Int. J. Man-Machine Studies (1969) **1**, 1–20

A Learning Machine With Monologue

J. H. ANDREAE AND P. M. CASHIN

University of Canterbury, Christchurch, New Zealand

(*Received 8 June 1968*)

The learning machine STeLLA has been developed considerably since it was first described in 1962. Although it has not been built in entirety, it has been simulated on computers in many different forms and with many different problems; special circuits have been developed for its construction.

The need to give the machine a "monologue" ability arose because the machine cannot learn to solve problems for which the input to the machine is inadequate to distinguish successive steps made by the machine. A simple example of this is the problem of learning to perform a sequence of actions when the sequence is not related to information received by the machine from its problem environment.

In this paper an introductory description of the STeLLA machine is given with the help of a particular problem which is then used to illustrate the generation of control policies by a dual machine. The dual STeLLA comprises two interdependent STeLLA machines or STeLLAments; one interacts directly with the problem environment, while the second STeLLAment interacts with an auxiliary "vocal" environment to provide monologue. Monologue is used to supplement the information from the problem invironment with information from the vocal environment. The STeLLAment interacting directly with the vocal environment has its input supplemented by information from the control policy of the other machine. The two machines are co-ordinated further by giving reward to the dual machine as a whole.

The procedure of counting is the kind of monologue which can be used to distinguish a sequence of steps that cannot be distinguished by successive inputs from the problem environment. This is illustrated in the paper by considering a problem that requires STeLLA "to walk across a dark courtyard". Monologue is not restricted to such counting of steps, but can take the form of more sophisticated "symbol" sequences. By allowing an operator to inject sequences into the monologue of the machine, an elementary form of dialogue could be set up. It would be a special feature of the dialogue that no predetermined language was imposed on the participants.

Examples of the use of monologue have been worked out and more complicated situations are being programmed for computer simulation both in the monologue and dialogue form.

1. Introduction

The learning machine STeLLA has been developed considerably since it was first described in 1962 (Andreae, 1964; Andreae & Joyce, 1965; Gaines

& Andreae, 1966). Although it has not been built in entirety, it has been simulated on computers in many different forms and with many different problems; special circuits (Gaines, 1967), like the ADDIE (Andreae, 1968; Gaines, 1968), have been developed for its eventual construction. In the meantime, new concepts are being introduced into the design, one of which is the subject of this paper.

The design of the learning machine has proceeded according to the following effective strategy.

(i) Try out a design.
(ii) Find classes of problem that it cannot tackle which, as humans, we feel it ought to be able to tackle.
(iii) Modify the design so that it can tackle one or more of these additional classes of problem.
(iv) With this new design, return to (i).

The need to give the machine "monologue" arose in this way. It was found that the machine could not tackle problems which were independent of the input to the machine. The monologue requires a "vocal" environment into which the machine makes "noises", which it then "hears". Monologue is necessary to augment the inadequate input to the machine and, at the same time, it represents a first step towards a natural machine language, man–machine dialogue, conversation, or what-you-will. In suggesting this, it must be emphasized that we are not giving the machine a language, but merely providing sufficient variety of structure for a primitive kind of language to develop.

STeLLA is a general problem-solver or versatile learning machine because her one aim is to "get more reward" and almost any task can be given to her with an appropriate procedure for reward.

The machine is a sampled-data system which interacts with a problem environment (or plant to be controlled) by sampling an input channel to obtain an input pattern vector $\mathbf{x}$ and by commanding through an output channel an action y once each sampling interval. Reward is a binary signal R which is given ($R = 1$) or not ($R = 0$) for each sampling interval. The machine is designed to obtain reward as often as possible (Fig. 1).

The problem of the learning machine can be appreciated more readily if it is reformulated for a human problem-solver:

> There is a row of lights in front of you (input pattern), each light being ON or OFF at any instant. Below the lights a row of push-buttons (actions) lie within your reach and these may be pressed, one at a time, in any sequence. There is nothing else but a single

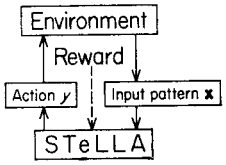


FIG. 1. The interaction of STeLLA and an environment.

light (reward) below the buttons and your task is to cause this light to flash on as often as possible.

The problem is ill-defined, but so are the real problems for which the machine is designed.

2. The Hundred-point Problem (HPP)

2.1. In order to focus the discussion on a meaningful situation, assume that the actions of STeLLA, unbeknown to her, cause her to move about the environment of Fig. 2 from point to point. When she commands action (1) (i.e. $y = 1$) of her three allowed actions ($y = 1$, 2 or 3), her position is changed to the next point on the left—except when this would take her "over the edge". Any move which would take her over the edge lands her in the position Ah marked with a black triangle (▲). Commands for actions (2) and (3) produce in each case alternatives with 50% probability as indicated in Fig. 3. Reward ($R = 1$) is given for each action which moves STeLLA into one of the positions connected by the upright cross or the sloping zig-zag.

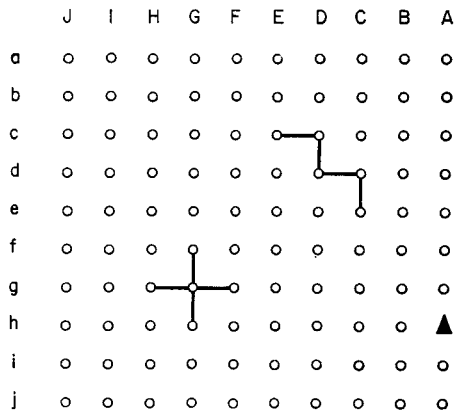


FIG. 2. HPP Environment. Reward is given at the points marked with a cross (Fg, Gf, Gg, Gh, Hg) and a zig-zag (Cd, Ce, Dc, Dd, Ec).

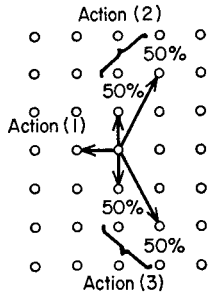


FIG. 3. Actions.

The binary word or vector, **x**, sampled by the machine in each of the 100 points is determined by the column (upper case) and row (lower case) letters according to the following code:

The first five binary digits are given by the column letter and the second five binary digits are given by the row letter to make up a ten bit input pattern. A code is used in which

a or A	means	10000	f or F	means	01111
b	B	11000	g	G	00111
c	C	11100	h	H	00011
d	D	11110	i	I	00001
e	E	11111	j	J	00000

Thus the black triangle point, Ah, has the code 1000000011 and this is the input pattern which the machine would sample at this point.

The environment described above will be called the hundred-point problem and will be abbreviated to HPP. It bears a close relationship to the car-steering environment used for many of the computer simulations of STeLLA (Gaines & Andreae, 1966). It is not a problem for which STeLLA needs monologue, but a modified HPP will be used later to illustrate the need for monologue.

2.2. Normally the learning machine is started on a problem without any initial information about either its task or its environment. This is not essential and a sudden transfer from one environment to another or from one task to another is allowed. In such cases the machine would be primed with information, even though it were irrelevant to the new environment. The simpler strategies embodied in the design of the machine are, however, most easily described in terms of the initial interaction of the machine with an environment after the machine has had its memory erased.

With no information on which to base decisions, the machine can do no better than select its actions at random. The result is a "random walk" or "searching trajectory" in the input space of the problem environment as illustrated in Fig. 4. During this time the machine's *predictor* is busy modelling the environment with a view to being able to answer the question: "If such-and-such an input pattern is sampled and such-and-such an action is commanded, what will be the next input pattern received?"

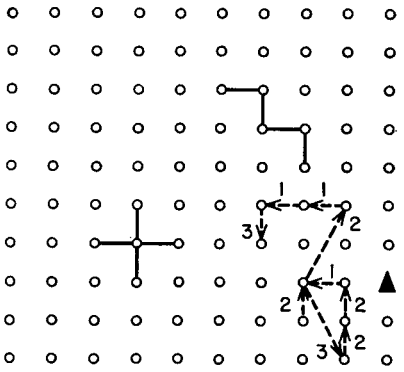


FIG. 4. Random search trajectory.

Then comes the time when the next action of STeLLA gains her reward and this causes her to store—in that part of her memory called the *control policy*—the last pattern sampled (Dg = 1111000111 in Fig. 5) and the action (2) which she commanded. These are stored together in the first *policy*

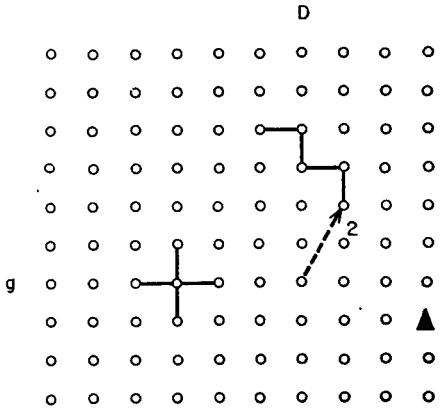


FIG. 5. Step to reward.

element (PE_1) with an initial estimate† (e.g. $r_1 = 0{\cdot}3$) of the probability of that action giving her reward again if she samples the same pattern:

PE_1: pattern $\mathbf{x}_1 = 1111000111$; action $y_1 = 2$; probability $r_1 = 0{\cdot}3$

After a further random search trajectory, STeLLA happens to sample again pattern $\mathbf{x}_1$ stored in policy element PE_1 (see Fig. 6). Before she commands action (2) ($= y_1$) in the hope of obtaining reward again, she stores a second policy element PE_2 to record the fact that she reached PE_1 from pattern $\mathbf{x}_2$ (Ee = 1111111111) by means of action y_2 (= 3). s_{21} is the machine's initial estimate of the probability that PE_1 can again be reached by action y_2 after sampling $\mathbf{x}_2$:

PE_2: pattern $\mathbf{x}_2 = 1111111111$; action $y_2 = 3$; probability $s_{21} = 0{\cdot}3$

Now STeLLA commands action (2) as required by y_1 and, depending upon whether she receives reward ($R = 1$) or not ($R = 0$), she alters her estimate r_1 to a new value

$$r_1(\text{new}) = (1-f_1\beta)\, r_1(\text{old}) + f_1\beta R.$$

β is an arbitrary constant (positive and small compared with unity) and f_1 is an *age factor* associated with the policy element which increases the precision of the estimation process as evidence accumulates. This process of exponential averaging ensures that r_1 will converge to an unbiassed estimate of the true probability in so far as the latter is meaningful.

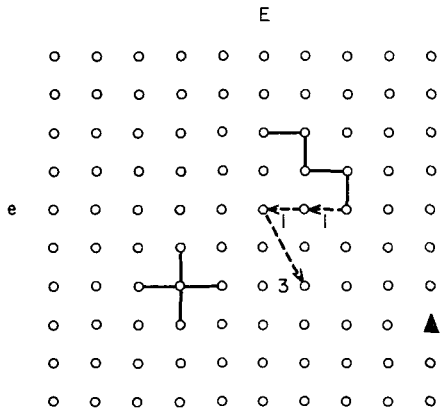


FIG. 6. Further search trajectory.

† Initial estimates of probability are parameters of the machine which have a strong effect on initial behaviour, but lose significance rapidly as the estimating processes progress.

By storing more policy elements with probability of reward and with probabilities of being followed by other policy elements, the machine can build up a simple control policy like that of Fig. 7, or a much more complex control policy like that illustrated in Fig. 8.

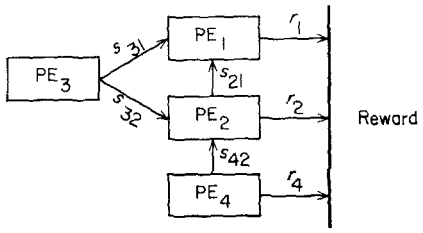


FIG. 7. Simple control policy.

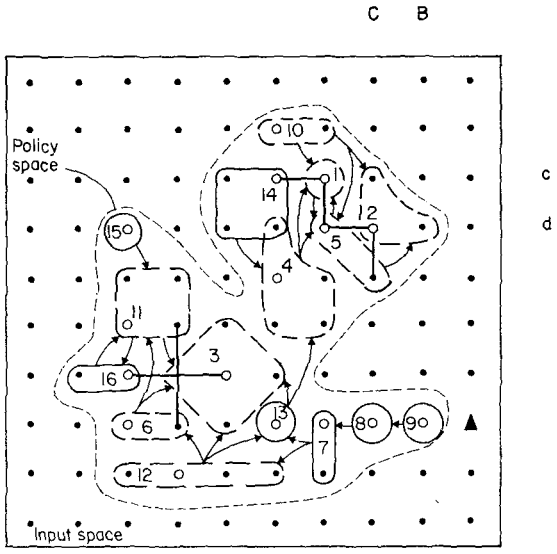


FIG. 8. Complex control policy.

2.3. In Fig. 8 the input space associated with each policy element is no longer restricted to a single point or stored pattern, but can be a cluster of patterns. It is not of particular significance in the simplified arguments of this paper that the STeLLA learning machine should be able to generalize over clusters of patterns in this way. We shall confine ourselves to a simple description of how a policy element embraces more than one input pattern in its stored pattern vector.

Policy element (2) in Fig. 8 may be taken as an example. The initially stored pattern of this policy element is that of the point Cd which is coded as 1110011110. With each bit (binary digit) of this pattern there is an associated adaptive weight. At some moment in time we may suppose that the associated weights have the values:

Pattern	1	1	1	0	0	1	1	1	1	0
Weights	0·9	0·8	0·3	0·9	0·8	0·7	0·9	0·9	0·2	0·8

The association of patterns with this policy element depends upon the value of a machine parameter called the *latitude*. Suppose this to have the value 0·4. Any input pattern which is such that the sum of the weights of those digits which do not match the input pattern exceeds the latitude is *not* associated with (recognized by) the policy element.
e.g.

Input pattern (Cc)	1	1	1	0	0	1	1	1	0	0
Stored pattern	1	1	1	0	0	1	1	1	1	0
Unlike digits									x	
Weights									0·2	

Sum of weights of unlike digits (= 0·2) is less than latitude (= 0·4). Therefore this pattern is recognized by policy element PE_2 as indicated in Fig. 2.8.

Input pattern (Bd)	1	1	0	0	0	1	1	1	1	0
Stored pattern	1	1	1	0	0	1	1	1	1	0
Unlike digits			x							
Weights			0·3							

Sum of weights (= 0·3) is less than latitude (= 0·4) so the pattern is recognized.

Input pattern (Bc)	1	1	0	0	0	1	1	1	0	0
Stored pattern	1	1	1	1	0	1	1	1	1	0
Unlike digits			x						x	
Weights			0·3						0·2	

Sum of weights (= 0·5) is greater than latitude (= 0·4) so the pattern is not recognized by the policy element. Indeed, Fig. 8 shows that pattern Bc is not within latitude of policy element PE_2.

2.4. We have shown how a policy element uses its adaptive weights to enable it to recognize input patterns that are in some way similar to its stored pattern. This situation leads to competition among the policy elements for recognition of patterns which give "success". To judge the competitors and ascribe priorities, two parameters are estimated for each policy element.

(i) Expectation

This parameter estimates the probability of reward for the policy element. Unlike the direct probability (r), the expectation (e) takes into account the probabilities of getting reward via other policy elements. Thus policy element PE_3 in Fig. 7 will have quite a high expectation provided the probabilities s_{31} and s_{32} are high.

(ii) Success fraction

This parameter estimates the proportion of successful steps contributed by each particular policy element.

These parameters are estimated by exponential averaging, following the procedure of Section 2.2 for r_1.

Not only does each policy element have an expectation which changes from sampling interval to sampling interval but the machine as a whole has an expectation corresponding to the policy element chosen to command the action for that step. A successful step is defined as one that results in immediate reward or in an increased expectation of the machine.

When several policy elements have recognized an input pattern they are subjected to a random, competitive, selection process biassed by the product of that policy element's expectation and success fraction. Thus, an unpromising policy element always has the chance to prove itself, but a policy element with a high expectation (probability of future reward) and also a record of success (high success fraction) becomes a strong favourite.

The results of this competitive selection are used to adjust the adaptive pattern weights, decreasing the weights of unlike digits for the winner and increasing the weights of unlike digits for the losers.

2.5. We have seen how policy elements are built up as "steps-back-from-reward" and how a policy element "generalizes" its stored pattern to a cluster of patterns by establishing the importance of each bit of the pattern through the use of adaptive weights. The competition between policy elements engenders optimal use of computation time and of the limited storage available to the control policy. The predictor is continually attempting to improve its model of the environment. It is employed both for predictive search (i.e. a random walk guided by the predictions of the predictor) to find a path into an area where a policy element will recognize the input pattern, and also as a check on the use of the policy in particular circumstances.†

† For detailed information about the STeLLA learning machine, the reader is referred to a Memorandum entitled "Computer Simulation of the Learning Machine STeLLA".

Although STeLLA has considerable power to learn how to get more reward and to adapt to changes in the environment (or task), she relies all the time on the input pattern containing information about "where she is" in the environment. If the input pattern contains insufficient information about her environment in relation to the current task, then STeLLA needs monologue to form an effective control policy.

3. The Need for Monologue

3.1. The STeLLA machine as described above is incapable of learning to tackle a very simple class of problems which are likely to arise frequently and in many forms.

Suppose the machine is connected into an environment in which it receives reward for performing three of its actions in a particular order, say:

action (1), action (2), action (3), Reward, action (1), action (2), action (3), Reward, action (3), action (1), action (2), (no reward) action (3), Reward, . . . etc.

Since the solution to this problem is independent of the input to the machine from the problem environment, it cannot construct an effective control policy by associating actions with input patterns. The stimulus–response solution is of no avail when the stimuli are irrelevant!

We can readily change the situation so as to enable STeLLA to tackle this problem by arranging for her last action to be remembered and included as extra digits in her input pattern vector. Now all she has to do is to associate with action (2) those patterns which contain in the extra digits action (1), and to associate with action (3) those patterns which contain action (2).

Indeed, it is true that so long as we—the designers—know that a problem has a particular input-independent characteristic, we can arrange for STeLLA to "see" it as being input-dependent by adding suitable remembered information to her input pattern vector from her past actions and patterns. This will not necessarily help her with problems which have input independent characteristics that we did not anticipate. Can we provide her with an ability to tackle input-independent problems for which she has not been specially prepared?

This will be answered by three examples. The first (Section 3.2) shows the need for monologue. The second (Section 4.2) provides a detailed but simple description of how monologue is established in a dual machine. The third (Section 4.4) is a suggestive and more complicated situation which is being investigated by computer simulation in our current research programme on STeLLA.

3.2. Consider the state-transition diagram for the problem environment shown in Fig. 9. There are two states P and Q, and transitions between these states are effected by the actions M and N.

The learning machine is given the problem of maximizing the frequency of reward, when reward is received for every transition from state Q to state P. The machine is allowed to command actions M or N and it is provided with a binary input x which distinguishes the current state of the system:

$x = 0$ when system is in state P
$x = 1$ when system is in state Q

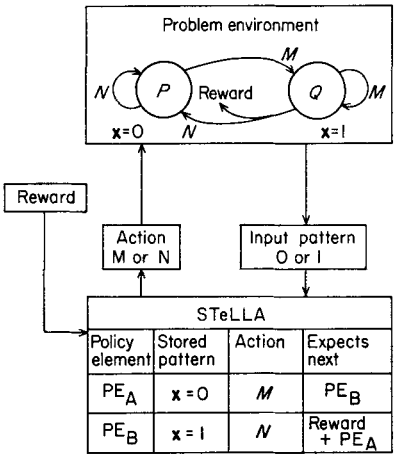


FIG. 9. Solution of a trivial problem.

The problem is trivial and STeLLA readily learns the control policy shown in Fig. 9:

Policy element (A) recognizes pattern $x = 0$ and commands action M, expecting policy element (B) to be selected next.

Policy element (B) recognizes pattern $x = 1$ and commands action N, expecting immediate reward and that policy element (A) will be used next.

If, however, $x = 0$ in both states P and Q, STeLLA can learn an appropriate control policy only if the input $\mathbf{x}$ is augmented by x', the recollection of the last action commanded:

$x' = 0$ when last command was action M
$x' = 1$ when last command was action N

This information can be provided entirely within the design of the learning

machine and the problem becomes solvable again through the control policy shown in Fig. 10:

Policy element (A) recognizes input pattern $x = 0$, $x' = 1$ and commands action M, expecting policy element (B) to follow.

Policy element (B) recognizes input pattern $x = 0$, $x' = 0$ and commands action N, expecting immediate reward and that policy element (A) will be selected next.

We are concerned with the possibility that the designer of the learning machine does not know in advance that the input will have to be augmented for a control policy to be learned. If he provides the machine with a particular augmentation, it may prove to be the wrong one for the task.

The machine needs to be able to augment its own input according to its tasks.

After cautiously mentioning a human analogy, we shall go on to show how a dual machine having a "vocal" environment can learn to augment its own input by trial and error.

3.3. Human analogies are dangerous. One is given here to aid the argument, not to carry it.

To walk across an open courtyard on a black night I count my footsteps and slow down on reaching the forty-third; then I carefully feel my way forward for the next two or three steps until I touch the expected door on

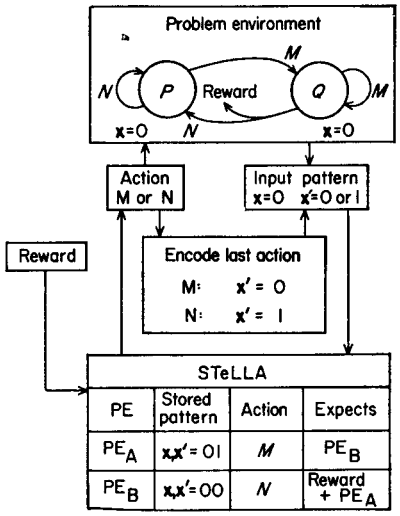


FIG. 10. Solution by recollection of last action.

the far side. My counting distinguishes the forty-third step from the earlier ones in the absence of any other distinction. When accompanied, my companion can count for me and tell me when the critical stage is reached.

The human use of verbalization (in a wider sense than just counting) and its relationship to "thinking" has been the subject of much discussion (Koestler, 1964). There seems to be little doubt that verbalization is invaluable in problem-solving, but not much is certain about the overall importance of language in human thought. From the point of view of adding monologue to a learning machine, it is sufficient to suggest that any ability to manipulate symbol-sequences is likely to extend the scope of the machine in more ways than we can predict.

4. Monologue in a Dual Machine

4.1. The idea is to couple two identical STeLLA machines together so that one interacts with the problem environment while the other provides the first with an augmented input by its own interactions with a "vocal" environment (Fig. 11). (A STeLLA used as part of a larger machine will be called a "STeLLAment".)

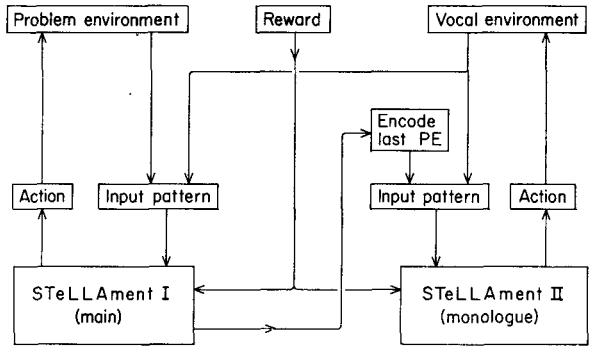


FIG. 11. The dual STeLLA.

The first STeLLAment will tackle tasks in the problem environment with the help of the "monologue" provided by the second in the vocal environment.

The two STeLLAments are coupled through their inputs and in the way they are rewarded; both are rewarded for satisfactory performance of either; each one augments the input of the other. The first has its input supplemented

from the environment of the second. The second has its input supplemented by a (e.g.) binary coding of the number of the last policy element whose decision was implemented by the first.

This will be explained now in detail in the simplest possible example.

4.2. The operation of the dual STeLLA will be illustrated by a detailed simulation of a simple case (Fig. 12). This is an extreme example in which monologue is all important and the problem environment provides no input pattern at all. This is equivalent to its providing an unchanging input pattern.

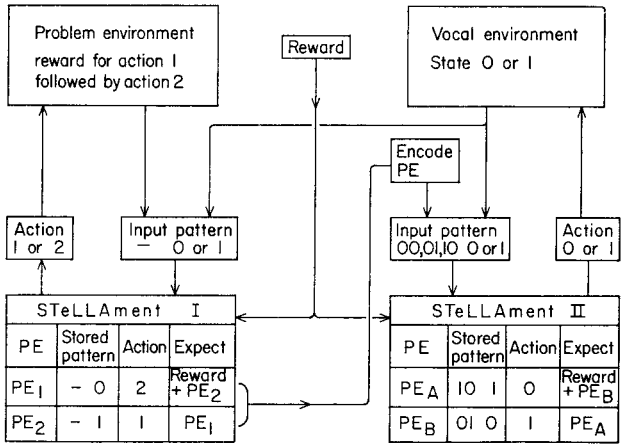


FIG. 12. Solution by monologue.

The machine should learn to perform actions (1) and (2) alternately because it is rewarded for performing action (2) after action (1)—the alternate sequence gives maximum rate of reward.

The reason for describing the auxiliary environment as a "vocal" environment becomes clear if we interpret the machine's ultimate solution to the problem as follows. Consider the two "vocal" actions to be sound "0" and sound "1". The sounds are "heard" by both STeLLAments since they appear in both input patterns. The machine's solution, shown diagrammatically in Fig. 13, is:

(i) .... make sound "1" by policy decision (*B*)
(ii) hear sound "1" and do action (1) by policy decision (2)
(iii) following PE_2 make sound "0" by policy decision (*A*)
(iv) hear sound "0" and do action (2) by policy decision (1)
(v) following policy element PE_1 .... continue with (i).

With this introduction to the problem, we can now proceed with the detailed description of the simulation.

4.3. The main STeLLAment (ST I) comprises two policy elements labelled PE_1 and PE_2 only. Pattern digit weights are omitted because in the short run simulated these weights would not change appreciably. ST I has actions "1" or "2". The monologue STeLLAment (ST II) has two policy elements, PE_A and PE_B, and has two actions, the "sounds" "0" and "1" (Fig. 12).

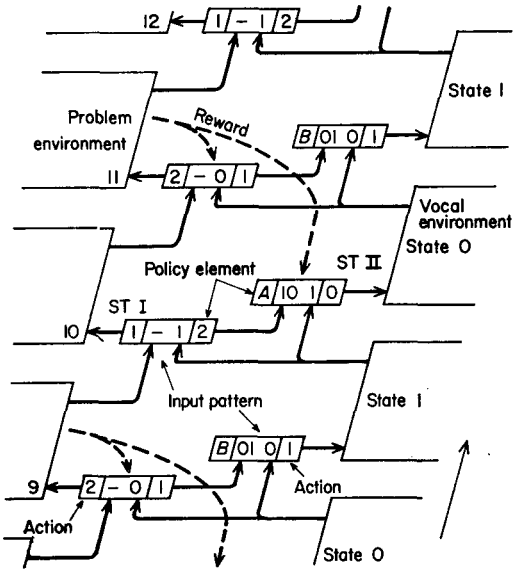


Fig. 13. Steps 9 to 12 of Table 1 in schematic form.

The input pattern to ST I consists of just one bit from the vocal environment indicating the last action or sound from ST II. The input pattern to ST II contains the same monologue bit plus two binary digits coding the policy element just selected by ST I. "00" means no policy element selected, "01" means policy element PE_1 and "10" means PE_2.

Reward is given to ST I and ST II immediately after ST I produces the action sequence "action (1) followed by action (2)" and before ST II makes its new action decision. That is, ST II is rewarded for the last action it commanded.

It should be noted that no policy element is stored in ST II when an input pattern "00_" indicates that no policy element has been selected in ST I. Actions selected at random are marked with an asterisk in Table 1, where 12 consecutive steps of the machine are listed.

TABLE 1

Simulation of a dual STeLLA with monologue

Input pattern	Policy element	Action	Reward	STeLLAment I control policy	Input pattern	Policy element	Action	Reward	STeLLAment II control policy	Step no.
−0	—	2*	—		000	—	1*	—		1
−1	—	1*	—		001	—	0*	—		2
−0	—	2*	R					R		3
				PE_1)–0/R-30, 1-00, 2-00/2)					no PE stored because pattern 001	
					010	—	1*			
−1	—	2*	—		001	—	1*	—		4
−1	—	1*	—		001	—	0*	—		5
−0	1			PE_2) _1/R-00, 1-30, 2-00/1)	101					6
		2	R	PE_1) _0/R-34, 1-00, 2-00/2)				R	PE_A)101/R-30, A-00, B-00/0)	
					010	—	0*			
−0	1			PE_1) _0/R-34, 1-30, 2-00/2)						7
		2	—	PE_1) _0/R-32, 1-30, 2-00/2)	010	—	1*			
−1	2			PE_1) _0/R-32, 1-28, 2-30/2)						8
		1	—		101	A	0	—	PE_B) 010/R-00, A-30, B-00/1)	
−0	1			PE_2) _1/R-00, 1-34, 2-00/1)						9
		2	R	PE_1) _0/R-36, 1-28, 2-30/2)				R	PE_A) 101/R-34, A-00, B-00/0)	
					010	B	1		PE_A) 101/R-34, A-00, B-30/0)	
−1	2			PE_1) _0/R-36, 1-26, 2-34/2)						10
		1	—		101	A	0	—	PE_B) 010/R-00, A-34, B-00/1)	
−0	1			PE_2) _1/R-00, 1-38, 2-00/1)						11
		2	R	PE_1) _0/R-40, 1-26, 2-34/2)				R	PE_A) 101/R-38, A-00, B-30/0)	
					010	B	1		PE_A) 101/R-38, A-00, B-34/0)	
−1	2			PE_1) _0/R-40, 1-24, 2-38/2)						12
		1	—		101	A		—	PE_B) 010/R-00, A-38, B-00/1)	

Probability estimates are shown as percentages in the control policies. Thus, R-32 means 0·32 or 32% probability of immediate reward, while 2-28 means 28% probability that policy element selected in the next step will be PE_2.

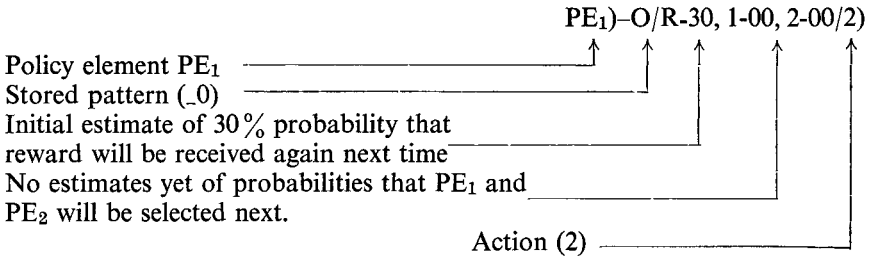
Incrementing and decrementing have been carried out according to a simple rule which is easier to follow than the correct exponential averaging process. The rule is: "add or subtract 4% if moving towards 50%, otherwise add or subtract 2". Examples are 26 to 30% (moving towards 50%), 26 to 24 (away from 50) and 73 to 69 (towards 50). The rule is, of course, used only for this illustration and never in a computer simulation.

Step-by-step commentary on simulation given in Table 1.

Step 1. ST I hears "0" from the vocal environment and selects action (2) at random. The input pattern "000" of ST II also contains the "0" from the vocal environment together with "00" coding the fact that no policy element has been selected by ST I. It selects action (sound) "1" at random.

Step 2. Input patterns now contain "1" from the vocal environment and random actions are selected again, action (1) by ST I and sound "0" by ST II.

Step 3. ST I hears sound "0" and randomly selects action (2), immediately getting reward because of its action (1)-action (2) sequence. Policy element PE_1 is stored with the information that, after pattern "0", action (2) obtained reward:



ST II shares the reward but does not store a policy element because its input pattern contains 00_ indicating that ST I had not selected a policy element. This rule prevents ST II from storing a great deal of useless information.

Steps 4 and 5. Both STeLLAments select actions at random.

Step 6. ST I hears "0" and recognizes this as being the pattern in PE_1. This causes a new policy element, PE_2, to be stored with the previous pattern (_1) and an initial estimate of 30% for the probability that PE_1 is followed by PE_2. The storage of PE_2 is equivalent to ST I having selected PE_2 on the last step so the input pattern to ST II is immediately modified from "001" to "101", where the "10_" is the coded input meaning "PE_2 selected last". Action (2) is now selected by ST I on the basis of PE_1 and reward is obtained. Therefore the estimate of the probability of reward in PE_1 is increased from R-30 to R-34, i.e. from 30 to 34%. Also ST II stores a new policy element PE_A with the last (modified) pattern (101), a 30% probability of reward and the action "0". ST II is receiving the input pattern "010", which is not recognized and so action "0" is selected at random.

Step 7. ST I uses PE_1 without obtaining reward so estimate of probability is reduced.

Step 8. ST II stores a second policy element, PE_B.

In the remaining steps detailed the dual control policy, illustrated in Fig. 13, becomes further established.

4.4. The problem of "walking in the dark" has been chosen for the last illustration of how the vocal environment can be used by STeLLAment II to enable STeLLAment I to perform in an input-independent situation.

The problem environment for STeLLAment I is the HPP (see Section 2.1); but this time the central area has been "blacked-out". The blacked-out area is marked in Fig. 14, all points in this area being indistinguishable from one another. The input pattern for STeLLA in the dark area is 1111111111 and will be referred to as Zz.

The vocal environment (Fig. 14) acted upon by STeLLAment II is such that each action produces a particular state of the environment. Action (1) results in state (1), etc. This is why it is called a vocal environment. When a sound is made in a silent place, that sound is then heard. When a word is spoken, the word is heard. In other kinds of environment it may take a whole sequence of actions to move from a present state to some particular desired state. The HPP is an example of this latter type of environment since, to move to some particular state (position in the HPP) takes a sequence of actions which are dependent on the present state (position). Problem environments are usually of this type, but there appear to be special advantages in using the vocal environment for monologue. This was not appreciated when it was suggested in Andreae (1968) that a kind of "notebook" environment would be suitable. Unless STeLLA has a direct, vocal type of environment for monologue, her chances of establishing monologue sequences by trial and error are slight. This is because the state of the problem environment at the start of any monologue-assisted sequence of actions has to be matched by an appropriate state in the monologue environment. The vocal environment can be placed immediately in a desired state by a single action.

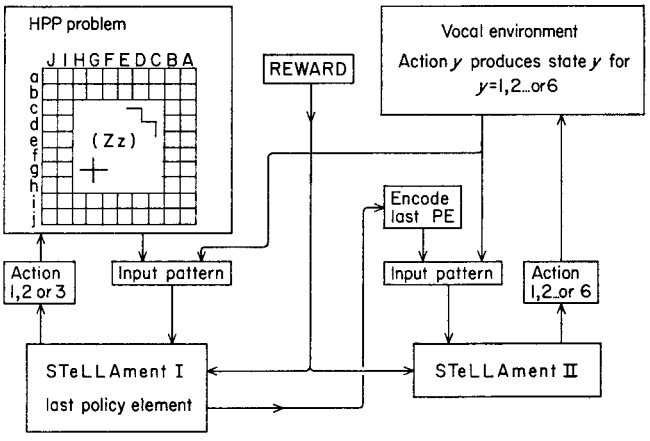


FIG. 14. The HPP problem requiring monologue.

The input pattern to STeLLAment I is shown in Fig. 14. The pattern is exactly as used in section 2 (except for the blacked-out area) supplemented by the state (i.e. "sound") of the vocal environment. For example, Fj-2 would indicate that STeLLA was in position Fj in the HPP and that the vocal environment was in state 2. The position Fj would be coded with the same code as before.

The input pattern to STeLLAment II comprises the policy element (coded) used by STeLLAment I, followed by the state of the vocal environment. An example would be 1–3, indicating that PE_1 had been used by STeLLAment I and that the vocal environment is in state 3. Although talked of as 1–3, etc. the input patterns have binary codes and bit weights estimating their importance just as described for the input patterns of a single STeLLA machine.

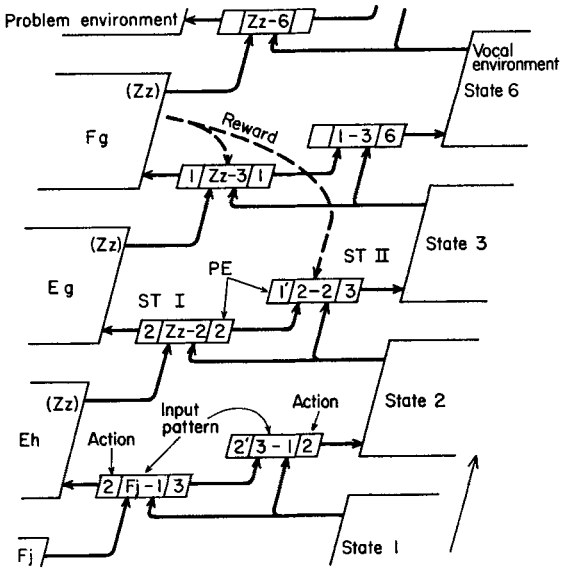


FIG. 15. Schematic solution of HPP problem by monologue.

STeLLA could learn policy elements that represent paths around the outside of the black-out area and moves into reward in one step, but without monologue no policy elements could be formed inside the blackout area. Figures 14 and 15 show a 3-policy-element path which the machine could learn from position Fj into the black-out area and then, via two steps in the dark, to reward.

5. Conclusion

Computer simulation of a dual STeLLA will shortly enable further experiments to be carried out on a machine with monologue.

Monologue is not restricted to the simple counting of steps, but could take the form of more sophisticated "symbol" sequences. By allowing an operator to inject sequences into the monologue of the machine, an elementary form ot dialogue would be set up. It would be a special feature of the dialogue that a predetermined language was not imposed on the participants.

It would be unreasonable to claim that this represented a first step towards unravelling the processes of language formation and development or towards the simulation of human learning. It is fundamental, however, to the design strategy which has led us along this path that we ask ourselves constantly: What is this kind of machine unable to do?

References

ANDREAE, J. H. (1964). STeLLA: a scheme for a learning machine. In *Automatic Remote Control*, ed. by V. Broïda, p. 503. Proc. 2nd IFAC Congress. London: Butterworths.

ANDREAE, J. H. (1968). Learning machines: a unified view. In *Encyclopaedia on Linguistics, Information and Control.* Oxford: Pergamon Press.

ANDREAE, J. H. & JOYCE, P. L. (1965). British Patents 1,011,685–7; German Patent 1234062.

GAINES, B. R. (1967). Techniques of identification with the stochastic computer. Proc. Symp. Problems of Identification in Automatic Control Systems, Prague. June 12–19, 1967.

GAINES, B. R. (1968). Stochastic computing. In *Encyclopaedia on Linguistics, Information and Control.* Oxford: Pergamon Press.

GAINES, B. R. & ANDREAE, J. H. (1966). A learning machine in the context of the general control problem. Proc. 3rd IFAC Congress, London: Inst. Mech. Engrs.

KOESTLER, A. (1964). *Act of Creation*, Bk. 2, Ch. XIV. London: Hutchinson.