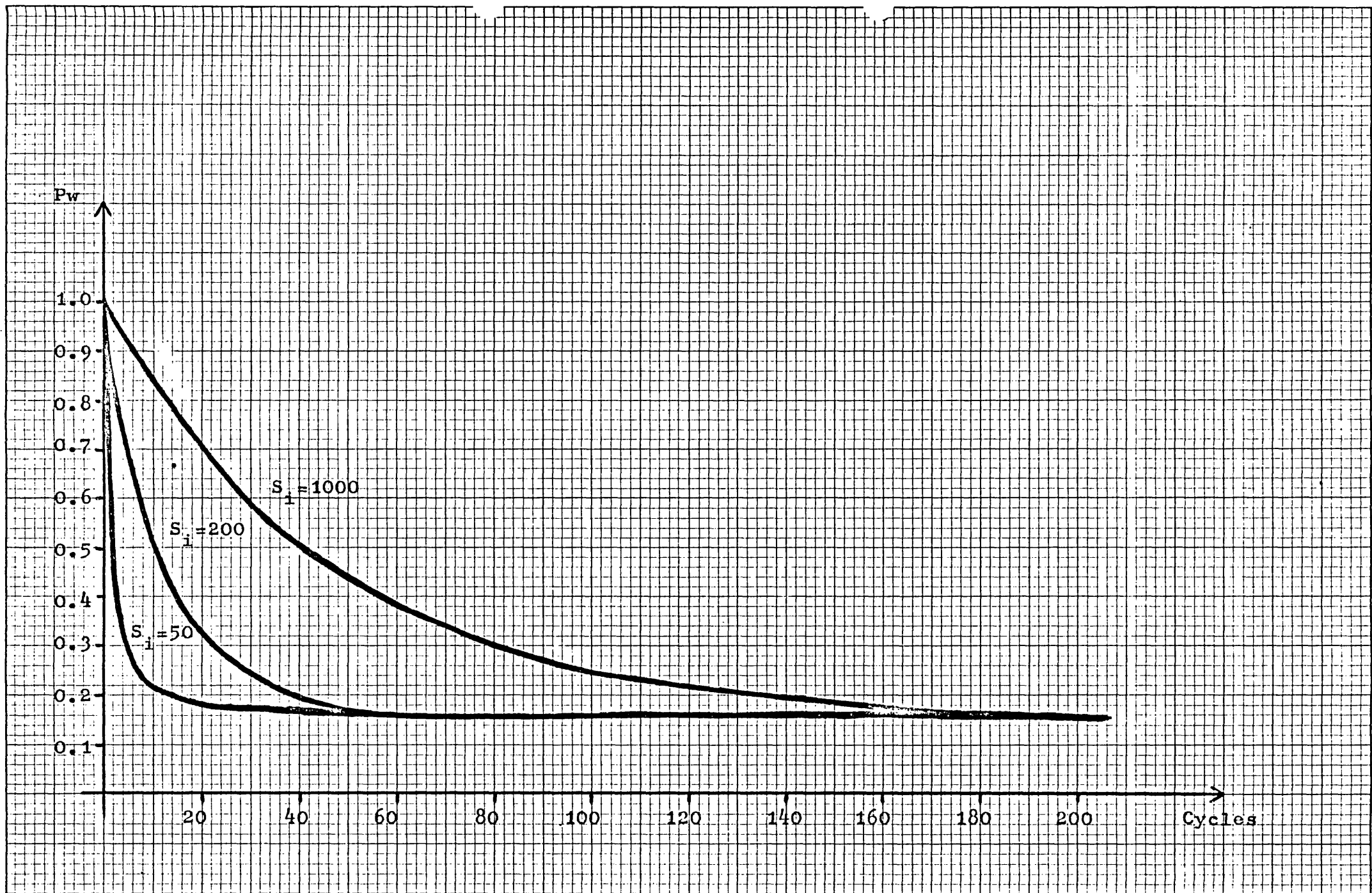


Figure 5.2 Influence of S_i on the probability of a wrong prediction P_w for $M=20$ (type 1 system).

are based on experiments with all three levels of systems and the other parameters were constant: $N_1 = 3$ for level 1; $N_1, N_2 = 3, S_1 = 50$ for level 2; $N_1, N_2, N_3 = 3, S_1, S_2 = 50$ for level 3. The only use of the parameter T - in these and all other experiments - was to stop the sequence after the relevant period.

5.3 The type 2 system

There are more parameters in a type 2 system than in a type 1 system, and there are in addition parameters for the organism model. The parameters in the environment generator are: the S/N pairs, M, W, C, P and V . The best way to see the affect of these parameters is by changing them one at a time, while keeping the others constant. We will see how these changes influence the system response.

The discussion on the S/N parameters applies to all systems and therefore, as in type 1, the only parameter that influences the time response is S_i , for the number of characteristic patterns (for a level i system). The response of the type 2 system as a function of S_i is given in Figure 5.3. The effect of S_i from the beginning of the sequence to the cycle of the first modification in rules is similar to that in type 1: a large S_i means more succession rules and therefore more cycles needed for a certain number of them to occur at least once. The number of versions per rule (V) is the same for all rules and is

constant, therefore an increase in S_i means an increase in the total number of versions by the same factor and hence a decrease in the probability for the occurrence of each one of them, and in particular the main version.

It may happen of course that it is not the main version which occurs the first time for some of the rules, but the influence of such cases is eliminated when taking the average results of several experiments because the probability for the occurrence of the main version is higher than for any of the others. After the first occurrence the selection by the organism model of a certain version depends on the relative frequency of occurrence of that version (the secondary value of each version in the list being ultimately a function of its frequency). The average results for several experiments are therefore again in favour of the main version due to its higher frequency, although in individual sequences there may be periods when the frequency of another version is higher and makes it the dominant version in spite of the slow response produced by the A/B organism parameters.

When there is a modification in the succession rules after W cycles, there is a sudden increase in the probability for a wrong prediction. The influence of S_i on this increase is in conjunction with the number of rules changed: this number being constant (parameter C), an increase in the total number of rules S_i results in a smaller probability for the occurrence of a rule which had

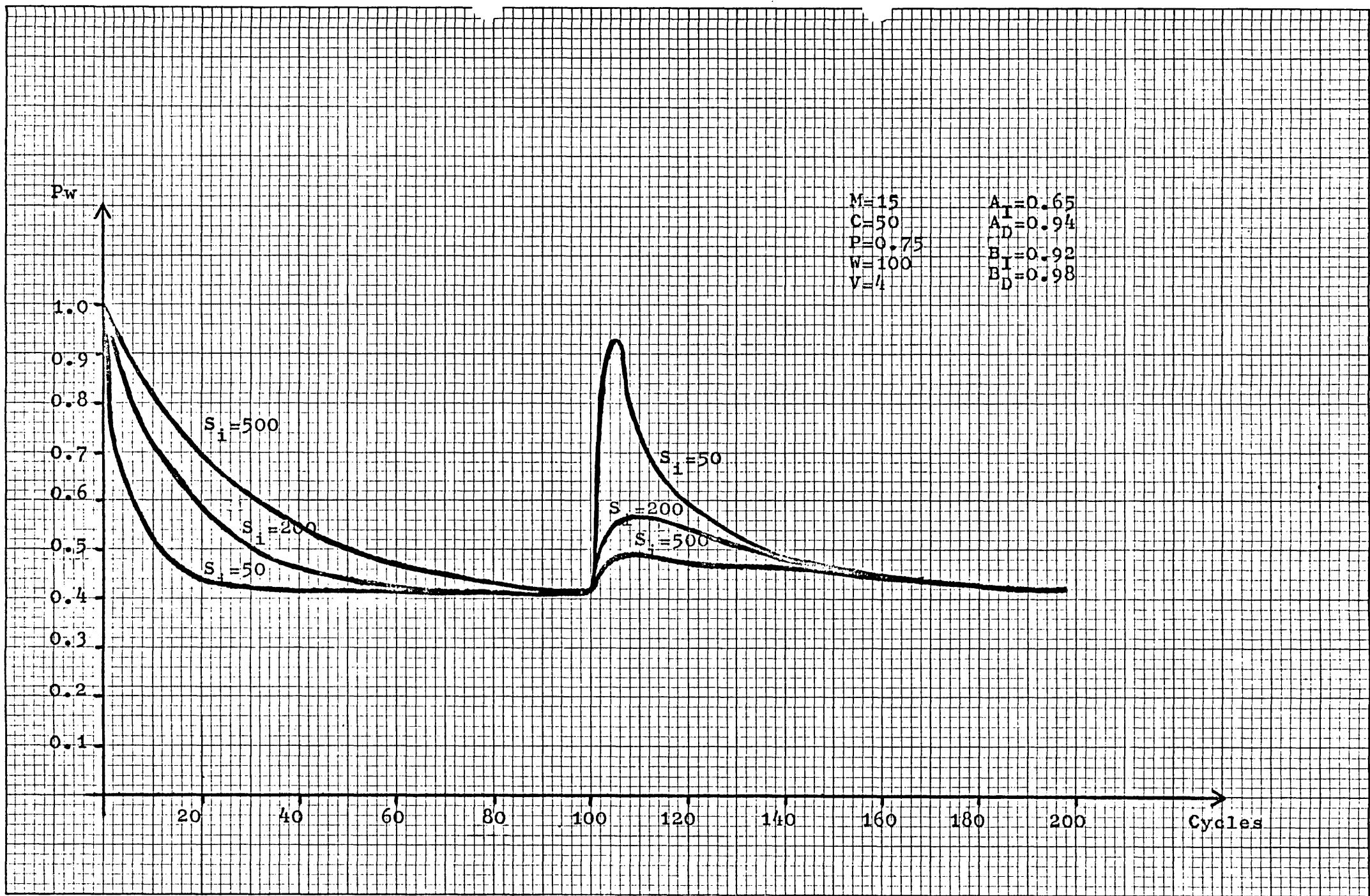


Figure 5.3 Influence of S_i on the probability of a wrong prediction P_w (type 2 system).

been modified, and therefore a smaller probability for a wrong prediction. If the value of S_i is so large that the system does not reach its steady-state, i.e. its lowest probability of errors, before the modification in rules (this does not happen in Figure 5.3), then after the modification there are two sources for wrong predictions - the new rules and those whose prediction had also been wrong before. The probability for the latter decreases at the same rate as before the modification but for the new rules it is different, involving the change in the values associated with the old and new main versions, which depend on the A/B parameters of the organism model. S_i influences both sources similarly, by affecting the frequency of occurrence of the main versions as explained for the earlier response.

The graphs in Figure 5.3 are for $S_i = 50, 200$ and 500 , with the values of the other parameters unchanged, as shown.

The parameter M (the upper bound on the number of characteristic patterns in a cycle) affects all the parts of the response graph, as seen in Figure 5.4 for $M = 5, 15$ and 25 . The reason is of course the fact that this parameter provides the link between the number of cycles (which are the unit the response is based on) and the frequency of all rules and versions (on which the predictions are based). It affects the early part, before a modification, similarly to the type 1 system, by influencing the average number of rules per cycle and thus the probability for the main version of a certain number of rules to occur at least

once in a certain number of cycles. As explained for S_i , taking the average results of several experiments eliminates the influence of those cases where the main version is not the first one to occur, or occurs less frequently than another one for short periods.

The parameter M also affects the response (including the steady-state if reached before a modification takes place) in connection with the fact that there are always two wrong predictions per cycle irrespective of its length, as explained for the type 1 system. Lower values for M increase therefore the influence of these two errors. M also affects the response after a modification, by deciding the number of occurrences of the main version for the new rules in a certain number of cycles (and thus the speed with which the change from the old to the new main versions takes place).

The parameter P (for the proportion of usage of the main version out of the total number of occurrences of each rule) also affects all the parts of the system response. This is not surprising, since correct predictions mean predictions of the correct main version, and the choice of a certain version by the organism is influenced by its relative frequency of occurrence. The choice is always for the main version anyway, after its higher frequency had made it the dominant version in the list. An increase in P then obviously results in a higher proportion of cases the prediction turns out to be correct. There is in addition an indirect influence:

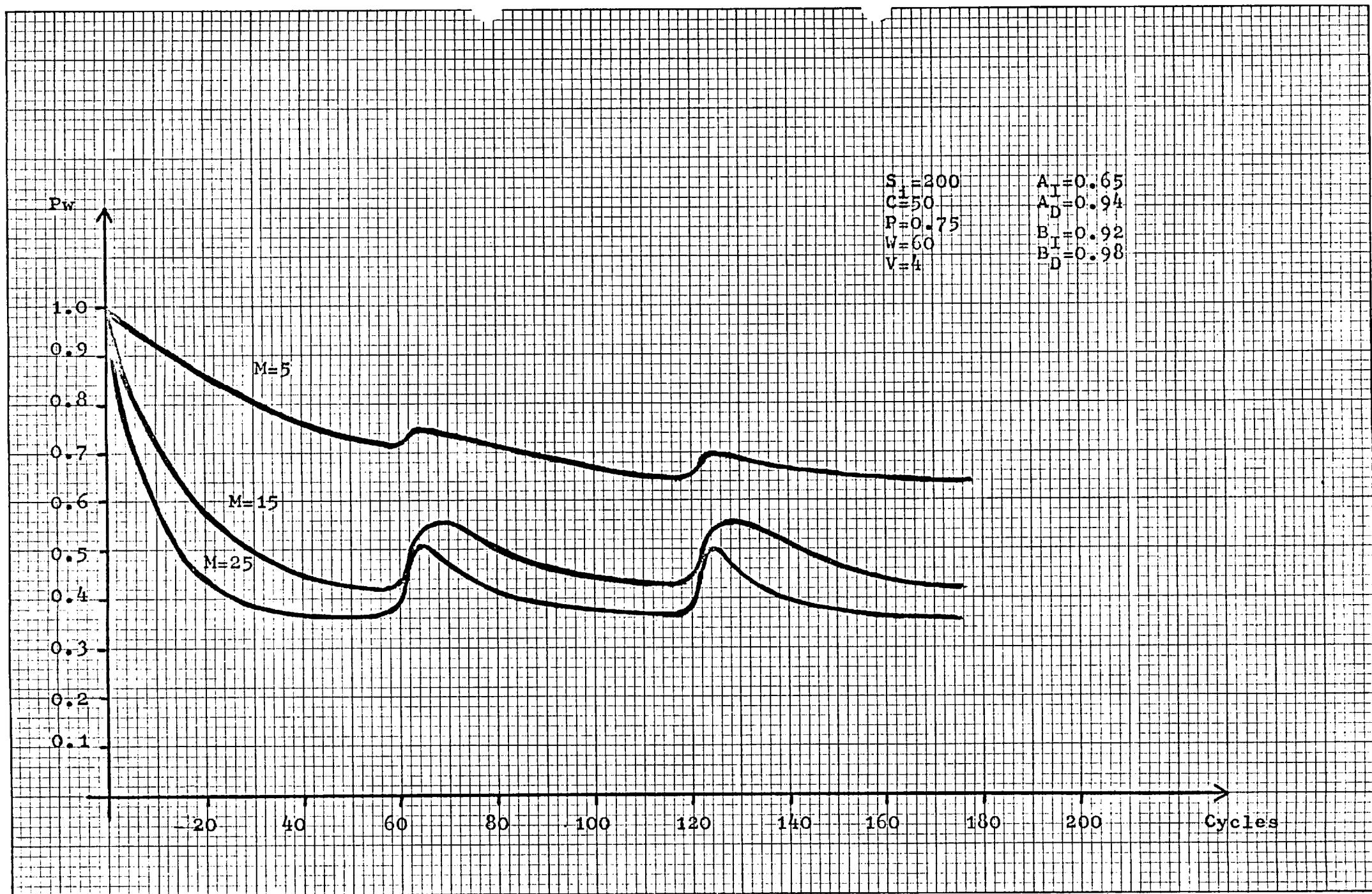


Figure 5.4 Influence of M on the probability of a wrong prediction P_w (type 2 system).

an increase in P reduces the probability of the main version losing its dominance for transient periods, due to the randomness in the sequence.

The influence of P can be noticed in particular in the steady-state response, when the system had reached its lowest possible probability for a wrong prediction. M and P are the only parameters which affect this part of the response and if it was not for the two permanently wrong predictions per cycle, the performance in the steady-state would be directly proportional to P and unaffected by M . The response of the system as a function of P is shown in Figure 5.5 for $P = 0.60, 0.75$ and 0.90 .

The only effect the parameter C has on the response is when a modification occurs. C is the number of succession rules being modified (for each rule all its versions are modified) and it is the proportion of the new rules out of the total number of rules S_i , that is reflected by the new probability for a wrong prediction immediately after a modification. We saw the influence of S_i with C constant in Figure 5.3, and in Figure 5.6 the influence of C is shown for $C = 20, 50$ and 100 , and $S_i = 200$ and constant.

The value of the parameter V (for the number of versions for each rule) was kept constant ($= 4$) in all experiments. This parameter has no significant influence on the system response in any practical situation. It has a powerful effect when its value is 2 (the lowest,



Figure 5.5 Influence of P on the probability of a wrong prediction P_w (type 2 system).

since $V = 1$ implies a type 1 system) and at the same time the value of P is low ($P < 0.65$). In this case the main version may lose very frequently its dominance on the only other version, in any of the rules. However we are not interested in such situations because the additional versions must be a simulation of disturbances only, each one's proportion being significantly lower than P .

The parameter W sets the number of cycles between consecutive modifications, and its effect is trivial. If its value is large enough for a certain combination of values for the parameters S_i , M and P , then the system had already reached its steady-state before the first modification, otherwise the effect of the still wrongly predicted versions will add to that of the newly modified rules (this can be seen in Figure 5.4). If the system reaches its steady-state before the first modification, it does not mean that it will reach it again before the second, or third, etc. This depends on the number of changes (parameter C) and the parameters A/B of the organism model. But the probability of a wrong prediction in the steady-state, if reached, must be the same throughout the sequence, since it only depends on the parameters M and P .

*

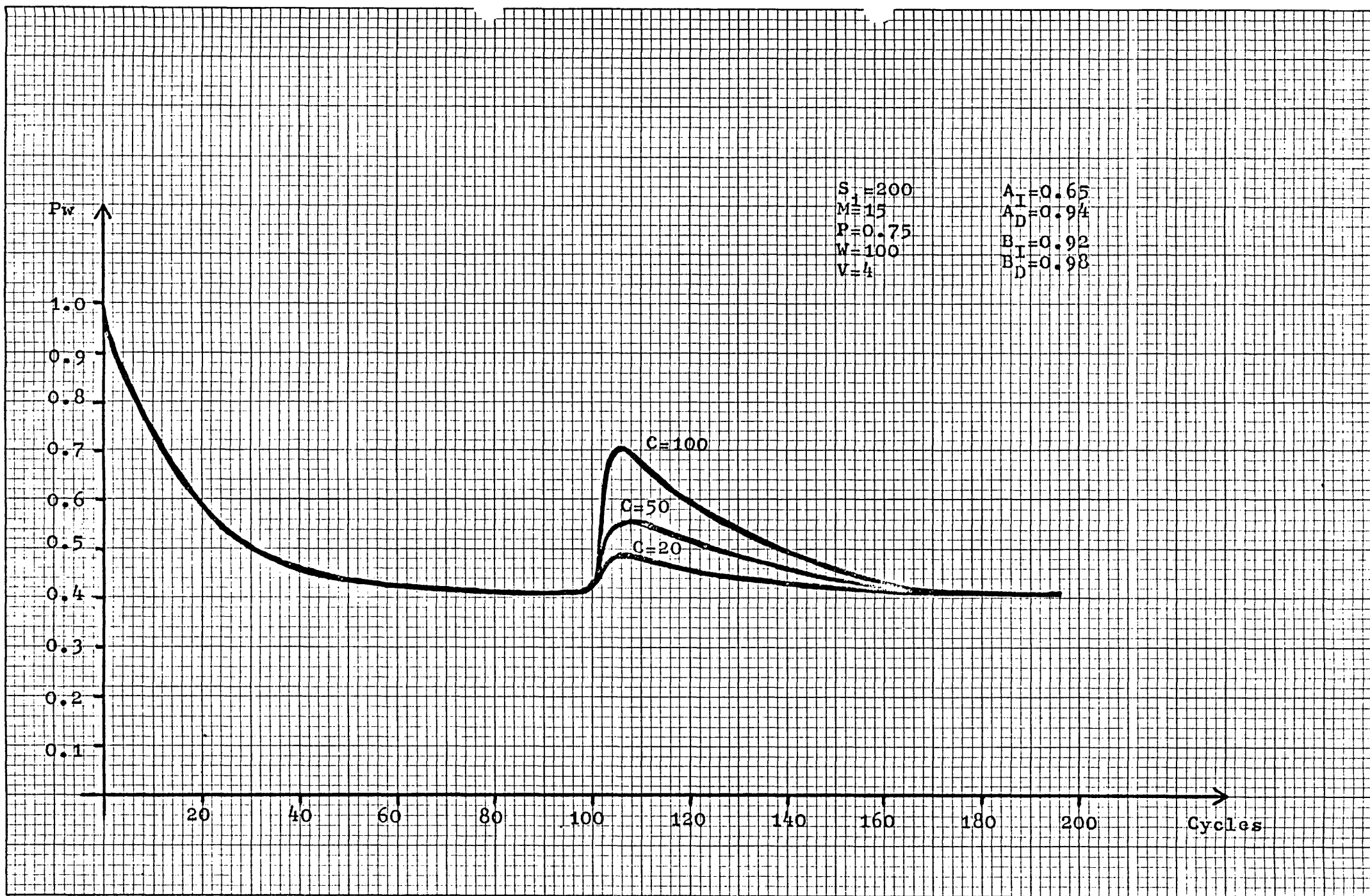


Figure 5.6 Influence of C on the probability of a wrong prediction P_w (type 2 system).

The frequency of modifications is most important in conjunction with the parameters A/B which control the primary and secondary values for all the versions in the organism model. These parameters effectively control the system's speed of response and if the response is too slow for a certain combination of values for the other parameters, in particular C and W , then the adjustment may be very poor: a new modification takes place before the main versions of the new rules from the previous modification had all become dominant. At the other extreme, if the response is too fast then the organism model will not keep the dominance of the main version when its frequency diminishes for transient periods, again resulting in a poor performance, particularly with low values for P and V .

In Figure 5.7 the response is shown for three sets of values for the parameters A_I , A_D , B_I and B_D , ranging from a fast to a slow one. These parameters only affect the response between consecutive modifications, but this part is also influenced by the parameters S_I , M and P (and indirectly by C , which affects the initial starting value following a modification). The organism parameters therefore can only be decided in relation to a particular set of values for the environment parameters. In practical terms, the reason for this is that the formulas which control the primary and secondary values of the various versions (the secondary values ultimately control the selection, and thus the performance) are time functions:

these values are influenced by both the organism parameters and the frequency of occurrence (relative and absolute) of these versions, which is set by the environment parameters.

For a given set of environment parameters and a required speed of response for the system, the range in which each one of the A/B parameters can lie is usually very narrow, but less so for A_I/A_D than for B_I/B_D (because the latter have a direct influence on the selection of the versions). It is possible to some extent to maintain the same response by changing the values of two or more of these parameters at the same time, but there are two features which are always present.

First, we must have the relations $A_D > A_I$ and $B_D > B_I$. As seen in the values given with the graphs, the difference is not large, but since the parameters are used in exponential functions it is significant. The reason for these relations is the fact that in all practical situations the total number of rules (and thus of versions) is very large compared with the average number of rules used in a cycle ($V \times S_i \gg M$). This means that even for the main version the frequency of occurrence in the sequence is very low, many cycles being required on the average for each occurrence. Both the primary and the secondary values must therefore decrease much more slowly than they increase, otherwise they would drop to zero soon after their rise (in fact if the primary value drops immediately to zero, the secondary value will hardly rise

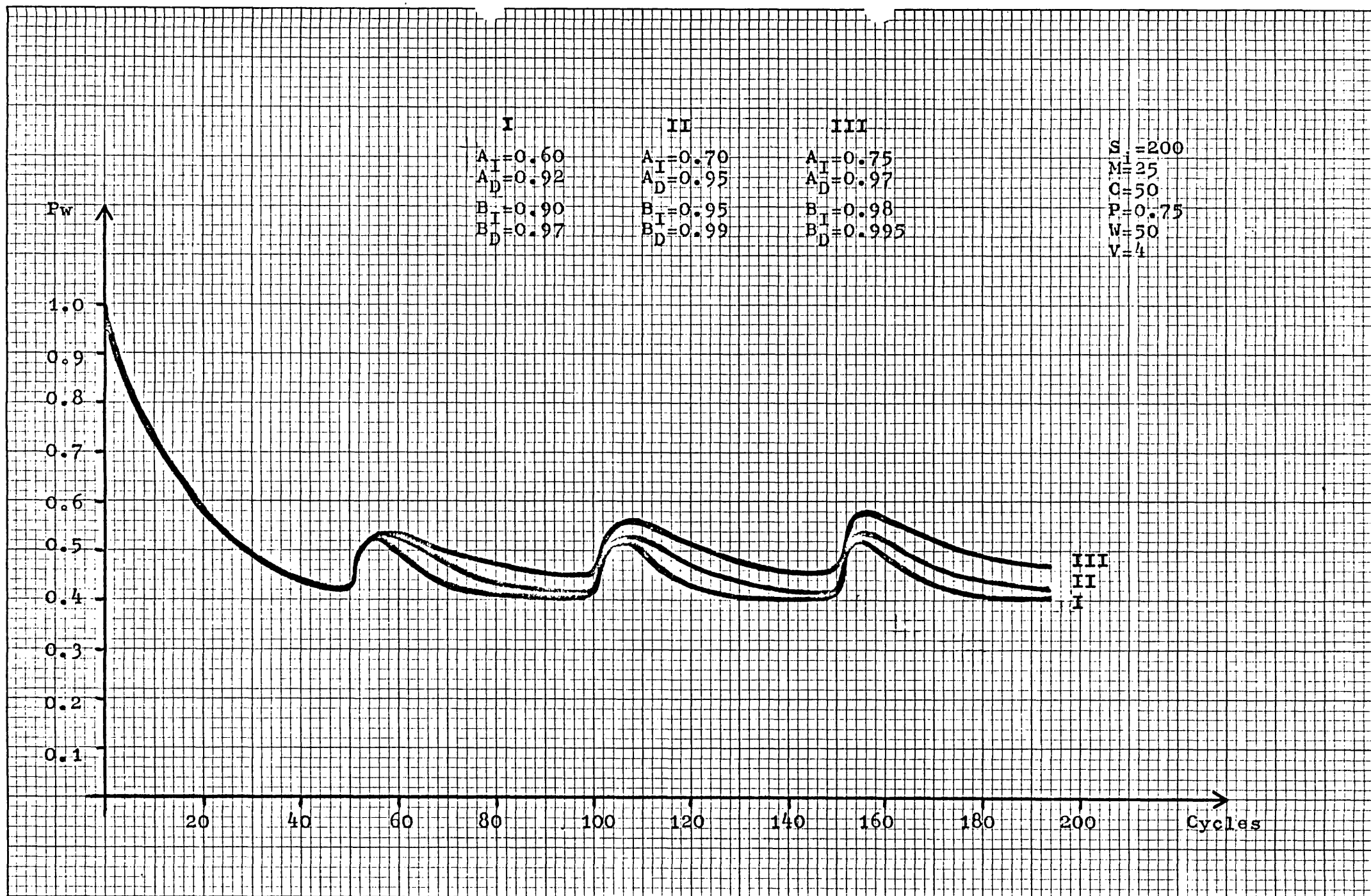


Figure 5.7 Influence of $A_I/A_D/B_I/B_D$ on the probability of a wrong prediction P_w (type 2 system).

at all). Clearly no meaningful comparisons among the secondary values can be made for selection purposes, unless they are above a certain minimum and there are significant differences. (These requirements are also needed for the decision of which versions to delete from the list, for efficiency purposes - a secondary value below a certain minimum must mean that the version had not occurred for a long time and had probably been changed).

The second common feature is the relations $A_I < B_I$ and $A_D < B_D$. This means that the changes in the primary values are much faster than in the secondary values (again, although the difference is not big, the exponential functions make it significant). These relations are implied by the different tasks the two values have. The primary value has to closely follow the changes in the frequency of occurrence of all the versions in the sequence, so that none of these changes is overlooked. The secondary value however, which is used for selecting the version to be used, must be more discriminating: the dominant version should be exchanged for a new one only if the frequency of the latter had been higher for a relatively long period, and this slow process requires relatively high values for B_I and B_D .

*

We must reemphasize the interpretation of the graphs

for the system response. As explained earlier they were obtained by calculating the average values (for the proportion of wrong predictions in each cycle) from several experiments made with the same parameters but with different random values, where such values are used in the environment sequence. (The averages were calculated by a special program which received the results from the individual experiments.)

Such a graph does not represent then the actual system response for a particular set of parameters, but rather the probability for such a response in each cycle, specifically the probability of a wrong prediction for any characteristic pattern - or any state - in that cycle. It is important to realize that the response on any one occasion is never as regular as in these graphs, because of the randomness in the sequence of states. When taking the average proportions, the differences between individual sequences are eliminated and only the common, general tendency remains.

In particular, when there is a modification in the succession rules the graphs show a sudden increase in the probability of a wrong prediction, followed by a gradual improvement. This is true only for the average of course - in each individual sequence the proportion of new rules (and thus of errors) in the cycles immediately following the modification may be smaller or larger than the average. For a specific case then, the graph may for instance rise gradually after a modification, before

it decreases, or show a sudden increase even higher than the average. Even in the steady-state (the type 1 system included) there has to be some irregularity in the individual cases because of the random length of each cycle (M only sets the upper bound).

The average response is essential for studying the behaviour of the system with each set of parameters, because individual cases are meaningless for this purpose. On the other hand, such a response should not be expected from the system if we are only interested in one particular experiment.

*

In Chapter 2 we showed that the system organism-environment belongs to the category of feedback control systems, which are all considered in cybernetics as equivalent, whether they are natural or artificial. The common feature of these systems is their tendency to achieve stability, and the natural organism-environment system that we are simulating is called ultrastable because of its capability not just to reach a stable state but also to use information from the past interaction between its two parts for better performance in the future. We also showed that from the three levels of stability in this system - the homeostatic processes, the organism's adjustment to the environment and the environment's adjustment to the organism - we are

only simulating the second one, this being the characteristic responsible for the higher mental activities observed in such systems.

It is interesting to see now after the implementation of the simulated system, that its behaviour closely resembles that of a feedback control system. We have designed it so that the requirement for the organism's adjustment to the environment is fulfilled - by requesting it to learn to predict the environment states. But since this is equivalent to requesting it to ensure the system's stability, the resulting simulated system can be regarded as a feedback control system, both from the viewpoint of its implementation and of its response (we are referring to the type 2 system, but with sequences of either type). The feedback needed for stabilization is provided by the values associated with the various versions of the succession rules and by the selection mechanism; the error to be corrected is obviously the number of wrong predictions. The role of the A/B parameters can then be naturally interpreted as setting the correct speed of response of this control system (a problem important in all feedback control systems, for precisely the same reasons we gave in Section 4.1.2 for the organism response).

The peculiarity of this system, as mentioned in Chapter 2, is the fact that the observed errors are not used for the correction of the immediate response. Each succession rule is treated separately and the improvement in the system's response applies to the occurrences

of individual rules, which are spread throughout the sequence. The result is that the correctness of the response at each time step is unrelated to the response at the previous or next time steps. This feature is intrinsic to the relation organism-environment and is an accurate simulation of the natural system, where the occurrences of any particular situation that the organism has to master are spread over long periods of time and interspersed with other, unrelated situations.

Here again we see the interpretation we have to give to the average results of the response in several, similar experiments. The response in an individual sequence is composed from the separate responses to each one of the rules, and this makes it irregular because all these responses are interlinked throughout the sequence. These irregularities are smoothed in the average response graph where the separate rules lose their influence and the general trend for all of them becomes clearer.

CHAPTER 6

CONCLUSIONS

We will summarize here the approach we have proposed for a general model in artificial intelligence, and see what problems still have to be solved. We started from the assumption that any task an artificial intelligent system would have to solve lies within the capabilities of the natural intelligent system, and therefore it is better to simulate directly this natural system rather than try to find separate solutions for the simulation of each task. In addition to the usual advantage of a general model - that of providing a solution for several problems at the same time - this approach is helpful because it reveals some general aspects which cannot be recognized when dealing with these problems separately.

These common aspects become apparent when we consider the various intelligence tasks as an interaction between the organism and its environment. There is no need to simulate the actual internal processes in the natural organism - it is enough to know that the object is always to achieve an efficient adjustment to the environment. This requirement is essential for the survival of the organism and we adopted it as our general definition of intelligence. We have seen that this can also be interpreted as a requirement for the stability of the system, if we regard the organism-environment as an "ultra-stable"

feedback control system. The problem of simulating any aspect of intelligent behaviour can then be reduced to that of designing a model whose behaviour exhibits a process of adjustment to the simulated environment.

The interaction between the organism and the environment in the natural system is achieved through several organs and we showed that in the simulated system these input/output signals can be coded into the convenient representation of organism and environment states, and that the role of the organism is then reduced to learning to predict correctly the sequence of environment states. The states, which are only defined at specific time steps when the signals are relevant, contain all the information necessary on the particular process being simulated.

Our next step was to find a way for representing all the environment sequences the organism is likely to encounter. We needed this for two reasons: first for studying the behaviour of the integrated, simulated system, and second for having a ready environment generator for simulating the real environment in an actual application.

Again, rather than analysing all possible situations in the real world, we defined several types and levels of complexity which must logically include all real environments. These definitions were based on those features which must exist in any environment - uncertainty and predictability - both being required at the same time if the environment is to provide novelty and complexity while still being

understandable to the organism. The repetitiveness provided by the succession rules, the different versions for these rules, their periodic modification, and the various random factors are the way those common features are simulated, while the parameters provide the means for producing sequences with different characteristics, to match a specific process.

The extension to the higher levels of environment is based on time patterns. At the first level the elements in the construction of the sequence are the states, which represent the pattern of the relationship organism-environment at certain instances in time. Both the repetitiveness and the uncertainty are at the level of these simple patterns. For higher levels, the elements are patterns formed by sequences of states, i.e. patterns in time; at each level these elements are based on the patterns of the previous level, producing thus an increasingly higher complexity. The construction of the environment sequence remains unchanged as far as the repetitiveness and the uncertainty are concerned, but these features always apply to the characteristic time patterns of each level.

The logic behind these extensions is that since the individual states contain all the possible information that can be transmitted instantaneously between the organism and the environment (for a given set of input/output channels and ranges of signals), the only way to increase

the amount of meaningful information is by including the time dimension. While the individual states still represent the smallest logical step in the simulated process, the higher level sequence is based on more complex time patterns, each one thus containing more information. The general format of the environment, although at a higher level, must be the same because the increase in complexity should not detract from its predictability: a higher level environment must need a higher level organism to cope with it, but the system as a whole should be able to achieve the same degree of stability as a lower level system.

We have designed organism models for the different levels of environment and provided parameters for enabling them to adjust to the various combinations of characteristics set by the environment parameters. We studied then the behaviour of these integrated systems and ascertained that the only restrictions imposed by the higher levels of complexity are on the implementation.

In spite of its apparent logic, this development cannot be considered of course as a proof that any real-world interaction between an organism and its environment can be simulated in this way, particularly for the higher mental activities of man. It is not known whether it is at all possible to find a formal representation for all kinds of such interactions. The types and levels of environment we have defined can clearly be used for

simulating many real-world processes, although it might be necessary to simplify these processes to some extent in order to put them within the capabilities of the environment generators. Those processes which cannot be simulated must be converted into sequences resembling our predefined environments at a certain level, and then the organism models can be used in the same way.

An obvious and straightforward application could be for pattern recognition and a simple example of this is given in the Appendix. A "training" sequence must be provided, with the patterns and their identification represented by the two elements of the succession rules; after that the model can identify similar patterns, and new rules can be inserted in the sequence constantly. A preprocessing stage is needed for reducing the patterns to a subset of possibilities to match the number of rules allowed. The complexity of the patterns themselves can be translated into either complexity of states or of time patterns. Such a system can learn by itself to recognize similar patterns within a preset limit of accuracy and without any previous knowledge (and of course the interpretation of any of the patterns can be modified at any time).

Other, less obvious applications which can use a similar principle are game playing, the solution of logical problems and maybe highly complex behaviour such as understanding natural languages. The interaction with the environment can always provide all the necessary knowledge

and amendments and it is not even necessary to distinguish between the "training" and the "production" periods - the performance of the model will simply depend at any time on the proportion of new data.

It is best initially to test this method for specific applications only, in order to simplify the problems of interface with the environment. But such a system can be adapted to any number of different applications, and can cope with them simultaneously, by using the same internal mechanism. Since the internal representation of the environment sequence is in the form of patterns irrespective of the form of the original interaction (number of channels, type of signals, etc.), the only part of the system that has to be specialized for each application is the coding and decoding between the external signals and the states. Apart from this, the resulting patterns and rules can be processed and stored together without any danger of confusion, the patterns being in fact self-identifying units: each state represents a unique combination of signals transmitted between the organism and the environment.

Such a system would be truly general and a direct result of the approach we have been following, namely a simulation of the integrated natural intelligent system. It is very unlikely that the brain uses different parts or different methods for dealing with each aspect of intelligent behaviour and there is no reason we should scatter research effort in different directions for each

such aspect. Not only is a research which is concentrated on a well-defined concept more efficient and the final multi-purpose system a more attractive goal; as pointed out in Section 1.2, only by studying the intelligent behaviour as a whole is it possible to detect the basic processes which govern it.

*

The use of the approach we have proposed in practical applications must be the subject of further research. It cannot provide an instant solution to all artificial intelligence problems. Rather, it must be viewed in a more general sense: all such problems can be considered as an interaction between an organism and an environment, but in each case the most convenient and efficient way of implementation has to be found. The environment is the process towards which an intelligent response must be shown and it should be either simulated or translated into an adequate form. The organism is that part of the system which must "solve" the problem of intelligence, by learning to predict the environment sequence. It may or may not be possible to simplify the actual environment into one of the sequences we have defined; if it is possible, the corresponding system can be used, otherwise a similar system must be designed, with an organism model to match the required environment sequence, and with a role identical

to that of the organism model we have studied.

We can see then that even if our simulation proves to be an oversimplification for some of the real-world processes, the fact we have already emphasized several times still holds, namely that the organism and the environment can only be considered together as an integrated system, and that the intelligent behaviour is not a function of the level of complexity, or of the kind of process simulated. This is of course a result of our original decision to regard intelligence as equivalent to the basic requirement for adaptation, common to all organism-environment systems. We have seen the implications of this fact for our implementations - the prediction mechanism was the same at all levels and naturally the responses were similar. The features which are particular to a certain level of complexity are internal to the system and recognized by both the organism and the environment, but they cannot be identified from outside the system and do not affect its behaviour.

All this does not mean of course that it is equally easy to achieve a certain level of intelligence, i.e. a certain accuracy in the prediction of the environment sequence, regardless of the level of complexity. The price paid for achieving the same response at a higher level is a decrease in performance from the implementation viewpoint. This is a direct result of the fact that larger amounts of information have to be manipulated while

the process is basically the same. The generality of this approach must therefore be viewed in the context of its implementation.

*

Traditionally there have been two views on the future of research in artificial intelligence. The negative view maintains that computers will never attain the level of human intelligence either because of practical difficulties or simply because an artificial system, being produced by man, will always lag behind man's intelligence. The positive view on the other hand maintains that artificial intelligence programs will in time be able to cope with any aspect of intelligent behaviour and after a certain point a self-development can be initiated, enabling thus the artificial intelligence to surpass the human one. This argument is not going to be settled for a long time yet and it is important to understand its meaning. In the context of the conventional approach in artificial intelligence research which we have discussed in Section 1.2, the argument usually refers not to the advance in our understanding of the natural intelligent behaviour, but only in our ability to simulate it. In other words the argument is whether or not it is possible to develop a software and hardware technology which is powerful enough to match the natural intelligent system. And this is in

fact a translation in terms of artificial intelligence, of a common problem valid in all fields of computer science.

The same argument has a different meaning however when the research is done according to the approach we have proposed. Whereas the conventional specific-task research is mainly concerned with the practical means of simulating partial effects of intelligent behaviour, our object is first to understand the mechanism of this behaviour and then to attempt a straightforward simulation of the whole process. In the former, a successful implementation also means a solution for that particular problem (though not necessarily for other aspects of intelligence). These solutions depend therefore directly on the advance in hardware and software technology, and this is actually the process we have been witnessing in the past. In the latter, the solution itself is independent of the advance in the other fields and only its implementation will have to rely on them; but this, if at all feasible, should be a trivial matter.

While looking for a quick solution to separate aspects of intelligent behaviour, the adepts of the conventional approach tried to ignore the natural mechanism, trusting in the power of their machines. They went deeper and deeper into technical details by trying to find in the separate aspects something which is only apparent in the integrated system. The more they simplified the problems, the farther they got from their solution and what started

as a short detour into software development has become a full-time occupation. The approach we are proposing avoids these pitfalls by concentrating on the intelligent behaviour of the integrated natural system. This is a difficult project which cannot provide a short-term solution, but when one is found it will be for all aspects of intelligence.

CHAPTER 7

RELATED WORK

The brain is the best information processing device we know of. This has been recognized long ago and it is not surprising that, as interest in computers increased also the research into the mechanism of the brain increased - either within artificial intelligence, or in psychology. This common research contributed to the creation of cybernetics. There are several approaches in this research and a favourite one, especially in the early days of the computer, was based on the apparent analogy between the brain and the computer from the viewpoint of their operating principles. The usefulness of this approach is doubtful, as explained in Chapter 1; even if we understand the biological mechanism of the brain it will be inefficient to implement it on a digital computer. It is advantageous to study only the high level activity of the brain and to regard the computer merely as a tool for simulating this activity, without caring whether or not the low level operations are similar to those of the brain.

The classic work on the brain-computer analogy is probably that of von Neumann²⁰. He discusses in detail the similarities and differences between the two systems from the viewpoint of their construction, mode of operation

and capabilities. Although there are many common features, there are some fundamental discrepancies which produce strikingly different overall performances. He points out in particular the basically different internal representation of information - the natural system uses much more redundancy and sacrifices accuracy for the capability of generalization, complex pattern processing and protection from damage. Wooldridge²¹ on the other hand emphasizes a fundamental similarity, namely the fact that the operation of both the natural and the artificial systems is based on the execution of a large number of simple, primitive steps, at high speeds. Other similarities are the presence of innate and learned functions and of feedback mechanisms for controlling the performance at several levels.

Sutherland²² discusses this subject from the viewpoint of the machine, showing that since it works according to principles basically similar to the brain it should be able to perform the same intelligent tasks and it is only a matter of time before ways are found for simulating all aspects of intelligence. George²³ approaches the subject from the viewpoint of the brain. He shows how the various facts normally accepted in psychology can be observed by viewing the brain as an artificial device, similar to a general purpose computer.

Much work has been done involving the use of computers for demonstrating and testing theories of the action of the brain. Although originally started by

psychologists, this research is recognized now as a field of artificial intelligence and it is extremely useful for increasing our understanding of the nervous system. Those who work in this direction are not concerned with practical applications of their computer models; the purpose of these models is mainly to simulate in a convenient way theories which would be more difficult or impossible to validate by other means.

A good example is the work of Rochester et al.²⁵ who showed by implementing the model proposed in Hebb's²⁴ cell assembly theory that some results are different from those expected. Hunt²⁷ describes the research being done on models derived from psychological studies and Holland²⁸ shows how we can use our knowledge on the structure of the brain in order to design more efficient artificial intelligent systems.

Doran²⁶ suggests that we should pay more attention to the natural intelligent system before designing artificial ones. Very little is known of its way of operation and little research is done in the way of seeing the brain in the general class of intelligent machines. Furthermore, we know virtually nothing about the relation between the structure of the brain and its working procedure.

The study of brain models and their simulation is very important in its own right but, as explained in Chapter 1, we prefer a more direct approach to the problem of building

an artificial intelligent system. We prefer to study the interaction between the brain and the environment and the behaviour of this system as a whole rather than regarding the brain as a separate part. We are using therefore an inductive method for understanding its mechanism rather than the deductive methods necessary when the brain is studied on its own. We are less concerned with an accurate simulation of the actual action of the nervous system, and more with the simulation of its interaction with the environment.

Closer to our approach are the studies made with learning models, i.e. models of the nervous system - at either a low or a high level - which modify their internal structure following their interaction with the environment so that their performance improves in time. This idea has been used in many applications but the principle is basically the same - the problem can always be described as one of concept formation. The model is presented with a sequence of signals in which it has to identify certain rules and use them in future instances. Hunt²⁹ discusses the requirements in such learning models. They need an internal representation of the features of the environment and the various objects or goals, which depend on each particular application. Decision trees may be formed and heuristics used for managing the information collected from the sequence and for deciding on the response.

A simple application is described in Feldman³⁰,

namely a model for the simulation of the binary choice experiment, one of the most famous in psychology. The model is presented with a sequence of signals and there are only two alternatives for its response. There is no previous knowledge - as in experiments with human subjects the sequence of signals is the only source of information. Learning models however have been used mainly in conjunction with the problem of pattern recognition, i.e. the model has to learn to recognize patterns by analyzing their features, sometimes with and sometimes without previous knowledge. The program of Selfridge and Neisser³¹ attempts to recognize hand printed capital letters but although it can learn from its own experience by modifying a number of operators, it is very inflexible in the specification of the features looked for in the letters. The program of Uhr and Vossler³² is more sophisticated: in addition to the learning achieved by the modification of operators according to past experience, the program also develops its own test features. It relies thus on less preset knowledge and the performance is improved by the additional levels of learning.

As explained in Chapter 2, the main characteristic of a model which relies more on its interaction with the environment than on previous knowledge is the fact that it can display adaptive behaviour. In other words it can not only acquire knowledge about the environment, but also modify this knowledge at any time still using the same

mechanism. Sears³³ and Kazmierczak³⁴ discuss this concept and although their works are in the field of pattern recognition the same principles apply to all aspects of learning. Indeed, the field of pattern recognition - which parallels that of perception in psychology - has been a favourite one for studies of learning models, as it has been for most other problems of artificial intelligence. The experimental results from psychology were particularly useful for the theories on which these models are based. White³⁵ and Steinbuch³⁶ emphasize this close relationship and present the common aspects of adaptive models, as well as techniques for solving the basic problems.

In all these models there is a certain amount of preset, "innate" information. They can all improve their performance by interacting with the environment but they have some knowledge at the beginning; this knowledge is usually in the form of an increase in the sophistication of the model through more detailed specification of the features it has to extract and analyze from the sequence of patterns, and the way it has to modify its operators as a function of its interaction with the environment. This is another way of applying the same principle we have followed in our approach, namely the organism model cannot be studied separately and we must take into account the expected environment features when designing the method by which it has to adapt. We must distinguish of course between this form of previous knowledge which is indispensable

for the organism (and comes in the form of "hardware" specifying what and how to learn rather than specific information) and the innate knowledge which, although may also be in the form of "hardware", is similar in nature to that acquired later. It is this kind of innate knowledge that we chose to ignore in our model and which is generally irrelevant in all studies of adaptive systems.

The neuron networks are another field of research in the general area of learning machines. These networks - usually called perceptrons - mimic the internal structure of the brain which is viewed as a large number of randomly interconnected simple devices. The learning process in such networks is achieved by modifying the connections between the individual elements. The neuron networks followed the classic "cell-assembly" theory of Hebb²⁴ and the theory of neural nets proposed by McCulloch and Pitts³⁷, both based on the principle that the stored information is distributed throughout the brain rather than being localized.

The first results were very encouraging and much research was done with relatively simple networks, e.g. Rosenblatt³⁸ and Uttley³⁹, which were followed by many other works. Again perception, or pattern recognition, was usually the problem these machines had to solve. Farley⁴¹ discusses several aspects of the application of neuron models in pattern recognition, and Warmon and Lewis⁴² bring examples of their use in the general field of learning.

Unfortunately the later results were disappointing because the machines could not be extended to more complex networks and their performance deteriorated rapidly for the more complex tasks. Minsky and Papert⁴⁰ have suggested that this is due to our failure to understand some fundamental principles - which in fact apply to computer science in general - and they set out to study the general theory of perceptrons, disregarding any practical applications they may have in specific situations. They argue that it is useless to experiment with the simple networks ignoring the limiting factors which assert themselves when the model is expanded in a practical application.

This problem is of course common to all simulations - unless they are based on a solid formal background it is impossible to prove, or even predict their performance for the various extensions and ramifications until these are implemented. Our approach is not immune either and it may very well prove to be impracticable for the simulation of complex intelligent behaviour without introducing some simplifications in the real-world environment. This is one of the reasons we consider that future research must concentrate on its implementation for specific applications since at this stage it is necessary to take into account those aspects which cannot be foreseen in a general model.

Nevertheless we believe that this approach is more likely to succeed in complex situations, because it is only based on the simulation of the observed, high level behaviour

of the natural system, whereas such models as the perceptron are based on certain assumptions on the low level mechanism of the brain. It had been assumed that it is possible to simulate the entire structure of the brain following the understanding of its elementary parts but this was not the case. It is pointless to try to implement a system before fully understanding it and at the present stage we only understand the high level behaviour of the brain. Rather than trying to correlate it to fragmentary knowledge of the low level operations we must concentrate on its simulation from an external viewpoint and disregard whether or not this is the way the brain actually works. Studying the behaviour of the system is more important than studying its internal structure.

*

There are several studies which discuss the behaviour of a learning device in a changing environment. Hunt²⁹ points out that the crucial problem is that of a device which interacts continuously with the environment and reacts to it, improving its performance not only from one instance to the other but also from one problem to the other. For a general purpose model like the one we have developed there is no distinction between different problems because they are all presented in the same way in the environment sequence, as described in Chapter 6. Such a general model would however be impracticable and it has to be tailored for each individual application.

Young⁴⁴ only discusses the natural system but he analyzes its behaviour cybernetically. The brain is part of the control system of a homeostat and as such its task is, like in all other control systems, to make a choice between alternatives, this being a process basic to life in general. The result is an increase in the correspondence between the system and its environment. Although we know that all adaptive changes depend on such processes of choice, we do not fully understand how they operate at the level of the brain. Ashby¹⁶ also presents this subject from the same point of view and in addition he describes the mechanical implementation of an "ultra-stable" homeostat. We have explained in Chapter 2 how our definition of intelligence corresponds to the requirement for stability of the system organism-environment. Another implementation which demonstrates the principle of stability was made by Walter¹⁷ in the form of mechanical "tortoises" which simulate the behaviour of primitive animals.

Apter⁴⁵ shows that the computer models of behaviour have to be viewed as simplified forms of the natural system. He emphasizes the importance of studying the behaviour of the organism as a whole when inside its environment and shows that evolution and development can be considered as a form of learning when the organism and the environment are viewed as an integrated system. Pringle⁴³ also points out the similarity between learning and evolution - both

involve an increase in the complexity of the system. We have pointed out in Chapter 2 however that in our model we ignore the changes which act at the level of the species, since they are too slow to affect an individual organism.

Klir and Valach⁴⁶, although regarding the organism as a special part of the system and the environment as an external, separate part to which the organism is connected, reach the same conclusion we have reached in our definition of intelligence. They show that all living systems can be viewed as information processors, their only purpose being in effect to anticipate the future and so prepare for expected changes in the environment. The learning process is then the acquirement of this capability through interaction with the environment. It is this quality that is called "self-organization" or "ultra-stability" - this is a higher level of adaptation, or an improvement in the adaptation capabilities.

The same subject can be approached from the viewpoint of an artificial intelligent system rather than a natural one. Pedelty⁴⁷ and de Latil⁴⁸ show that a machine can display intelligent behaviour when presented with a changing environment. Since they both regard the machine-environment cybernetically their conclusions are similar to those for a natural system. Ashby⁴⁹ also discusses aspects of the interaction between a machine and its environment and emphasizes that it is impossible to study them separately. Indeed, any closed system can in fact

be regarded as composed from two parts - one corresponding to the machine, or organism and the other to the environments.

Doran^{50,51} describes the computer simulation of the behaviour of an animal in its natural environment. Both the organism and the environment are made as life-like as possible, with various states defined to represent situations which would occur in the real world, e.g. hunger, obstacles and temperature. The organism's behaviour is based on seeking to satisfy its needs and it can learn from its previous decisions. Similar simulations are described in Jacobs⁵² and Findler⁵³. In all these models the organism starts with a blank memory and learns by itself from its failures and successes how to adapt to its environment. The simulation includes input and output for the organism, several levels of memory and a mechanism for evaluating its interaction with the environment. The environment is represented as a matrix with certain values for the various objects and features simulated and the organism acts inside this matrix.

There are some obvious differences between the approach adopted in these organism-environment studies and the approach we followed for our model. Most simulations are concerned with real life aspects of the interaction with the environment and therefore they are specific simulations of the behaviour of an hypothetical organism. In addition, while the main object in our model is a continuous adaptation to a changing environment, the object in the above studies

.

is usually for the organism to achieve a certain goal given certain constraints, hence its behaviour is more a display of problem solving than of adaptation; this is unavoidable if the project is regarded as a specific-task situation. Most important of all, the simulation is of the behaviour of the device in an environment as seen by the device itself, while we have stressed that we regard the organism and the environment as similar parts (at least externally) and used the full knowledge of one part's features when simulating the other.

*

Theoretical models and computer simulations in particular have always been recognized as a convenient way of representing theories of learning, perception and intelligent behaviour in general. The common view is that they are simplified but accurate forms of the systems they are modelling and more concrete forms of abstract ideas⁴⁵. Nonetheless the use of models for developing a theory must be justified because it is not always obvious whether its extension to more general aspects or to a practical application will be possible. Hilgard and Bower⁵⁴ for instance, while showing the importance of models in the study of behaviour, point out the danger involved in this approach and already mentioned in Chapter 1. It was thought that complex behaviour can be understood by studying first separate aspects under simplified conditions. It is now recognized however that this approach has failed and

we must try to model directly the complex intelligent behaviour.

Popper⁵⁵ also argues that models of theories cannot be proved to be true. What can be done is only show that they are consistent with the observed facts. Different models can only be compared according to their performance in relation to the modelled system. Klir and Valach⁴⁶ define the model as an instrument - the fundamental concepts remain the system, its behaviour and structure.

In spite of their limitations models are invaluable in the study of complex systems like those representing intelligent behaviour, as in the case of Rochester et al. mentioned earlier. In addition, one can chose to concentrate on the simulation of those aspects which are of interest, while representing the others in the most convenient way. Chomsky⁵⁶ for instance makes a clear distinction between the observed linguistic behaviour and the actual linguistic competence and this probably applies to any other mental behaviour. The implication is that we can build models of the mental competence and if these models are correct, we ought to obtain the same external behaviour displayed by the real system. This view is of course identical to the one we have expressed when we decided to ignore the internal mechanism of the natural system and to simulate only its behaviour towards the environment.

Several models of human problem solving have been proposed and most of them have also been implemented.

They are all based on the principle of goal-directed behaviour and the division of the major goals into subgoals, etc. Newell and Simon⁵⁷ describe their long project which resulted in many implementations. Problem solving can be understood in terms of a task environment, a space in which the problem solver represents the environment, the task and the knowledge he accumulates, and finally the program the problem solver uses for approaching the task. Information is extracted from the task environment and the solution is found by means of selective search through the problem space. Several examples are given of the performance of the simulated system for various applications. The drawback of this model, like all other general purpose models, is that it cannot by itself find the common internal representation for the different problems. Nevertheless it is an example of how a general model can be designed to accurately represent a theory of intelligent behaviour.

A similar attempt is described in Reitman^{58,59} where the model is derived from an earlier version of the GPS and from Hebb's cell assembly theory. This project introduces such concepts as parallel processing and loss of information in the general goal directed behaviour, in order to account for evidence observed in human problem solving. Miller et al.⁶⁰ and Klix and Sydow⁶¹ suggest a theory in which the goal directed behaviour is structured hierarchically. The most elementary goals

should be considered as the units of behaviour and thus any problem solving activity can be represented by a temporal arrangement of subgoals, at several levels, the same strategy being used at each stage. As already mentioned, although this idea has been used in many implementations there is still no proof that complex mental behaviour can be represented in this way.

For simpler forms of behaviour however, or for representing only certain aspects of intelligence, the models and their implementation have proved to be quite adequate. Willshaw and Longuet-Higgins⁶² for example describe a model for pattern recognition in which the emphasis is on the simulation of features observed in the natural system, namely influence of damage and forgetting and the way these effects are overcome. Klahr⁶³ also concentrates on perception and describes experiments with a model based on grouping the information from input and applying certain quantification operators for producing the output.

*

An important problem we have encountered when designing our model was that of generalization, or abstracting of the patterns in the environment sequence. This is a fundamental problem in any intelligent system, whether the application is pattern recognition, learning or any other form of intelligent behaviour. As explained in Chapter 4 we bypassed this problem in the simulation

of the general system by evaluating the organism's predictions on a yes-or-no basis rather than according to "closeness". This followed our decision to work directly with states, ignore the interface with the environment and assume that a pattern can also represent a class of patterns. We pointed out that the problem of generalization is best solved for individual applications because it is in effect a problem of optimization - a trading of accuracy for efficiency - and therefore subject to attributes which cannot be considered in a general system. (In the Appendix we show an example of this problem in a specific application.)

Hunt and Hovland⁷⁴ argue that the reason the problem of generalization appears in all aspects of intelligent behaviour is that such behaviour always involve some form of concept formation - the machine has to map stimuli from predetermined subsets into particular responses. Although theoretically it is the same problem that must be solved in all cases, for practical reasons it is usually solved individually. The concept we have called "closeness" is discussed as "property classes" in Farley⁴¹ and as "strong discrimination" in Keller⁷⁵. Johnson⁷² also mentions that although the codes the brain uses when processing information seem to be unique, they are evaluated on the basis of closeness and not by a yes-or-no comparison. And Pedelty⁴⁷ who uses the term "level of awareness" for

the concept of generalization argues that the information interface between a machine and its environment must be adequate for the level of sophistication of the machine. In other words the capability of classifying the input signals must be correctly related to the internal processing capabilities.

When we defined the structure of the environment in Chapter 3 we saw that the main problem was to represent all the main features found in a real-world environment. It had to contain a certain amount of repetitiveness in order to be predictable but it also had to contain some randomness to represent the changes to which the organism has to adapt. The levels and types defined made it possible to represent various degrees of complexity, a more complex environment containing in effect a larger amount of information. There is a strong evidence that the interaction between the brain and the environment is based on such an organization, i.e. the natural intelligent system is so built that it recognizes in the environment constructions similar to those we have defined. Miller^{64,65} shows that the brain is quite limited in its capacity to process information and that the input from the environment is accepted in "chunks". The efficiency is increased by means of coding these short sequences which are effectively processed by the brain as time patterns. The division of the environment sequence in our model into strings of states intersperced with zero-states was based

on these observations and the use of the same idea for higher levels of time patterns was a natural extension of the coding mechanism. The same idea is expressed by van Emden⁶⁶ who shows that a higher level of complexity can be obtained by combining several systems of a lower level and producing thus a larger amount of information transfer.

The theory of information grouping, or "chunking" of the input sequence, came largely as a result of the problems found when trying to apply the simple associative serial behaviour theory to experimental evidence. Lashley⁶⁷ who first discussed these problems noted that there are complex interdependencies among items in a behavioural sequence which are best described not as associations but as a coding process applied to groups of data and at several levels. Johnson^{68,72} discusses the coding process and shows that it depends on the temporal position of each item in the sequence. The codes are therefore true time patterns employed by the brain in order to facilitate the information processing. Cermak⁶⁹ and Herriot⁷⁰ present the concept of several levels of coding and suggest that the information contained in the input sequence is preserved and transferred through each stage of encoding up to the highest level. This results in a more efficient storage and retrieval of information than would be possible with a purely associative mechanism, but they admit that it is not fully understood how this transfer of information is performed. Posner⁷¹ shows that there is some experimental evidence

for the existence of parallel, multiple codes, i.e. the information is stored in more than one form and there are several encodings for each item in a sequence, in different contexts.

Since this coding theory encounters some problems which can be solved by the associative theory, it has been suggested that a combination of both might provide the answer. Estes⁷³ develops a model which can explain all empirical evidence and manages to avoid to some extent the main objections against the coding theory, namely that the encoding/decoding processes are never specified and the various levels are obscure. In this model there is an element of sequential learning, while the coding mechanism is retained for an increase in the information processing capabilities.

It is this latter approach that was adopted in our model. Simple associations cannot produce a complex intelligent behaviour but they seem to exist in all real-world environments and account for their predictability. They appear in the simulation in the form of succession rules at the level of characteristic patterns and the coding mechanism is employed for producing the time patterns at various levels. An environment of any complexity can thus be obtained by adding levels of encoding and increasing the amount of information transferred in the interaction with the organism. The learning process always applies

at one level only - that at which the associations occur.

It must be emphasized however that, although based on psychological theories, our model represents a generalization. The psychological findings refer to the brain's response to an input sequence of signals, whereas in our model the sequence is a sequence of states (as defined in Chapter 2) and represents the entire interaction between the brain and the environment.

APPENDIX.

AN EXAMPLE

We have indicated in Chapter 6 that more research is necessary in order to ascertain the practical value of this approach for solving problems in artificial intelligence. Although the approach is general and applicable to any problem, it will probably be best initially to test it in a more limited way, for obvious implementation considerations. The final goal is however to have a system capable of dealing with a variety of problems, the only part specific to each problem being the stage of interaction with the outside world.

Even when applying this approach to a specific, restricted problem only, the only way to see whether for practical applications it has advantages over the conventional specific-task methods is by regarding it as a serious project in its own right and devoting enough effort to ensure the best possible implementation. The example we describe here is only a small scale demonstration and no attempt was made to achieve a certain level of performance. Therefore it would be meaningless to compare the results with those obtained by other programs.

The example involves a simple application in pattern recognition, namely the recognition of handwritten digits

(0, 1, ... 9). The problem is to design a program that learns by itself to recognize the digits from a sequence of patterns and their interpretation. The program has no previous knowledge on these patterns, their number or variety. The digits are contained in a matrix of 30 x 20 elements which can have the binary values 0 or 1, with their obvious meaning. For each digit, 3 to 5 patterns were produced to represent typical, widely different hand drawings. These patterns were subsequently distorted in various ways by a special program, while still retaining their original format. For each digit many different patterns were thus obtained (50 - 100 depending on the digit). Each such pattern was paired with another one which was simply its interpretation as a digit and these pairs (which are the rules in the environment sequence for this system) were stored in a table. The part of the system which acts as environment generator chooses at random pairs from this table and presents them to the part which acts as the organism.

This system is still a simulation of the real-world situation, but this is unavoidable if we want to bypass the various technical details. As noted above, it is only intended as an indication on the use of our approach and the sequence produced is a very good approximation of many different hand-written digits as they would be represented internally after a preprocessing stage.

For this simple application a level 1 type 1 system

is sufficient. There is no need for a type 2 system because we do not expect the interpretations given to the various digit patterns to be changed later (there may always be new patterns in the sequence, but the type 1 organism can of course cope with this by adding the new rules to its table). Also, the disturbance-type changes can be considered as slightly different, new patterns and are dealt with as such in a special way described below.

For complex patterns and interpretations in this kind of application it is worth considering a higher level system, but the patterns we are using are simple enough for level 1. There is no reason to subdivide them into simpler patterns (rows of elements, or submatrices) and therefore they will also be the environment states. The digit interpretations following the patterns as the second part of the rule must also be considered as states but there is no need to bring them to a common internal representation, since they can be easily distinguished due to their fixed order in the sequence. This fixed order also allows us to make another simplification, namely to dispose of zero-states. Since the patterns and their interpretation always come in pairs which have no connection with one another (the recognition is of digits, not of numbers), the length of the cycles in the sequence is always 2 states and the organism needs no zero-states for recognizing them.

The comparisons of the interpretations in the environment

sequence with those predicted is made on a yes-or-no basis, as in the theoretical models. But the comparison of the patterns themselves when searching the table of rules during the recognition process is made according to the original definition, i.e. they only have to be as "close" as possible. This is desirable because there are only ten different interpretations, the recognition process being effectively a many-to-one mapping. Since some of the patterns for the same digit may only differ to a small extent from one another, the comparison based on "closeness" minimizes the number of different rules the organism will have to form and it also takes care of any disturbance-type, small random changes in the patterns, as well as of "noises" in the matrix in general.

In practice the comparison is done simply by counting the number of corresponding elements in the two matrices, whose binary values differ from each other. If the total exceeds a preset maximum bound, then the two patterns are considered to be different. If no rule is found whose pattern satisfies the criterion of closeness, then no prediction is made and the new pattern with its interpretation are added in the table; otherwise the rule containing the pattern closest to the one in the sequence is used in prediction. Clearly a high value for the bound set for closeness means less cases when new rules have to be defined and more cases of wrong predictions. On the other hand a low value results in more accurate predictions but more cases when no prediction is made because the pattern is

considered to be new, this leading to a slower learning process.

In order to further help reduce the overall number of different rules without detracting from the accuracy of the predictions, and also for increasing the flexibility of the bound used as criterion of closeness, the digit patterns are not employed directly as they appear in the environment sequence. Instead they are blurred, i.e. the lines forming the digits are made thicker (from 1 to 3 matrix elements and from 2 to 4). Only the blurred patterns are used in comparison and this is the way they are stored in the table. The result is that patterns which would be considered as very different from one another in their original form, become "closer" because the proportion of matrix elements with different values (out of the elements with the value 1) is smaller. The criterion of closeness can accordingly be relaxed more freely without producing a large increase in the number of wrong predictions.

The system was tested like the models described in Chapter 5, by calculating the average results from several experiments. Here the only difference among the various experiments was the order in which the patterns were presented (always random). The results, which show therefore the probability of a wrong prediction as a function of the number of pairs presented, are similar of course to those of a type 1 system (Figures 5.1 and 5.2). The difference is that for this application there are predictions

only for the states which are interpretations of the patterns, therefore there are no permanent wrong predictions due to zero-states and the random first state in a cycle. The lowest probability that can be achieved in the steady-state is a function of the bound set for the criterion of closeness in the recognition of patterns. It can be as low as 0 but in this case the number of rules is very large (practically each different pair becomes a new rule) and the learning process is very slow. More interesting are the cases when we are willing to sacrifice some accuracy for a better overall performance. For example, for a permanent probability of a wrong prediction of 0.05, only about one third of the different pairs are considered as different rules and the graph reaches the steady state after less than 600 pairs.

REFERENCES

1. Turing, A.M. - Intelligent Machinery, in Meltzer M. and Michie D. eds. Machine Intelligence 5, Edinburgh University Press, 1969.
2. Michie, D. - Machines and the Theory of Intelligence, in Michie D., On Machine Intelligence, Edinburgh University Press, 1974.
3. Uhr, L. - Pattern Recognition, in Uhr. L. ed., Pattern Recognition, Wiley, 1966.
4. Feigenbaum, E.A. - Artificial Intelligence: Themes in the Second Decade, in Information Processing 68, North-Holland Publishing Company, 1969.
5. Samuel, A.L. - Some Studies in Machine Learning Using the Game of Checkers, in Feigenbaum E.A. and Feldman J. eds., Computers and Thought, McGraw-Hill, 1963.
6. Ullman, J.P. - Idealized Pattern Recognition, in Watanabe S. ed., Frontiers of Pattern Recognition, Academic Press, 1972.
7. Ernst, G.W. and Newell, A. - GPS: A Case Study in Generality and Problem Solving, Academic Press, 1969.
8. Kaneff, S. - Pattern Cognition and the Organization of Information, in Watanabe S. ed., Frontiers of Pattern Recognition, Academic Press, 1972.
9. Michie, D. - Programmer's Gambit, in Michie, D., On Machine Intelligence, Edinburgh University Press, 1974.
10. Turing, A.M. - Computing Machinery and Intelligence, in Feigenbaum E.A. and Feldman J. eds., Computers and Thought, McGraw-Hill, 1963.
11. Restle, F. - Mathematical Models in Psychology, Penguin Books, 1971.
12. Apter, M.J. - The Computer Simulation of Behaviour, Hutchinson, 1970.

13. Butcher, H.J. - Human Intelligence, its Nature and Assessment, Methuen, 1968.
14. Wiener, N. - Cybernetics, 2nd edition, MIT Press, 1961.
15. Singh, J. - Information Theory, Language and Cybernetics, Dover, 1966.
16. Ashby, W.R. - Design for a Brain, Chapman & Hall, 1960.
17. Walter, G. - The Living Brain, Duckworth, 1953.
18. Pask, G. - An Approach to Cybernetics, Hutchinson, 1968.
19. IBM, Data Processing Techniques, C20-8011-0 - Random Number Generation and Testing.
20. von Neumann, J. - The Computer and the Brain, Yale University Press, 1958.
21. Wooldridge, D.E. - The Machinery of the Brain, McGraw-Hill, 1963.
22. Sutherland, N.S. - Machines like Men, Science Journal, 4, 10, Oct. 1968.
23. George, F.H. - The Brain as a Computer, Pergamon Press, 1961.
24. Hebb, D.O. - The Organization of Behaviour, Wiley, 1969.
25. Rochester, N., Holland, J.H., Haibt, L.H. and Duda, W.L. - Tests on a Cell Assembly Theory of the Action of the Brain Using a Large Digital Computer, IRE Trans. Inform. Theory, IT-2, 1956.
26. Doran, J.E. - Some Recent Models of the Brain, in Meltzer B. and Michie D. eds., Machine Intelligence 6, Edinburgh University Press, 1971.
27. Hunt, E. - Computer Simulation: Artificial Intelligence Studies and their Relation to Psychology, Ann. Rev. Psych. 19, 1968.
28. Holland, J. - Outline for a Logical Theory of Adaptive Systems, Journal A.C.M., 9, 1962.

29. Hunt, E.B. - Concept Learning: An Information Processing Problem, Wiley, 1966.
30. Feldman, J. - Computer Simulation of Cognitive Processes, in Borko H. ed., Computer Applications in the Behavioural Sciences, Prentice-Hall, 1962.
31. Selfridge, O.G. and Neisser, U. - Pattern Recognition by Machine, Scientific American, 203, 2, August 1960.
32. Uhr, L. and Vossler, C. - A Pattern Recognition Program that Generates, Evaluates and Adjusts its own Operators, in Uhr L. ed., Pattern Recognition, Wiley, 1966.
33. Sears, R.W. - Adaptive Representations for Pattern Recognition, 1965 IEEE International Convention Record.
34. Kazmierczak, H. and Steinbuch, K. - Adaptive Systems in Pattern Recognition, IEEE Transactions on Electronic Computers, Vol. E.C. 12, Dec. 1963.
35. White, B.W. - Studies of Perception, in Borko H. ed., Computer Applications in the Behavioural Sciences, Prentice-Hall, 1962.
36. Steinbuch, K. - Technical Models of Human Information Processing, in Klix F. ed., The Organization of Human Information Processing, Akademie-Verlag, 1967.
37. McCulloch, W.S. and Pitts, W. - A Logical Calculus of the Ideas Immanent in Neural Nets, Bull. Math. Bioph., Vol. 5, 1943.
38. Rosenblatt, F. - Principles of Neurodynamics, Spartan Books, 1962.
39. Uttley, A.M. - Conditional Probability Machines, in Automata Studies, Princeton University, 1956.
40. Minsky, M. and Papert, S. - Perceptrons, MIT Press, 1969.
41. Farley, B.A. - Self-organizing Models for Learned Perception, in Yovits M.C. and Cameron S. eds., Self-organizing Systems, Pergamon Press, 1960.

42. Harmon, L.D. and Lewis, E.R. - Neural Modelling, Psychol. Rev., 46, 1966.
43. Pringle, J.W.S. - On the Parallel Between Learning and Evolution, Behaviour, 3, 1951.
44. Young, J.F. - A Model of the Brain, Clarendon Press, 1964.
45. Apter, M.J. - Cybernetics and Development, Pergamon Press, 1966.
46. Klir, J. and Valach, M. - Cybernetic Modelling, Iliffe, 1967.
47. Pedelty, M.J. - An Approach to Machine Intelligence, Spartan Books, 1963.
48. de Latil, P. - Thinking by Machine, Houghton Mifflin Company, 1957.
49. Ashby, W.R. - Principles of the Self-organizing System, in von Foerster H. and Jopf G. eds., Principles of Self-organization, Pergamon Press, 1962.
50. Doran, J.E. - Experiments with a Pleasure-seeking Automaton, in Michie D, ed., Machine Intelligence 3, Edinburgh University Press, 1968.
51. Doran, J.E. - Planning and Generalization in an Automaton/Environment System, in Meltzer M. and Michie D. eds., Machine Intelligence 4, Edinburgh University Press, 1969.
52. Jacobs, W. - A Structure for Systems that Plan Abstractly, Proc. SJCC, 38, 1971.
53. Findler, N.V. - A Computer Model of Inductive and Deductive Inference Making, in Dalenoort G.J., Process Models for Psychology, Rotterdam University Press, 1973.
54. Hilgard, E.R. and Bower, G.H. - Theories of Learning, Appleton-Century-Crofts, 1966.
55. Popper, K.R. - The Logic of Scientific Discovery, Hutchinson, 1959.
56. Chomsky, N. - Language and Mind, Harcourt, Brace and World, 1968.
57. Newell, A. and Simon, H.A. - Human Problem Solving, Prentice-Hall, 1972.

58. Reitman, W.R., Grove, R.B. and Shoup, R.G. - Argus: an Information Processing Model of Thinking, Behav. Sci. Vol. 9, 1964.
59. Reitman, W.R. - Cognition and Thought: an Information Processing Approach, Wiley, 1965.
60. Miller, G.A., Galanter, E. and Pribram, K.H. - Plans and the Structure of Behaviour, Holt, Rinehart and Winston, 1960.
61. Klix, F. and Sydow, H. - The Organization of Information Processing in Problem-solving Behaviour, in Klix F., The Organization of Human Information Processing, Akademie-Verlag, 1967.
62. Willshaw, D.J. and Longuet-Higgins, H.C. - Associative Memory Models, in Meltzer M. and Michie D. eds., Machine Intelligences 5, Edinburgh University Press, 1969.
63. Klahr, D. - Quantification Processes, in Chase W.G. ed., Visual Information Processing, Academic Press 1973.
64. Miller, G.A. - Human Memory and the Storage of Information, IRE Trans. Inf. Theory, IT-2, 1956.
65. Miller, G.A. - The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information, Psych. Rev., Vol. 63, 2, March 1956.
66. van Emden, M.H. - Hierarchical Decomposition of Complexity, in Meltzer M. and Michie D. eds., Machine Intelligence 5, Edinburgh University Press, 1969.
67. Lashley, K.S. - The Problem of Serial Order in Behaviour, in Jeffress L.A. ed., Cerebral Mechanisms in Behaviour, Wiley, 1951.
68. Johnson, N.F. - Organization and the Concept of a Memory Code, in Melton A.W. and Martin E.J. eds., Coding Processes in Human Memory, V.H. Winston & Sons, 1972.
69. Cermak, L.S. - Human Memory - Research and Theory, The Ronald Press Company, 1972.
70. Herriot, P. - Attributes of Memory, Methuen, 1974.

71. Posner, M.J. - Coordination of Internal Codes, in Chase W.G. ed., Visual Information Processing, Academic Press, 1973.
72. Johnson, N.F. - Chunking: Associative Chaining versus Coding, Journal of Verbal Learning and Verbal Behaviour, 8, 1969.
73. Estes, W.K. - An Associative Basis for Coding and Organization in Memory, in Melton A.W. and Martin E.J. eds., Coding Processes in Human Memory, V.H. Winston & Sons, 1972.
74. Hunt, E.B. and Hovland, C.I. - Programming a Model of Human Concept Formulation, in Feigenbaum E.A. and Feldman J. eds., Computers and Thought, McGraw Hill, 1963.
75. Keller, H.B. - Finite Automata, Pattern Recognition and Perceptrons, Journ. A.C.M., Jan. 1961.