

First Investigations of Dream-like Cognitive Processing using the Anticipatory Classifier System

Julian Holley¹

Clares MHE Ltd, Wells, Somerset, UK.

Learning Classifier Systems Group Technical Report UWELCSG04-002

Abstract

The cognitive abilities of the anticipatory classifier system (ACS) have already been successfully shown in earlier work (Stolzmann et al 2000). This report takes inspiration from some philosophical ideas for the purpose of dreaming in animals and humans during REM sleep. This is supported by recent neurological studies that show that rats revisit recent situations in a way that suggests dreaming (Wilson et al 2001). A simple extension is made to the ACS that uses the incomplete information contained in the classifier list as a basis for an abstract world model in which to interact or 'dream'. The abstract thread or dream direction is an emergent property of the selection process, this can be used to recycle around well known states and reduce real world interaction. The system is applied to two simple problems, the random walk and T-maze experiment and demonstrate that they require considerably less interactions with the real world to develop confident world models. Further models and extensions are proposed to advance the system, such as environmental directed generalisation and speculative rule creation.

¹ E-mail: julian@holley.uklinux.net

1 Introduction

There is increasing evidence (Stickgold et al 2001) that sleep and in particular rapid eye movement sleep (REM) or paradoxical sleep² (PS) where dreaming mostly occurs has important roles in memory and learning amongst both warm blooded animals and humans³. A recent neuro-physiological study (Wilson et al 2001) has demonstrated that rats do 'dream' or at least replay or rehearse maze experiments while asleep⁴. This confirms the results of earlier work (Wilson et al 1994) and also the obvious physiological similarities between human REM sleep and animal REM sleep. Psychological studies with animals and humans have shown that REM deprivation (REMD) can impair the learning of complex tasks (Smith 1995). Studies particularly with rats (Smith et al 1986) have demonstrated a distinct correlation between task complexity and onset and longevity of REM sleep. This report attempts to relate philosophical concepts of dream sleep with physiological and psychological evidence to improve learning in the machine learning arena (Sutton & Barto 1998). Specifically a tentative start is made by exploring a modified latent learning architecture, the Anticipatory Classifier System (ACS) (Stolzmann 1998) to simulate simple dreaming. Early results indicate a reduction in real world interaction to learn two simple maps, the random walk (Sutton & Barto 1998) and T-maze problem (Tolman 1932).

2 The Dream Paradox

What is the function of dreaming ? The fascination with the human dreaming experience has had a long and controversial history, the earliest evidence of interest dates from about 18,000 years ago in the form of cave wall drawings from our Cro-Magnon ancestors. Ever since the discovery of REM sleep in 1953 (Aserinsky et al 1953) there has been a considerable sleep and dream research activity. More recently there has been renewed interest following from the advancement in and availability of non-invasive brain imaging techniques. Whilst the necessity for sleep for the majority of creatures is obvious, the underlying mechanisms and development are still unclear, the role that dreaming plays in sleep is particularly controversial. Contemporary views on dream function are divided. The division can be broadly expressed between the functional and the non-functional theories. In the former, functionalist argue for the case that dreaming is part of continuous cognitive development, whereas in the later, the non-functional theories argue that dreaming is a secondary effect of underlying biological brain mechanisms or a genetic legacy and has no direct function. There are many theories on both sides of the debate none of which can convincing tip the balance either way. A conclusive theory on the purpose of dreaming and other associated sleep activities may be a combination of functional and non-function theories. As with much brain research in higher mammals much of the difficulty is down to the sheer complexity and redundancy of the organ coupled with the blurring of mind and brain, however in this work a more pragmatic approach is taken.

The purpose of this research is not to demonstrate the function or non-function of dreaming in animals and humans, although results may contribute to either side of the argument. It is to take inspiration from psychological studies as well as suggest philosophical ideas for dreaming and apply them to a machine learning architecture. It is proposed that this will in turn improve certain areas of performance, in particular to improve reaction to novel events, to generalise past knowledge and speculate on the future in preparation for interaction in a dynamic world. It is suggested in part that dreaming may be a form of abstract play, in a similar way that play prepares animals and humans physically for dangerous but unusual events, dreaming is a cognitive strategic version of physical play.

Based on the evidence⁵ so far, it is suggested that dreaming is not a direct mechanism for memory storage or reduction, but a method of either generalisation, speculative preparation or emotional play. Whilst the evidence for dreaming being non-functional or a side effect of another perhaps a neurological brain process is still the subject of on going research, the logical and philosophical ideas are still valid. For example, consider that the dreaming phenomena did not exist within sleep, or that an artificial learning agent had a long period of time when it is either not possible to operate or not required to operate (completely satiated). In such a situation, with sophisticated cognitive abilities or computing processes available, it seems pertinent for this resource to be utilised in some way. Such cognitive off-line maintenance could involve memory generalisation, memory or behavioural regeneration, or speculative behavioural generation (Stickgold et al 2001).

2 REM or rapid eye movement sleep is also called :- Active sleep, D-state, Desynchronised sleep, Dreaming state, Emergent state 1, Fast sleep, Paradoxical stage of sleep, Rhombencephalic phase of sleep and Stage 1 REM sleep.

3 The dolphin and duck billed platypus being the exception. Dolphins are naturally interesting due to their obvious intelligence and yet dream-less sleep. However dolphins suffer from 'Ondine's curse' which means that they must breathe voluntarily which is unfortunate for a sea dwelling mammal when it comes to sleeping. The dolphin has evolved an interesting solution to this problem. Dolphins do sleep but only with one half of the brain at a time, respiratory control swapping sides with the sleeping side of the brain.

4 Fascinating research (Morrison et al 1969) has shown that cats appear to dream as we imagine they might. Cats are the most prolific dreamers known, experiencing an average of 200 minutes a day of REM sleep. In Morrison's experiments the area of the brain responsible for inhibition of the motor system during REM sleep was disabled allowing the subject cats to act out their dreams as REM sleep occurred. The cats behaved completely normally with the exception that, at the onset of REM sleep, the cats would physically act out their dreams. The dreams consisted of visual exploration head motions and common emotional postures such as stalking, play, rage, fear and cleaning. During these episodes the cats were completely devoid of any their situation ignoring all sensory inputs (Morrison et al 1969).

5 See Hobson (2002), Jouvet (1994), Lawton (2003), Rock (2004) for reviews.

3 The Anticipatory Learning Classifier System (ACS)

The Anticipatory Classifier System (ACS) is a type of learning classifier system (LCS). Classifier systems were first proposed by Holland (1976). A learning classifier system is a rule creation, storage, execution and rule adaptation system initially designed so the rule base could be manipulated by genetic competition amongst the rules (Holland et al 2000). Although it is not necessary to utilise genetic modification of the rule base in order to solve problems (Gerard et al 2001, Gerard et al 2003), it is currently the most successful form of generalisation and was the original motivation behind the structure. Learning and adaptation in the ACS can be broken down into the following areas :-

- Knowledge structure.
- Action selection.
- Rule generation (classifier creation).
- Anticipatory learning.
- Environmental learning.
- Rule generalisation.

Essentially a learning classifier system consists of a list of simple production rules of 'IF state X THEN DO action Y' constructs that are each associated with a quality value based on the historical success of that rule in the execution environment. Most classifier systems consist of a rule creation scheme, a rule selection scheme, a reinforcement scheme and a generalisation process performed by genetic competition amongst existing rules. The genetic manipulation of the rules occurs periodically and is asynchronous to the rule learning process.

The Anticipatory Classifier System (ACS) introduced by Stolzmann (1998) differs from the original classifier system in that each rule is not only associated with a current state and action but also the expected next state. So production rules in the ACS have the form 'IF state X THEN DO action Y AND expect next state = X'. There are two quality values associated with each rule, that of prediction quality of the rule and that of environmental reward. The prediction quality is a measure of the ability of the rule to predict the next state after execution of the rule action. The environmental quality takes on the same role as the quality measure in traditional learning classifier systems, a measure of rule performance based on the environmental target of the system. Adaptation of these quality values are independent, but they can be combined in various ways to form rule selection and rule adaptation.

The ACS can therefore learn 'latently', learning associations between states and actions independent of the environmental reward. This is very similar to earlier ideas by Holland (1990) and systems by Riolo (1991) and Wilson (1995). Holland's (1976) original system was designed to operate in a genetic schema (Holland 1975) and thus state representations are discrete as opposed to real valued, as used in the machine learning and neural network arenas.

The basic ACS structure therefore takes the following form :-

$$\text{state}(n) \rightarrow \text{action}(n) \rightarrow \text{state}(n + 1) \quad Q_p \quad Q_e$$

In comparison to a traditional classifier system :-

$$\text{state}(n) \rightarrow \text{action}(n) \quad Q_e$$

Where :-

n = The current system step.

State = A representation of the environmental state.

Q_p = The current estimate that state(n + 1) will follow state(n) after action(n). Based on historical performance.

Q_e = The current estimate of environmental pay-off. Based on the historical performance within the context of the environmental target, (i.e. finding food, increasing wealth etc.)

State representations consist of an application dependent alphabet with a special generalisation letter or wild card, often represented as '*' in computational search expressions but represented in this context by the hash symbol '#'. An example state representation schema would be :-

```
state 1  AAAB
state 2  AABA
state 3  AABB
.
.
state N  CCCC      etc.
```

Listing 1.

The alphabet consists of symbols A, B, C and #, where '#' has a state field dependent meaning. In the state(n) field the '#' terms refers to any attribute (A, B or C). In the expected state field, state(n + 1) the '#' term refers to the 'pass through' function. The pass through function means that the state attribute in state(n) does not change from step n to step n + 1, the state attribute is *passed through* to the next step. In this example each environmental situation can be described by four discrete attributes of which each part consists of another three discrete symbols, the maximum state representation is therefore $X \wedge Y$ where X is the 'state depth' and Y is the 'state width' (in this example $3 \wedge 4 = 81$ states and with the additional hash symbol $4 \wedge 4 = 256$ possible rules). Some typical ACS rules are :-

C	state(n)	->	action(n)	->	state(n + 1)	Qp	Qe
1	#AB#	->	F	->	#BA#	0.9	0.2
2	####	->	G	->	####	0.5	0.5

Rule 1 (classifier C1) is valid for states 2 and 3 from listing 1. and classifier 2 is valid for all states. Also described in the classifier rule are the historical prediction and environmental values, Qp and Qe. Values tending towards 0 indicate a poor prediction capability or poor environmental reward. Values tending towards 1 indicate a good prediction capability or good environmental reward. Thus classifier 1 has a 90% prediction that the next state following action 'F' is the current state with the middle two attributes exchanged, but the action 'F' has a low environmental reward of 0.2.

Rule 2 (classifier C2) matches all states and believes that there is no environmental change following action 'G', (the current state is passed through unchanged). This is a special case of a classifier from which all others are derived, these are often called 'root' classifiers. Root classifiers cannot be deleted and serve as a catch all rule. There is one root classifier for each possible action of the system. When no other condition parts in the classifier list match, the root classifiers have a chance to be selected since the condition part matches all situations.

3.1 A Description of the ACS Operation⁶

Assume an Anticipatory Classifier System with one effector with two actions (1 and 0), four binary coded state inputs (detectors) and thus four binary coded expectations. The system is initially equipped with the most general classifiers for each action with no predisposition for the next anticipated state (the non removable 'root' classifiers).

Initial classifier list :-

	C	A	E	M	Qp	
Classifier 1	=	####	0	####	####	0.5
Classifier 2	=	####	1	####	####	0.5

Where :-

- C = The Condition part – used to match classifiers to current inputs.
- A = The Action part – action executed for matched input state.
- E = The Expectation part – prediction of next state, specifying changing attributes.
- M = The Mark part (see below) – used to record incorrect selections.
- Qp = The Quality value of the classifier to predict the next state.

The system is to be applied to an unknown plant and learns to predict the next state based on the current state and the action selected. For the moment there is no environmental reward present.

The plant has an initial input state of 0000, the '#' symbol in the condition part of the classifiers means 'don't care' and matches any attribute of the input, in this case '0' or '1'. Therefore both the shown classifiers condition parts match the input state. In this case only one action can be performed at each step so one of the classifiers must be selected. The matching classifiers are thus selected to form a 'match list' (both of them) and a selection from this list is based on their historic prediction quality average and the generality of the classifier (number of non '#' symbols in the condition part). The prediction quality value is stored in the Qp value, where low values tending towards zero indicate a poor prediction history and values tending towards 1.0 indicate a successful history. Both classifiers make a 'bid' to perform the action based on the historic quality and their generality. Selection between bids is proportionally probabilistic. The larger the bid the higher the chance of selection (roulette wheel selection).

⁶ A detailed description of the ACS can be found in Stolzmann 2000 and Butz 2001.

$$\text{bid}(c) = (1 / Q_t) * Q_p(c)$$

Where :-

- c = Classifier id.
- Q_t = Total of all Q(c)'s in the match list.
- Q_p = Classifier prediction quality.

Of course other methods of selection can be used to drive latent exploration/exploitation behaviour. In Stolzmann (1998) the bids were biased with an specificity term Spec(x) which encouraged selection of classifiers that contained less '#' terms. The bid for each classifier is thus :-

$$\text{bid}(c) = (1 / Q_t) * Q_p(c) * 2^{\text{Spec}(c)}$$

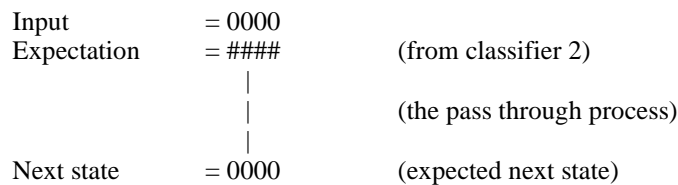
Where :-

- c = Classifier id.
- Q_t = Total of all [Q_p(c) * 2^{Spec(c)}]'s terms in the match list.
- Q_p = Classifier prediction quality.
- Spec = Quantity of non '#' terms in the classifier⁷.

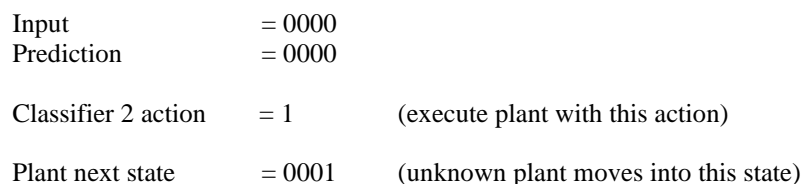
In this case each bid is identical and are thus equally likely to be selected. Assume that classifier 2 was selected.

Input 0000 -> classifier 2

The task now is, for the classifier to predict the next state after the associated action is executed on the plant. The prediction is the next state based on the current input state and modified by the information stored in the expectation part by the pass through process. The current input state (0000) is taken as the next state modified by any preference stored within the expectation part. No preference within the expectation part is indicated again by the '#' (or more appropriately for this state field, no change).



In this case the classifier has no predetermination of the next state and thus predicts by the pass through mechanism no change, i.e. the next state after executing the associated action remains same as the current state. To discover if the classifier was correct the action part of classifier 2 is applied to the plant and next real state acquired.



The classifier incorrectly predicted the next state of the plant from input 0000 with action of 1 the plant does not stay the same, but moves to state 0001. From this interaction with the plant the following facts have been determined. With a current plant state of 0000 a state of 0001 follows action 1 and also classifier 2 incorrectly predicted the next state 0000.

Intuitively this classifier should not be selected in this situation again. Two courses of action are open, firstly the generation of another classifier which is more likely to be selected and to decrease the likelihood of classifier 2 being called again in this situation.

⁷ The heavy weighting towards specific classifiers biases the system to prefer specific classifiers for selection over more general ones. In Stolzmann (1998) T-maze experiment this could lead to cyclitic behaviour of the simulated rat in the maze.

So a new classifier could be generated thus :-

C	A	E	M	Qp	
0000	1	0001	####	0.5	(most specific)

Matching the previous conditions completely, however whilst correct, this makes the classifier very specific and it could never be used in other situations, where the information held could have been useful. With respect to the pass through mechanism that is used to generate the next state prediction, the minimum preference that needs to be set in the expectation part of the classifier would be ###1. In other words the next state stays the same as the current input state, with the exception of the last attribute, which needs to be a 1, hence :-

C	A	E	M	Qp	
###0	1	###1	####	0.5	(most general)

This classifier can still be selected for an input of 0000 and using the pass through mechanism still produces the correct prediction of the next state following the action 1.

Input	= 0000	
Expectation	= ###1	(from classifier 2)
		(the pass through process)
Next state	= 0001	(expected next state)

This classifier could then be utilised in possible future situations, for example if a state of 1001 followed 1000 after an action of 1, then this classifier is also correct, i.e. :-

Input	= 1000	
Expectation	= ###1	(from classifier 2)
		(the pass through process)
Next state	= 1001	(expected next state)

Whereas the highly specific classifier :-

C	A	E	M	Qp	
0000	1	0001	####	0.5	(too specific)

This case would have been excluded (because of the first condition attribute specification of 0) and information held repeated elsewhere in another classifier.

So in general whenever a new classifier rule is generated it is done so in a way that preserves maximum generality of the rule, specifying the minimum possible for the correct selection. This process is also called 'the specification of changing components'.

The original classifier in this case (classifier 2) is also penalised for the incorrect prediction by a reduction of the quality strength Qp by moving Qp proportionally towards zero by a small preset increment (the Widrow Hoff delta rule) :-

$$Qp(t + 1) = Qp(t) + \beta(\text{Error})$$

$$Qp(t + 1) = Qp(t) + \beta(0.0 - Qp(t))$$

Where, Error is (Target - Qp(t)) and β is a small empirical value ($0 < \beta < 1$) (typically $\beta = 0.1$)

If the classifier is bad and the Qp value tends towards zero then the classifier can be considered useless and become deleted if the Qp value becomes lower than a certain deletion threshold, since the value will theoretically never reach zero.

In this case a new classifier has been generated and the quality value Qp of the selected classifier has been reduced, there is also another mechanism than can be applied when an incorrect classifier is chosen, that of 'marking'.

As shown above the generated classifiers are attempting to stay maximally general and therefore open to incorrect selections, indeed as the learning cycle processes a particular classifier may be called more than once incorrectly. Incorrect selections can be stored as useful information in so much as the next time a classifier is correctly selected examination of incorrect selections can be used to prevent further incorrect selections. So in this case the incorrect selection input of 0000 is stored as the classifier mark, thus classifier 2 becomes :-

C	A	E	M	Qp
####	1	####	0000	0.5

With the mark of 0000 indicating a case where the inputs were 0000 but this was not the correct selection.

To summarise so far, for a plant state of 0000 all classifiers (1 and 2) could match the input both having condition parts of ####. After a bidding process which in this case results in identical bids, classifier 2 was chosen. With no predisposition of the next state held in the expectation part of classifier 2, the pass through mechanism results in the next expected state being identical to the current state, 0000. The action 1 of classifier 2 is applied to the plant from the current input state of 0000 which results in a movement to the state 0001 not 0000 as predicted, therefore the classifier was incorrect. In the incorrect case the following is performed :-

The classifier is made so as to be less likely to be selected in this situation again by :-

- If possible, the formulation of a new maximally general classifier based on the selected input, action and resulting next state.
- The classifier is penalised for the incorrect selection by moving the prediction quality value Qp closer to zero. If the value of Qp goes below some predetermined threshold the classifier is completely deleted.

The information that this classifier was selected incorrectly in a given state is stored as the 'mark' i.e. the mark becomes 0000. If the classifier is selected in the future and gives a correct prediction, the mark can be used to mean :- "In the past this classifier was selected incorrectly with the input state stored in the mark, therefore to prevent that event reoccurring generate a new classifier that is different in the condition part such to exclude selection of the input state indicated by the mark". The technique that performs this process is called 'the specification of unchanging components'.

The classifier list has now grown to 3 :-

	C	A	E	M	Qp	
Classifier 1 =	####	0	####	####	0.5	(not selected yet)
Classifier 2 =	####	1	####	0000	0.45	(penalised and marked)
Classifier 3 =	###0	1	###1	####	0.5	(new classifier)

To further demonstrate the marking mechanism, assume that with a plant state of 1111 a next state of 1111 exists after an action of 1. Assume also that by the same selection and bidding process, classifier 2 has been successfully selected and makes the following prediction of the next state :-

Input	= 1111
Expectation	= #### (from classifier 2)
	(the pass through process)
Next state	= 1111 (expected next state)

The action of 1 is applied to the plant and the next real state is also 1111 therefore the classifier correctly predicted the next state. However the classifier is marked. The mark indicates that in state 0000 this classifier was previously selected, but that this did not lead to a successful prediction. At this point a new classifier could be generated such which allows selection in this current case (the correct selection) and also prevents selection in the case indicated by the mark (the previous incorrect selection).

So classifier 2 is :-

C	A	E	M	Qp
####	1	####	0000	0.45

Classifier 2 is correct when the input 1111 causes a selection, but has been incorrect in the past at input '0000' as stored in the mark. Therefore a new classifier could be generated such that input state 1111 is selected but that ensures input 0000 cannot be selected, hence :-

Again, attempting to stay maximally general only the minimum alteration to the classifier need be applied in order to achieve the desired outcome. Therefore a change can be made in the condition part at any of the points where hash terms exist. In this case the first opportunity (first don't care '#') where the mark differs from the current input state is used, but it could be any. The first '#' symbol occurs at the first attribute as does the first difference (that of 1 to 0).

Correct input case	= 1111	
Previous bad case	= 0000	(stored in the mark).
Current condition	= #####	
New condition	= 1###	

So the new classifier generated from mark is :-

C	A	E	M	Qp	
1###	1	1###	#####	0.5	(classifier 4 – generated from mark)

This new classifier will still be selected for the case of 1111 but is excluded from being selected in the case of 0000 by the specification of the first attribute. As stated above there is more than one option for specifying the new classifier, i.e.

C	A	E	M	Qp	
#1##	1	#1##	#####	0.5	(classifier 4 – generated from mark)

or

C	A	E	M	Qp	
##1#	1	##1#	#####	0.5	(classifier 4 – generated from mark)

or

C	A	E	M	Qp	
###1	1	###1	#####	0.5	(classifier 4 – generated from mark)

Any of these specifications will prevent the selection in the case where 0000 was the input (as indicated in the mark). In this case the first available option for specification has been utilised however given that there is no further knowledge on which one of these attributes to select a random position for specification should be chosen.

During the learning process it may be the case that a classifier is selected more than once and makes erroneous predictions, in such cases the mark strip may already be occupied with a previous example of an incorrect selection. Using classifier 2 again :-

C	A	E	M	Qp	
#####	1	#####	0000	0.45	(mark showing incorrect at state 0000)

Assume that this classifier is selected again for input state 1001 and predicts the next state :-

Input	= 1001	
Expectation	= #####	(from classifier 2)
		(the pass through process)
Next state	= 1001	(expected next state)

With the action of 1 applied to the plant, the actual next state is say 1011, the classifier has made another incorrect prediction. The mark is already occupied with a previous incorrect input state that of 0000. Another mark store could be created for every incorrect selection, however in a binary attribute scheme the marks can be combined in the same mark space if another third symbol is used to indicate '1 or 0' i.e. the binary coding scheme for the mark now becomes a tertiary system :-

0 = 0
 1 = 1
 0 or 1 = @

So two incorrect input states of 0000 and 1001 now become a mark of :-

first incorrect input state = 0000
 second = 1001
 new mark = @00@

This scheme is convenient for the binary coding string used in this example, but becomes cumbersome when more than two symbols are used and a list of marks may be more appropriate. The frequency of attribute changes in the mark are lost and the '@' attribute will be specified equally even when there is a significantly higher portion of one attribute with respect to the other.

Using this multiple mark (@00@) and the previous example when classifier 2 correctly predicted the next state i.e. :-

C	A	E	M	Qp	
####	1	####	@00@	0.405	(mark showing incorrect at state 0000 and 1001)

With an input state of 1111 the classifier correctly predicts an output of 1111 but from the mark of @00@ establishes that this classifier has been selected incorrectly with input state 0000 and 1001 and needs to generate a new classifier to prevent this reoccurring and isolate these cases.

Specifying the first or last attribute space of the condition would only exclude one case but not both. So the new classifier is made specific at either point of commonality between the correct input and the mark, i.e. the first shared attribute (2), or the second (3) which are both zero. In this case the first (most left) is selected randomly hence :-

New classifier generated from classifier 2 with mark @00@ :-

C	A	E	M	Qp	
#1##	1	#1##	####	0.5	(classifier 5)

This classifier excludes future selections from 0000 and 1001 but will still be selected for 1111.

Also when a classifier is correctly selected the prediction quality of the classifier is also increased in a similar method to that of punishment. The quality Qp value is shifted proportionally closer to 1, hence :-

$$Qp(t + 1) = Qp(t) + \beta(1 - Qp(t))$$

If the classifier is successful the Quality value tends towards 1 and if the value becomes greater than some predetermined small value below 1 the classifier is termed 'sure'.

To summarise the ACS learning strategy so far :-

1 – With a certain input state, a list of classifiers is selected from all available classifiers where the input state match the condition part to make a match set.

2 – Based on some application dependent selection criteria, a classifier from the match list is chosen to be executed. The scheme used in Stolzmann's ACS is a roulette wheel based scheme. Bids from the match list are proportional to their prediction quality, environmental reward and an empirical (assumed) bias based on generality (quantity of '#' symbols). The more specified classifiers (less hash terms in the whole classifier) are much more likely to be selected than lesser specified classifiers.

3 – The winning classifier makes a prediction of the next state that is expected after the associated classifier action is executed.

4 – After the action is executed and the next real state from the plant has been attained the classifier is modified by a processes termed the Anticipatory Learning Process (ALP) as described previously, such :-

If the prediction was correct :-

- The quality of the classifier is increased.
- If the classifier is marked a new classifier (if possible) is formed based on 'unchanging components'.

If the prediction is incorrect :-

- The quality of the classifier is decreased.
(Classifiers with the exception of root classifiers are deleted if their prediction quality reduces below a preset value).
- The classifier is marked with the current input state.
- If possible a new classifier is formed based on the 'changing components'.

5 – goto start.

3.1.1 Environmental Reward Learning

The description so far has concentrated only on anticipatory learning, i.e. learning to predict the next state and the associated mechanisms to compliment this process (ALP). By using the previous process to generate a classifier list that can successfully predict successive states, the system can be used with an environmental reward to drive a behaviour. The system can be augmented with another parameter of environmental reward (Qe in this description) which is a measure of reward associated only with the environment and not with the predictive capability of the classifier, i.e. :-

	C	A	E	M	Qp	Qe
Classifier 1 =	####	0	####	####	0.5	0.5
Classifier 2 =	####	1	####	####	0.5	0.5
.	=
.	=
.	=
Classifier n =	####	1	####	####	0.5	0.5

Where :-

- C = The Condition part – used to match classifiers to current inputs.
- A = The Action part – action executed for matched input state.
- E = The Expectation part – prediction of next state, specifying changing attributes.
- M = The Mark part (see below) – used to record incorrect selections.
- Qp = The Quality value of the classifier to predict the next state.
- Qe = The historic environmental reward or payoff.
- n = Length of classifier list.

There are numerous methods and strategies for distributing environmental rewards between the classifier rules, (Sutton & Barto 1998) but essentially the environment reward is independent of classifier prediction, i.e. classifiers exist with high predictions and low environmental rewards. Generally the ACS utilises an on-policy bucket brigade reward algorithm whereby the environmental reward is updated based on a discounted future reward. This can also be distributed to an action set as opposed to a single classifier. An action set is a sub-set of the match set, where classifiers that form the match set are re-selected again based on their action. As shown previously the match set is formed from classifiers whose condition part matches the current plant state. From this list a bidding process takes place and a winning classifier rule is selected. The winning classifier has a specific action, the action list is created by all other classifiers within the match list that also have the same action. Environmental rewards can be proportionally distributed amongst the match list, also non action matching classifiers could be proportionally penalised. The bidding and selection process can be augmented with the current environmental reward to alter behaviour based on some target, i.e. from previous :-

$$\text{bid}(c) = (1 / \text{Btu}) * \text{Qp}(c) * \text{Qe}(c) * 2^{\wedge} \text{Spec}(c)$$

Where :-

- c = Classifier id.
- Btu = Total of all [Qp(c) * Qe(c) * 2 ^ Spec(c)]'s terms in the match list.
- Qp = Classifier prediction quality.
- Qe = Classifier environmental reward quality.
- Spec = Quantity of non '#' (hash) terms in the classifier.

In Stolzmann (1998) an off-policy reward learning was applied, whereby the ACS interacted with the environment without an environmental reward until a certain level of anticipatory quality of the classifier list existed. The environmental reward was then introduced and the system invoked again. When the environmental reward was encountered classifiers that matched the reward state (terminal state) where rewarded proportionally based on their specificity and prediction quality. In other words the environmental reward 'flooded' the reward quality Qe based on classifiers that could find a predictable route to the reward state.

4 Improved Convergence Through Model Learning Using the ACS

The long term goal of this work is to improve an agent's cognitive abilities inspired by developing theories from dream research. As an interim step toward this goal the ACS has been employed to reduce learning interactions with the real world by abstractly or cognitively revisiting situations and confirming or weakening state relationships. As the ACS interacts with the environment a generalised model is developed, this knowledge can be exploited to perform some model learning Dyna style (Sutton 1991) and reduce environmental interaction. As with the Dyna architecture the goal is to minimise environmental interaction and utilise information held in the model in order to obtain an accurate model with the least real world interactions. However the ACS does not hold an explicit world model but a generalised representation and this represents a challenge when employing the model to generate an abstract representation of the real environment. This interim step, model learning⁸ is applied to the ACS by switching ACS experiences between the real world and the developing model. The model is a snapshot of the classifier list at the time between leaving interactions with the real world and entering the abstract model world. Consider the following example :-

An ACS system has been interacting with an arbitrary plant and has formed the following classifiers :-

	C	A	E	Qp
Classifier 1 =	####	0	####	0.2
Classifier 2 =	####	1	####	0.2
Classifier 3 =	01##	0	10##	0.9
Classifier 4 =	10##	1	01##	0.9

For classifier 3, the list indicates that with a high confidence (Qp = 0.9) if the agent is currently in a state where the first (left most) attributes are '01' and a '0' action is performed, then the first two attributes of the following state will be '10', the rest of the attributes remain unchanged, i.e. :-

Step 1.

Input state = 0100 (which matches all the classifiers with the exception of classifier 4).
 |
 C3 expectation field = 10##
 Expected next state = 1000 (last two attributes passed through).

Step 2.

Input state = 1000 (which matches all the classifiers with the exception of classifier 3).
 |
 C4 expectation field = 01##
 Expected next state = 0100 (last two attributes passed through).

To present the ACS with a 'model' of the world, an algorithm must extract this information from the classifier list and present the same format back to the ACS, the current input and the next state as though action 'x' is executed. The difference being that the real world response is reality whilst the model is an estimate, an estimate that improves with experience on a stationary problem world (See figure 1 & figure 2).

⁸ Model learning is well established in the machine learning arena (Sutton & Barto 1998) and this has been transferred to the learning classifiers by several researchers (Gerald et al 2000) (Roberts 1993).

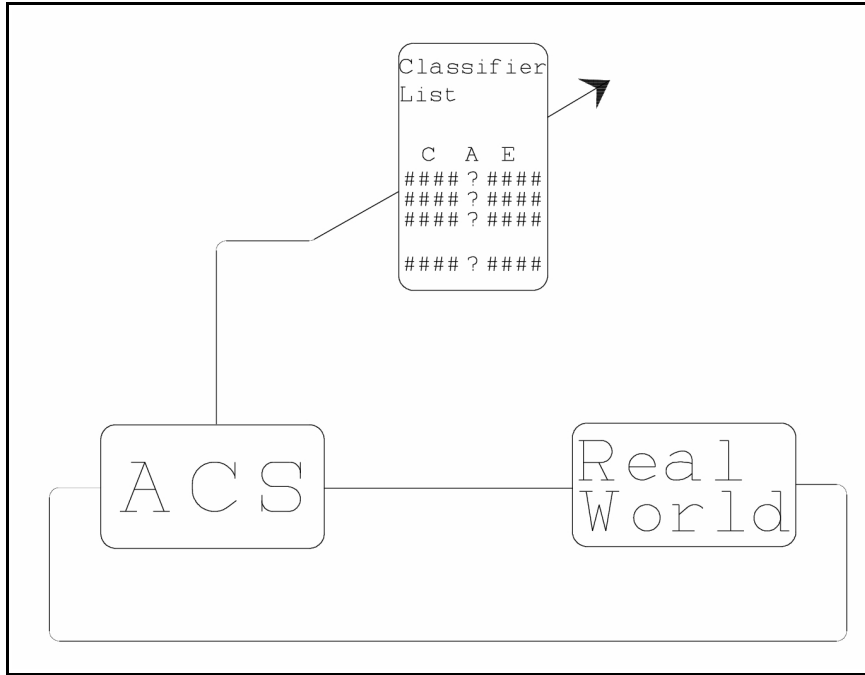


Figure 1
The basic ACS interaction. This can be compared to figure 2 showing the extension employed.

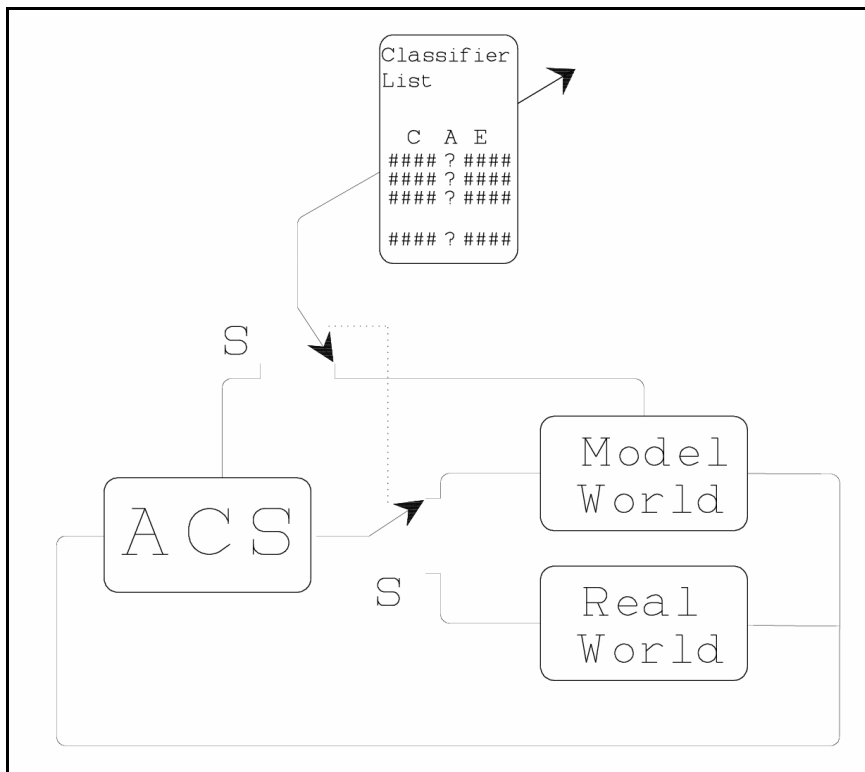


Figure 2
The 'ACS' switching between the real world and the model of the real world. The model is able to generate responses by using the information held in the classifier list. The model incorporates another selection strategy to select classifiers by condition and action. Missing information in the hash '#' terms are filled by passing through the current state of those attributes effected.

The algorithm for the model world operation and response is similar to that of the normal ACS selection process. From the snap copy of the classifier list the model must select the most appropriate next state given the current state and action. Consider that the ACS is running with the previous classifier list. Assume the current state is 0100 and through the normal selection process classifier 3 is selected with the action '0' then the classifier expects the next state to become 1000. At this point before executing the action on the plant (real world) the switch 'S' changes state to allow the ACS to interact with the model world.

The model algorithm takes a copy of the current classifier list and waits for an input and action pair from the ACS. The ACS then executes action '0' on the model world instead of the real world without being able to differentiate between the two worlds. The model must now reply to the ACS as the real world would by making a 'best choice' response in the form of the next state. Then the system must select from the best classifier from the list to use for that input condition and action. This can be achieved by using exactly the same selection process amongst the classifiers as the normal ACS would use (i.e. generation of a match list, bid generation and final selection) except that the classifiers able to make a bid must also match in the action part to that of the requested action currently presented from the ACS.

From the previous list the only two classifiers to match in the condition part and the action part are classifier 1 and classifier 3. If (as with the ACS) a high bias is given to classifiers that are more specific the likelihood that classifier 3 will win the bid is high (good prediction history, $Q_p = 0.9$ and contains less hash terms). If classifier 3 wins the bid, the model algorithm simply responds to the ACS with the next expected state by presenting the current input with a pass through operation with the expectation part of classifier 3 to form '1000' as the response. If classifier 1 wins (even with the small bid), then the current input state will be passed through as the next state (the model incorrectly presents no change).

When the ACS receives the 'predicted' response from the model, this is taken as the actual response and learning continues. This can dramatically reduce the amount of interactions that are required for the ACS to build up an accurate model in a static environment. When a novel situation is presented to the model and no input-condition and action pair exists, the root classifiers will always win the bid and predict no change, if the ACS is expecting the action to have a consequence and as such ignores the response and the ACS tries another action.

5 The Experiments⁹

To explore the operation of this model two simple learning problems are explored, the random walk and the T-maze. Both of these problems have been used to explore other issues such as reward propagation and latent learning, however in these experiments they are initially used as two simple mazes. The object of the experiments is to record the length of total real interactions that are required to reach the same prediction accuracy when ACS interaction is split between interacting with the real world and the model. Sessions run until a complete sweep of each state with each action predicts the next state correctly and with a classifier that has a high prediction quality ($> 90\%$ aka 'sure'). Table 1 illustrates the split between real world and model world interactions. The first entry in table 1 illustrates that no model learning occurs in the first session. The second entry indicates that 90% of ACS interactions are with the real world and 10% with the model. The interactions are distributed throughout the session, so that in session 2, 9 real world interactions occur, followed by 1 model world interaction. In the last entry in table 1 there is only 1 real interaction followed by 9 model interactions.

Session	% Real interactions (s-real)	% Model interactions (s-model)	Effect
1	100	0	"No model interaction"
2	90	10	
3	80	20	
..	
..	
9	20	80	
10	10	90	"Little real interaction"

Table 1

An experimental session consists of running the ACS system with each of the above switching points and averaging the results over 100 trials. The balance between real and model interactions is slightly asymmetric because no learning can take place without any interactions with the real world at all, hence the real interactions stop at a minimum of 10%.

In each session (a row in table 1) the following sets of results are recorded as illustrated in table 2. :-

1	2	3	4	5	6	7	8
Run-Real %	Run-Model %	Real	Model	Restarts	Correct	Baseline	Ave Classifiers
100	0	955.53	0	0	100	5.83	11.88
..
10	90	69.41	578.46	22.89	100	5.86	7.58

Table 2

The layout of the results scheme.

A description of the parameters that are recorded are listed below :-

Real world interactions, column 1 – 'Run-Real'

The percentage of ACS interaction on the real world, with respect to 'Run-Model' below.

Model world interactions, column 2 – 'Run-Model'

The percentage of ACS interaction on the model world, with respect to 'Run-Real' above.

⁹ The terms 'model' and 'dream' are used interchangeably, however the term 'model' best describes the off-line interactions in the experiments reported herein rather than the term 'dream'. The term 'dream' is mainly used to describe on-going and future work, whereby off-line interactions are less deterministic than that generated by the abstract selection process used in these experiments.

Real cycles executed to solution, column 3 – 'Real'

The quantity of steps during execution that have involved interaction with the real world.

Model cycles executed to solution, column 4 – 'Model'

The quantity of steps during execution that have involved interaction with the model world.

Bad model replies, or restarts, column 5 – 'Restarts'

If during ACS interaction with the model a useless response to an action is received, i.e. no change, then a new random input start state is selected and model interaction continues. The 'Restarts' trace represents the amount of times that the modified APL running during the model phase 'explores' and the proposed new classifier is ignored and a new session is started.

Model accuracy, column 6 – 'Correct'

Every 10 steps the ACS is tested by selecting a random state and action pair and comparing the ACS response (expected state) with the correct plant response.

Random accuracy base line, column 7 – 'Baseline'

The identical test is performed as described for the model accuracy (above) except another random next state is chosen to find the rate at which random input states, actions and next states occur by chance.

Classifier list length, column 8 – 'Ave classifiers'

The average length of the ACS classifier list.

Four experiments are conducted with different selection policies for both the ACS and the model selection policy. This is expressed in the following table 3.

Experiment	ACS Selection Policy.	Dream Selection Policy
1	Probabilistic bid	Deterministic
2	Probabilistic bid	Probabilistic bid
3	Select best	Deterministic
4	Select best	Probabilistic bid

Table 3

The 4 experiments with different ACS and model selection policies during the model or 'dream' phase.

A Probabilistic bid is the roulette wheel selection technique, whereby qualifying classifiers each make a bid, bids are normalised and the winning classifier is selected by a generating a linear random number between 0 and 1.

A Deterministic selection is whereby classifier make the same bid as in a probabilistic selection, with the exception that the highest bid wins.

5.1 Simple Random Walk

In this problem an agent has to learn to remain on a raised level surface that is terminated by a 'cliff edge' at either end of the surface. At every time step the agent must make a decision to move either left or right and cannot remain still. Successive actions in one direction will terminate with the agent falling off the edge of the cliff, which must be avoided. The agent is blind to the edge of the cliff and can only learn by falling off the edge. To allow the agent to learn a punishment is given every time a fall is encountered, at all other times there is no environmental feedback. Sutton and Barto (1998) amongst others have used this simple problem to illustrate various machine learning concepts. A diagram illustrates the problem :-

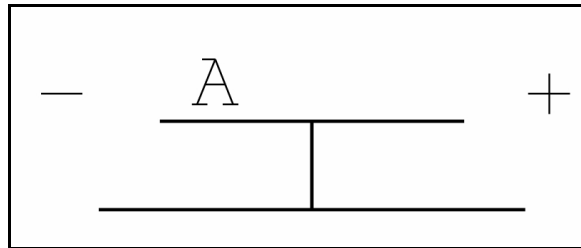


Figure 3

Possible actions are right (+) and left (-) and the top surface is numerically represented and coded into a binary form. The most left hand side is coded '0000' and the most right hand side is coded '1111'.

This can be also be viewed as learning sequential binary numerical associations. For simplicity no environmental learning is considered and if the agent moves over the edge of the cliff (goes beyond or below the numerical limits) then the agent simply appears at the other edge (other numerical limit). The agent is considered to be successful when a successful sweep (walk) of the state space can be achieved with the two actions left or right (plus and minus) and the classifiers that were used had a high prediction quality or a 'sure' classifier (a prediction quality above a predetermined threshold). In other words, not only must there be a successful numerical sequential walk, but that successful walk must have been generated by classifiers that were all sure. A typical final classifier list can be illustrated in table 4.

Pos.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Code	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Table 4

Coding of the random walk. State 0 and state 15 wrap around to respective states.

Agent action.	Description.
+	Move right (increment).
-	Move Left (decrement).

Table 5

Definitions of the possible ACS actions.

Id	C	E	M	A	Qe	Qp	sr	sl	i	di	d	dd
C0	####	####	mmmm	(+)	0.50	0.21	0	17	0	0	17	0
C1	####	####	mmmm	(-)	0.50	0.22	0	16	0	0	16	0
C2	###0	###1	####	(+)	0.50	1.00	1	99	98	0	0	0
C3	###1	###0	####	(-)	0.50	0.97	1	58	58	0	0	0
C4	#011	#100	####	(+)	0.50	1.00	1	91	90	0	0	0
C5	##01	##10	####	(+)	0.50	1.00	1	154	152	0	0	0
C6	0111	1000	####	(+)	0.50	0.96	1	50	50	0	0	0
C7	1000	0111	####	(-)	0.50	0.96	1	47	47	0	0	0
C8	0000	1111	####	(-)	0.50	0.96	1	47	47	0	0	0
C9	1111	0000	####	(+)	0.50	0.91	0	34	33	0	0	0
C10	##10	##01	####	(-)	0.50	1.00	1	124	122	0	0	0
C11	#100	#011	####	(-)	0.50	0.99	1	70	70	0	0	0

Table 6

A typical classifier list for the random walk problem, with no model learning.

Id	C	E	M	A	Qe	Qp	sr	sl	i	di	d	dd
C0	####	####	mmmm	(+)	0.50	0.17	0	8	0	0	8	13
C1	####	####	mmm0	(-)	0.50	0.15	0	5	0	0	4	20
C2	###0	###1	####	(+)	0.50	1.00	1	38	38	112	0	0
C3	0000	1111	####	(-)	0.50	0.98	1	13	12	52	0	0
C4	##01	##10	####	(+)	0.50	1.00	1	55	51	186	0	0
C5	#011	#100	####	(+)	0.50	1.00	1	26	25	109	0	0
C6	#100	#011	####	(-)	0.50	1.00	1	20	20	90	0	0
C7	0111	1000	####	(+)	0.50	0.98	1	12	11	50	0	0
C8	1111	0000	####	(+)	0.50	0.96	1	10	10	39	0	0
C9	##10	##01	####	(-)	0.50	1.00	1	28	27	94	0	0
C10	1000	0111	####	(-)	0.50	0.95	0	10	9	35	0	0

Table 7

A typical classifier result for the random walk problem, with mostly model learning.

Where :-

- Id = Assigned classifier identity.
- C = Condition part.
- E = Expectation part.
- M = Mark part.
- A = Action.
- Qe = Environmental reward (not used).
- Qp = Prediction quality.
- sr = Sure classifiers.
- sl = Run selection quantify.
- i = Real world increments.
- di = Model world increments.
- d = Real world decrements.
- dd = Model world decrements.

Where states :-

- 0000 = Most left hand side of walk.
- 1111 = Most right hand of walk.
- (-) = Move to the left.
- (+) = Move to the right.

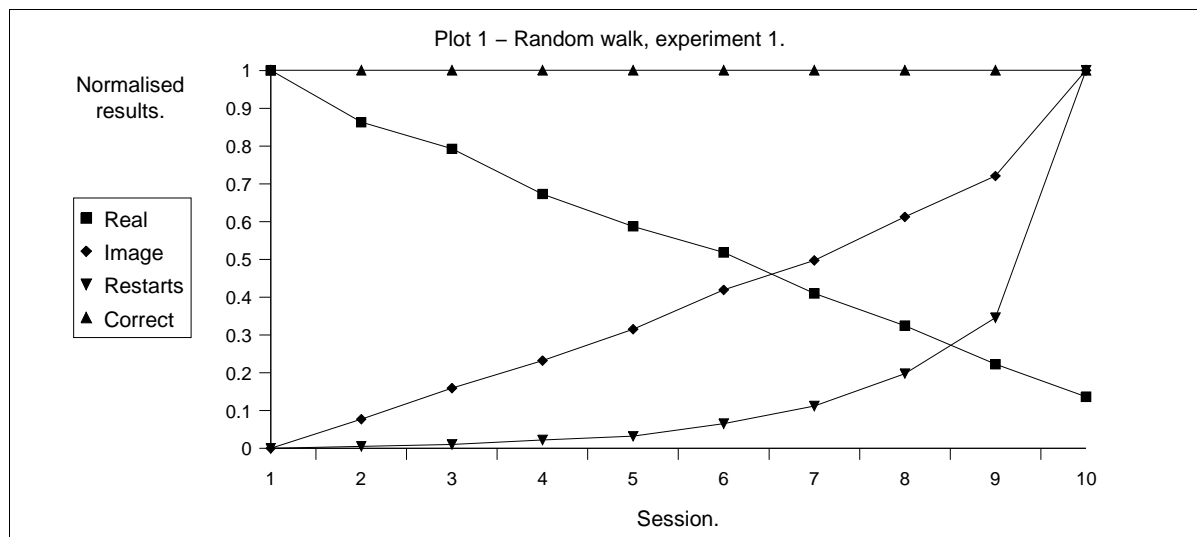
5.2 Results of Learning the Random Walk Maze

Experiment R-Walk 1

ACS classifier selection process of probabilistic bidding operates consistently throughout interacting with the real and model worlds. The model (dream) selection process is based on the selecting the best classifier on which to form the response (deterministic response).

Run-Real %	Run-Model %	Real	Model	Restarts	Correct	Baseline	Ave Classifiers
100	0	1152.18	0.00	0.00	100.00	5.87	10.87
90	10	994.23	105.36	0.36	100.00	5.88	10.60
80	20	913.00	218.29	0.76	100.00	5.85	10.50
70	30	775.42	317.38	1.59	100.00	5.84	10.35
60	40	676.82	431.72	2.31	100.00	5.87	10.16
50	50	597.54	573.55	4.73	100.00	5.80	9.99
40	60	472.58	680.75	8.10	100.00	5.87	9.60
30	70	373.81	837.96	14.30	100.00	5.84	9.32
20	80	256.48	985.96	24.95	100.00	5.87	8.54
10	90	156.93	1368.54	72.25	100.00	5.83	7.56

Table 8
Absolute results of experiment R-walk 1.



Plot 1

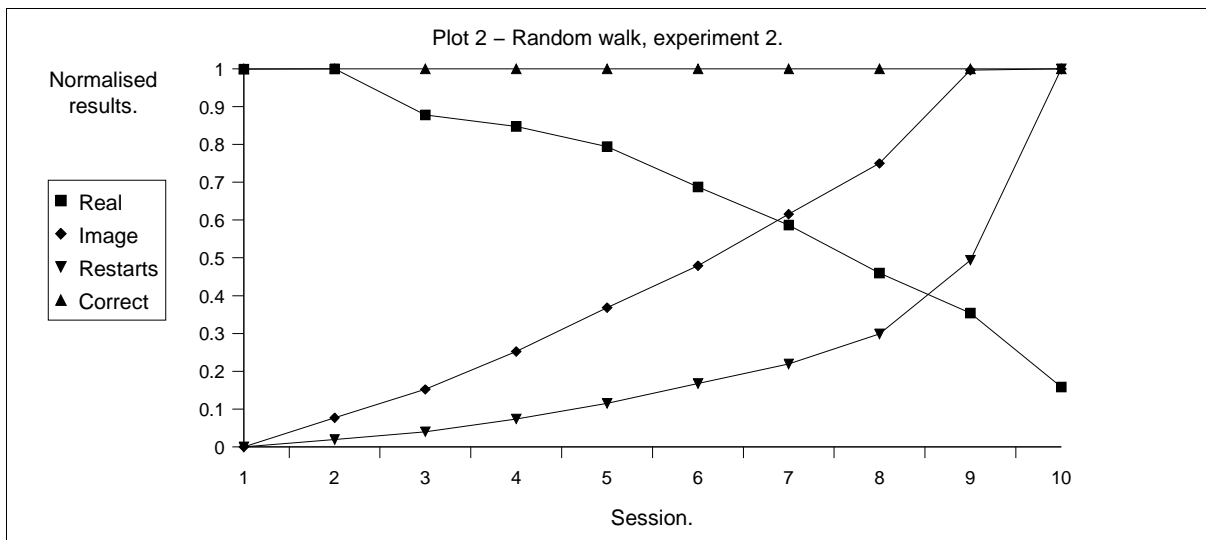
The graphical representation of table 8. The x-axis represents the results of the 10 sessions. Session 1 starts with 100% real world interactions (normal ACS) and session 10 ends with just 10% real world interactions and 90% model. The Y-axis illustrates the decreasing requirement to interact with the real world, (trace 'Real') and the corresponding increase in the requirement for model learning (trace 'Model') to maintain the system accuracy (trace 'Correct'). With the decrease of real interactions and increase model reliance, the world model becomes less useful during the model/dream phase and this increases restarts (trace 'Restart').

Experiment R-Walk 2

ACS classifier selection process of probabilistic bidding operates consistently throughout interacting with the real and model worlds. The model (dream) selection process is probabilistic.

Run-Real %	Run-Model %	Real	Model	Restarts	Correct	Baseline	Ave Classifiers
100	0	1090.24	0.00	0.00	100.00	5.90	10.72
90	10	1090.80	116.33	1.82	100.00	5.94	10.80
80	20	957.82	229.84	3.75	100.00	5.89	10.61
70	30	924.44	381.55	6.90	100.00	5.81	10.56
60	40	866.13	557.09	10.72	100.00	5.86	10.47
50	50	749.87	724.30	15.65	100.00	5.89	10.26
40	60	639.71	930.15	20.46	100.00	5.91	10.08
30	70	501.44	1133.58	27.85	100.00	5.90	9.66
20	80	386.15	1506.32	46.07	100.00	5.86	9.13
10	90	172.82	1511.56	93.26	100.00	5.82	7.73

Table 9
Absolute results of experiment R-walk 2.



Plot 2

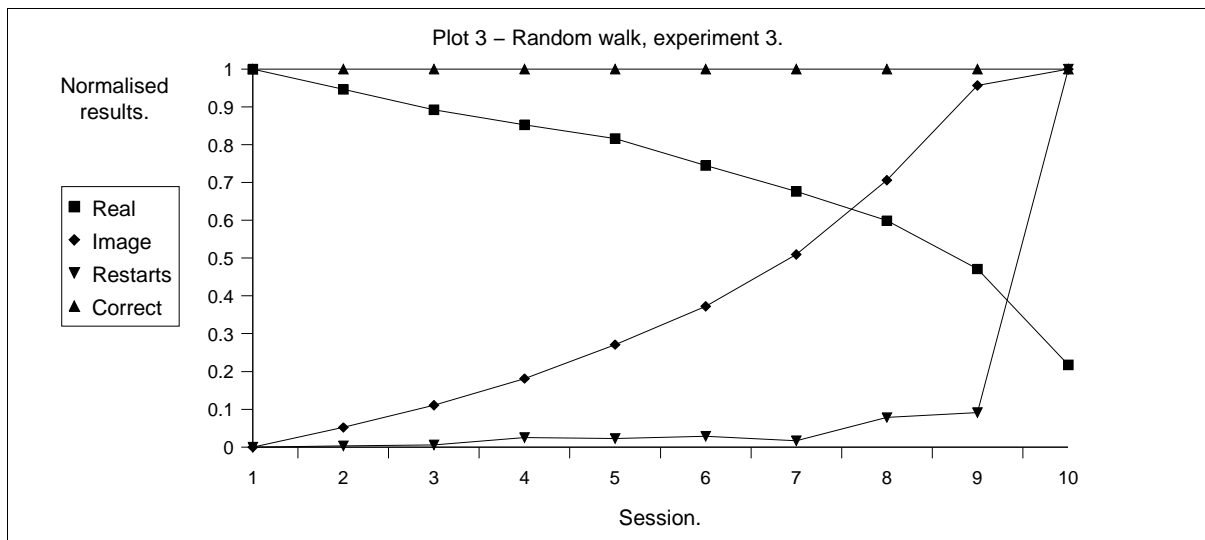
The graphical representation of table 9. The x-axis represents the results of the 10 sessions. Session 1 starts with 100% real world interactions (normal ACS) and session 10 ends with just 10% real world interactions and 90% model. The Y-axis illustrates the decreasing requirement to interact with the real world, (trace 'Real') and the corresponding increase in the requirement for model learning (trace 'Model') to maintain the system accuracy (trace 'Correct'). With the decrease of real interactions and increase model reliance, the world model becomes less useful during the model/dream phase and this increases restarts (trace 'Restart'). With the model selection process occasionally choosing an exploitive response the 'Restarts' trace shows an increase of use-less responses in comparison to R-walk experiment 1 where the model selection policy always chooses the best response.

Experiment R–Walk 3

ACS classifier selection process of probabilistic bidding operates differently between interacting with the real and model worlds. During interaction with the real world selection is probabilistic, during the model phase selection is optimal. The model (dream) selection process is based on the selecting the best classifier on which to form the response (deterministic response).

Run–Real %	Run–Model %	Real	Model	Restarts	Correct	Baseline	Ave Classifiers
100	0	1116.56	0.00	0.00	100.00	5.89	10.90
90	10	1039.38	110.65	0.02	100.00	5.88	10.83
80	20	1017.68	244.69	0.04	100.00	5.85	10.76
70	30	985.67	407.80	0.12	100.00	5.89	10.79
60	40	928.64	598.93	0.40	100.00	5.78	10.83
50	50	828.63	802.30	0.10	100.00	5.87	10.78
40	60	781.09	1142.13	0.60	100.00	5.86	10.68
30	70	669.93	1530.53	0.97	100.00	5.88	10.67
20	80	517.60	2036.24	1.11	100.00	5.86	10.25
10	90	237.93	2116.99	4.59	100.00	5.82	9.10

Table 10
Absolute results of experiment R–walk 3.



Plot 3

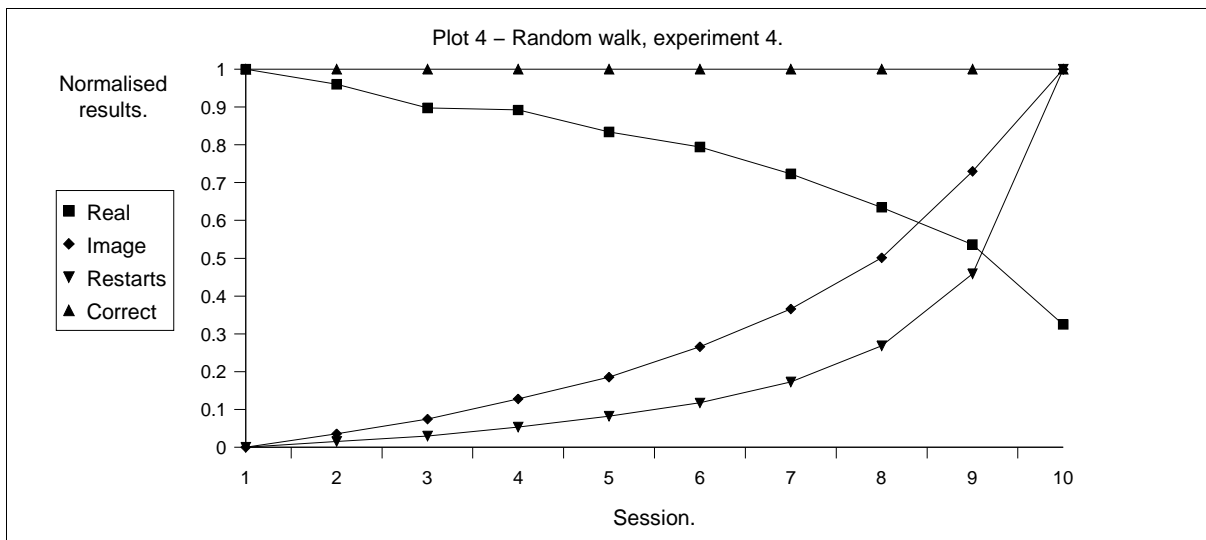
The graphical representation of table 10. The x-axis represents the results of the 10 sessions. Session 1 starts with 100% real world interactions (normal ACS) and session 10 ends with just 10% real world interactions and 90% model. The Y-axis illustrates the decreasing requirement to interact with the real world, (trace 'Real') and the corresponding increase in the requirement for model learning (trace 'Model') to maintain the system accuracy (trace 'Correct'). With the decrease of real interactions and increase model reliance, the world model becomes less useful during the model/dream phase and this increases restarts (trace 'Restart'). The ACS occasionally explores during interaction with the real world and then switching to exploit only during the model phase. The model selection policy operates only in exploit, choosing the best responses. This has the effect of the system cycling about state pairs which reduces use–less responses as represented by the 'Restarts' trace. However this rapidly starts to break down as the model becomes less complete.

Experiment R–Walk 4

ACS classifier selection process of probabilistic bidding operates differently between interacting with the real and model worlds. During interaction with the real world selection is deterministic, during the model phase selection is probabilistic. The model (dream) selection process is probabilistic.

Run–Real %	Run–model %	Real	Model	Restarts	Correct	Baseline	Ave Classifiers
100	0	1106.77	0.00	0.00	100.00	5.90	10.70
90	10	1062.93	113.07	0.94	100.00	5.85	10.82
80	20	993.84	238.52	1.87	100.00	5.89	10.69
70	30	987.46	408.34	3.30	100.00	5.80	10.82
60	40	923.33	594.92	5.12	100.00	5.86	10.73
50	50	878.77	852.93	7.30	100.00	5.88	10.84
40	60	800.22	1171.21	10.77	100.00	5.91	10.82
30	70	702.39	1605.29	16.72	100.00	5.88	10.74
20	80	593.55	2337.69	28.57	100.00	5.81	10.64
10	90	359.39	3203.96	62.23	100.00	5.86	10.13

Table 11
Absolute results of experiment R–walk 4



Plot 4

The graphical representation of table 11. The x-axis represents the results of the 10 sessions. Session 1 starts with 100% real world interactions (normal ACS) and session 10 ends with just 10% real world interactions and 90% model. The Y-axis illustrates the decreasing requirement to interact with the real world, (trace 'Real') and the corresponding increase in the requirement for model learning (trace 'model') to maintain the system accuracy (trace 'Correct'). With the decrease of real interactions and increase model reliance, the world model becomes less useful during the model/dream phase and this increases restarts (trace 'Restart'). The 'Restarts' trace highlights the occasional exploration response of the model selection policy.

5.4 The T-Maze Problem

The T-maze experiment has its roots in animal psychology experiments originally devised to explore the latent learning capabilities of rats (Blodgett 1929) (Tolman 1932). The T-maze is so described due to the shape of the maze. The maze starts at the base of the 'T' and follows up to the bridge of the 'T' where an agent in the maze will have to make a decision to turn left or right toward either extent of the top of the 'T'. The decision at the 'T' junction is the key aspect of the maze. In the animal experiments rats could not see the differently discriminated chambers (coloured or textured) located at the end of the horizontal top bar section (figure 4). The rats had to make a visit to the extents of the top 'T' bar in order to discover the different end chamber colours or textures. Rats were allowed to explore the maze without any reward for a period of time. Rats were then directly placed in one of the end chambers with a reward. After the rats had consumed the reward they were then replaced at the start of the maze and their behaviour monitored, the theory being that, if the rats built up a cognitive map of the maze during the UN-rewarded exploration phase, then during post reward trials the rats would head straight back to the particular coloured/textured chamber where the reward had been experienced.

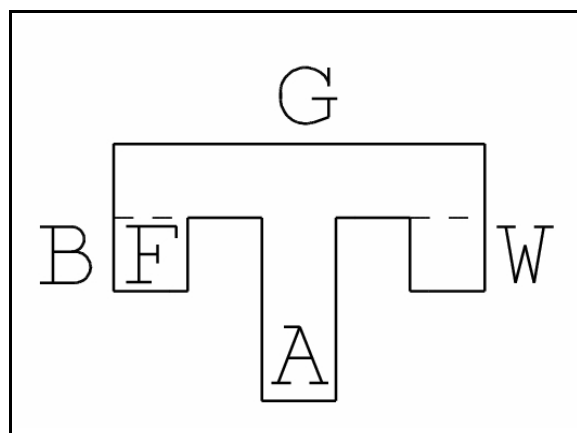


Figure 4

Schematic of the classic T-maze experiment, where 'A' represents the agent and 'F' represents the reward or food in the maze. All the walls are gray 'G' with the exception of the two end chambers that have black 'B' and white 'W' walls. The agent is shown at the start of the maze with the food located in the black end chamber.

The ACS was originally applied to this experiment to illustrate the latent learning abilities of the ACS and similarities to the original rat experiments (Tolman 1932). In this instance the experiment is not fully repeated, in that environmental motivation is not included. The experiment here simply uses the 'T' maze as a structure to learn, and illustrate the benefit of model interactions. As with the previous random walk problem, the maze is simply explored and terminated when a certain confidence level is attained, when the maze has been latently discovered.

010g	<- L	011g	R->	001g
		^		
v L		F		v R
000b	W->	100g	<-W	000w

Table 12

Stolzmann's coding of the T-maze.

Forward move	Left move	Right move	Wall colour	Possible motions.
1	0	0	g	Possible only to move forward, gray walls.
0	1	1	g	Possible only to move left and right, gray walls.
0	1	0	g	Possible only to move left, gray walls.
0	0	1	g	Possible only to move right, gray walls.
0	0	0	b	End chamber, black walls.
0	0	0	w	End chamber, white walls.

Table 13
Example T-maze state representations.

Agent action.	Description.
F	Move forward.
L	Move left.
R	Move right.
W	Move to start of maze.

Table 14
Definitions of the possible ACS actions.

Id	C	E	M	A	Qe	Qp	sr	sl	i	di	d	dd
C0	####	####	100g	(F)	0.50	0.22	0	16	0	0	16	0
C1	####	####	01mg	(L)	0.50	0.20	0	19	0	0	18	0
C2	####	####	0m1g	(R)	0.50	0.26	0	14	0	0	13	0
C3	####	####	000m	(W)	0.50	0.19	0	19	0	0	19	0
C4	100#	011#	####	(F)	0.50	1.00	1	258	255	0	0	0
C5	#1##	#0##	####	(R)	0.50	0.95	1	96	78	0	18	0
C6	#01g	#00w	####	(R)	0.50	1.00	1	108	107	0	0	0
C7	##1#	##0#	####	(L)	0.50	0.92	0	72	65	0	7	0
C8	0##w	1##g	####	(W)	0.50	1.00	1	99	99	0	0	0
C9	#10g	#00b	####	(L)	0.50	0.99	1	82	81	0	0	0
C10	0##b	1##g	####	(W)	0.50	0.99	1	85	i83	0	0	0

Table 15

A typical classifier result for the T-maze problem, with no model learning.

Id	C	E	M	A	Qe	Qp	sr	sl	i	di	d	dd
C0	####	####	100g	(F)	0.50	0.02	0	7	0	0	6	55
C1	####	####	01mg	(L)	0.50	0.02	0	8	0	0	8	52
C2	####	####	0m1g	(R)	0.50	0.02	0	10	0	0	9	54
C3	####	####	000m	(W)	0.50	0.02	0	4	0	0	4	55
C4	100#	011#	####	(F)	0.50	1.00	1	96	93	794	0	0
C5	0##b	1##g	####	(W)	0.50	1.00	1	35	35	364	0	0
C6	#10g	#00b	####	(L)	0.50	1.00	1	50	36	373	0	0
C7	#1##	#0##	####	(R)	0.50	0.99	1	30	27	202	0	32
C8	#01g	#00w	####	(R)	0.50	0.99	1	12	8	79	0	0
C9	0##w	1##g	####	(W)	0.50	0.95	1	2	2	43	0	0

Table 16

A typical classifier result for the T-maze problem, with model learning.

Where :-

Id = Assigned classifier identity.
C = Condition part.
E = Expectation part.
M = Mark part.
A = Action.
Qe = Environmental reward (not used).
Qp = Prediction quality.
sr = Sure classifiers.
sl = Run selection quantify.
i = Real world increments.
di = Model world increments.
d = Real world decrements.
dd = Model world decrements.

Where states :-

Bit 0 = Wall colour, g = gray, w = white and b = black.
Bit 1 = Possible to move right.
Bit 2 = Possible to move left.
Bit 3 = Possible to move forward.

Actions are :-

F = Move forwards.
L = Move left.
R = Move right.
W = Move to start position when in end chambers.

Where state coding is identical to Stolzmann (1998).

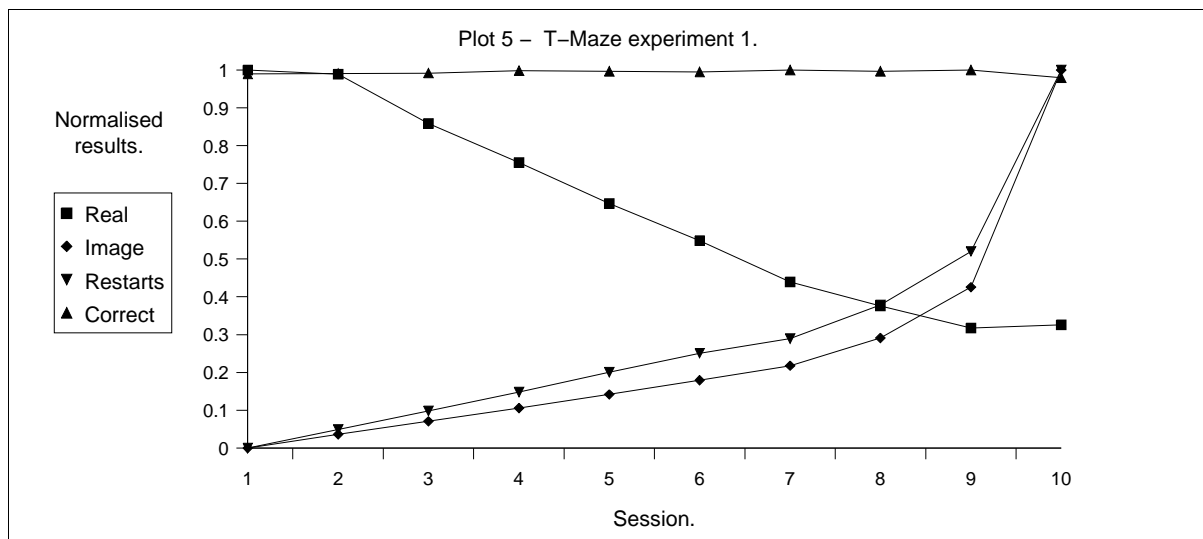
5.5 Result from Learning the T-Maze

Experiment T-maze 1

ACS classifier selection process of probabilistic bidding operates consistently throughout interacting with the real and model worlds. The model (dream) selection process is based on selecting the best classifier on which to form the response (deterministic response).

Run-Real %	Run-Model %	Real	Model	Restarts	Correct	Baseline	Ave Classifiers
100	0	916.03	0	0	99.01	33.26	11.58
90	10	905.68	95.92	20.98	99.04	33.7	11.46
80	20	786.67	187.15	41.83	99.18	33.31	11.32
70	30	691.81	281.01	63.15	99.85	33.05	11.13
60	40	592.66	375.58	85.69	99.67	33.45	11.27
50	50	502.28	474.74	106.98	99.51	33.26	11.15
40	60	402.82	577.3	123.5	100	33.36	11.03
30	70	343.95	771.29	161.3	99.66	33.43	11.02
20	80	290.93	1127.52	221.85	100	33.42	11
10	90	298.39	2649.79	426.4	97.94	33.38	10.32

Table 17
Absolute results of experiment T-maze 1.



Plot 5

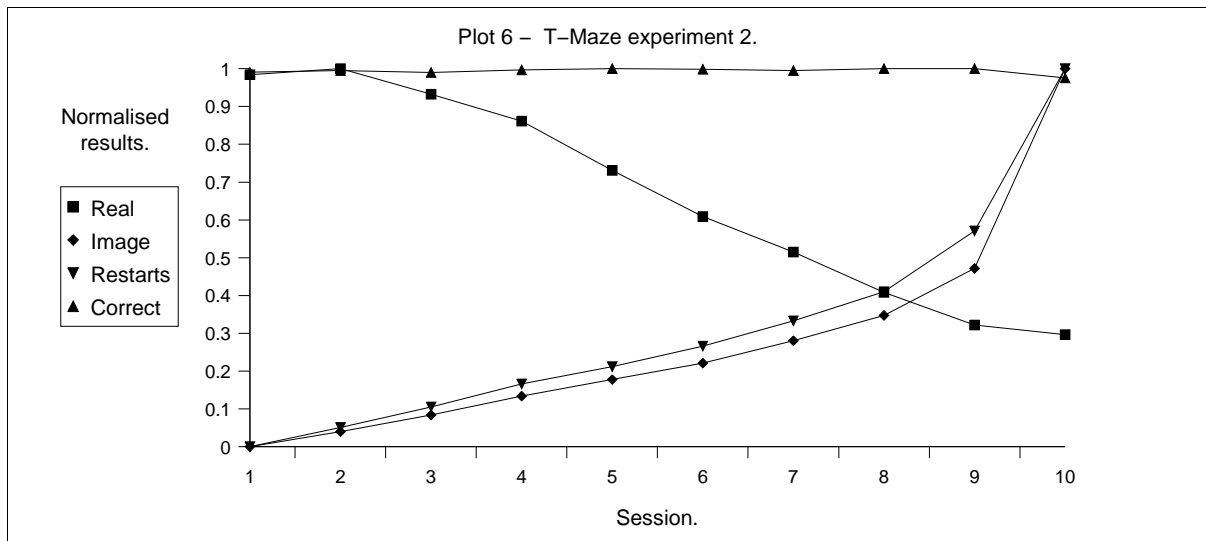
The graphical representation of table 17. The x-axis represents the results of the 10 sessions. Session 1 starts with 100% real world interactions (normal ACS) and session 10 ends with just 10% real world interactions and 90% model. The Y-axis illustrates the decreasing requirement to interact with the real world, (trace 'Real') and the corresponding increase in the requirement for model learning (trace 'model') to maintain the system accuracy (trace 'Correct'). With the decrease of real interactions and increase model reliance, the world model becomes less useful during the model/dream phase and this increases restarts (trace 'Restart').

Experiment T-maze 2

ACS classifier selection process of probabilistic bidding operates consistently throughout interacting with the real and model worlds. The model (dream) selection process is probabilistic.

Run-Real %	Run-Model %	Real	Model	Restarts	Correct	Baseline	Ave Classifiers
100	0	935.01	0	0	99.04	33.19	11.62
90	10	950.31	100.74	18.76	99.49	33.55	11.33
80	20	886.41	210.74	38.89	99.01	33.52	11.26
70	30	818.61	335.92	61.36	99.69	33.57	11.13
60	40	694.84	444.65	78.13	100	33.54	11.07
50	50	579.02	553.66	98.02	99.83	33.23	11.08
40	60	489.31	701.52	122.7	99.49	33.31	11.1
30	70	387.72	870.38	151.13	100	33.28	11
20	80	305.97	1182.14	210.55	100	33.33	10.99
10	90	282.15	2504.3	368.73	97.52	33.53	10.36

Table 18
Absolute results of experiment T-maze 2.



Plot 6

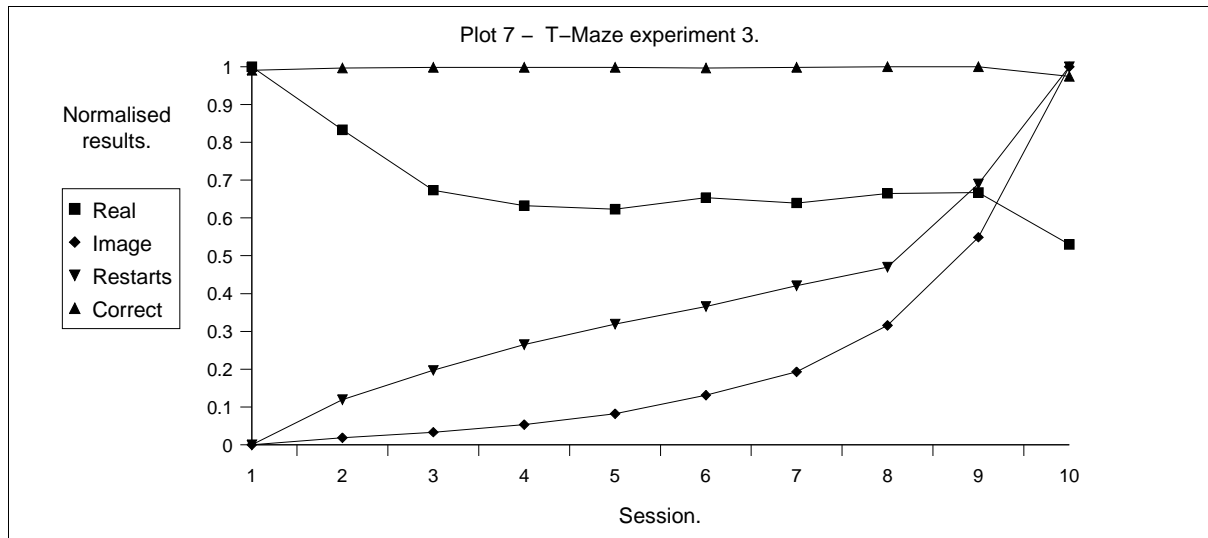
The graphical representation of table 18. The x-axis represents the results of the 10 sessions. Session 1 starts with 100% real world interactions (normal ACS) and session 10 ends with just 10% real world interactions and 90% model. The Y-axis illustrates the decreasing requirement to interact with the real world, (trace 'Real') and the corresponding increase in the requirement for model learning (trace 'Model') to maintain the system accuracy (trace 'Correct'). With the decrease of real interactions and increase model reliance, the world model becomes less useful during the model/dream phase and this increases restarts (trace 'Restart'). The occasional explorative responses from the model selection policy slightly improves performance in comparison to the purely exploitative responses in T-maze experiment 1.

Experiment T-maze 3

ACS classifier selection process of probabilistic bidding operates differently between interacting with the real and model worlds. During interaction with the real world selection is probabilistic, during the model phase selection is optimal. The model (dream) selection process is based on the selecting the best classifier on which to form the response (deterministic response).

Run-Real %	Run-Model %	Real	Model	Restarts	Correct	Baseline	Ave Classifiers
100	0	922.25	0	0	99.03	33.17	11.66
90	10	768.21	80.26	16.36	99.65	33.31	11.66
80	20	621.21	146.21	26.93	99.84	33.2	11.91
70	30	583.23	234.95	36.24	99.82	33.49	11.83
60	40	574.73	359.12	43.67	99.85	33.22	11.61
50	50	602.77	574.59	50.06	99.67	33.22	11.65
40	60	589.97	848.62	57.55	99.84	33.74	11.7
30	70	613.44	1387.58	64.32	100	33.41	11.82
20	80	615.47	2409.8	94.44	100	33.18	11.79
10	90	488.96	4389.45	136.82	97.49	33.1	11.42

Table 19
Absolute results of experiment T-maze 3.



Plot 7

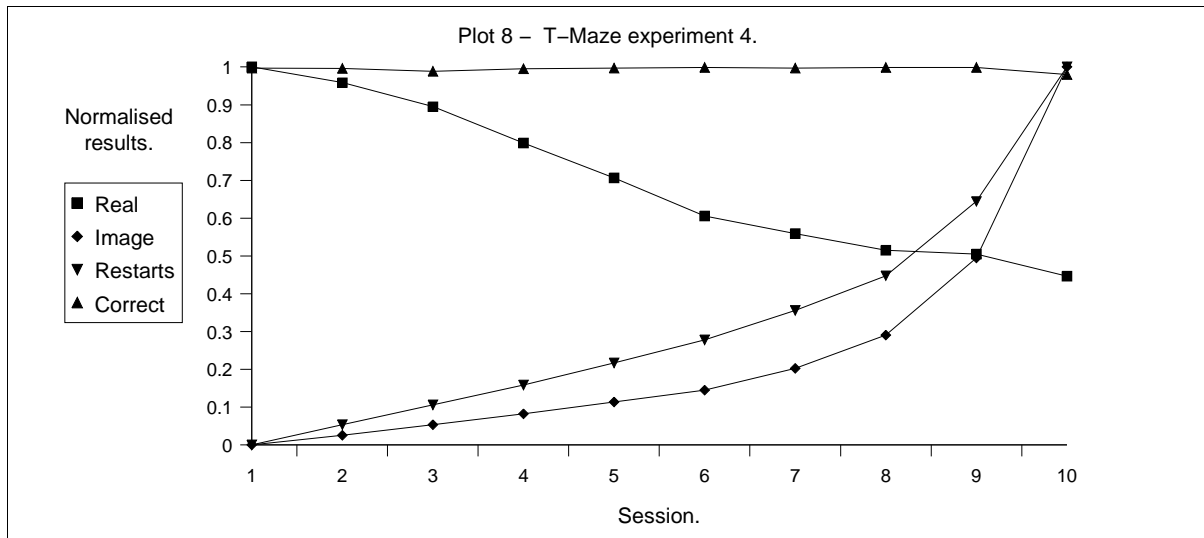
The graphical representation of table 19. The x-axis represents the results of the 10 sessions. Session 1 starts with 100% real world interactions (normal ACS) and session 10 ends with just 10% real world interactions and 90% model. The Y-axis illustrates the decreasing requirement to interact with the real world, (trace 'Real') and the corresponding increase in the requirement for model learning (trace 'Model') to maintain the system accuracy (trace 'Correct'). With the decrease of real interactions and increase model reliance, the world model becomes less useful during the model/dream phase and this increases restarts (trace 'Restart'). The larger problem space of the T-maze in comparison to the R-maze is highlighted by the consistent quantity of real interactions that are required. Exploitation on the part of the ACS and the model selection policy during the model phase fails to explore the larger space of the T-maze.

Experiment T-maze 4

ACS classifier selection process of probabilistic bidding operates differently between interacting with the real and model worlds. During interaction with the real world selection is deterministic, during the model phase selection is probabilistic. The model (dream) selection process is probabilistic.

Run-Real %	Run-Model %	Real	Model	Restarts	Correct	Baseline	Ave Classifiers
100	0	943.94	0	0	99.66	33.4	11.63
90	10	904.74	95.68	14.95	99.62	33.2	11.33
80	20	844.72	200.96	29.73	98.8	33.24	11.25
70	30	754.1	308.89	44.29	99.5	33.3	11.28
60	40	666.77	427.56	60.56	99.67	33.41	11.18
50	50	571.38	545.66	77.72	99.84	33.21	11.21
40	60	527.89	760.22	99.39	99.69	33.4	11.27
30	70	486.23	1091.7	125.14	99.84	33.22	11.28
20	80	476.8	1859.7	180.05	99.83	33.28	11.28
10	90	421.26	3758.33	279.47	98.02	33.49	10.91

Table 20
Absolute results of experiment T-maze 4.



Plot 8

The graphical representation of table 20. The x-axis represents the results of the 10 sessions. Session 1 starts with 100% real world interactions (normal ACS) and session 10 ends with just 10% real world interactions and 90% model. The Y-axis illustrates the decreasing requirement to interact with the real world, (trace 'Real') and the corresponding increase in the requirement for model learning (trace 'Model') to maintain the system accuracy (trace 'Correct'). With the decrease of real interactions and increase model reliance, the world model becomes less useful during the model/dream phase and this increases restarts (trace 'Restart'). The occasional exploration during the model phase improves performance in comparison to T-maze experiment 3.

6 Interpretation of the Results

6.1 Random Walk Maze

Each session that is run with the differing ratios of real plant interaction and model interaction are only terminated when the classifier system can predict the correct response for both actions in every state, all of the classifiers that are used to generate this response must have a high confidence, a prediction quality of 90% or above, aka 'sure' classifiers. All the results indicate the primary expected outcome, interaction with the real world can be reduced if that reduction is replaced by interaction with a suitable model of the world.

More interesting are the subtle differences caused by the alternate selection policies for both the generation of the model response (dream selection policy) and the usual ACS selection policy. The differences are emphasised as the real world interactions are very much smaller than the imaginary interactions.

In R-walk experiment 1 and 2, where the ACS selection policy did not change between interacting with the real or model world, the least real world interactions occur then the model selects only the best responses and more real world interactions are required if the model occasionally chooses an exploratory response; with the useless responses increasing as indicated with the 'restarts' plot.

In R-walk experiment 2 and 3, the ACS selection policy does change between interacting with the real or model world. In R-walk experiment 3 both selection policies are greedy. The ACS does not explore during the model phase and the model selection policy only returns the best estimates. This strategy requires more real world interactions than when the ACS continues to explore during the model phase. The restarts metric indicates that there are few useless responses from the model and thus indicates that more real world cycles are required because the two policies are causing oscillations between adjacent classifier rules and failing to move about the environment. In the final R-walk experiment the ACS continues to select greedily whilst the model explores with alternate responses. As real world interaction demises the exploration around the incomplete model causes many useless responses.

These results then are as to be expected, the best result in terms of least interaction with the real world to achieve the same result is achieved when the ACS continues as normal occasionally exploring whilst the model replies with the best possible response. The worst response occurs when the ACS never explores and the model does, producing many useless responses.

Straightforward though these results are, they illustrate two key facets of on-going and future work, how the agent (ACS) makes use of the model interaction and how the interaction develops the model selection strategy.

6.2 Interpretation of the Results from the T-Maze

Whilst returning similar results to that of the random walk experiments this slightly larger problem has emphasised two results. Firstly, in experiment T-maze 2, where the model selection policy does not always return the best prediction of the next state but explores amongst close predictions and proves to perform better in the long term than always returning the best expectation. The slight deviations from the best predictions are improving the ACS model in the long term. Secondly in experiment T-maze 3, more real interactions are consistently required when the ACS exploits as does the model selection policy, indicating a cyclic behaviour which has a long term detrimental effect on learning the whole maze.

7 Discussion and Future Work

The structure presented reduces real world interaction in a similar method to other model learning systems most noticeably Dyna-Q and derivatives (Sutton & Barto 1998) and other model featuring classifier systems (Riolo 1991, Roberts 1993, Wilson 2000, Bull 2001) of which some have combined Dyna style extensions (Gerard et al 2000, Stolzmann et al 2000). However these systems have developed from a machine learning perspective (with perhaps the exception of Stolzmann et al 2000). Two important concepts contrast within this work, firstly in the dream motivation and secondly in the structure. The structure employs an addition *dream* agent that utilises the generalised model representation held in the classifier list to generate a dynamic model representation of the real world.

The selection of classifier rules during the dream phase was biased in the same way as that of real waking interaction, a bias towards more specific good classifiers. The dream phase circles around known good positions, reiterating associations. This particular dream policy is useful in the static problems tested. The function of generating useful threads and how the thread modifies existing rules is the subject of continuing work. It is difficult to imagine explicitly generating dream threads that are useful to the agent by hand. However, with this architecture, dream threads are an emergent property of the selection process. The selection process could easily be modified further by various methods, such as mutating attributes from classifiers that have a common condition part (in a match or action set) or filling in missing attributes from other similar classifiers in uncertain situations. During the dream thread execution several actions could be taken to generate new rules, assimilate rules or delete rules. Various strategies for different problems could be hand crafted to explore these ideas and this is certainly a good starting point, however ideally this dream concept should in itself be adaptive. The dream policy and rule adaptation scheme should be able to modulate behaviour between a static safe environment and a dynamic dangerous environment, figure 5 and figure 6 illustrate the architectural operation of the next stage of work.

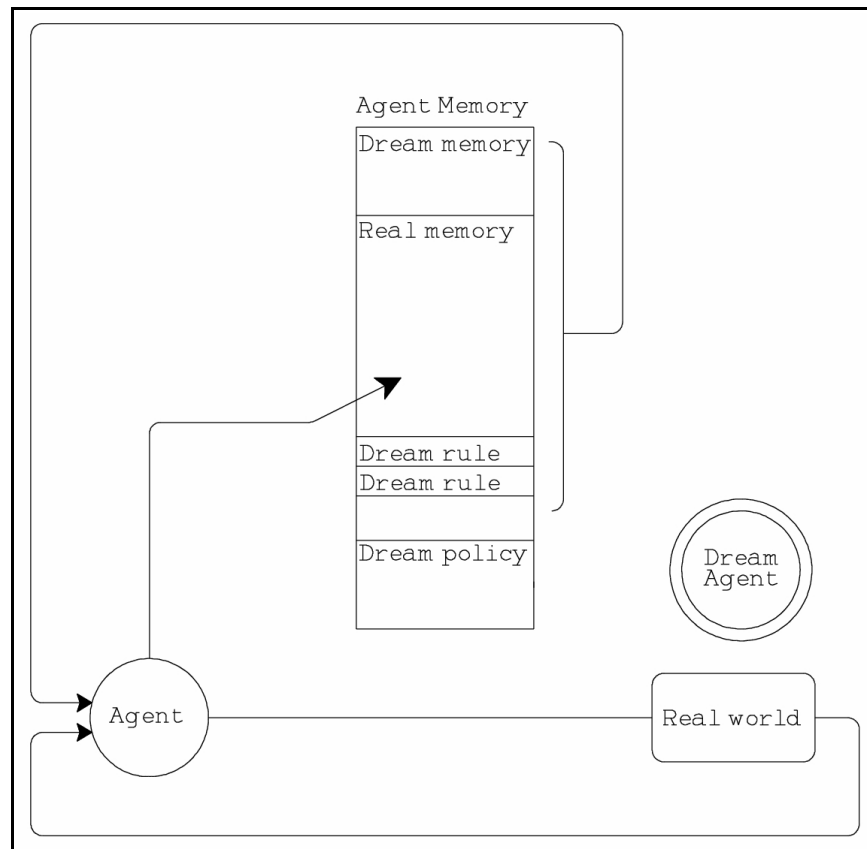


Figure 5

Agent interacting with the real world. In some situations (such as uncertainty) rules are used from the dream memory, if successful these can be incorporated into the real memory (long term memory). The dream agent is shown inactive during this 'waking phase'.

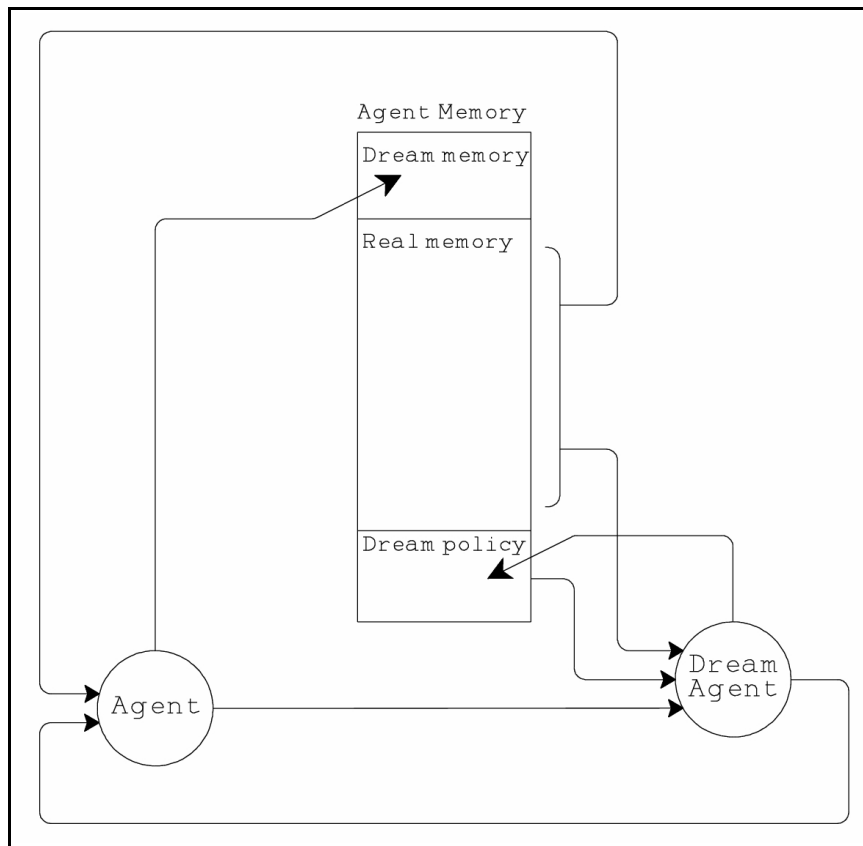


Figure 6

Agent interacting with the dream world. The agent is interacting with the dream world (dream agent) as though interacting with reality with the exception that any learning (new rules) are stored separately. When interacting with reality (figure 5) these rules can be explored and if successful incorporated into the real memory (long term memory). The dream agent is shown modifying rule creation policy, whereby performance is based on the inclusion and or value rate of the dream rules created. The merit of the dream thread or story line is guided by the dream policy which can be adapted to improve the quantitative and or qualitative value of dream rule creation.

More specifically with respect to the ACS further work will be focused on two areas, that of dream thread generation and rule adaptation. Modifications to the ACS structure are also required, such as, automatic state generation schemes. This is where internal states or indistinguishable external states are not stored in explicit alternative memory structures, but as additional attributes of the condition and expectation part of the classifier rules.

The ACS and other similar classifier systems rely on 'rough' rule creation scheme and then operate a genetic competition amongst the rule base with a goal to generate an accurate but maximally general rule set. Genetic competition has been successfully applied to the ACS (Butz et al 2000, Butz 2001) and in some aspects can be compared to the proposed system described herein. The difference between the proposed system and an GA augmented ACS is the route and not the final destination, indeed comparisons between the two in a static problem space would ideally show that the proposed structure only transiently out performed ACS/GA, but consistently performed better overall on a dynamic problem.

Problems upon which to base performance will obviously be important, perhaps most of all will be problems that require a 'leap of imagination', problems where goal states are distant and unguided. Problems that require an intermediate dream stepping stone in order to become solvable. It is just these types of procedural problems that dream deprivation seems to effect, and by incorporation of dream style wanderings that will give further insight to solutions to such problems.

No environmental reward was introduced in either of the problems, but this will clearly be an important aspect of future work. Incorporating fight or flight in a predator-prey situation, for example, simulating primitive desires or emotions. Dream content in humans and other animals tends to be emotionally charged, maintaining consistent agent drive through dream scenarios is a simple and possibly powerful method of experience maintenance and speculation.

8 Conclusion

Essentially this report has proposed a general idealistic extension to the ACS inspired by proposed dream function that is reinforced by recent neurological dream studies. More specifically a simple extension that allows the ACS agent to replay state to state transitions as though real on two small problems, that of the random walk and the T-maze. The results have shown that this reduces real world learning interactions. Clearly much more work needs to be done to modify the ACS or similar model building architecture in order to create a system that proves a benefit solving dynamic problems.

The ACS was chosen as a test bed architecture partly because the initial structure avoided use of genetic manipulation of the rule base as with other classifier systems. There is no intention to incorporate genetic competition amongst rules, which may be surprising since this is key to much learning classifiers systems work. In some areas the proposed work may replicate some of the benefits of genetic competition, indeed comparisons will be interesting. Through further work it may transpire that the classifier framework may not be the most suitable, but the simplicity and transparency should provide a sound platform from which to start.

In respect to the long term goals of this current research it interesting to conclude with a quote from neurological researchers Kenway Louie and Matthew Wilson (Wilson et al 2001) on results from their research into rodent sleep behaviour. In their recent study they were able to detect temporal reactivation of spatial hippocampal place cells during REM sleep periods that were very similar to activations when awake :-

".....This reactivation of previous behavioural representations may be important for the learning of procedural tasks, which is dependent upon REM sleep (Karni et al., 1994). Mnemonic information that may have shared characteristics along a particular behavioural axis such as emotion could be juxtaposed and evaluated for common causal links, allowing adaptive behaviour change based on prior experience (Hobson et al., 1998)"

(Wilson et al 2001)

9 Experimental details

The ACS employed was similar to that of Stolzmann's original version used in the T-maze experiment, unless stated otherwise procedures and parameters were identical (Stolzmann 1998).

System parameters were set as follows :-

- Initial classifier prediction quality value, $initial_predictionq (Q_p) = 0.5$;
- Initial classifier environmental reward quality value, $initial_predictq (Q_e) = 0.5$ (not used)
- Anticipatory learning coefficient, $bant (B) = 0.05$;
- Reward learning coefficient, $brei = 0.2$; (not used)
- Classifier deletion threshold, $dl = 0.1$;
- E-sure value, $es = 0.9$;

No environmental reward learning was used in either problem.

Classifiers could be generated by specification of changing components and specification of unchanging components.

Classifier creation during the model phase was inhibited.

Each session always started by interacting with the real world first. For example in last session only 10% of the interactions were with the real world, the first 10 steps were with the real world, followed by 90 interactions with the current world model. This split between the real and model world interaction continues until the exit criteria is satisfied or the maximum session steps are exceeded.

Each session ran until the exit satisfaction was met or session steps of both real and model interactions exceeded 5000 steps.

Each session consists of a repetition of 100 trials, over which results are averaged.

The session exit criteria is only satisfied when the classifier list is able to classify a complete sweep of the problem space with classifier rules that had a high prediction confidence (sure).

Random number routines were platform independent taken from Press et al 1988 and seeded from the system clock at the start of each experiment.

A simplified flow chart of the experiment procedure employed is illustrated in figure 7.

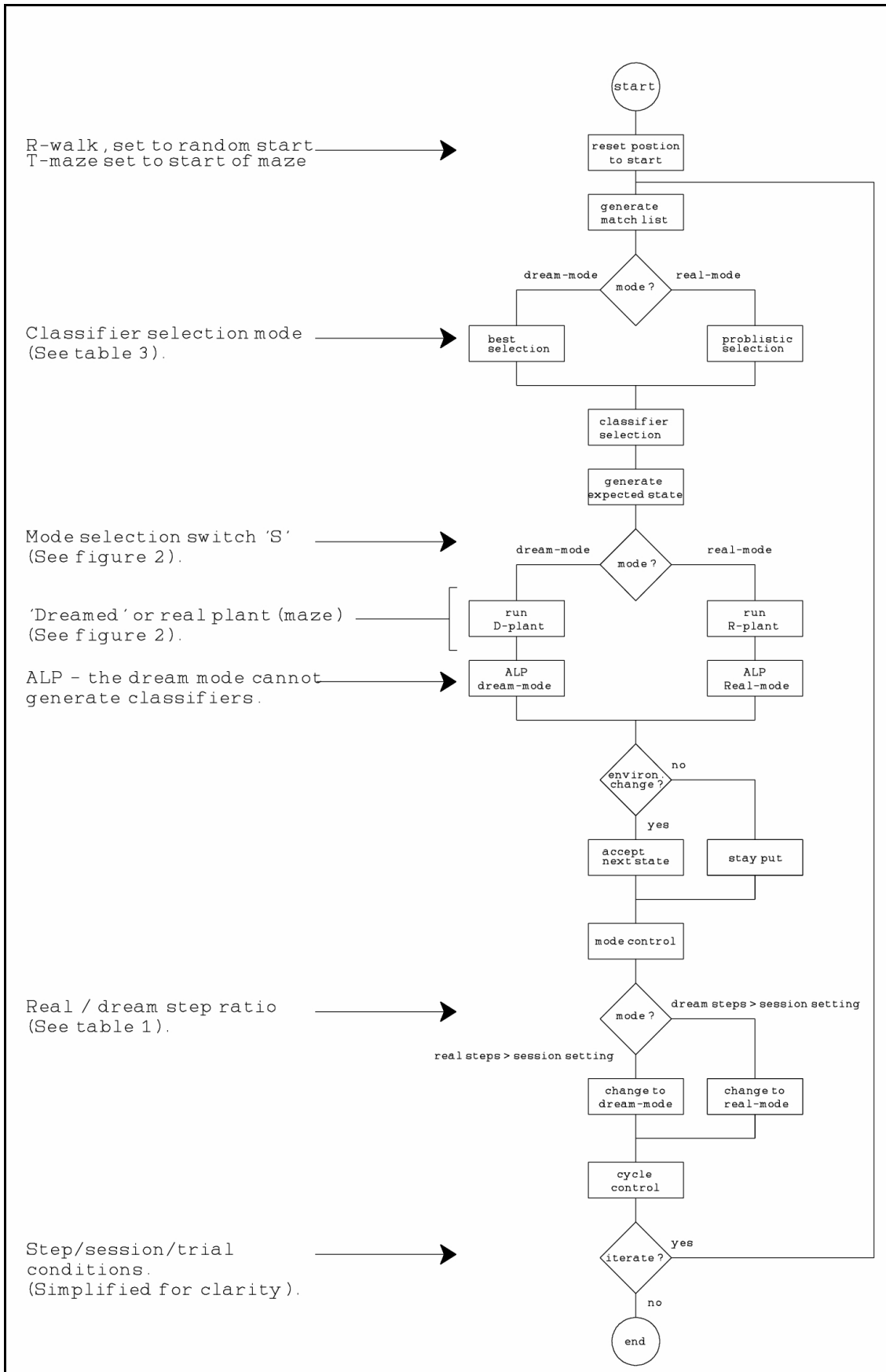


Figure 7
Simplified flow chart of the experimental procedure.

References

- Aserinsky, E. and Kletmen, N. (1953) Regularly Occurring Periods of Eye Mobility, and Concomitant Phenomena, During Sleep. *Science* 118: Pages 273–274.
- Blodgett, H. (1929) The Effect of Introduction of Reward Upon the Maze Performance of Rats, *Psychological Review*, Vol. 4. Pages 113–134.
- Bull, L. (2001) Lookahead and Latent Learning in ZCS, Learning Classifier Systems Group Technical Report UWELCSG01–004, University of the West of England.
- Butz, M. V. (2001) Anticipatory Learning Classifier Systems, *Genetic Algorithms and Evolutionary Computation*, 4. Kluwer Academic Publishers. ISBN 0–792–37630–7.
- Butz, M. V., Goldberg, D. E. and Stolzmann, W. (2000) Introducing a Genetic Generalisation Pressure to the Anticipatory Classifier System: Part 1 – Theoretical Approach. Submitted to the Genetic and Evolutionary Computation Conference (GECCO–2000).
- Butz, M. V., Goldberg, D. E. and Stolzmann, W. (2000) The Anticipatory Classifier System and Genetic Generalisation. Technical Report 2000032, Illinois Genetic Algorithms Laboratory.
- Gerard, P. (to appear, 2000). Combining Anticipation and Dynamic Programming in Classifier Systems. In Stolzmann, W., Lanzi, P.–L., and Wilson, S. W., (Eds.), *Third International Workshop on Learning Classifier Systems*, Paris, France.
- Gerard, P. and Sigaud, O. (2001) YACS: Combining Dynamic Programming with Generalization in Classifier Systems. In *Advances in Classifier Systems*, Vol. 1996 of LNAI, Springer–Verlag. Pages 52–69.
- Gerard, P., Meyer J. A. and Sigaud, O. (2003) Combining Latent Learning with Dynamic Programming in the Modular Anticipatory Classifier System. *European Journal of Operation Research* (submitted 2003).
- Hobson, J. A. (2002) *Dreaming, An Introduction to the Science of Sleep*. Oxford University Press Inc., New York. ISBN 0–19–280304–2.
- Hobson, J. A., Pace–Schott, E.F., Stickgold, R. and Kahn, D. (1998) To Dream or Not to Dream ? Relevant Data from New Neuroimaging and Electrophysiological Studies. *Current Opinions in Neurobiology*, 8. Pages 239–244.
- Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, Michigan. ISBN 0–262–08213–6.
- Holland, J. H. (1976) *Adaptation, Progress in Theoretical Biology IV*, Editor Rosenblatt, R. F. New York Academic Press.
- Holland, J. H. (1990) Concerning the Emergence of Tag–Mediated Lookahead in Classifier Systems. In Forest, S. (Ed.), *Emergent Computation. Proceedings of the Ninth Annual International Conference of the Center for Nonlinear Studies on Self–organizing, Collective, and Cooperative Phenomena in Natural and Artificial Computational Networks. A Special Issue of Physica D.*, Elsevier Science Associates. Vol. 42, Pages 188–201.
- Holland, J. H., Booker L. B., Colombetti, M., Dorigo, M., Goldberg, D. E., Forrest, S., Riolo, R. L., Smith, R. E. Lanzi, P. L., Stolzmann, W. and Wilson, S. W. (2000) What is a Learning Classifier System ? *Learning Classifier Systems. From Foundations to Applications*, Vol. 1813 of LNAI. Springer–Verlag, Berlin. Pages 3–32.
- Karni, A., Tanne, D., Rubenstine, B. S., Askenasy, J.J. And Sagi, D. (1994) Dependence on REM Sleep of Overnight Improvement of a Perceptual Skill. *Science* 265. Pages 679–682.
- Jouvet, M. (1994) *The Paradox of Sleep: The Story of Dreaming*. Translated by Laurence Garey (1999). The MIT Press, Cambridge MA. ISBN 0–262–10080–0.
- Lawton, G. (2003) To Sleep, Perchance to Dream, *New Scientist*, Vol. 178, No. 2401. Pages 34–39.
- Morrison, A. and Henley, K. (1969) Release of Organised Behaviour During Desynchronised Sleep in Cats with Pontine Lesion. *Psychophysiology* 6, 245.
- Press, W. H., Teukolsy, S. A., Vetterling, W. T. and Flannery, B. P. (1988) *Numerical Recipes in C: The Art of Scientific Computing*. ISBN 0–521–43108–5

- Riolo, R. L. (1991) Lookahead Planning and Latent Learning in a Classifier System. Proceedings of the First International Conference on Simulation of Adaptive Behavior. Cambridge, MA: The MIT Press. Pages 316–326.
- Roberts, G. (1993) Dynamic Planning for Classifier Systems. Proceedings of the 5th International Conference on Genetic Algorithms (ICGA93). Morgan Kaufmann. Pages 231–237.
- Rock, A. (2004) *The Mind at Night*, Basic Books, Cambridge MA. ISBN 0–7382–0755–1.
- Smith, C. and Lapp, L. (1986) Prolonged Increases in both PS and Number of REMs Following a Shuttle Avoidance Task, *Physiological Behavior*, Vol. 43. Pages 1053–1057.
- Smith, C. (1995) Sleep States and Memory Processes, *Behavioral Brain Research*, Vol. 69. Pages 137–145.
- Stickgold, R., Hobson, J. A., Fosse, M. and Fosse, M. (2001) Sleep, Learning and Dreams: Off–line Memory Processing, *Science*, Vol. 294. Pages 1052–1057.
- Stolzmann, W. (1998) Anticipatory Classifier Systems. *Genetic Programming 1998: Proceedings of the Third Annual Conference*, July 22–25, 1998, University of Wisconsin, Madison, Wisconsin. San Francisco, CA: Morgan Kaufmann. Pages 658–664.
- Stolzmann, W. (2000) An Introduction to Anticipatory Classifier Systems. *Learning Classifier Systems: From Foundations to Applications*, Vol. 1813 of LNAI. Springer–Verlag, Berlin, 2000. Pages 175–194.
- Stolzmann, W., Butz, M. V., Hoffmann, J. and Goldberg, D. E. (2000) First Cognitive Capabilities in the Anticipatory Classifier System, *From Animals to Animats 6*, Proceedings from the Sixth International Conference on Adaptive Behavior. Pages 285–296.
- Sutton, R. S. and Barto, A. G. (1998) *Reinforcement Learning*. The MIT Press, Cambridge, MA. ISBN 0–585–02445–6.
- Sutton, R. S. (1991) Dyna, An Integrated Architecture for Learning, Planning and Reacting. *SIGART Bulletin*, 2: ACM Press. Pages 160–163.
- Tolman, E. C. (1932) *Purposive Behavior in Animals and Men*. New York: Appleton.
- Wilson, M. A. and Louie, K. (2001) Temporally Structured Replay of Awake Hippocampal Ensemble Activity During Rapid Eye Movement Sleep, *Neuron*, Vol. 29. Pages 145–156.
- Wilson, M. A. and McNaughton, B. L. (1994) Reactivation of Hippocampal Ensemble Memories During Sleep, *Science* 265. Pages 676–679.
- Wilson, S. W. (1995) Classifier Fitness Based on Accuracy, *Evolutionary Computation*, Vol. 3, No. 2. Pages 149–177.
- Wilson, S. W. (2000) State of XCS Classifier System Research. *Learning Classifier Systems. From Foundations to Applications*, Vol. 1813 of LNAI. Springer–Verlag, Berlin, 2000. Pages 63–81.
- End.