Chapter 5

5. Experimental Design and Approach

The implementation of the SRS/E program may be considered to be in two separate parts. The first part encodes the behavioural and learning activities of the algorithm discussed in the previous chapter. The second part provides an emulation of an experimental environment that may be used to investigate the properties of the algorithm. This chapter considers the nature of this simulated "world" and describes some of the facilities available in the SRS/E program to assist the experimenter investigating the algorithm as implemented.

Communication between these two parts of the program is primarily via a single sub-routine call, "`DoWorldAction`()". This is a manifestation of the abstract activity described by the "`DoAction`()" construct of step six of the SRS/E algorithm (figure 4-13). The "`DoWorldAction`()" call takes two important parameters. The first parameter passes an action from the animat to the environment parts. The action takes the form of a `response_string`, an ASCII string extracted from an element of $\mathcal{R}$ representing the action to be taken by the animat on the current execution cycle. The second parameter returns a sequence of tokens representing sensory events detected by the animat following completion of the action supplied in the first parameter. Tokens are returned from the "environment part" to the "animat part" of the SRS/E program via an input buffer and recorded in the *Input Token List*, $\mathcal{I}$. Each token is defined as a sequence of characters from the ASCII set. Each token is separated from others in the input stream by a delineation character. Tokens have no embedded meaning to SRS/E, but the naming policy that has been adopted here is convenient for experimenter analysis of the generated trace and log files. Certain of the user interface utilities exploit this specific token format, and the adoption of an arbitrarily named but otherwise equivalent token would disrupt their operation.

5.1. Experimental Design

The experimental design used here follows that devised by Sutton (1990) to investigate the performance of the *Dyna-PI*, *Dyna-Q* and *Dyna-Q+* algorithms. Environments for his series of experiments took the form of simple grid mazes, through which the animat may progress from some starting point to some other finish or goal point. The animat may not leave the boundaries of the environment (there is no wrap around), and obstacles may be placed into the maze, making certain locations unattainable. Blockages may be added at any point throughout the experimental procedure to test the response of the animat to changing circumstances. In Sutton's design the animat may make one of four basic actions, each translating the animat into an adjacent cell in the grid. These actions, registered into the *Response List* $\mathcal{R}$ as "N", "S", "W" and "E", are equivalent to the actions "UP", "DOWN", "LEFT" and "RIGHT" defined by Sutton. Each action is assigned an *action cost* value of 1.0.

The simplest of these environments is shown in figure 5-1. It will be referred to as *DynaWorld/Standard* in the current work. Several other researchers have used this environment. Booker (1990) and Riolo (1990) have both described extensions to classifier systems tested with this environment. Peng and Williams (1992) describe extensions to the original Dyna framework. Littman (1994) investigated "memoryless" policies, where actions are based solely on current sensation - "traceless" in SRS/E terms. Each investigator is at liberty to adapt or create their own new or variant environment, and there are consequently a wide variety of designs in use.

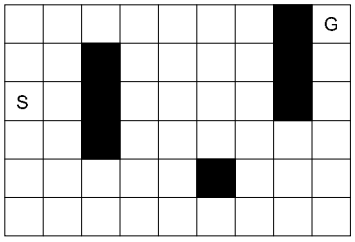

Figure 5-1: Sutton's DynaWorld/Standard Environment

DynaWorld represents a constrained and restricted experimental environment. While not appearing overly demanding as a learning task the maze environment follows in a long tradition of utilising highly controlled experimental environments. They have been espoused, in particular, by the behaviourist and instrumental conditioning schools of research. The latter group in particular place experimental animals in repeatable situations (as typified by the *Skinner box*) with the specific aim of investigating learning phenomena in isolation from other aspects of the subjects naturally occurring behavioural repertoire. The choice of a maze environment is also particularly resonant of the research methods employed by Tolman, of which a number of emulations follow in later sections. All the investigations performed here use simulated environments.

Several other pre-defined environments are available in the currently implemented SRS/E program. At the beginning of each experimental run the experimenter may select from a number of predefined maze patterns. Besides the DynaWorld/Standard environment Sutton (1990) defined an environment of the same size, but which is divided into two parts by a row of obstacles. This "Changing-World" environment is shown in figure 6-12. By selectively removing or adding blocks the behaviour of the animat may be investigated under several conditions where previously known paths disappear, and where new paths become available. A separate maze environment, not due to Sutton, is used in the latent and place learning experiments described later. This environment is shown in figure 6-22. It provides the animat with three distinct paths from the start to goal points, each of different length. The experimenter may, optionally, define environments of arbitrary size, and add or remove blocked cells as required. The experimenter may also elect to allow the animat to translate in all eight directions, "N", "NE", "E", "SE", "S", "SW", "W" and "NW". Diagonal actions attract an action cost of 1.414 (i.e., $\sqrt{2}$). An action that would cause the animat to leave the boundaries of the environment, or to enter a blocked cell, leaves the animat position unaltered, but incurs the cost associated with the action. Following Sutton's definition every cell in the maze is uniquely and reliably identified. In this implementation cells are identified by a single token of the form "XnYn", where "n" is substituted with the cell's X (or Y) co-ordinate. Cell co-ordinates are measured from (0,0), the bottom left hand corner.

Animals are notoriously variable in their performance, even in the most controlled of experimental conditions. The simulated environment holds a number of significant advantages over experimentation with live animals. First among these is the ability to maintain a high degree of control over the environment and the animat. Various aspects of the behavioural repertoire can be suppressed where they would interfere with the progress of the experiment. Motivation, in terms of goal setting, can be controlled independently of the underlying requirement (for instance hunger initiating food seeking activity) by manipulating the equivalent drive. The simulated animat also demonstrates a considerable degree of variability, arising from the nature of the randomising conditions used. Fortunately a ready supply of subject animats may be created to achieve a significant or reliable demonstration of highly variable performance phenomena at little or no cost and far less inconvenience than their naturally occurring counterparts.

The SRS/E program offers a repeatable experimental situation. From identical starting conditions the program will run identically over successive trials. Once any condition varies the course of an experiment will diverge. This facility may be used in several ways. First animats may be effectively "cloned", taken to some fixed point in the procedure, which is then modified according to the experimental schedule to investigate the specific effects of each variation. Second the procedures may be used in bulk, without constant monitoring and interesting instances identified from the logged record and replicated for further, more detailed, investigation. Finally the SRS/E program offers a high degree of visibility. The use of on-screen information display, in conjunction with the recorded logged information allows the experimenter access to a record of the internal processes that gave rise to specific behaviours and actions. The type and quantity of information displayed and recorded has been refined over a period of time to best reflect what is required for a full analysis. Examples exploiting these facilities occur throughout the next chapter.

5.2. The User Interface

With an environment defined and major parameters selected, the investigator may intervene during each experimental run to control the conditions required by the schedule. At the conclusion of each execution cycle the investigator may utilise the

command interface to make changes, to request diagnostic or analytical information, or continue the experiment. This interface is presented on a command line basis, and the options available are shown in the printout of figure 5-2. The interventions available to the investigator fall into five categories: (1) controlling execution cycles; (2) displaying and recording list information; (3) managing goals; (4) managing the animat and environment; and (5) accessing SRS/E program utilities.

```
Command: ?
        <enter>: run for one cycle
        <number>: run until cycle <number>
        @<number>: run for <number> cycles
        ! <x> <y>: break when animat reaches <x> <y>
        t: show Token List
        s: show Sign list
        s<token_id>: show signs using <token_id>
        h[<number>]: show Hypothesis <number> or List
        e[filename]: Export hypothesis List
        p: show current Prediction list
        g: show goal list
        g <sign_number>: set goal (G: save temp tally)
        g-<sign_number>: clear goal
        M: show policy Map (m: valence level map)
        w: show World (W[-]: temp tally [and clear]; WT: world tally)
        = <x> <y>: Move Animat to X,Y
        r: move animat to random starting location
        + <x> <y>: Set obstacle at X,Y
        - <x> <y>: Clear obstacle at X,Y
        u: update system settings
        ; (or *): record comment in trace file
        f: - not available (no trace file)
        #: Show partial path
        ?: this Help
        q: quit

Command:
```

Figure 5-2: The SRS/E Experimenter Command Options

5.2.1. Controlling Execution Cycles

Many experimental schedules to be described call for periods where the animat is free to roam the environment, alternating with goal directed activities. The investigator may single step ("<enter>") through the experiment, giving time to absorb the information about the previous cycle's activity, or may allow the experiment to run up to a specified cycle ("<number>"), or for a specific number of execution cycles ("@<number>"). Certain experimental regimes call for the animat to be allowed to locate the goal by random walk exploration, prior to detailed investigation. SRS/E allows the investigator to specify an interrupt condition ("!<x> <y>") which returns the program to manual control once the named location given by the co-ordinates "<x>" and "<y>" has been visited by the animat.

In the current program this is tied to the animat entering a specific named cell (defined by its co-ordinates). Future versions could, more generally, interrupt on the detection of a specific token, sign, or some combination of these types.

5.2.2. Displaying and Recording List Information

At any command cycle the investigator may display the contents of the Token List ("t"), the Sign List ("s"), the Hypothesis List ("h"), the Prediction List ("p"), or the Goal List ("g"). Additionally the investigator may inspect individual signs associated with a specific token ("s<token_id>"), and individual μ-hypotheses ("h<number>"). The complete Hypothesis List may be exported at any command cycle in a form suited to later importation to a standard spreadsheet utility ("e[filename]").

5.2.3. Managing Goals

In addition to viewing the goal list, the investigator may, at any command cycle, assert ("g<sign_number>") or clear ("g-<sign_number>") any goal. Goals may also be asserted from behaviours coded into the Behaviour List, and are automatically cleared when the goal is satisfied, or when extinguished by the extinction process. When asserting a goal the investigator is also prompted to supply a goal priority for that goal. Whenever a goal is asserted the temporary *world tally* is cleared. The world tally records the number of times each cell has been visited since last reset. The use of the command "G" in place of "g" to assert a goal leaves the tally unchanged to accumulate values.

5.2.4. Managing the Animat and Environment

The shape and size of the experimental environment is fixed at the start of each experiment, however the investigator may add ("+<X> <Y>") or remove ("-<X> <Y>") obstructions in the environment. The animat may be moved to a named location ("=<X> <Y>"), or moved at random to a new starting location ("r"). The animat may not be placed on a blocked location. When using the random relocation command the investigator must be careful not to create any unintentional enclosed pockets of cells into which the animat might become inappropriately trapped.

At any command cycle the investigator may display the “World Tally”, showing the number of visits to each cell either since the experiment started (“WT”), or the temporary tally (“W”), which records visits to each cell since the goal was last asserted. The temporary world tally is automatically initialised when a goal is asserted, or it may be explicitly initialised with the “W-” command. Figure 6-10 demonstrates the use of these commands to record the general movements of the animat between stages in a single experiment. The “w” command shows the shape of the environment, currently obstructed cells and the animat position to confirm the investigator has performed the required steps in the experimental schedule correctly.

A representation of the current Dynamic Policy Map may be obtained with the command “M”. An example of this data is shown in figure 6-7. Information about the μ-hypotheses with the best estimated cost path to the top-goal for each cell in the maze is mapped onto environment co-ordinates. There may be many μ-hypotheses associated with each of the cells that are not represented. The information presented in the first line of each cell shows the individual μ-hypothesis name and the valence level at which it appears in the DPM. The second line shows the response action associated with the μ-hypothesis. The third shows the estimated cost for the action according to the prevailing evaluation function. The last line in each cell the total estimated cost of the valenced path to the goal.

Each cell represents the “s1” component of the selected μ-hypothesis. Any cell that has not been visited (through, for instance, insufficient exploration), or which is blocked is shown blanked. The DPM displayed is that resulting from the most recent build. If no goal has yet been activated during the current experiment, no DPM is available and none can be shown. A short form display of the current DPM is also obtainable with the “m” command, which displays only the μ-hypothesis identity and valence level.

5.2.5. Accessing Utilities

The investigator may change the values of the important system settings at any point during an experiment using the “u” command. Comments may be recorded to the trace file (“*” or “;”). The trace file may be temporarily suspended, and

subsequently reactivated if required ("f"). An experiment is concluded with the quit ("q") command.

5.3. The System Execution Trace Log

Each time the SRS/E program is run the experimenter has the option to create a record of all significant activities that occur during that run (the *log file*), which may be inspected and analysed in detail after the experiment is concluded. The log file records the following information. (1) The creation and modification of all μ-hypotheses. (2) All predictions made, at the time of their corroboration. The resultant `cpos`, `bpos`, `recency` and other significant values for the predicting μ-hypothesis are recorded. (3) Periodic summaries of numbers of μ-hypotheses created and modified. (4) A copy of the *valenced path* (as figure 4-1) each time the DPM is recomputed. Trial and error actions are not recorded, but valenced (and unvalenced) actions are. (5) The experimenter may request at any time a log record of the complete (or selected elements of) the Token, Sign, Hypothesis, Prediction or Goal Lists. (6) The system automatically logs important activities, such as goal activations, satisfactions and extinctions, changes in goal priority, and actions by the experimenter, that change the environment or animat. The user may also write "freeform" textual comments to the log at any point.

At the conclusion of the experiment the complete final Sign and Hypothesis Lists are logged. Log files are automatically "watermarked" with the start and finish times of the experiment. The current SRS/E program has been augmented with several routines to display information about the DPM in a manner that relates the internal representations to the layout of the simulated environment. Where such representations are recorded in the log they are specific to the simulated environments, not a general feature of the SRS/E system. They will be introduced as appropriate when experimental run results are considered.

5.3.1. Processing Log Results

A utility program, *filter.exe*, has been prepared to extract relevant information from SRS/E log files to facilitate their analysis. Log files (as they are in human readable form) can grow to an unwieldy size. "filter.exe" contains options to

prepare a more concise format for review, as well as to extract data in a form suited to graphing and tabulating utilities.

5.4. Important Schedule Variables

At the start of each experimental run the investigator is able to define a number of parameters in addition to specifying the form of the environment. The three most significant of these variables are: (1) the *action repetition rate*, abbreviated to Arep; (2) the *action dispersion probability*, abbreviated to Adisp; and (3) the *learning probability rate* (Lprob, λ). At the start of each experimental run the investigator will also be required to select a seed (*rseed*) for the pseudo-random number generator[26] used. The selection of the same seed allows an experimental run to be replicated while all other conditions remain equal.

5.4.1. Action Repetition Rate (Arep)

Many of the experiments to be described call for actions to be selected at random during an initial period of *trial and error* exploratory behaviour. Sutton adopts the term *random walk* to describe this activity. A true random walk can lead to the animat doubling back on itself to such an extent that exploration of a maze of any size may take an excessive number of execution cycles, with specific areas becoming "over explored". Some of the experiments to be described require that the animat has the opportunity to partially explore most of the environment. The action repetition rate parameter increases the probability that the animat will select a new action at each cycle. With Arep set to 0.0 a new action is selected every cycle, with Arep set to 1.0 the system would always use the same action. An Arep value of 0.5 indicates that the same action as the previous one will be selected with a probability of 0.5 and a new one with a probability of 0.5. Higher values of the action repetition rate increase the tendency for longer sequences of the same action.

[26]The random number generator ("rand()") supplied with the compiler has been used.

Figure 5-3 summarises the effect of the action repetition rate on random walk length over 2,500 trials for each of four settings, where the animat must traverse the maze (of figure 6-1) from start to goal in each trial. The figure shows the number of individual trials (a single traversal) falling into "buckets" of 100 steps. The minimum possible path length is 14 steps. The distribution is skewed, but it may be seen that an Arep value of 0.0 (new action always) leads to a higher average path length (841.9 steps), and a considerable number of instances where the path length reaches a large value than when higher values of Arep are selected. The average path length for Arep = 0.25 was 589.1, for Arep = 0.5 was 419.5 and for Arep = 0.75 was 343.4. The minimum random walk length achieved in the 10,000 trials was 19 (Arep = 0.75). Any advantages gained by increased exploration rates are somewhat offset with higher values of Arep by a tendency for the animat to become trapped at edges and corners, an effect that has detrimental effects in some experimental situations.

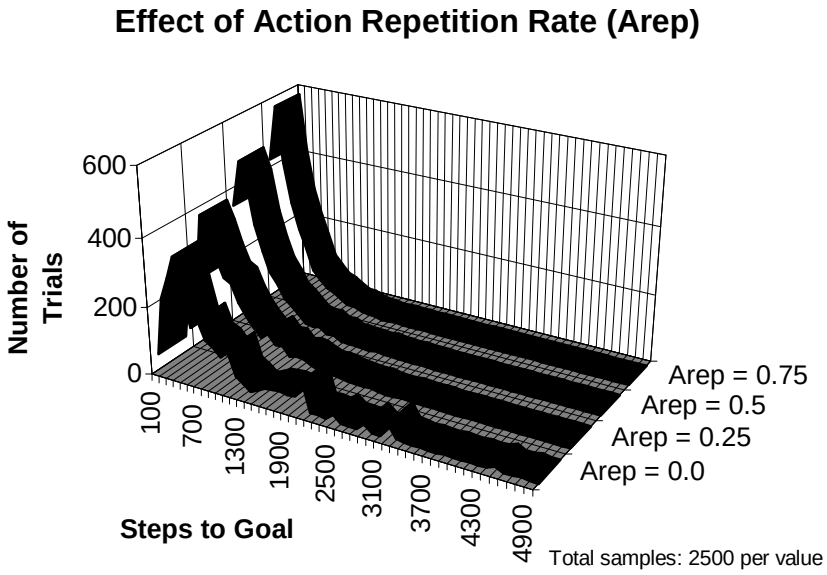

Figure 5-3: Effect of Arep on Random-Walk Path Length

5.4.2. Action Dispersion Probability (Adisp)

Sutton defines a class of noise for the Dyna environments in which actions made by the animat are translated into another action (at the interface between animat and environment) with a given probability p. Actions are translated into either the

action one segment clockwise or the one segment anti-clockwise. So, for example, "UP" will be converted to "LEFT" with a frequency defined by (1-p)/2, or to "RIGHT" with a frequency defined by (1-p)/2, or left unchanged with a frequency defined by (p), similarly for the other actions available. The probability with which this translation occurs is controlled in the SRS/E program by the Adisp parameter. When Adisp is set to 1.0 all actions are unmodified. With Adisp set to 0.5, 50% of actions would be unmodified, 25% converted to the action viewed clockwise, 25% to the action viewed anti-clockwise. The source and destination states are still recognised correctly in Sutton's definition. Other forms of "noise" might also be defined.

5.4.3. Learning Probability Rate (Lprob)

This schedule variable equates directly with the *learning probability rate* (Lprob, λ) described previously. The implementation and properties of the learning probability rate were described in chapter four (section 4.12.8).

5.5. Fixed Schedule Experiments

Several of the experimental procedures to be described call for an intricate or highly repetitive sequence of steps to be performed so as to appropriately demonstrate the properties of the system. Where this is the case the SRS/E suite incorporates program code to set up each trial within the overall experiment and to record the results obtained for subsequent analysis. Typically, the investigator will be required to establish basic parameters for the experiment, but will not be required to directly monitor or intervene in its progress.

Three such *fixed schedule experiments* have been used in obtaining the results described later. The first schedule sets the animat to a defined starting position and counts the number of steps (execution cycles) required for the animat to reach a defined goal location. Having reached the goal the animat is returned to the start location and the run restarted. This may be repeated as many times as required. This fixed schedule is used to provide the comparative results relative to Sutton's Dyna algorithms (section 6.2, next chapter), and to investigate the effects of noise (section 6.3). These procedures were used to determine the results presented in

figure 6-3, and to generate control data for a range of subsequent experiments. The second fixed experimental schedule automates the path-blocking experiments, presenting results in the style of a cumulative reinforcement curve to allow easy comparison with Sutton's results (section 6.5.7). The third fixed schedule automates the latent learning task (section 6.6), and accumulates results such that they may be presented in a manner facilitating comparison with those of Tolman and Honzik (1930). As every fixed schedule experiment starts from known parameters, it is possible to replicate any particular experimental trial up to a known point before transferring to manual control. In this way a particular outcome may be investigated in greater detail or pursued for additional steps if required. The full trace file may be disabled during the fixed schedule phase to avoid recording unnecessary detail and subsequently re-enabled during the manual phase to monitor results in detail.

The next chapter describes and discusses a number of experiments performed with the SRS/E program.