Chapter 4

4. The SRS/E Algorithm

This chapter describes the *SRS/E* computer algorithm. SRS/E is derived directly from the *Dynamic Expectancy Model* postulates of learning and behaviour developed in the previous chapter. SRS/E follows in the tradition established by Becker's JCM, Mott's ALP and Drescher's systems by providing an intermediate level cognitive model based on the context-action-outcome triplet. As with these previous systems, SRS/E offers a sensory-motor view of learning. It is not, however, to be considered as a re-implementation of any of these existing systems. As with Mott's ALP and Drescher's algorithm, and indeed the majority of extant animat control algorithms, SRS/E is based on a repeating cycle of sensory acquisition from the environment, processing and taking overt actions into the environment.

Each model is a reflection of the times in which it was created. Becker's JCM proposal and Mott's ALP implementation adopt an associative net structure for schemata LTM; consistent with prevailing theories from psychology and cognitive science, for example, Norman (1969). Adopting a net structure served to contain the computational search and matching load inherent in these designs, bringing distinct practical advantages to Mott's implementation in the context of a time-sharing ICL mainframe. Drescher's later (1991) system adopted a "neural crossbar architecture", consistent with the revival of interest in connectionist thinking at that time. Availability of the massively parallel *Connection Machine* made the brute force approach of the marginal attribution algorithm feasible. In turn, SRS/E arises as a reaction to an upsurge of interest in reinforcement learning and related behaviourist concepts. SRS/E's name, an abbreviation of Stimulus-Response-Stimulus/Expectancy, pays passing tribute to the life's work of E.C. Tolman, and defines the positioning of the work. Various other items of terminology, notably the use of *Sign*, *Valence*, *Hypothesis* and (*Cognitive*) *Map*, are derived from the vocabulary developed by Tolman and his contemporaries.

In contrast to these other systems SRS/E is primarily an algorithm that manipulates lists of data. This chapter is divided into two main parts. In the first part the various types of data list are described. The second part presents the algorithm used to manipulate the lists, perform the learning tasks and generate overt behaviours, either from the animat's predefined ethogram, or as a consequence of learned information.

4.1. Encoding the Ethogram: SRS/E List Structures

SRS/E currently defines seven internal data structures. These data structures will be referred to as *lists*. Each list encapsulates an aspect of the animat's ethogram, and so record the instantaneous "state" of the animat. At defined points in its execution cycle the SRS/E algorithm will inspect the contents of these lists and generate behaviours based on the prevailing contents of those lists. Equally the SRS/E algorithm will add, modify or delete information stored on the lists by processes derived from the Dynamic Expectancy Model postulates described in chapter three. These processes will be defined later in this chapter. Each of the seven lists is composed of *list elements*. In turn each element of each list is itself composed of *list element values*, which record items of information relevant to each list element. So, for example, the Hypothesis List is composed of many individual μ-hypotheses, the elements of that list. Each μ-hypothesis has attached to it various hypothesis values, which are created and initialised at the same time as the individual μ-hypothesis, and may be updated each time the algorithm utilises the individual μ-hypothesis. All list element values (or "values") are updated by the SRS/E algorithm as a result of events impinging on the animat and actions the animat makes. The list structures, list elements and list element values are summarised in table 4-1, and described in the sub-sections that follow. List elements may be defined by the originator before the creation of an individual animat, as would be the case with the Response and Behaviour Lists. Otherwise, as would be typical for all the other lists, lists are empty at the point the animat becomes a free standing individual. In which case the SRS/E algorithm creates individual list element entries as the need arises.

4.1.1. List Notation

Throughout this chapter each of the seven lists will be represented by a single calligraphic character. Upper-case characters represent complete lists ($\mathcal{I}$, $\mathcal{S}$, $\mathcal{R}$, $\mathcal{B}$, $\mathcal{H}$, $\mathcal{P}$ and $\mathcal{G}$). Lower case characters represent individual elements in the respective list (i, s, r, b, h, p and g). Table 4-1 summarises this notation. A superscript notation will be adopted to indicate some property of a list or a list element. In particular the use of an asterisk will indicate "active" elements, those whose attributes match the prevailing circumstances on the current execution cycle. For instance $\mathcal{I}^*$ will refer to all those elements of $\mathcal{I}$ where the corresponding token has been detected in the sensory buffers $\mathcal{I}^* \subseteq \mathcal{I}$, therefore $\mathcal{I} - \mathcal{I}^*$ will refer to all those elements of $\mathcal{I}$ where no corresponding input token has been detected. A number of additional superscripted forms will be introduced later; each will indicate some sub-set of a list, or specify some attribute of a list element. A notation in which the list element value name is used to refer to or access a list element or sub-list will also be employed.

As with JCM, ALP and Drescher's system every element in each SRS/E list has attached to it a number of numeric and other values. These values are updated as the algorithm executes and are in turn used by the algorithm in selecting overt behaviours and to guide the learning process. SRS/E is intended primarily as a platform for experimentation. List element values are therefore variously available for use in the algorithm as presented, and by reporting and analysis software created with the specific purpose of analysing and presenting experimental results. The list element values used by SRS/E are shown in table 4-1. Their functions and purposes are described following a detailed description of each list type. Such values will be shown in a different font "`thus`". List element value names shown in this different font are chosen to directly reflect the variable names employed in the current implementation of SRS/E used to conduct the experiments described in chapter six. The character in brackets associated with each value shown in table 4-1 indicates the data type selected for that value in the current implementation. A calligraphic character, "($\mathcal{S}$)" for example, indicates a pointer or reference to a list element of the indicated type; "(i)" indicates an integer type; "(t)" a "time" value, and "(b)" a bit-sequence. The types "(i)", "(t)" and "(b)" are all encoded conveniently as (long) integers. Time values are recorded as discrete intervals

corresponding to execution cycles of the algorithm. ASCII encoded strings are indicated "(s)", real or floating point values as "(f)". The range of some floating point values will be restricted within the program.

List Symbol	List Description	List Element Symbol	List Element Values
$\mathcal{I}$	**Input Token List**. Binary, atomic input items from sensors. Associates input items to arbitrary internal symbols	i	`token_string` (s) `token_identifier` (i) `token_first_seen` (t) `token_last_seen` (t) `token_count` (i), `token_prob` (f) `token_activation_trace` (b)
$\mathcal{S}$	**Sign List**. Descriptions of an environmental "state", defined by a conjunction of tokens (i) and other internal symbols	s	`sign_conjunction` (see text) `sign_identifier` (i) `sign_first_seen` (t) `sign_last_seen` (t) `sign_count` (i), `sign_prob` (f) `sign_activation_trace` (b) `best_valence_level` (i)
$\mathcal{R}$	**Response List**. All available actions (simple and compound)	r	`response_string` (s) `response_identifier` (i) `response_cost` (f) `response_activation_trace` (b)
$\mathcal{B}$	**Behaviour List**. (condition,action) defined innate behaviour patterns (condition $\in \mathcal{S} \rightarrow$ action $\in \mathcal{R}$).	b	`condition` (s) `action` (r) `behaviour_priority` (f)
$\mathcal{G}$	**Goal List**. Actual or potential system goals, prioritised by $\mathcal{B}$.	g	`goal_sign` (s) `goal_priority` (f) `time_goal_set` (t)
$\mathcal{H}$	**Hypothesis List**. List of μ-hypotheses in the form (s1,r1,s2) s1 $\in \mathcal{S}$, r1 $\in \mathcal{R}$, s2 $\in \mathcal{S}$.	h	`s1` (s), `r1` (r), `s2` (s) `time_shift` (t) `hypo_identifier` (i) `hypo_first_seen` (t) `hypo_last_seen` (t) `hypo_activation_trace` (b) `recency` (i), `hypo_bpos` (f) `hypo_cpos` (f), `hypo_cneg` (f) `hypo_age` (t), `hypo_maturity` (i) `hypo_creator` (h) `valence_level` (i) `cost_estimate` (f) `policy_value` (f)
$\mathcal{P}$	**Prediction List**. List of predictions awaiting confirmation.	p	`predicting_hypo`(h) `predicted_sign`(s) `predicted_time` (t)

Table 4-1: SRS/E Internal Data Structures

4.1.2. Summary of Lists

The **Input Token List** records binary atomic input items from system sensors and assigns each one a unique, but arbitrary, internal symbol such that each subsequent appearance of the same input item will generate the same internal symbol. The Input Token List implements the "token" of definition T0.

The **Sign List** provides the system with partial or complete descriptions of the environmental "state". A sign is defined as a conjunction of input tokens and other internally generated symbols, and their negations, providing the structure to implement the sign of definition S0.

The **Response List** defines the set of all the actions available to the animat, to implement the action of definition A0. Simple actions are defined by the ethogram. Compound actions (postulate A3) may be formed by the concatenation of simple actions.

The **Behaviour List** explicitly defines the innate behaviour patterns for the animat as an integral part of the ethogram (definition B0). Fixed, pre-programmed, behaviour patterns (postulate B2) may subsequently be subsumed by learned, goal-seeking behaviour. For simple animat ethogram definitions the Behaviour List will also be responsible for setting goals (postulate B3) and so balancing the priorities between fixed and learned behaviour.

The **Goal List** records none, one or more possible goals being sought by the animat at any particular time (definition G0). The animat only pursues one goal at any one time, the *top-goal*.

The **Hypothesis List** records learned expectancies (μ-hypotheses) in the form "s1,r1,s2". Context "s1" and consequence "s2" are elements from the Sign List. Action "r1" is an element from the Response List. Each element of the Hypothesis List equates directly to a single μ-hypothesis, a small, isolatable fragment of knowledge about the animat's existence, well defined in terms of the other list types (definition H0). To be of value to the system each μ-hypothesis must make a clear and verifiable prediction. Corroborated μ-hypotheses are subsequently used

by the animat to generate useful goal-seeking behaviours. The SRS/E algorithm provides the algorithmic resources to create, verify, modify, delete and use μ-hypotheses.

The **Prediction List** records expectations made by activated μ-hypotheses for confirmation or denial at a defined time. This structure retains time tagged predictions until they are verified (postulate H1).

4.2. Tokens and the Input Token List

SRS/E employs a grounded symbol approach to behaviour and learning and has much in common with the notion of *deictic representation*[21] (Agre and Chapman, 1987; Chapman, 1989; Whitehead and Ballard, 1991). *Deictic markers* point to aspects of the perceivable environment. Ideally each marker will point to only one object or event, or to one well-defined class of objects or events, in the environment. This allows the animat to respond appropriately to the presence of the object or occurrence of the event, or to learn the significance of the object or event with minimal ambiguity (the FDMSSE assumption).

Typically input tokens either directly reflect the value of some sensor, or are derived from sensor values to define a partially or wholly complete state descriptor. Thus SRS/E will equally accept ALP style kernels, such as "<LOW>S" or "<BRIGHT>S", derived directly from the transducer values from the robot, or Drescher's (1991, p117) *primitive items* "hp11", "vp11", or "fovf00-33" denoting partial state descriptors from the simulated environment. As with Mott and Drescher, SRS/E input tokens are binary in nature, present or absent. SRS/E does not employ the predicate and value representation described by Becker.

The SRS/E algorithm accepts sequences of tokens from the environment. During each execution cycle none, one or many tokens may be presented to the algorithm from a sensor sub-system integral with the animat. The first appearance of any token is registered into the Input Token List, $\mathcal{I}$, and the new token is assigned a unique internal code. This realises the *tokenisation* process, described in postulate

[21](OED) deictic: a & n, Pointing, demonstrative, [Gk: deiktikos]

T1. For every subsequent appearance of that token the unique code will be generated from the list. At each execution cycle the Input Token List I will be partitioned into those tokens that have appeared in the input stream on the current cycle and hence are active, and all the others that have not appeared and are not active. As indicated in section 4.1.1 the active partition is denoted I^*.

Tokens may be registered into I by the originator as part of the initial ethogram definition and subsequently employed in generation of innate behaviour patterns. Apart from this, tokens have no inherent "meaning" to the system. Once registered into the Input Token List, token identities are permanently retained. SRS/E will accept new additional tokens at any point in the lifecycle of the animat. The appearance of novel tokens also drives the learning process. There is no generalisation over input tokens; non-identical input token strings are treated as wholly distinct.

The Input Token List is implemented as a *hash table* (Knuth, 1973), the internally generated token symbol value being set equal to the index position in the hash table. Initially the hash table is given a fixed size, but is grown automatically and the symbols re-hashed when the table is close to overflow. As part of this process all internal token symbol values are updated to reflect their new position in the table.

4.2.1. Input Token List Values

In addition to the `token_identifier`, the internal symbol, and the external representation of the token string `token_string`, the Input Token List maintains four additional numeric values for each Input Token List element. As an aid to the analysis of experimental data the input `token_string` is retained in the Input Token List and is shown in preference to the anonymous internal symbol in output trace and log files. The list element value `token_first_seen` records which execution cycle the token ι was first detected. The value `token_last_seen` records the execution cycle when the token was most recently detected. The value `token_count` records the total number of cycles that the token ι has occurred on I^*. The raw probability of occurrence (`token_prob`) for any token may be derived according to the equation:

$$\texttt{token\_prob} \leftarrow \frac{\texttt{token\_count}}{\texttt{now - token\_first\_seen + 1}} \qquad \text{(eqn. 4-1)}$$

This raw token probability may be used as a measure to determine the degree to which the sensory sub-system is able to differentiate the phenomena indicated by the token from others. Generally, tokens with a relatively low raw probability measure facilitate the behavioural and learning process.

A record of recent past activations for each element ι is maintained in the variable `token_activation_trace` according to the assignments:

$$\texttt{token\_activation\_trace}^{t\text{-n-1}} \leftarrow \texttt{token\_activation\_trace}^{t\text{-n}} \qquad \text{(eqn. 4-2a)}$$

$$\texttt{token\_activation\_trace}^{\text{now}} \leftarrow \mathcal{I}^{\,\text{activation\_state}} \qquad \text{(eqn. 4-2b)}$$

These trace values, and those for other list element types, are used in sign definitions to record past activations and provide a mechanism to implement *temporal discrimination*, an aspect of the μ-hypothesis differentiation process (postulate H6). The activation traces are of finite length, newer values entering the trace displace older values which are lost to the algorithm.

In the current implementation of SRS/E, n of equation 4-2a takes the values 1 to 32. The token activation trace is therefore conveniently represented as individual bit positions in a long integer. The operation described by equation 4-2a is achieved in the current SRS/E implementation as an arithmetic shift left by one bit position. The operation described by equation 4-2b by setting (or clearing) the lowest order bit of the integer recording the trace values according to the current activation value of the token.

4.3. Signs and the Sign List

Signs encapsulate one or more tokens into a single item (this is derived from postulate S1). They are identified within the system by a unique symbolic identifier.

The total Sign List is designated as S. The subset of signs that are active at the current time are designated as S^*. Sign activation was described by postulate S2.

4.3.1. Representing Signs

As with the schema representations of Mott and Drescher, SRS/E signs are a conjunction of primitive tokens, where the token must be present for the conjunction to be active, or negated tokens, where the token must not be present for the conjunction to be active. Drescher's representation is severely restricted with respect to Mott's in that the schema left hand side in ALP allowed inclusion of kernels from any position in *Short Term Memory* (STM), whereas Drescher's did not. Mott's use of the little arrow notation, with its strict time sequence information, imparts further contextual information to the schema left hand side. SRS/E also adopts an explicit time representation to tokens, so:

ALP: [<BRIGHT>S → <FRONT>S ¬ <CHARGE>S ....]

becomes:

SRS/E: (bright$^{@t\text{-}1}$ & front & ~charge ....)

In SRS/E all timings are considered to be relative to the current cycle (*t*=0 or, equivalently, *t*=now), negative from the past, positive into the future. Thus the notation "@*t*-1" is conveniently read as "at the current time minus one", or "on the cycle before the current one". *Token negation* is represented by the tilde character ("~"). The representation of past events in ALP is limited to the length of STM (typically six cycles), in SRS/E by the length of the activation trace (typically 32 cycles). Unlike Becker, but like SRS/E, Mott did not permit recycling of kernels from the end of STM into the input register as essential timing information is lost. Drescher offered no equivalent to a Short Term Memory in his system.

By convention an input token incorporated into a sign will be automatically dereferenced to its external form from the internally represented symbolic form whenever it is displayed or printed. Sign conjunctions may also incorporate other symbolic information contained within the SRS/E system. So a sign conjunction

may include the symbolic name for another sign (from $\mathcal{S}$). Similarly actions (from $\mathcal{R}$) may be included. Thus past actions by the system are available for inclusion into the "s1" conjunction. μ-Hypothesis activation (from $\mathcal{H}$) may also be recorded in a sign, by including the symbolic name of the hypothesis (to be described in a later section). The inclusion of the hypothesis form into the sign conjunction may give the system limited access to its own operation and hence the possibility of predicting, seeking as a goal, and creating hypotheses about aspects of its own learning behaviour. The ramifications of this ability are beyond the experimental investigations of SRS/E presented here. This construct is broadly equivalent to Mott's proposal for an *internal kernel* and Drescher's notion of a *synthetic item*, but more concise and manageable than the latter as only the symbolic name is required. SRS/E does not, however, at present have any explicit support for the notion of *object permanence*.

The `sign_conjunction` may be more concisely defined as:

$$\mathcal{S} \in \mathcal{S}^* \text{ iff } conjunction_{n=1}^{k}(\chi_n^s) \qquad \text{(eqn. 4-3)}$$

where *k* gives the number of terms in the conjunction. Each of the items χ_n^s may substitute for one of four forms:

$$\chi_n^s \equiv \varkappa \in \mathcal{X}^* \qquad \text{form 1}$$

or

$$\sim\chi_n^s \equiv \varkappa \notin \mathcal{X}^* \qquad \text{form 2}$$

or

$$\chi_n^{s@-t} \equiv \varkappa \in \mathcal{X}^{*@-t} \qquad \text{form 3}$$

or

$$\sim\chi_n^{s@-t} \equiv \varkappa \notin \mathcal{X}^{*@-t} \qquad \text{form 4}$$

allowing for the presence of symbol of type $\varkappa$ (form 1), the absence of symbol of type $\varkappa$ (form 2), the recorded presence of symbol of type $\varkappa$ at time (now-*t*) in the

past (form 3) and the recorded absence of symbol of type $\aleph$ in the past (form 4). In these forms the symbol $\aleph$ (and hence $\mathcal{X}$) may substitute for elements from any of the lists $\mathcal{I}$, $\mathcal{S}$, $\mathcal{R}$ or $\mathcal{H}$.

The sign definition adopted in SRS/E has no *don't care* ("#") representation of the form employed in classifier systems. If a symbol is not explicitly included its condition is taken as irrelevant. This is generally consistent with Popper's view that an 'experiment' should define all its relevant preconditions, but exclude all those inconsequential to its outcome. This representation is not as concise where small, bounded sets of features are to be considered, but offers significant advantages where small subsets of a very large feature set are to be represented and where past values of features are to be included. Many other representational schemes have been proposed to enable machine learning systems to represent left hand side preconditions completely or conveniently. In particular, Michalski (1980) describes a condition form for the VL_{21} logic system that includes enumeration, variabilisation and hierarchical descriptions; but not past events.

In the SRS/E implementation the Sign List is held as an indexed list of sign elements. The index is used to create the sign identifier (thus: "Snnnn", where nnnn is the index number). This designation for a sign symbol appears in the log and analysis information from the experimental runs of SRS/E. Individual conjuncts in a `sign_conjunction` definition are recorded as a triple: conjunct identifier, a negation flag, and time offset. In the current implementation they are recorded in a canonical form for efficient access. Also in the current implementation negation is indicated by recording the conjunct identifier (for instance `token_identifier`) with a negative value. Attempts to create a new sign that duplicates an existing sign are rejected by SRS/E.

4.3.2. Other Sign List Values

Each element of the Sign List is assigned a unique `sign_identifier`, as described, and each sign has associated with it `sign_first_seen`, `sign_last_seen`, `sign_count` and `sign_activation_trace` values. The derivation and use of each of these mirrors the derivation and use described for the

equivalent Input Token values. Sign probability, `sign_prob`, is calculated in an analogous manner to `token_prob`:

$$\texttt{sign\_prob} \leftarrow \frac{\texttt{sign\_count}}{\texttt{now - sign\_first\_seen + 1}} \qquad \text{(eqn. 4-4)}$$

An additional measure, `raw_sign_prob`, may be derived from the individual probability (*p*) values of the component parts of the sign conjunction:

$$\texttt{raw\_sign\_prob} \leftarrow \prod_{n=1}^{k} (p(\chi_n^s)) \qquad \text{(eqn. 4-5)}$$

Where `sign_prob` >> `raw_sign_prob` the SRS/E algorithm may use this as an indication that the sign conjunction is a significant combination of component parts, and not just a combination of random or "occult" occurrences.

4.4. Actions and the Response List

The Response List, $\mathcal{R}$, records the basic actions available to the animat. For any SRS/E controlled animat, the originator "registers" a list of basic actions and their associated costs as part of the initial ethogram definition. Actions will be required to serve the needs of both the innate behavioural and the learning components of the SRS/E system, though the same actions may well be adequate for both purposes. In SRS/E the actions defined in $\mathcal{R}$ serve as instructions or commands to the actuation sub-system, whether physical or simulated. Selection and description of the actions in $\mathcal{R}$ are an integral part of any experimental run discussed in chapter six. SRS/E supports both simple (molecular) and compound (molar) actions. A compound action is one built from the concatenation of two or more simple actions, as described by postulate A3. Compound actions run to completion once initiated. This definition of compound action is therefore distinct from Drescher's definition of a *composite action*, which may be seen as an intermediate stage between the SRS/E compound action and the Dynamic Policy Map.

In the current implementation each action is held as an element in the indexed list $\mathcal{R}$. Individual actions are registered into the list before the start of each

experimental run. Additional entries may be registered into the list at any time, to implement a *maturation* strategy, for instance. On each execution cycle SRS/E will select a single action from $\mathcal{R}$ to be reified (derived from postulate A1) and delivered to the actuator sub-system. The reified action is placed on the $\mathcal{R}^*$ list for the cycle in which it is active. Output actions take the form of an ASCII string (entered at the time of registration) to be interpreted by the actuator system as an instruction to perform some defined activity. Trace and log information arising from the use of SRS/E will automatically dereference the action index to this string for ease and clarity of analysis, as with Input Token List entries.

4.4.1. Response List Values

In addition to the anonymous internal symbolic value, `response_identifier` and the external string representation of the action, `response_string` stored with each action in $\mathcal{R}$, the SRS/E algorithm records `response_cost`, an estimate of the effort that will be expended whenever that action is taken (the action cost, from postulate A2). This is the estimate provided by the originator at the time the action is registered. It may reflect the energy required to perform the action, a notional amount of resource depleted by the action, or the time taken to complete the simple or compound action, or some combination of these and other attributes. This is broadly in keeping with Tolman's (Tolman, 1932, Ch. 7) observations that rats generally choose paths through experimental mazes that minimise delay or effort.

On a practical note this value also provides the Dynamic Policy Map generation algorithm a metric by which to evaluate the appropriateness of alternative paths through the map. The originator is required to specify `response_cost` values of unity or greater, and that these values be proportioned according to the relative effort across all actions in $\mathcal{R}$. The `response_activation_trace` maintains a transient record of past actions (a record of $\mathcal{R}^*$), computed as for `token_activation_trace` and `sign_activation_trace`.

4.5. Innate Activity and the Behaviour List

The *Behaviour List* $\mathcal{B}$ defines the innate behaviours for the animat. This definition is an essential part of the ethogram, and built into the animat at the time of its definition by the originator. Such behaviours will react to situations, events, and changes in the environment as prescribed by the originator. In the main these activities will be mediated and modified by internally generated and detected needs, drives or motivations differentially selecting or inhibiting aspects of *innate behaviour* patterns. Innate behaviours need not be fixed over the life-cycle of the animat and may vary according to a *maturation* schedule or *imprinting* regime. This section does not intend to revisit the mechanisms by which behaviours are formed and selected, nor to further consider the arguments over which of the many proposed strategies most effectively or closely model observed natural behaviours. It will, however, be primarily concerned with how the overt behaviour of the animat will be apportioned between the innate and learned parts of the mechanism.

4.5.1. Behaviour List Structure and Selection

The Behaviour List is a notional list of condition-action pairs (`condition` $\in \mathcal{S} \rightarrow$ `action` $\in \mathcal{R}$), fully in the tradition of the stimulus-response behaviourist camp. At each execution cycle every element b of $\mathcal{B}$ is evaluated against $\mathcal{S}^*$, and a list of applicable candidate actions, $\mathcal{B}^*$, formulated. The selection of behaviours on each cycle is thus made based on the evidence for their applicability. To achieve the required balance of innate and learned behaviours the Behaviour List will be considered to be in two parts. The first part, $\mathcal{B}^r$, lists condition-action pairs from which action candidates will be selected ($\mathcal{B}^{r*}$). This part of the list realises the *primary behaviours* of postulate B2. The second part, $\mathcal{B}^g$, lists condition-action pairs determining which, if any, goals the animat should pursue given the prevailing circumstances. This second part of the list realises the *goal setting behaviours* of postulate B3. During each execution cycle several possible actions, and several goals could be applicable. SRS/E makes its selection from $\mathcal{B}^{r*}$ and $\mathcal{B}^{g*}$ on a priority basis.

Each potential innate behaviour in the animat is assigned a priority by the originator, which is initially set within the ethogram according to its significance.

Thus in an animal simulation, predator avoidance might be assigned a high priority, and therefore be made manifest whenever the conditions that indicate the approach or presence of a predator. Other behaviours, those initiated by, say, the onset of hunger (detected, perhaps, by lowered "blood sugar levels") having a lower overall priority and so being interrupted by the avoidance behaviour. SRS/E must also adjudicate between innate and goal seeking behaviours, those derived from the Dynamic Policy Map. To achieve this, elements of $\mathcal{B}^g$ (and so $\mathcal{B}^{g*}$) are also assigned a priority in the ethogram. At each cycle SRS/E will either select the highest priority element from $\mathcal{B}^{r*}$, if this priority is higher than that for the highest priority element from $\mathcal{B}^{g*}$. Otherwise a Dynamic Policy Map will be created, or the existing one used, to generate a behaviour from stored μ-hypotheses.

Where none of the defined innate behaviours has an effective priority, it is inappropriate for the animat to pursue any of those behaviours. So, if it is not threatened, hungry, thirsty, tired or dirty, etc., then there is little to be gained by fleeing, eating, drinking, sleeping or preening, etc., just because one of these behaviours is slightly less irrelevant than the others. Therefore the SRS/E algorithm places a lower bound, the *basal level threshold* (ε), on behaviour activation, below which none of the behaviours defined in $\mathcal{B}$ will be selected. Yet the animat is expected to perform some activity on each cycle. Where no innate behaviour or goal behaviour is active the animat performs exploratory actions selected from $\mathcal{R}$. These implement the third, and mandatory class of innate behaviour pattern, the Default (exploratory) Behaviours (realising postulate B4). The learning mechanism is still actively monitoring the actions taken and their outcomes and learning continues during these periods of apparently undirected activity.

The Behaviour List as defined for the present version of SRS/E places restrictions on what may be effectively represented by the originator. It is adequate to generate the reflexive behaviours described for ALP. Any scheme by which behaviours are controlled through the presence of only binary releasers provides little useful analogue with the natural world, and gives rise to a range of difficulties in providing a useful simulation of innate behaviour. The default exploratory ("trial and error") behaviour is present in SRS/E as an inherent component of the system and requires no additional intervention by the originator. For the purposes of the

experimental regimes to be described in chapter six the experimenter is able to activate goals externally.

4.5.2. Behaviour List Values

In addition to the `condition` and `action` values, each element of $\mathcal{B}$ has associated with it the value `behaviour_priority`, which defines the pre-assigned importance of the behavioural component. There is a fundamental difference between actions on the $\mathcal{B}^r$ and $\mathcal{B}^g$ parts of the Behaviour List. In the former case the action is selected from those available on the Response List. In the latter case the "action" taken is to place a sign onto the Goal List, or to manipulate the priority of the goal because circumstances have altered.

Potential exists to extend the $\mathcal{B}^r$ part of SRS/E to respond to a conventional external reward schedule. A separate reinforcement strategy may be put in place to re-prioritise elements of the Behaviour List relative to desirable outcomes, either employing a straightforward immediate reward mechanism or some variant of the *Q*-learning or *bucket-brigade algorithms*.

4.6. Goals and the Goal List

The *Goal List* is a sub-set of the Sign List ($\mathcal{G} \subseteq \mathcal{S}$). Any sign, whether created by the originator or formulated during the learning process, may be designated as a goal state (`goal_sign`). The structure of the SRS/E sign offers a single representational type which provides (1) a symbolic name, such that the goal can be conveniently identified internally within the system; (2) a description of what is relevant to the definition of the goal (and so what is not relevant); and (3) a test enabling the system to recognise when the goal has been achieved. Signs are attached to the Goal List under the control of the Innate Behaviour List ($\mathcal{B}^{g*}$), as previously described (postulate B3). The goal sign having the highest associated priority (`goal_priority`) is designated g^1 and so forms the seed to build the current Dynamic Policy Map. This is the *top-goal*. SRS/E supports many signs on the Goal List, after the top-goal these are designated g^2, g^3 and so on, ordered according to their given priority.

Goals are deemed *satisfied* when they appear on $\mathcal{G}^*$ (and so $\mathcal{S}^*$), realising postulate G3. The SRS/E algorithm automatically cancels satisfied goals by removing them from $\mathcal{G}$, and remaining goals on the Goal List are moved up the list automatically. As a consequence of this the Dynamic Policy Map is recomputed with the new seed and the observed behaviour of the animat changes accordingly. The change in behaviour is in effect instantaneous, and may lead to a completely different set of responses being employed by the animat in apparently identical circumstances. This is a significant departure from the reinforcement and *Q*-learning approach, where a single goal is repeatedly sought and a network of paths (a graph) constructed, dedicated to achieving the designated goal. When the Goal List becomes empty, use of the Dynamic Policy Map as a behaviour generator ceases. Until a new goal of sufficient priority is again placed on $\mathcal{G}$ observable behaviour reverts to innate actions drawn from the Innate Behaviour List $\mathcal{B}^{r*}$ or default behaviour mechanisms.

Under these circumstances the originator bears some responsibility for ensuring the stability of the Goal List ordering. SRS/E builds the DPM according to the top-goal g^1. It may be that $\mathcal{B}^{r*}$ gives rise to two goals of very similar priority, because, for instance, they are derived from sensors currently giving signals of equivalent significance. Under these circumstances the priority of the multiple goals may be unstable, swapping between the alternatives. The DPM is automatically recomputed at each priority swap causing changes or reversals of observed behaviour leading, in turn, to the inability of the animat to reach any of the enabled goal states. This is equivalent to the problem faced by any of the *Action Selection Mechanisms* (ASM) described earlier, where each must ensure that coherent patterns of behaviour are established to meet the needs of the animat.

4.7. The Hypothesis List

The Hypothesis List is the primary repository of learned knowledge within the SRS/E algorithm. Each element of the list, a *μ-hypothesis*, encapsulates a small, well-formulated, identifiable and verifiable fragment of information. A μ-hypotheses is not an unequivocal statement about the animat or its environment, but is an assertion about the nature of things - it may be true or it may be false. A μ-hypotheses may be partially complete and so true in some proportion of instances

in which it is applicable. Every μ-hypothesis is an independent observation. SRS/E supports the notion of competing hypotheses, several hypotheses that share identical pre-conditions or which share identical conclusions. SRS/E accepts mutually inconsistent μ-hypotheses, to be resolved following corroboration[22]. SRS/E does not allow the installation of duplicate copies of identical μ-hypothesis.

The originator is, of course, at liberty to incorporate into the ethogram or controlling algorithm whatever consistency checking and verification mechanisms he or she considers appropriate. To do so takes the construction of the animat controller back to the realms increasingly referred to as traditional AI (Cliff, 1994) or *GOFAI* (Good Old Fashioned Artificial Intelligence, Boden, 1994). This is a valid approach, but not the one adopted here, and moves the animat definition towards the category (3) intelligence of chapter one. In SRS/E ambiguity is resolved by application and testing of the μ-hypotheses in the form of μ-experiments, which are conducted by the SRS/E system whenever the opportunity arises to do so. In turn, μ-experiments take the form of making verifiable predictions about the perceivable state of the animat or its environment at some defined time in the future.

All μ-hypotheses in SRS/E take the form of a triplet of component parts:

$$\text{Sign1} + \text{Response} \rightarrow \text{Sign2}^{@+t} \qquad \text{(eqn. 4-6)}$$

The first sign (Sign1 or just "s1") provides a context in which the performance of the action (Response or just "r1") is hypothesised to result in the appearance of the second sign (Sign2 or "s2") some specified time in the future (at '@' the predicted time, $+t$ cycles in the future). The signs "s1" and "s2" are drawn from $\mathcal{S}$, the response "r1" from $\mathcal{R}$. Response "r1" is the action to be taken on this cycle, "s1" is the current value of the context sign. However "s1" may include token values drawn from the various activation traces, and so inherently defines a temporal as well as a spatial context. In Tolman's terms, "s2" is set as an expectancy whenever "s1" and "r1" are present. This expectancy relationship is the basis of the means-

[22]Or, if the animat is in a genuinely inconsistent environment, or in one which is unresolvably ambiguous, to remain inconsistent in perpetuity. Vershure and Pfeifer (1993) develop these issues further.

ends capability of SRS/E. If "s2" is an end, or goal, to be achieved, then “s1” and “r1” provide a means of achieving that end. In considering any μ-hypotheses with “s2” as its desired end, the corresponding “s1”, if it is not currently active and so available, may become an end, or sub-goal, in its own right. Developing a cognitive map of *means-ends-readiness* from many individual expectancies was a central component of Tolman’s expectancy theory. *Means-Ends Analysis* has developed into a cornerstone concept in traditional Artificial Intelligence from its introduction by Newell and Simon (1972) in the form of the *General Problem Solver* (GPS).

In a perfect μ-hypothesis “s1” defines exactly those conditions under which the response “r1” leads to the appearance of “s2” at the designated time. In an incompletely specified μ-hypotheses the relationship will hold on some occasions, but not others. A μ-hypothesis created as the result of an *occult occurrence* should hold very rarely (specifically, at a frequency of occurrence commensurate with the computed raw probability derived from its component parts). The evidence for *superstitious learning* was reviewed earlier. The conditions under which the μ-experiment may be performed occur whenever “s1” and “r1” are on their respective active lists ($\mathcal{S}^*$ and $\mathcal{R}^*$) at *t*=now, regardless of whether or not “r1” had been actively selected to achieve “s2”. Drescher (1991) refers to the latter case as *implicit activation*.

4.7.1. Other Hypothesis List Values

As with other list types, SRS/E μ-hypotheses have associated with them a number of values. These values record corroborative evidence about each μ-hypothesis and retain information used by the three main processes involved in the management of μ-hypotheses. These processes are: (1) μ-hypothesis corroboration and reinforcement (realising postulates H3 and H4); (2) building the Dynamic Policy Map (realising postulates P1 and P2); and (3) μ-hypothesis list maintenance (realising postulates H6 and H7). Some of the list element values associated with each μ-hypothesis are described next, and the three main processes and the μ-hypothesis values associated with them in the sections that follow. As each of the three processes are intimately interrelated, the order of these sections is somewhat arbitrary chosen.

Each μ-hypothesis on the Hypothesis List is assigned a unique `hypo_identifier`, created from the list index number. Index numbers are created in sequential order, and so indicate the relative age of the μ-hypothesis. The designation "Hnnnn" appears in the output log and analysis information, where nnnn is the list index number. The values `hypo_first_seen` and `hypo_last_seen` respectively record the cycle on which the μ-hypothesis was created and the most recent cycle on which the μ-hypothesis was active. A μ-hypothesis is defined as active when the following conditions are met on any given execution cycle:

$$h \in \mathcal{H}^* \textbf{ iff } s1(h) \in S^* \textbf{ AND } r1(h) \in \mathcal{R}^* \qquad \text{(eqn. 4-7)}$$

These conditions define when a μ-hypothesis will perform a μ-experiment by making a verifiable prediction. The value `hypo_activation_trace` records the most recent activations for the μ-hypothesis. The value `time_shift` records the number of cycles between an activation of a μ-hypothesis and the time that the "s2" sign is predicted to occur. The derived value `hypo_age` indicates the number of cycles elapsed since the μ-hypothesis was created. It is calculated from `hypo_first_seen` and the system variable "now".

The remaining values associated with each Hypothesis List entry may be characterised into serving one of three purposes. (1) Corroborative values recording the performance of the predictive ability of a μ-hypothesis. These values reflect the confidence the system may place in the effectiveness of the μ-hypothesis when building the Dynamic Policy Map, and in calculating when to modify or delete individual μ-hypotheses. These values broadly reflect the notion of *schema confidence weight* adopted by Becker and Mott. (2) Values computed, and re-computed, each time the Dynamic Policy Map is prepared. These values provide the action selection mechanism with the basis to determine which μ-hypothesis (and hence which action "r1") should be passed to the actuation sub-system during goal seeking behaviour. (3) Administrative values, recording information relevant to the creation and subsequent modification of individual μ-hypotheses. Major section headings will now be given over to the discussion of these values, reflecting their importance to the operation of the SRS/E algorithm.

4.8. Corroborating μ-Hypotheses, Predictions and the Prediction List

Every time a μ-hypotheses is activated it will perform a μ-experiment and so make a prediction, which will be verified on a later execution cycle. Each prediction is placed on the *Prediction List*, $\mathcal{P}$. As predictions are all of the form where a known sign is expected at a known time, the validation process is a straightforward matter of matching the elements of $\mathcal{P}$ which were predicted for the current execution cycle against the active Sign List $\mathcal{S}^*$. Alternative interpretations are available as to how "credit" for a correct or "debit" for an incorrect prediction should be assigned to the individual μ-hypotheses responsible for the prediction. These alternatives are reflected in the corroboration (H3) and reinforcement (H4) postulates. SRS/E maintains four values for each μ-hypotheses for this purpose.

Following Popper's notion that it is the absolute frequency of outcome that provides the appropriate measure of a hypothesis, the values `hypo_cpos` (cumulative positive, `cpos`) and `hypo_cneg` (cumulative negative, `cneg`) record the number of successful and unsuccessful predictions respectively. Specifically:

$$\texttt{cpos} \leftarrow \texttt{cpos} + 1 \textbf{ iff } \texttt{s2}(\mathcal{h})^{@t=\text{pred}} \in \texttt{predicted\_sign}(\mathcal{P})^{@t=\text{pred}} \qquad \text{(eqn. 4-8)}$$

$$\texttt{cneg} \leftarrow \texttt{cneg} + 1 \textbf{ iff } \texttt{s2}(\mathcal{h})^{@t=\text{pred}} \notin \texttt{predicted\_sign}(\mathcal{P})^{@t=\text{pred}} \qquad \text{(eqn. 4-9)}$$

These two equations compare predictions made at some point in the past (*t*=pred) to the appearance of actual signs at that predicted time. These two measures reflect the overall effectiveness of the μ-hypothesis over its span from the point of creation (the execution cycle recorded in `hypo_first_seen`), to the current execution cycle (less any predictions made, but not yet verified). The overall probability that the expectation defined by the μ-hypothesis will hold is therefore defined by:

$$\texttt{hypo\_prob} \leftarrow \frac{\texttt{cpos}}{\texttt{cpos} + \texttt{cneg}} \qquad \text{(eqn. 4-10)}$$

This is the corroboration measure (Ch of postulate H3). By definition every μ-hypothesis is assumed to represent a successful prediction at the time of its creation. This assumption is considered reasonable when using the pattern

extraction creation process described later, even though the μ-hypothesis may subsequently be determined to denote an *occult occurrence*. This initial fillip to a new μ-hypothesis' confidence value will be referred to as the *creation bonus*.

In a changeable environment the validity of any given μ-hypothesis may also change with time. To reflect this the value `hypo_bpos` (`bpos`) is updated according to a discounting factor, thereby giving precedence to the effects of recent activations at the expense of those further in the past, specifically:

$$\texttt{bpos} \leftarrow \texttt{bpos} - \alpha(\texttt{bpos} - 1) \textbf{ iff } \texttt{s2}(\mathcal{h})^{@t=\text{pred}} \in \texttt{predicted\_sign}(\mathcal{P})^{@t=\text{pred}} \quad \text{(eqn. 4-11)}$$

or

$$\texttt{bpos} \leftarrow \texttt{bpos} - \beta(\texttt{bpos}) \textbf{ iff } \texttt{s2}(\mathcal{h})^{@t=\text{pred}} \notin \texttt{predicted\_sign}(\mathcal{P})^{@t=\text{pred}} \quad \text{(eqn. 4-12)}$$

otherwise

`bpos` unchanged

where:

α is the positive *reinforcement rate*, $(0 \leq \alpha \leq 1)$
and
β is the negative *extinction rate*, $(0 \leq \beta \leq 1)$

This implements the *reinforcement measure* (Rh of postulate H4). Long sequences of successful predictions for a single μ-hypothesis will asymptotically tend its `bpos` values to 1.0, long sequences of failed predictions will similarly tend `bpos` values towards 0.0. This notion of an asymptotic *negatively accelerating curve* is ubiquitous throughout the conditioning and behaviourist literature, and forms the basis of MacCorquodale and Meehl's (1954, p. 237) *strength of expectancy* measure. This procedure is similar to those used in most recent reinforcement and the *Q*-learning mechanisms.

The last value in this group is `recency`, which specifically records the outcome of the most recently completed prediction for each μ-hypothesis. The `recency` measure represents an alternative approach to Drescher's modelling of *object permanence*. The `recency` value remains asserted for any individual μ-hypothesis after a valid prediction about "s2" is detected. It is cleared when the prediction next fails. It acts as one form of event memory. Unlike Drescher's system SRS/E contains no inherent mechanism supporting the representation or manipulation of a "physical object".

The different measures `cpos`, `cneg`, `bpos` and `recency` serve different purposes in the generation of the Dynamic Policy Map (cost estimation) and in the management of the Hypothesis List (differentiation and deletion of ineffective μ-hypotheses). These differently computed values may reflect different views of the predictive effectiveness of μ-hypotheses. SRS/E may represent permanent (`hypo_prob`), semi-permanent or recurring (`bpos`), and transient (`recency`) phenomena. In this context the term "permanent" may equally be applied to an immutable physical law as to any phenomena that remains consistently predictable throughout the lifetime of the animat. For example, an animal, or animat learning to seek nourishment may locate a source that is habitually available, which may reliably be returned to. Equally a source of nourishment may be identified, which only comprises a finite quantity of sustenance. Finally the creature may happen across a single item of nourishment, which once consumed is finished. No second order effects are proposed for SRS/E to further classify individual μ-hypotheses into these various categories based on longevity of the phenomenon underlying the prediction. Such a strategy might properly be included in later implementations.

4.8.1. Prediction List Element Values

Each element of the list is created from the "s2" of any activated μ-hypothesis. Each element retains only three items, `predicting_hypo`, the identity of the μ-hypothesis responsible for the prediction, `predicted_sign` and `predicted_time`, the sign expected and the execution cycle on which it is predicted to occur. Elements of $\mathcal{P}$ are deleted as soon as the prediction they define has been verified against $\mathcal{S}^*$. As each prediction is held separately, any μ-hypothesis may have several predictions waiting for confirmation (as each μ-hypothesis may make at

most one prediction on each execution cycle this is limited by the number of cycles between now and t^{pred}). There may equally be more than one prediction of a given sign for each future execution cycle, as many different μ-hypotheses may predict the same outcome.

4.9. The Dynamic Policy Map (DPM)

Whenever $\mathcal{B}^{g*}$ is not empty and the priority of the top-goal is greater than that for the highest priority candidate action from $\mathcal{B}^{r*}$ the SRS/E algorithm will attempt to construct a *Dynamic Policy Map* (DPM, after definition P0) from knowledge accumulated in the Hypothesis List. The effect of the Dynamic Policy Map is to categorise entries in the Sign and Hypothesis Lists according to an estimate of their effectiveness as being on a path of actions that will lead to the satisfaction of the top-goal. The SRS/E algorithm builds the Dynamic Policy Map by the process of *spreading activation*, based on repeated application of the *spreading valence* postulate (postulate P2). Individual μ-hypotheses, h, which lead directly to the top-goal, g^1, are selected (where $s2(h) = g^1$). This selection and binding process will be referred to as "valencing", following Tolman's use of the term. Context signs in these μ-hypotheses may then act as "sub-goals", allowing another sub-set of the Hypothesis List to be incorporated into the Dynamic Policy Map. The SRS/E algorithm stops building the DPM once all the entries in the Hypothesis List have been incorporated or there are no more μ-hypotheses that may be chained in this way. Signs and μ-hypotheses incorporated in the DPM are termed *sub-valenced*. The *valence level* of each μ-hypothesis incorporated into the DPM indicates the estimated minimum number of sub-goals that must be traversed to reach the designated goal sign.

The Dynamic Policy Map may be considered as a graph structure. Signs from the Sign List act as nodes, μ-hypotheses from the Hypothesis List the arcs. One special sign, the top-goal, acts as the seed or start point for the spreading activation process to create the graph. Development proceeds on a breadth-first basis, μ-hypotheses at each valence level are selected at the same step in the spreading activation process. This is implemented as a variant of the well-established *graph-search* procedure (Nilsson, 1980, Ch. 2).

Every arc has associated with it a cost estimate. An arc is traversed by selecting the action, "r1", from the μ-hypothesis. The true cost of traversing the arc is given by the `response_cost` value assigned to each action (the *action cost* of postulate A2). This is simply the "effort" expended in taking the action, as provided in the Response List. The estimated cost of traversing the arc to a node at the next valence level takes into account the true cost of the action and the relative effectiveness of the μ-hypothesis in actually achieving its expected outcome, based on past experience. This `cost_estimate` for each μ-hypothesis is prepared from:

$$\texttt{cost\_estimate} \leftarrow \frac{\texttt{response\_cost}}{\texttt{hypothesis\_confidence}} \qquad \text{(eqn. 4-13)}$$

This realises the Cost Estimate postulate (P3). The `hypothesis_confidence` value is in turn prepared from:

$$\begin{aligned}\texttt{hypothesis\_confidence} \leftarrow\ & (\texttt{hypo\_prob} * \gamma^1) + \\ & (\texttt{hypo\_bpos} * \gamma^2) + \\ & (\texttt{recency} * \gamma^3) + \\ & (|\texttt{oscill}| * \gamma^4)\end{aligned} \qquad \text{(eqn. 4-14)}$$

where:

$$(\gamma^1 + \gamma^2 + \gamma^3 + \gamma^4) = 1$$

and

$$(0 \leq \texttt{oscill} \leq 1)$$

The `hypo_prob`, `hypo_bpos` and `recency` values are those previously described. The `oscill` component is an essentially random factor designed to perturb the path selection process. This has the dual effect of adding an element of uncertainty to encourage the use of other μ-hypotheses, and to allow the system to escape from potential behavioural loops. The effect of this parameter is intended to reflect the use that Hull describes for his *oscillatory* component, ${}_{S}O_{R}$, from which the current name is derived. In implementation the value of `oscill` is derived from the pseudo-random number sequence generator (and so is not really "oscillatory" at

all). While superficially similar in effect to Sutton's (1991) *exploration bonus* in *Dyna-Q+*, the balance of goal-seeking behaviour to exploration is ultimately achieved in a quite dissimilar manner in SRS/E. This is considered in detail in chapter six.

The cost estimate for each arc, ignoring the `oscill` component, reflects the given action cost scaled by the recorded probability that the causal relationship described by the μ-hypothesis is indeed responsible for the transition. Assuming for the moment that the *selection factor* γ^1 has been set to one (and so γ^2, γ^3 and γ^4 are all zero[23]) the `cost_estimate` for the arc is equal to the true (given) cost of the action "r1" when `hypo_prob` is at its maximum value. This condition only holds when the μ-hypothesis has never failed. Where a μ-hypothesis has been created as result of an occult occurrence the value of `hypo_prob` will tend to zero, and so the value of cost_estimate will tend toward infinity. The `hypo_prob` value will never reach zero, due to the initial *creation bonus*. Increasing the relative contribution of γ^2 (at the expense of γ^1) biases cost estimates toward more recent experiences. Values for the factors γ^1, γ^2, γ^3 and γ^4 are set by the experimenter before each experiential run, and are fixed for the duration of that run in the current implementation.

No account in the computation of the cost estimate is taken of the experience of the μ-hypothesis, as recorded in the `hypo_age` and `hypo_maturity` measures, in the current implementation. For the experiments described later the creation bonus serves to increase the likelihood that a new (and therefore inexperienced) μ-hypothesis will be selected and so appears to provide an adequate balance of new and old knowledge. A more sophisticated strategy may bias the estimate to more experienced μ-hypotheses where the importance or priority of the goal is high. Conversely newer, less experienced, μ-hypotheses may be favoured in *play* situations, where (apparently unimportant) goals are set for the explicit purpose of gaining experience and knowledge. Such considerations are left for future investigations.

[23] Note that these superscripts indicate the first γ, the second γ and so on; similarly g^1, g^2, etc.

4.9.1. Selecting actions from the DPM

Every μ-hypothesis implicated in the DPM is assigned a `policy_value`, the minimum sum of individual `cost_estimate` elements across all the arcs from the sign node associated with "s1" to the goal sign node g^1. This is a realisation of Postulate P4. During the graph building process the `policy_value` associated with each node is updated if a lower cost route to that node is discovered. Figure 4-1 shows a printout from an experimental log showing a *valenced path*, the lowest (estimated) cost path from the current situation to the desired goal. It records the individual μ-hypotheses (e.g. "H119") selected from the graph, the individual cost contributions from `cost_estimate` ("cost") and the cumulative `policy_value` ("total") values as the valence levels are traversed. It starts with a node ("X2Y0", the printout has automatically dereferenced signs to external names) that is currently on the active Sign List $\mathcal{S}^*$, and so defines the μ-hypothesis ("H126") which will contribute the reified action ("U") in the current execution cycle.

```
H126 predicts X2Y1 from X2Y0 (active) after U (cost = 1.818182, total = 15.006273)
H117 predicts X3Y1 from X2Y1 after R (cost = 1.290323, total = 13.188091)
H119 predicts X4Y1 from X3Y1 after R (cost = 1.059603, total = 11.897769)
H120 predicts X5Y1 from X4Y1 after R (cost = 1.290323, total = 10.838166)
H4 predicts X6Y1 from X5Y1 after R (cost = 1.290323, total = 9.547844)
H5 predicts X7Y1 from X6Y1 after R (cost = 1.290323, total = 8.257522)
H6 predicts X8Y1 from X7Y1 after R (cost = 1.290323, total = 6.967199)
H8 predicts X8Y2 from X8Y1 after U (cost = 1.126761, total = 5.676877)
H9 predicts X8Y3 from X8Y2 after U (cost = 1.078894, total = 4.550116)
H10 predicts X8Y4 from X8Y3 after U (cost = 2.351558, total = 3.471222)
H11 predicts X8Y5 (goal) from X8Y4 after U (cost = 1.119664, total = 1.119664)
Valenced path in 11 steps, estimated cost 15.006273
```

Figure 4-1: Log Printout of a Valenced Path

It is important to note that the valence path printout is not a set of prescribed actions to be performed to reach to goal state, as would be the case in *STRIPS* (Fikes and Nilsson, 1971), but rather a sub-set of the total DPM. It is presented to provide the experimenter with information about the current state of the animat under investigation. The action selected may, or may not, lead to the expected sign at the lower valence level on the valence path. On the next execution cycle a new assessment of the environment is made, as indicated by a new $\mathcal{S}^*$.

The next action is selected from the DPM on the basis of the new S^*. It may be that the next action on the existing valence path is selected. However the new S^* may indicate that a shorter route has, through fortuitous circumstance, become available; equally only longer routes may now be available. In each eventuality the DPM acts essentially equivalently to the *policy map* in *reinforcement* and *Q-learning* algorithms, recommending the best course of action relative to the current circumstances and the goal sought.

There is a pathological case where no intersection between S^* and the DPM exists and so no action can be selected from the DPM. Under this circumstance the current algorithm selects an exploratory trial and error action at random. A more sophisticated variant of the algorithm might balance the return to exploratory activity with a "faith" that the action was perhaps successful, but that the expected outcome had not been properly detected. In this way the animat may continue along a previously computed valence path and avoid the potential disruption caused by deflecting to exploratory actions.

4.9.2. Recomputing the DPM

There are several circumstances where the SRS/E algorithm must recompute the Dynamic Policy Map. When the top-goal, g^1, is satisfied, the next highest priority goal becomes the top-goal, and a new DPM must be computed before another action may be selected. Similarly innate behaviours from the Behaviour List may alter the priorities of the Goal List (realising postulate B3), also precipitating a recalculation of the DPM. At each execution cycle many μ-hypotheses may have their values updated, reflecting predictions they made in the past. At any cycle new μ-hypotheses may be added to the Hypothesis List, or existing ones deleted from the list. Any of these changes can have profound effects on the best paths through the graph. On the other hand, recomputing the DPM is a cost overhead not to be ignored. The SRS/E algorithm must recompute the DPM if the goal changes, but the experimenter may control the sensitivity of SRS/E to changes in the Hypothesis List.

The system variable `rebuildpolicynet` is cleared each time the DPM is rebuilt. It is incremented by some quantity Δ each time the Hypothesis List changes, and by

some (typically smaller) amount δ every time a μ-experiment prediction fails. Before each use of the DPM `rebuildpolicynet` is compared to the system constant `REBUILDPOLICYTRIP`, the DPM being recreated once this trip value is reached or exceeded by `rebuildpolicynet`. Apart from the effect these values have on the balance of resource utilisation by SRS/E on policy construction and other computational activities, they also have a profound effect on aspects of the animats observable behaviour. This effect is particularly apparent in the *dual path blocking* experiments described later. In the current implementation Δ and δ are selected such that the DPM is rebuilt following any change.

4.9.3. The DPM, A Worked Example

Figure 4-2 shows a graph generated from the model Hypothesis List shown embedded in the figure. For the purposes of this example a DPM comprising eight signs and 12 μ-hypotheses is created. In this instance the top-goal, g^1, is equated to sign number "S16". Only three actions are available on the Response List, "A1", "A2" and "A3" all with an actual cost of one. The third column shows some possible "cost estimate" values for the various μ-hypotheses following a period of behaviour. At each valence level in the graph the policy cost associated with each sign is the cumulative policy value of the lowest cost path through the graph to the chosen goal. Each arc is labelled with the μ-hypothesis responsible for the transition, with its action and associated cost estimate.

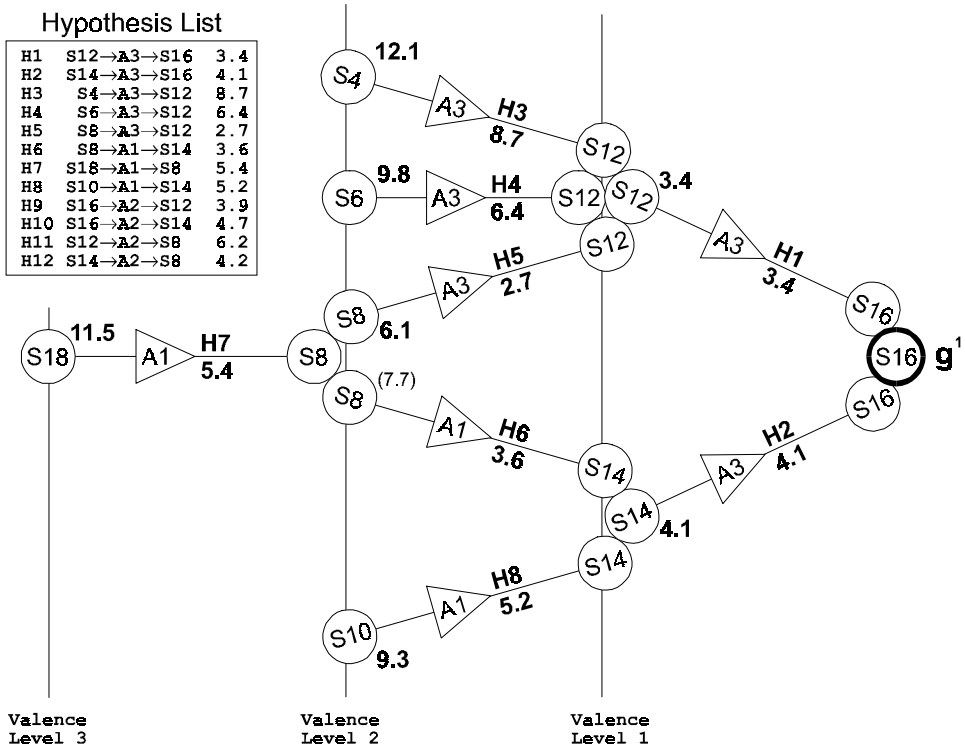

Figure 4-2: Model DPM Generated from Sample Hypothesis List

It may be that on the current execution cycle signs "S4" and "S18" are active and so on S^* (figure 4-3a). Policy cost for "S18" is lower than "S4", so SRS/E selects action "A1". The expectation is that "S8" will appear on S^* on the next execution cycle, and so action "A3" from μ-hypothesis "H5" would be selected. As a consequence these circumstances the `hypothesis_confidence` value of the successful μ-hypothesis "H7" would be strengthened, and that for the unsuccessful μ-hypothesis "H3" would be diminished (figure 4-3b). With "S8" on the active Sign List, SRS/E will choose the path described by "H5", performing action "A3", expecting sign "S12". If this expectation is met, "H5" is strengthened, and action "A3" (from "H1") will be selected on the next execution cycle; leading to goal satisfaction if that subsequent expectation is also satisfied.

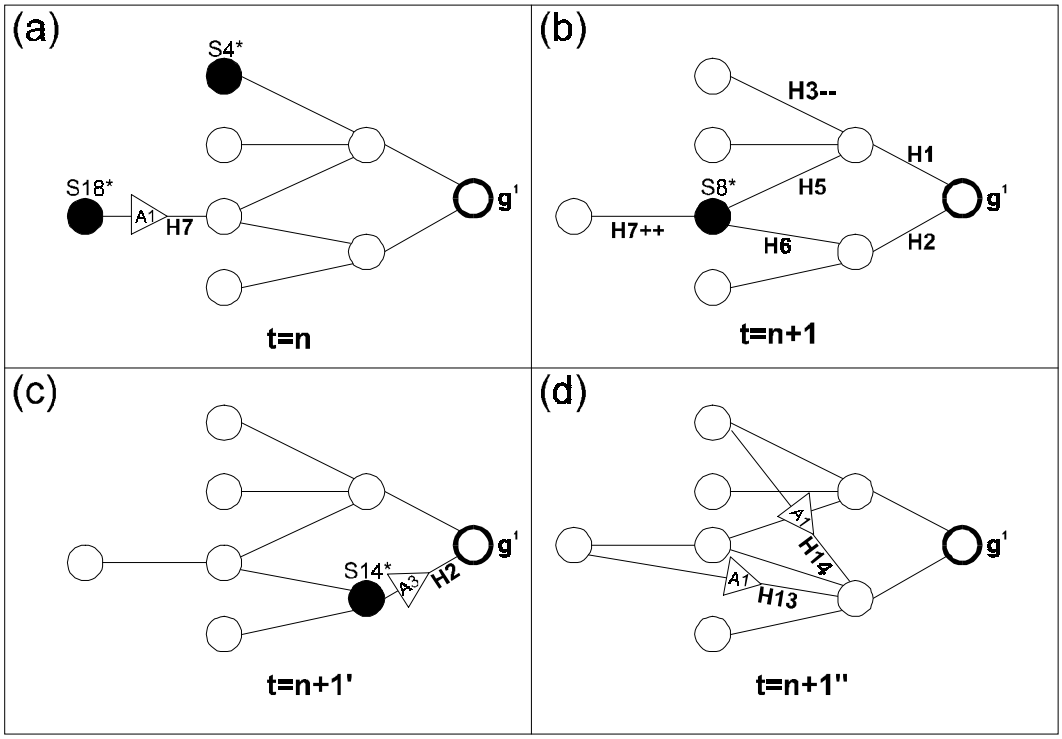

Figure 4-3: Various Outcomes for Model DPM

If, at the step indicated by figure 4-3b, the action "A3" did not lead to the expected sign "S12", but instead "S8" remained on $\mathcal{S}^*$ then confidence in "H5" would be weakened. Eventually the cumulative cost of the path "H5"-"H1" would exceed that for "H6"-"H2", at which point SRS/E would attempt action "A1" (from "H6"). Note that the confidence in "H6" was unaltered during the time "A3" actions were attempted, because it was not placed on $\mathcal{H}^*$ as its "r1" precondition was not matched and so it was not eligible to issue a prediction. The rate at which the estimated cost of any path rises under these circumstances is primarily controlled by the β extinction rate factor; though changes in estimated cost will not take effect until increments to δ (and Δ) cause the DPM to be recomputed.

What SRS/E hypothesises about the consequences of its actions in the environment, and what actually occurs may not hold true in practice. Considering again the situation described by figure 4-3a, it may be that rather than the expected activation of "S8", sign "S14" is activated (figure 4-3c), either through some previously unknown path, or by a previously undetected event. On this execution

cycle SRS/E would select the action "A3" associated with μ-hypothesis "H2". If this expectation subsequently holds the top-goal would be achieved, and so removed from the Goal List. As a side effect of this unexpected transition SRS/E may create the new μ-hypotheses "H16:(S4→A1→S14)" and "H17:(S18→A1→S14)" (figure 4-3d), employing the mechanism of postulate H5-2.

Under the initial conditions described by figure 4-3a, the new paths of lower estimated cost offered by "H16" and "H17" may be considered in future instances in preference to either "H3" or "H7" originally available. Where they are due to a genuinely repeatable phenomenon the confidences of these new μ-hypotheses will be strengthened, leading to the adoption of the lower cost estimate path. Where the μ-hypotheses were created due to occult or unrepeatable circumstances the use of the new, apparently preferable, path will fall into disuse following a number of unsuccessful applications. The experimental procedure adopted in chapter six can give rise to this phenomenon (for instance, the effect shown in figure 6-10c), and it will be considered further.

The effects of recomputing the Dynamic Policy Map can completely alter the response of SRS/E to incoming tokens. Figure 4-4 shows an alternative computation of the DPM graph using the same Hypothesis List as Figure 4-2, but where the goal definition has changed from "S16" to "S8". Note in particular that, although none of the cost estimates for the μ-hypotheses have changed, the response of the system to signs "S14" and "S12" is now completely different. This feature differentiates the behaviour SRS/E from the reaction of reinforcement and *Q*-learning systems in the manner highly reminiscent of Tolman's arguments in favour of *expectancy theory* over stimulus-response theorising.

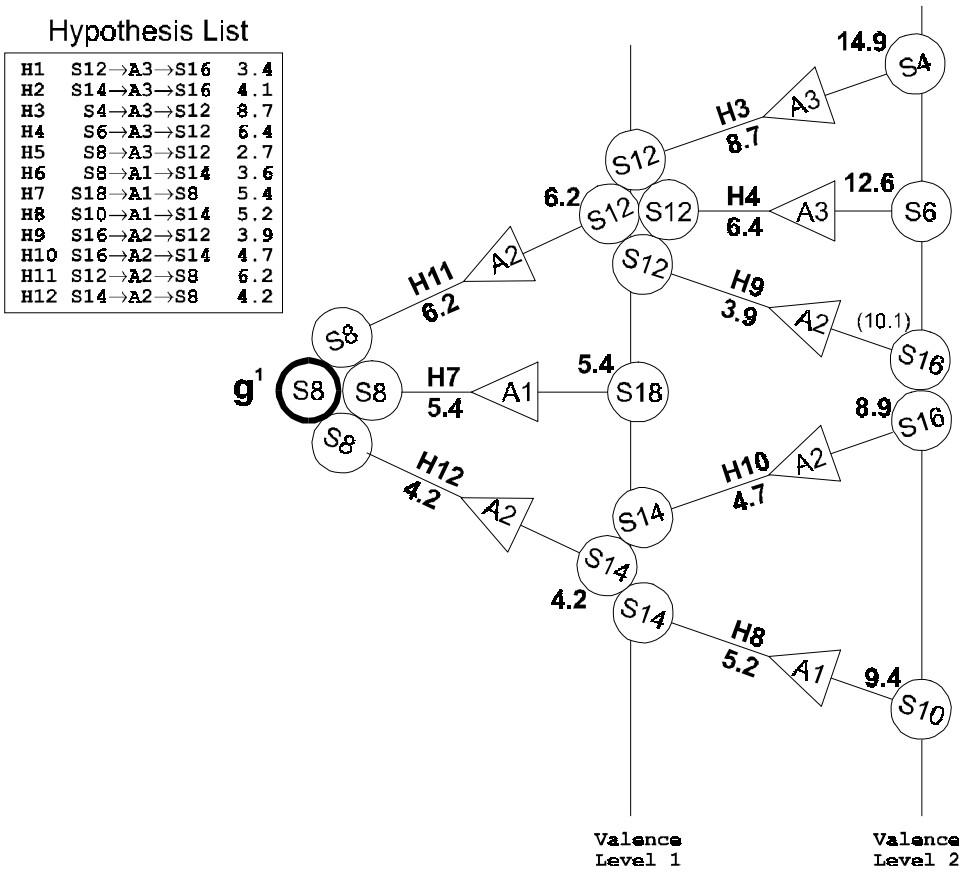

Figure 4-4: Model Graph Recomputed for Goal "S8"

4.9.4. Pursuing Alternative Goal Paths

The Dynamic Policy Map indicates the path with the currently most favourable estimated cost from an active sign state to the highest priority top-goal state. Actions are selected on the basis of this estimate. Consider the DPM graph shown in figure 4-5.

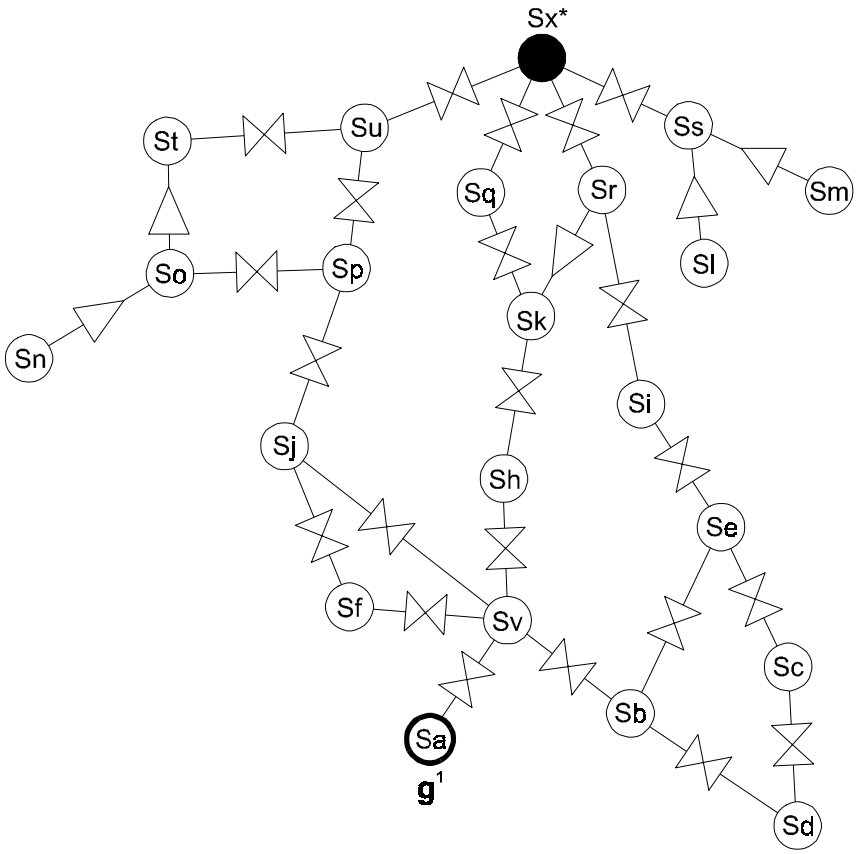

Figure 4-5: A Sample Dynamic Policy Map

The top-goal (g^1) is equated to "Sa", and the only active sign is "Sx*". Six distinct paths are available to the animat. These are summarised in table 4-2, with illustrative cost estimates. Individual signs are shown with letters, not sequence numbers, purely as a shorthand notation. The double arrow on an arc indicates a pair of μ-hypotheses, for instance a path is known both between "Sx" and "Su" and between "Su" and "Sx". No path is available through the loop formed by "Su"-"St"-"So"-"Sp" as no μ-hypothesis exists for the transition "St"-"So", as indicated by the unidirectional arrowhead.

Path	"Estimated Cost"
(1) Sx-Sq-Sk-Sh-Sv-Sa	18.4
(2) Sx-Sr-Sk-Sh-Sv-Sa	20.8
(3) Sx-Su-Sp-Sj-Sv-Sa	38.5
(4) Sx-Su-Sp-Sj-Sf-Sv-Sa	45.7
(5) Sx-Sr-Si-Se-Sb-Sv-Sa	67.9
(6) Sx-Sr-Si-Se-Sc-Sd-Sb-Sv-Sa	158.1

Table 4-2: Paths Through Figure 4-5 Graph

On the basis of the cost estimates shown, the animat will select the action "r1" associated with μ-hypothesis "Hxq" (indicating the transition from "Sx*" to "Sq"). If this expectation is met, the animat selects "Hqk", and so on. Should this path succeed, then sign "Sa" will be removed from the Goal List, and path (1) will be strengthened. If, while at node "Sq", the expectation described by "Hqk" failed, the cost of the remaining path "Sq*"-"Sk"-"Sh"-"Sv"-"Sa" would rise, due entirely to the increased estimate for "Hqk". In practice under these circumstances, the increase in cost for a single expectation failure is relatively small and it may be that the estimated cost of the remaining path is still below that for any alternative, so that "r1" from "Hqk" will be tried again. Even if the remaining path would have a greater cost, if the effect of δ (the expectation failure policy rebuild increment) is small the DPM may not be rebuilt, and the policy decision will remain unaltered.

At some point, the cost estimate would come to exceed that for the next lower estimated cost path, "Sq*"-"Sx"-"Sr"-"Sk"-"Sh"-"Sv"-"Sa" in a recomputed DPM, and the action associated with "Hqx" would be selected. If this is also blocked at some point, the next lowest cost estimate path would be attempted, starting from the currently active node. Each time the cost estimates indicate a new path, following a DPM recomputation, a new solution path is tried. The frequency with which the DPM is recomputed determines how persistent the animat will appear to be in pursuing a blocked course of action.

Individuals with values of Δ and δ that are small relative to `REBUILDPOLICYTRIP` will persist with one course of action longer than individuals where these values are

correspondingly larger. *Persistence of behaviour* may be an appropriate course of action. In the environment he describes, the probability of Mott's robot reaching the charger under the influence of the schema "<BRIGHT>S $\rightarrow$ <FORW>M $\Rightarrow$ <ON-CHARGE>S" is very low. It is nevertheless the best option available, and a persistent individual animat that did not swap between other alternatives frequently would be advantaged. In other circumstances the ability to change to a potentially better solution path may be advantageous, where there is serious competition from other individuals for limited resources, for instance. No second order learning phenomena are currently implemented in the SRS/E algorithm to determine an appropriate balance between persistence and fickleness in selecting a solution path.

4.9.5. Pursuing a Goal to Extinction

In the situation where all possible paths to a top-goal are unobtainable, continued attempts at the goal become a threat to the animat's survival by locking out other behaviours. The goal must be forcibly abandoned, this is the *goal extinction point* (postulate G4). Goal extinction is achieved in the SRS/E algorithm by removing the unsatisfied top-goal, g^1, from $\mathcal{G}$. The animat would then be free to pursue the next highest priority goal as top-goal, or other behaviours if there are no further elements on $\mathcal{G}$. Extinction of behaviours has been widely observed experimentally (section 3.6.3). Extinction does not, however, appear as an abrupt abandonment of the behaviour. Instead the behaviour persists for a time (the "on-period"), then suspended briefly (the "off-period") before being resumed for another on-period. This alternation of apparently goal directed behaviour with periods of some other activity persists for a time, until the goal directed behaviour finally appears to be completely suppressed. The relative lengths of the "on" and "off-periods" change in a characteristic manner, the periods "on" shortening and the periods "off" lengthening.

During goal directed behaviour SRS/E always takes the best possible estimated path, there is no explicit exploration during this type of behaviour. SRS/E does not attempt to locate new paths, but instead applies its resources to achieving the goal using the best known path. At the end of the first "on" period behaviour reverts to default *trial and error* actions. This period has the effect of exploring for new paths through the graph. If the animat "stumbles" upon the solution and arrives at

the goal it is satisfied in the normal way, and a new path is known for future use. Lengthening periods of exploration have the effect of widening the area of search in the graph space, increasing the likelihood of happening on a previously unknown path through the cognitive map and thereby reaching the top-goal[24]. The duration of the first "on-period" is determined from the initial cost-estimate of the best path in the graph. The *valence break point* (VBP, described by postulate P6), is set to some multiple of the initial lowest policy value cost estimate (`bestcost`) computed by the algorithm. This multiple is defined by the system constant `VALENCE_BREAK_POINT_FACTOR`, currently set to 10.

$$\texttt{VBP} \leftarrow \texttt{bestcost} * \texttt{VALENCE\_BREAK\_POINT\_FACTOR} \qquad \text{(eqn. 4-15)}$$

Thus in the example given by figure 4-5 (table 4-2), goal directed behaviour would continue until the estimated cost of the best available path exceeds a value of 184.0. The multiplier value is selected to give the animat ample opportunity to achieve the goal by direct use of the DPM, allowing a generous margin for failed expectations.

Once the *policy value* of the best path reaches the VBP value the goal is temporarily suppressed, and VBP is again multiplied by the valence break point factor (to 1840.0). On reaching each break point behaviour reverts to exploratory actions for a period determined by a `goal_recovery_rate` parameter, the *goal recovery mechanism*. Actions taken during this period are referred to as *unvalenced actions*, to distinguish them from purely trial-and-error exploratory activities. On the first suppression the goal recovery rate is high, and behaviour reverts to goal directed quickly after only a few unvalenced actions.

On reaching each subsequent valence break point the goal recovery rate is reduced (in the current implementation by a factor of two) and so the number of unvalenced actions during the off-period increases. Each time the blocked μ-hypothesis fails the estimated cost of the step increases at an exponential rate, and the time taken to

[24]Panic reactions may be an extreme form of this phenomena, wild or exaggerated actions being performed, possibly beyond the normal limits to physical well-being, in a final attempt to escape some intolerable condition. Indistinguishable behaviours may equally be part of the innate behavioural repertoire, unrelated to goal seeking.

reach the next VBP level decreases as a consequence. At some point the estimated cost of the path exceeds the *goal cancellation level*, Ω, and the unachievable top-goal is automatically deleted from the Goal List.

The extinction process will be demonstrated experimentally in chapter six, but it is most clearly shown when only a single path exists through the DPM to the goal. Such is in effect the case in *Skinner box* experiments. Only pressing the bar delivers the reward. Similarly the only known route to the goal definition sign "Sa" in figure 4-5 is via the path "Sv"-"Sa" (μ-hypothesis "Hva"). If the experimenter denies the animat access to "Sa", then "Hva" will be tried on every attempt to reach the goal (since there is no other known option), and the estimated cost of this step will rise until Ω is reached. On the other hand if there is some other, as yet unknown, route, then the periods of exploration give the animat the possibility of discovering it by growing the cognitive map. These effects are investigated in the *path blocking* and *alternative path experiments* of chapter six.

4.10. Creating New μ-Hypotheses

New μ-hypotheses are created under two specific circumstances, (1) the appearance of a completely novel sign, postulate H5-1 (*novel event*); and (2) the appearance of a sign that is known, but which was not predicted, postulate H5-2 (*unexpected event*). SRS/E may therefore operate under the *tabula rasa* conditions discussed previously. It is also a strong example of an *unsupervised learning* procedure, no intervention is required from the originator or experimenter to cause or guide the learning process. The originator may, of course, build behavioural patterns into the Behaviour List intended to advantage or bias the animat's learning process. The experimenter may equally establish situations that trigger or exploit the animat's innate learning ability to train or teach the animat. In the experiments to be described no such behaviour patterns are used. Conditions under which the experimenter intervenes are described were appropriate.

SRS/E uses a *pattern extraction* method for creating new μ-hypotheses. The detection of a novel or unpredicted sign, notated for the moment "s2", causes SRS/E to extract a recent action, "r1", from $\mathcal{R}$, as recorded in the

response_activation_trace values, and to extract a sign, "s1" from $\mathcal{S}$, as recorded in the sign_activation_trace values. The new μ-hypothesis:

$$h \leftarrow \mathcal{H}(s1,r1,s2^{@+t}) \qquad \text{(eqn. 4-16)}$$

is created from the components extracted from the various traces. Note the use of the notation "$\aleph \leftarrow \mathcal{X}(y)$" to denote the creation of a list element of type $\aleph$ from some (appropriately typed) element or elements "y". Note also that the action selected is to be drawn from at least one execution cycle in the past, and that the context sign "s1" shall be contemporary with the action "r1". As a convention, where "s2" follows "s1" and "r1" by exactly one execution cycle the use of the *"@" (at) notation* will normally be dispensed with, as this is the default relationship. Where all the component token parts for "s1" are drawn from their respective activation traces, then action selection and prediction by the μ-hypothesis will not depend on the current state of the system, only on the recorded past states.

In keeping with Popper's observation that the simplest means possible should be employed to describe the phenomena (*occam's razor*), the current implementation of SRS/E initially creates new μ-hypotheses to this notion, concurrent sign "s1" and action "r1" predicting the target sign "s2" on the next execution cycle. The exact combination of elements for the new μ-hypothesis are specified by a *hypothesis template*, which in the current implementation is coded into the structure of the SRS/E algorithm. As the size of $\mathcal{S}^*$ increases, the number of possible options for inclusion in the new μ-hypothesis will increase. Currently, SRS/E may limit the number of μ-hypotheses created for each novel or unpredicted sign appearance. This, in effect, creates a *sampling strategy* for the learning process. The mechanism for an explicit sampling strategy implemented in SRS/E is described later.

This is a form of *instrumental learning*, predicated on a fundamental notion of *causality* between the context in which the animat makes actions, the specific actions made by the animat and the consequences to the animat and its environment of those actions. It is an *animat-centric view*, but there may be other

active agents in the environment causing changes. These are only recorded by the animat in so far as they affect the animat's ability to manipulate its circumstances.

Shettleworth (1975) provides evidence that animals may be predisposed to utilise features from the environment selectively. With or without this innate bias it would be a reasonable alternative strategy to create many μ-hypotheses in an attempt to explain the occurrence of the novel phenomena, and allow the subsequent corroboration process to select useful μ-hypotheses and discard the remainder, a sub-set sampling assumption. In the absence of any underlying "theory" about the environment, which is the default assumption, each μ-hypothesis forming "guess" is as good as another[25].

4.10.1. Maintaining the Hypothesis List

Given the use of the *pattern extraction* (token selection from the various lists $\mathcal{I}$, $\mathcal{S}$, $\mathcal{R}$ and $\mathcal{H}$) method for creating new μ-hypotheses one of four outcomes will emerge following a period of corroboration. First, an individual μ-hypothesis may accurately predict its outcome. Second, a μ-hypothesis may accurately predict its outcome only in a fraction of the instances in which it is activated. Third, a μ-hypothesis may never, or very rarely predict correctly. Fourth, a μ-hypothesis may not be activated again, and so will make no predictions that may be corroborated.

The first of these outcomes needs no immediate action. The second outcome may indicate that the μ-hypothesis be a candidate for *specialisation*, one form of differentiation (postulate H6). By this process extra tokens are added to the context sign "s1", on the assumption that the μ-hypothesis is underspecified in its application. JCM and ALP both propose a specialisation mechanism. In the current definition, the Dynamic Expectancy Model isolates candidate μ-hypotheses which have intermediate corroboration values, and which have a maturity (`hypo_maturity`) value greater than the system defined *maturity threshold* level (Ψ). The use of the maturity criteria ensures that candidate μ-hypotheses have undergone a sufficient number of activations and hence corroborative predictions. Maturity is not equivalent to age.

[25]This cluster of hastily formed guesses contingent on a new phenomena may be related to the "first appearances" effect, widely, but often apocryphally, described. For instance King (1987).

For any of these candidates, which are currently on the active list $\mathcal{H}^*$, and where the confidence measure falls between a system defined *lower confidence bound* (θ) and *upper confidence bound* (Θ), an additional token term is added to the existing context sign "s1". In the current scheme this token is drawn from the record of token activations recorded in the respective activation traces. It is in essence another "guess", but (as with μ-hypothesis creation) one drawn from the population of extant observations. The original μ-hypothesis is retained, and a new one appended to the Hypothesis List. Duplicate μ-hypotheses are not installed by SRS/E. By appending the new, modified, sign "s1" to the Sign List a stream of novel signs is created to further activate the μ-hypothesis creation process.

The experiments described later make extensive use of the μ-hypothesis creation steps, but do not necessitate the use of this specialisation step. It is therefore largely speculative. However the intention is to create a population of μ-hypotheses, which attempts to improve its performance based on predictive ability within the lifespan of the animat. Where the initial μ-hypotheses were created from the simplest combination of parts, new μ-hypotheses will only be created when these minimalist interpretations of the environment are demonstrated inadequate through the corroboration process. Among other candidate approaches to this step in the SRS/E algorithm are the use of the cross-over and mutation techniques employed by *Genetic Algorithms* (GA), and the techniques used by the *machine learning by induction* schools of thought.

Both Becker and Mott also discuss *generalisation*, the converse operation to specialisation. In generalisation terms are removed from the context of ineffective schema on the premise that they contain irrelevant additional kernels which over specify and hence reduce the effectiveness of the μ-hypothesis. The Dynamic Expectancy Model does not provide any explicit mechanism for generalisation. It instead relies on the notion that less effective μ-hypotheses will be removed, after a suitable period of corroboration, by the deletion/forgetting process described below.

The third outcome indicates a candidate for deletion, as it apparently fails in its task as a hypothesis about the environment. The current definition for SRS/E selects a

candidate set of μ-hypotheses for deletion on the basis of their maturity (compared to the *maturity threshold*, Ψ) and confidence values from a sub-set of the population sharing a common consequence sign "s2". A reasonable minimum value for the lower confidence bound (θ, also the minimum bound for specialisation) would be one based on *joint probabilities* (Harrison, 1983):

$$\texttt{joint\_prob} = p(\text{“s1”}) * p(\text{“r1”}) * p(\text{“s2”}) \qquad \text{(eqn. 4-17)}$$

The joint probability value would be that value approximated by a μ-hypothesis created following a true chance or *occult occurrence*. The algorithm's readiness to delete μ-hypotheses must also be related to the number available for predicting "s2". Where only one, or a very limited number of μ-hypotheses are available it appears inappropriate to expunge this knowledge, even where it is demonstrated to be of restricted value. Experimental evidence from *Skinner box* experiments would appear to indicate that experimental animals do not erase operant behaviours even after full extinction, as evidenced by the spontaneous recovery of the extinguished behaviour after a period of rest. It may also be noted that where only a single action elicits reward its use may be particularly persistent during the extinction process.

The fourth outcome offers no information on which to base a decision, and so a pragmatic approach is indicated. In principle an old, untested, μ-hypothesis has no more nor less potential as a valuable item of knowledge than a more recently created one, which has yet to be tested. Where nothing else is known about the outcome there is a clear reason to retain the uncorroborated μ-hypotheses. Where other alternatives already exist, and space is becoming at a premium, a Hypothesis List element falling into this category is a clear candidate for deletion - but as a purely housekeeping consideration.

4.11. The SRS/E Execution Cycle

In the second main part of this chapter the *SRS/E* algorithm is considered in some detail as a series of interrelated computational processes. SRS/E must explicitly balance the demands placed upon it by definitions of innate behaviours provided in the animat's ethogram, goal-initiated behaviours, and by the requirement to

generate new behaviours. Goal-setting, goal-seeking and the learning processes are all defined or controlled as part of the total ethogram. The extent to which the animat can create learned behaviours and the degree to which it can override innate behaviours with learned ones are also defined in the original ethogram. In this way SRS/E can truly be described as implementing a "scheme for learning and behaviour".

4.11.1. Summary of Execution Cycle Steps

Whereas the first part of this chapter described the definition of the various list types and discussed much of the rationale behind various design choices in the construction of the current implementation of SRS/E, this part describes the algorithm primarily from the viewpoint of the manipulations performed on those lists during an individual execution cycle. Figure 4-6 summarises the main steps in each SRS/E cycle. Sub-sections summarise these list manipulations with a degree of formality, utilising the notation developed earlier. The intention of this algorithm is to create a situation where each of the lists is sustained on a continuing basis.

In **step one** the algorithm accepts tokens derived from the animat's sensors and transducers. These are converted to the internal symbol form using information recorded in the Input Token List, and used to evaluate the activation state of all Sign List elements.

In **step two** the Prediction List is inspected for any predictions made in the past which fall due on the current cycle. These predictions are compared with the active Sign List, and the hypotheses making the predictions are updated, for both successful and failed expectations. This is the corroboration and reinforcement of existing μ-hypotheses (from postulates H3 and H4).

In **step three** the algorithm evaluates the Behaviour List to prepare a candidate action and to determine which, if any, innate behaviours or goals are appropriate in the prevailing circumstances. The SRS/E algorithm requires that the Behaviour List provide a priority associated with each candidate activity or goal. When the highest priority activity is greater than the highest priority goal, no goal seeking behaviour is considered and the algorithm skips immediately to step 6 to perform the chosen

action. Whenever step three does not actively select any purposive behaviour or assert a goal a default, exploratory, action will be selected.

Step 1a) *Gather Input Tokens to form* $\mathcal{I}^*$
1b) *Update* $\mathcal{S}^*$
1c) *Cancel satisfied goals from* $\mathcal{G}$
Step 2) *Evaluate past* μ*-experiments from* $\mathcal{P}^t$
Step 3a) *Select default action candidate from* $\mathcal{R}$
3b) *Select innate action and priority from* $\mathcal{B}^{r*}$
3c) *Set goals* $\mathcal{G}$ *and priorities from* $\mathcal{B}^{g*}$
3d) *Innate priority > goal priority?* $\rightarrow$ *to step 6*
Step 4) *Build Dynamic Policy Map (*DPM*) relative to* g^1
Step 5) *Select valenced action from (*DPM $\cup$ $\mathcal{S}^*$*)*
Step 6) *Perform selected candidate action*
Step 7) *Perform* μ*-experiments from* $\mathcal{H}^*$*, update* $\mathcal{P}$
Step 8a) *Novel occurrence?* $\rightarrow$ *create hypothesis on* $\mathcal{H}$
8b) *Unexpected occurrence?* $\rightarrow$ *create hypothesis on* $\mathcal{H}$
8c) *Partially effective hypothesis?* $\rightarrow$ *differentiate to* $\mathcal{H}$
8d) *Ineffective hypothesis?* $\rightarrow$ *delete from* $\mathcal{H}$
Step 9) *To step 1*

Figure 4-6: Summary of Steps in the SRS/E Execution Cycle

In **step four** the algorithm builds (if required) a Dynamic Policy Map. This is performed as a spreading activation graph building algorithm. μ-Hypotheses that are known to lead directly to the top-goal are considered to have a valence level of one, and so define a set of sub-goals (their "s1" component), which in turn act as sub-goals at valence level two, and so on.

In **step five** the algorithm matches the current perceived situation, as expressed by the active Sign List from step one, with the Dynamic Policy Map generated in step four, to select a candidate action to be performed in step six. Step five must also cater for situations where there is no intersection between the current policy map

and any active signs, and for circumstances where the policy map proves ineffective at providing a goal path.

Having defined an action to take, either as a high-priority innate action, a goal directed action selected from the Dynamic Policy Map or a default action, this action is passed to the animat actuators in **step six**.

Once an action is selected, and given the active Sign List from step one, a sub-set of the Hypothesis List will be active, able to make a prediction. Active μ-hypotheses take part in μ-experiments. **Step seven** selects all the active μ-hypotheses and causes them to append their prediction about "s2" onto the Prediction List. A μ-hypotheses does not have to have contributed to the action selected in step six to be considered active (*implicit activation*).

Step eight concerns itself with the management of the Hypothesis List. In keeping with the principles defined in the previous chapter. μ-Hypotheses may be *created*, varied or removed within this step.

Having concluded one cycle (**step nine**), the algorithm returns to step one and begins the next. It might again be noted that SRS/E does not provide for any *terminating condition*, there is nothing inherent in the basic algorithm that concludes the continued execution of cycles.

The base SRS/E algorithm, coupled to any behavioural definitions provided by the originator in the ethogram, is expected to imbue the animat with an appropriate degree of *behavioural autonomy*. The new-born animal or human child may require protection and nurturing, the child may be tutored and educated, but these things do not compromise our notion that they are autonomous and so ultimately self-sufficient. Should the undamaged individual require continued nurture, not achieve a normal degree of self-sufficiency, or be unable to learn without continued tuition, then it might reasonably be concluded that an adequate level of autonomy had not been achieved within the ethogram definition. Similarly the ethogram design may call for a protected maturational period, and as an essentially autonomous learning system the animat may be teachable, but these do not undermine the defining behavioural autonomy properties for the ethogram or animat.

4.12. The SRS/E Algorithm in Detail

Figure 4-7 illustrates the major steps in the SRS/E algorithm, the most significant data pathways and their relationships to the various list structures. Individual steps in the algorithm are described in greater detail in the sections that follow. Steps which read from the list structures are indicated with a solid line termination ("•→"), those which add to a list structure by a "+" indicator ("⊕←"), and those which remove elements from a list by a "-" termination ("⊖←"). Each of the *subsumption points* (SP1 and SP2) indicates a stage in the algorithm where a previously selected candidate action may be replaced (subsumed) by an action of higher priority.

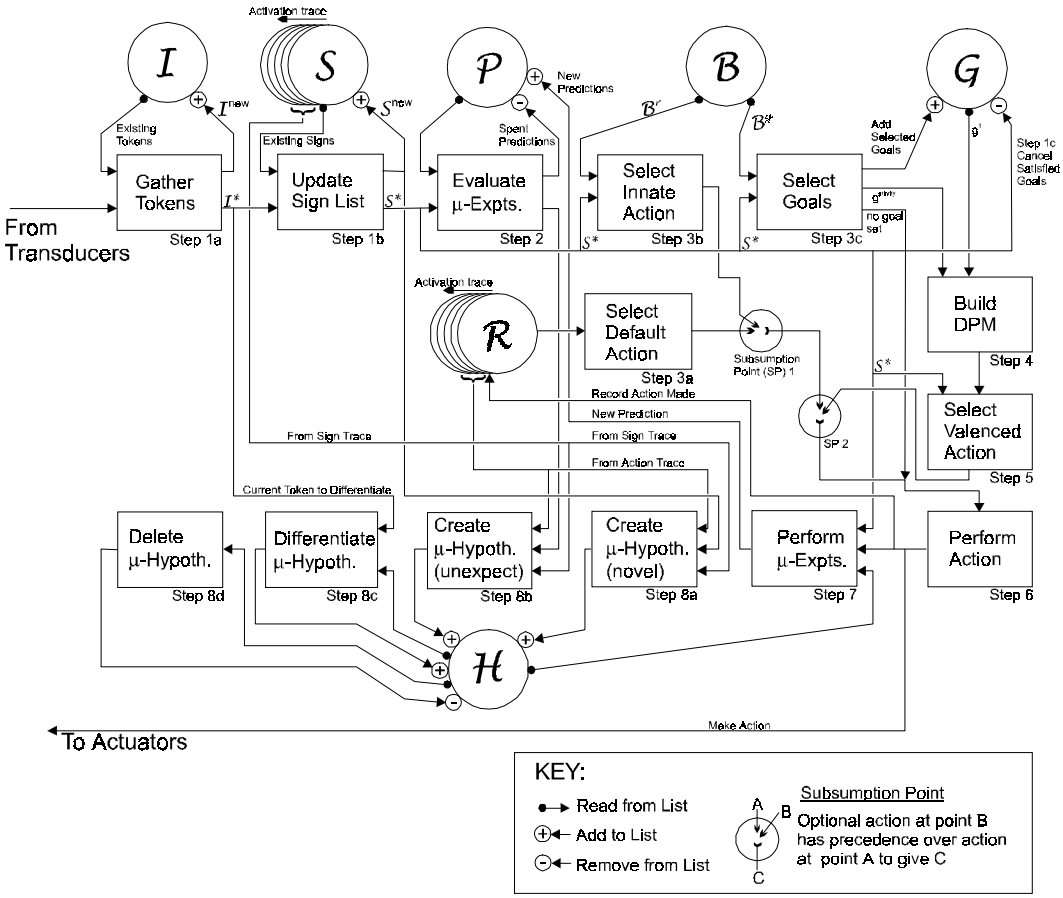

Figure 4-7: The SRS/E Algorithm

4.12.1. Step 1: Processing Input Tokens and Signs

Figure 4-8 shows the list management activities undertaken during **step 1.0** of the SRS/E cycle. In **step 1.1.1**, input token strings are accepted from the input buffer and converted into the internal token form (ι). **Steps 1.1.2** perform additional processing on input tokens previously unknown to the system (i.e., any not already on I). The novel token is appended to I (**step 1.1.2.1**). Additionally a new sign is created from each novel token (**step 1.1.2.2**) and appended to the temporary list S^{new}. Tokens present in the input buffer on the current cycle are assigned to the active Token List, I^*, (**step 1.1.3**). New signs created in step 1.1.2.2 are added to the Sign List (**step 1.2**). The temporary list S^{new} will be used to drive the learning process of step 8.1. Once all input tokens have been processed, each sign is evaluated according to the criteria laid down in equation 4-3, forms 1 through 4. Every sign meeting the criteria defined for activation are placed on the active Sign List S^* (**step 1.3**). **Step 1.4** matches elements on the Goal List (G) to any active signs (S^*), and automatically cancels satisfied goals.

Initialise $S^{new} \leftarrow \{\}$; $I^* \leftarrow \{\}$; $S^* \leftarrow \{\}$;

1.1 Accept tokens into buffer, for each `token_string` do

1.1.1 $\iota \leftarrow I$(`token_string`) [convert input string]

[note: $X(y)$ convert element of type y to element of type X]

1.1.2 if $\iota \notin I$ [a token previously unknown to the system]

1.1.2.1 $I \leftarrow I + \iota$ [append ι to I]

1.1.2.2 $S^{new} \leftarrow S^{new} + S(\iota)$ [create a sign containing ι]

1.1.3 $I^* \leftarrow I^* + \iota$

1.2 $S \leftarrow S + S^{new}$

1.3 For each s where $s \in S$

1.3.1 if (`EvalSignConjunction`(s))

$S^* \leftarrow S^* + s$ [eqn. 4-3]

1.4 $G \leftarrow G - (S^* \cap G)$ [cancel satisfied goals]

Figure 4-8: Step One, Token and Sign Processing

4.12.2. Step 2: Evaluating μ-Experiments on the Basis of Prior Prediction

Once active signs have been determined the algorithm may assess the accuracy of past predictions falling due on the current execution cycle and so update the individual μ-hypotheses responsible for those predictions (figure 4-9). **Steps 2.1** process each element of $\mathcal{P}$ where the `predicted_time` is equal to `now`. Where the `predicted_sign` is on $\mathcal{S}^*$ the μ-hypothesis identified by the Prediction List element `predicting_hypo` is updated according to equations 4-8 and 4-11 (**step 2.1.1.1**). The temporary list $\mathcal{S}^{pred}$ records each sign that was correctly predicted (**step 2.1.1.2**). Similarly **step 2.1.2.1** updates each μ-hypothesis responsible for an incorrect prediction falling due at the current time, according to equations 4-9 and 4-12. For each failed prediction the system variable `rebuildpolicynet` is increased by the amount δ (**step 2.1.2.2**). Spent predictions are removed from $\mathcal{P}$ (**step 2.1.3**). The temporary list $\mathcal{S}^{unexpected}$ records all active signs that were not predicted by any μ-hypothesis (**step 2.3**), these will be used to drive the learning process of step 8.2.

Initialise $\mathcal{S}^{pred} \leftarrow \{\}$;
2.1 for every p ($p \in \mathcal{P}$), such that `predicted_time`(p) = `now`, do
    2.1.1 if `predicted_sign`(p) $\in \mathcal{S}^*$ [prediction succeeds]
        2.1.1.1 Update `predicting_hypo`(p) [according to α, eqn. 4-11]
        2.1.1.2 $\mathcal{S}^{pred} \leftarrow \mathcal{S}^{pred}$ + `predicted_sign`(p)
    2.1.2 if `predicted_sign`(p) $\notin \mathcal{S}^*$ [prediction fails]
        2.1.2.1 Update `predicting_hypo`(p) [according to β , eqn. 4-12]
        2.1.2.2 `rebuildpolicynet` ← `rebuildpolicynet` + δ
    2.1.3 $\mathcal{P} \leftarrow \mathcal{P}$ - p [remove spent prediction]
2.2 $\mathcal{S}^{unexpected} \leftarrow \mathcal{S}^*$ - $\mathcal{S}^{pred}$ [record unpredicted signs]

Figure 4-9: Step Two, Evaluation of μ-Experiments

4.12.3. Step 3: Selecting Innate Behaviours and Setting Goals

The availability of $\mathcal{S}^*$ also allows the Behaviour List, $\mathcal{B}$, to be evaluated (figure 4-10). The default candidate action, `candidate_action`, for this cycle is selected from $\mathcal{R}$ in **step 3.1**. In the present scheme the default candidate action is selected at

random from those available. This forms the trial and error (or other default) action if no other candidate is selected during the current cycle. A list of active behaviours, $\mathcal{B}^{r*}$, is selected from the *primary behaviours* part ($\mathcal{B}^{r}$) of the Behaviour List on the basis of a match between the condition part and the active Sign List $\mathcal{S}^*$ (**step 3.2**). The action with the highest priority is selected from the active primary behaviours ($\mathcal{B}^{r*}$) and assigned to `innate_action` according to the stored `behaviour_priority` values (**step 3.3**). The actual priority of that behaviour is recorded in the variable `innate_priority` (**step 3.4**). If `innate_action` has a higher priority than the *basal level threshold* (ε) it is adopted as the candidate action, `candidate_action`, for the current cycle in preference to the one selected in step 3.1 (**step 3.7**). The Goal List is built from the *goal setting behaviours* part of $\mathcal{B}$ ($\mathcal{B}^{g}$) in **step 3.5**, and the Goal List priority ordered (according to `goal_priority`) in **step 3.6**. SRS/E selects between innate and goal seeking behaviours on each cycle according to the priority of the top-goal, g^1, and the value recorded in `innate_priority` (**step 3.8**). Where an innate behaviour is selected the algorithm skips directly to perform the candidate action in step 6 (**step 3.8.1**).

Initialise $\mathcal{B}^* \leftarrow \{\}$;
3.1 `candidate_action` ← `SelectRandomAction`($\mathcal{R}$)
3.2 for each b where `action`(b) ∈ $\mathcal{B}^{r}$ **AND** `condition`(b) ∈ $\mathcal{S}^*$
    3.2.1 $\mathcal{B}^{r*} \leftarrow \mathcal{B}^{r*} + b$;
3.3 `innate_action` ← `action`(max(`behaviour_priority`($\mathcal{B}^{r*}$))) [innate action]
3.4 `innate_priority` ← max(`behaviour_priority`($\mathcal{B}^{r*}$))
3.5 for each b where `action`(b) ∈ $\mathcal{B}^{g}$ **AND** `condition`(b) ∈ $\mathcal{S}^*$
    3.5.1 $\mathcal{G} \leftarrow \mathcal{G} + b$ [build Goal List]
3.6 $\mathcal{G} \leftarrow$ order(`goal_priority`($\mathcal{G}$)) [order Goal List by priorities]
3.7 if(`innate_priority` > ε) [above basal threshold?]
    3.7.1 `candidate_action` ← `innate_action`
3.8 if(`goal_priority`(g^1) < `innate_priority`) [select goal or innate]
    3.8.1 skip to step 6.0

Figure 4-10: Step Three: Select Innate Actions and Set Goals

4.12.4. Step 4: Building the Dynamic Policy Map

Steps 4.1 determine whether the *Dynamic Policy Map* is to be constructed on this execution cycle. If the goal g^1 is already satisfied, the goal is cancelled (**step 4.1.1**), and the next lower priority goal selected (**step 4.1.2**). If no goal remains on the Goal List control passes directly to step 6.0 (**step 4.2**). If the top-goal is unchanged since the last cycle and the `rebuildpolicynet` value has not exceeded `REBUILDPOLICYTRIP` no change is required and the algorithm skips directly to valenced action selection in step 5.0 (**step 4.3**).

Steps 4.4 (stage 1 of the construction) build the first valence level in the DPM. For all elements (h) of the Hypothesis List where the consequence "s2" is equivalent to g^1 the steps 4.4.n are taken. The estimated cost for the transition is obtained (equation 4-13) and held in $h^{£}$, the cost estimate value for μ-hypothesis h (**step 4.4.1**). The temporary list $S^{v=2}$ is built from the context signs "s1" for μ-hypotheses selected (**step 4.4.2**), these form the sub-goals at the next valence level. The temporary list $H^{£}$ records the estimated policy cost for the μ-hypothesis h as $h^{£}$ (**step 4.4.3**). Similarly the temporary list $S^{£}$ records the lowest cost solution found so far for each sign implicated in the construction of the DPM (**step 4.4.4**). If the context sign "s1" for any instance of h is already on the active Sign List S^{*}, then a path from the current situation to the goal has been found (**step 4.4.5**) and the flag `pathavailable` is set **`TRUE`**. The lowest cost path estimate `bestcost` is updated if the estimated cost of this new path is lower than any previously found solution path from this sign to the top-goal (**step 4.4.6**). Once `pathavailable` is asserted the algorithm might to skip to step 5.0 (i.e., perform the action associated with the element h with the active context sign), or it may continue to build the DPM to discover possible lower cost paths. Were the animat to be constrained to perform an action within a given time this flag is an important indicator that a path exists. The current implementation places no such time constraint on the algorithm.

Steps 4.5-4.8 (stage 2 of the construction) continue the spreading activation process for successive valence levels, `vn+1` (**step 4.5**), until there are no further nodes to expand (**step 4.6**) which terminates the DPM construction. Each node identified as a sub-goal at the previous valence level is expanded (**steps 4.7**) in the manner described for steps 4.4. The temporary list $H^{£}$ records the policy value for

each μ-hypothesis by adding the new cost estimate value for the transition to the previously computed lowest policy value for the sub-goal "s2" (**step 4.7.1**). The temporary list $\mathcal{H}^{£}$ is updated to reflect new policy values (**step 4.7.2**). Whenever a new sign node or a lower estimated policy cost to a sign node is discovered (**step 4.7.3**), the sign is established at the new valence level (**step 4.7.3.1**) and the new or lower cost is recorded (**step 4.7.3.2**). The net effect of this process is to categorise every μ-hypothesis, and so each sign "s1", which is implicated in the DPM by its lowest estimated policy cost to the top-goal. The flag `pathavailable` may be set at any valence level (**step 4.7.4**). The variable `bestcost` is updated whenever a new lowest estimated cost is encountered (**step 4.7.5**). If there is no intersection of sub-goal node and S^*, `pathavailable` remains **`FALSE`** and `bestcost` remains undefined.

Initialise $\mathcal{H}^{£} \leftarrow \{\}$; $\mathcal{S}^{v} \leftarrow \{\}$; $\mathcal{S}^{£} \leftarrow \{\}$;

`rebuildpolicynet` ← 0; `pathavailable` ← **FALSE**;

`bestcost` ← **MAXVALUE**; `vn` ← 1 [valence level one]

Rebuild map if goal changed or 'rebuild' greater than threshold

4.1 while ($g^1 \in \mathcal{S}^*$) [top-goal already satisfied]

4.1.1 $\mathcal{G} \leftarrow \mathcal{G}$ - g^1 [so remove]

4.1.2 $g^1 \leftarrow$ max(`goal_priority`($\mathcal{G}$)) [and select next highest]

4.2 if($\mathcal{G}$ = {}) skip to step 6.0 [no goals on Goal List]

4.3 (if $g^1 = g^{1@t-1}$ **AND** `rebuildpolicynet` < `REBUILDPOLICYTRIP`)

skip to step 5.0 [no need to rebuild DPM]

Stage 1 - create first valence level

4.4 for each h such that `s2`(h) = g^1

4.4.1 $h^{£} \leftarrow$ `GetCostEstimate`(h) [eqn. 4-13]

4.4.2. $\mathcal{S}^{v+1} \leftarrow \mathcal{S}^{v+1}$ + `s1`(h) [record valenced sub-goals]

4.4.3 $\mathcal{H}^{£} \leftarrow \mathcal{H}^{£} + h^{£}$ [cost of transition s1 to goal]

4.4.4 $\mathcal{S}^{£} \leftarrow$ s1($h^{£}$) [record sign cost]

4.4.5 if(`s1`(h) $\in \mathcal{S}^*$)

`pathavailable` ← **TRUE** [path solution found]

4.4.6 if(`bestcost` > $h^{£}$) `bestcost` ← $h^{£}$

Stage 2 - continue spreading activation until done

4.5 `vn` ← `vn` + 1

4.6 if($\mathcal{S}^{v}$ = {}) skip to step 5.0 [expansion complete]

4.7 for each h such that `s2`(h) $\in \mathcal{S}^{v=vn}$ [expand each sub-goal]

4.7.1 $h^{£} \leftarrow$ s2($\mathcal{S}^{£}$) + `GetCostEstimate`(h) [eqn. 4-13]

4.7.2 $\mathcal{H}^{£} \leftarrow \mathcal{H}^{£} + h^{£}$ [record total cost of path]

4.7.3 if(`s1`(h) $\notin \mathcal{S}^{v}$ **OR** s1($h^{£}$) > s1($\mathcal{S}^{£}$)) [new or better path]

4.7.3.1 $\mathcal{S}^{v+1} \leftarrow \mathcal{S}^{v+1}$ + `s1`(h) [new sub-goals]

4.7.3.2 $\mathcal{S}^{£} \leftarrow \mathcal{S}^{£}$ + s1($h^{£}$) [record lower sign cost]

4.7.4 if(`s1`(h) $\in \mathcal{S}^*$)

`pathavailable` ← **TRUE** [solution path found]

4.7.5 if(`bestcost` > $h^{£}$) `bestcost` ← $h^{£}$

4.8 return to step 4.5 [expand next valence level]

Figure 4-11: Step Four, Construct Dynamic Policy Map

4.12.5. Step 5: Selecting a Valenced Action

Steps 5 (figure 4-12) determine whether a valenced action is appropriate, and if so select the action. These steps also manage the *goal extinction* process. A value for the *valence break point* is determined first. If VBP is already set, this value is used (**step 5.1**). Where this is the first instance of a DPM, or the previous valence break point has been exceeded, a new value for VBP is computed according to equation 4-15 (**step 5.3**). The valence break point is cleared if no path is found (**step 5.2**). A temporary list of μ-hypotheses, $\mathcal{H}^{\#£}$, is formed from the intersection of those μ-hypotheses with valence (recorded on $\mathcal{H}^{£}$) and whose condition part "s1" is on the active Sign List $\mathcal{S}^*$ (**step 5.4**). The candidate valenced action, valenced_action, is extracted from the element of $\mathcal{H}^{\#£}$ with the lowest estimated policy cost to the goal (**step 5.5**). If the estimated cost of this proposed action is still less than VBP, this valenced action is selected as the overall candidate action, candidate_action, for the execution cycle (**step 5.7**). Where there is no intersection of valenced μ-hypotheses and the active Sign List, the candidate action selected in step 3 will be used. This summary of the algorithm does not detail the sub-steps for the *goal recovery mechanism* previously described. **Step 5.8** determines if the total estimated cost of the path has exceeded the *goal cancellation level*, Ω, and if so removes the current top-goal from $\mathcal{G}$.

5.1 VBP ← GetValenceBreakPoint() [establish VBP]
5.2 if (pathavailable = **FALSE**) VBP ← 0 [no path to goal]
5.3 else if (VBP ≤ 0 **OR** VBP > bestcost) [compute VBP]
VBP ← bestcost * VALENCEBREAKPOINTFACTOR
5.4 $\mathcal{H}^{\#£} \leftarrow \mathcal{H}^{£} \cap (\text{s1}(\mathcal{h}) \in \mathcal{S}^*)$ [candidate active signs]
5.5 $\mathcal{h} \leftarrow \min(\mathcal{H}^{\#£})$ [select least policy cost]
5.6 valenced_action ← r1($\mathcal{h}$)
5.7 if(policy_value($\mathcal{h}$) ≤ VBP) [break-point reached?]
candidate_action ← valenced_action [no, use valenced action]
5.8 if(policy_value($\mathcal{h}$) > Ω) [goal cancellation level?]
5.8.1 $\mathcal{G} \leftarrow \mathcal{G} - g^1$ [so cancel top-goal]

Figure 4-12: Step Five, Select Valenced Action

4.12.6. Step 6: Performing an Action

Figure 4-13 describes the action reification process. The action `candidate_action`, selected either as an innate response from the Behaviour List $\mathcal{B}^g$ (step 3.3), from the Dynamic Policy Map as a valenced action (step 5.7), or as a default trial and error action (step 3.1) is sent to the animat's effectors to be performed on the current cycle (**step 6.1**). The element of the Response List finally selected is recorded on the active Response List $\mathcal{R}^*$ for the current execution cycle (**step 6.2**).

6.1 `DoAction(candidate_action)` [reify candidate action]
6.2 $\mathcal{R}^* \leftarrow$ `candidate_action` [record in trace]

Figure 4-13: Step Six, Perform Action

4.12.7. Step 7: Conducting μ-Experiments

Figure 4-14 describes the steps taken to create the predictive expectations. The active Hypothesis List $\mathcal{H}^*$ is constructed from every μ-hypothesis where the context sign "s1" appears on the active Sign List $\mathcal{S}^*$ and the action "r1" appears on the active Response List $\mathcal{R}^*$ (**step 7.1.1**). SRS/E does not distinguish between actions made as part of the goal seeking process and those made due to innate behaviour definitions or for any other reason. As a consequence SRS/E corroborates μ-hypotheses whenever they establish an expectation. Such expectations are added to the Prediction List as a triple recording the μ-hypothesis responsible for the prediction, the predicted sign, the time at which that sign is predicted (**step 7.1.2**). The value *t* is recovered from the `time_shift` value associated with the μ-hypothesis. These predictions will be corroborated in step 2 of later execution cycles.

initialise $\mathcal{H}^* \leftarrow \{\}$;
7.1 for all h, such that s1(h) $\in \mathcal{S}^*$ **AND** r1(h) $\in \mathcal{R}^*$
    7.1.1 $\mathcal{H}^* \leftarrow \mathcal{H}^* + h$ [record activation]
    7.1.2 $\mathcal{P} \leftarrow \mathcal{P} + \mathcal{P}(h,$ s2(h), now + t) [make prediction]

Figure 4-14: Step Seven: Conduct μ-Experiments

4.12.8. Step 8: Hypothesis Creation and Management

Steps 8.1 (figure 4-15) are concerned with the creation of new μ-hypotheses when a *novel event* is detected. These steps are triggered when the temporary list $\mathcal{S}^{new}$ is not empty. Elements were placed on $\mathcal{S}^{new}$ in step 1.2. A new μ-hypothesis is created from a context sign ("s1") selected from the Sign List activation trace record (**step 8.1.2**), from an action ("r1") selected from the Response List activation trace (**step 8.1.3**), and the novel sign extracted from $\mathcal{S}^{new}$ ("s2"), (**step 8.1.4**). The newly formulated μ-hypothesis is added to the Hypothesis List (**step 8.1.5**) and its values set to reflect the *creation bonus* previously described. As the μ-hypothesis is created from a novel sign, there is no possibility that it will duplicate an existing μ-hypothesis. The *timebase shift* is achieved by predicting the occurrence of "s2" n cycles in the future, where the "s1" and "r1" values were previously extracted from the respective activation traces n cycles in the past. The relative time shift, +*t*, is recorded in the μ-hypothesis `time_shift` value.

The creation of a new μ-hypothesis may affect the structure of the DPM, and so the system value `rebuildpolicynet` is incremented by Δ to hasten or trigger a DPM rebuild (**step 8.1.6**). The novel sign is removed from $\mathcal{S}^{new}$ (**step 8.1.7**), and steps 8.1 repeated until this list is empty. An explicit sampling learning strategy is implemented by omitting steps 8.1.2 to 8.1.6 for one or more of the signs on $\mathcal{S}^{new}$ according to a frequency set by the *learning probability rate*. The learning probability rate will also be referred to by the abbreviation *Lprob* and by the symbol (λ). When the learning probability rate is 1.0 every opportunity to create a μ-hypothesis will be used, if it were set to 0.0 no μ-hypothesis creation would occur. In electing to implement a *sampling strategy* at this point any sign passed over will only seed a new μ-hypothesis as a result of the process described in steps 8.2, as it will not reappear on $\mathcal{S}^{new}$.

Steps 8.2 create new μ-hypotheses when unexpected signs are detected. Elements were added to the temporary list $\mathcal{S}^{unexpected}$ in step 2.2. The basic mechanism for μ-hypothesis creation is identical to that described in steps 8.1. In a sampling strategy ($\lambda < 1.0$) passed over signs can reappear on $\mathcal{S}^{unexpected}$ again (as they may remain unpredicted), and so be the subject of this process on a subsequent execution cycle.

Creation on the basis of novelty

8.1 for each $\mathcal{s}^{new}$ such that ($\mathcal{S}^{new} \neq \{\}$ **AND** $\mathcal{s}^{new} \in \mathcal{S}^{new}$)

8.1.1 if (rand(0.0 .. 1.0) > λ) skip to step 8.1.7

8.1.2 `s1` ← `Select`($\mathcal{s}^{x} \in \mathcal{S}^{*@-t}$)

8.1.3 `r1` ← `Select`($\mathcal{r}^{x} \in \mathcal{R}^{*@-t}$)

8.1.4 `s2` ← $\mathcal{s}^{new}$

8.1.5 $\mathcal{H} \leftarrow \mathcal{H} + \mathcal{H}$(`s1`,`r1`,`s2`$^{@+t}$), where `s1` ≠ `s2`

8.1.6 `rebuildpolicynet` ← `rebuildpolicynet` + Δ

8.1.7 $\mathcal{S}^{new} \leftarrow \mathcal{S}^{new} - \mathcal{s}^{new}$

Creation on the basis of unpredicted event

8.2 for each $\mathcal{s}^{unexpected}$ such that ($\mathcal{S}^{unexpected} \neq \{\}$ **AND** $\mathcal{s}^{unexpected} \in \mathcal{S}^{unexpected}$)

8.2.1 if (rand(0.0 .. 1.0) > λ) skip to step 8.2.7

8.2.2 `s1` ← `Select`($\mathcal{s}^{x} \in \mathcal{S}^{*@-t}$)

8.2.3 `r1` ← `Select`($\mathcal{r}^{x} \in \mathcal{R}^{*@-t}$)

8.2.4 `s2` ← $\mathcal{s}^{unexpected}$

8.2.5 $\mathcal{H} \leftarrow \mathcal{H} + \mathcal{H}$(`s1`,`r1`,`s2`$^{@+t}$), where `s1` ≠ `s2`

8.2.6 `rebuildpolicynet` ← `rebuildpolicynet` + Δ

8.2.7 $\mathcal{S}^{unexpected} \leftarrow \mathcal{S}^{unexpected} - \mathcal{s}^{unexpected}$

Figure 4-15: Step Eight, Hypothesis Creation

Steps 8.3 (figure 4-16) describe the *specialisation* process by which individual μ-hypotheses are made more specific in their application. Extra specificity is achieved by adding discriminant terms to the context sign conjunction ("s1"). The current definition selects μ-hypotheses that are: (1) active; (2) exceed the *maturity threshold* (Ψ), in that they have been tested many times; and (3) have an indeterminate confidence probability values (`hypo_prob`, or `bpos`) falling between the *upper* (Θ) and *lower* (θ) *confidence bounds*. A selected μ-hypothesis must be active to ensure that the additional elements added to the conjunction are drawn from the set of extant events at the time of modification (i.e., those falling within the range defined by the respective activation traces).

A new context sign is created by adding an additional term to the existing context sign conjunction (**step 8.3.1**). This new term may be drawn from the Input Token List, the Sign List, the Response List or the Hypothesis List. It may take any of the four forms described in equation 4-3. Action (**step 8.3.2**) and consequence (**step 8.3.3**) parts are copied from the existing μ-hypothesis. The new μ-hypothesis is appended to the Hypothesis List (**step 8.3.4**). The original μ-hypothesis is not removed, and will compete with the new one. The new sign created in step 8.3.3 is appended to the Sign List (**step 8.3.5**). On its first subsequent activation the new sign will appear as a candidate on $\mathcal{S}^{\text{unexpected}}$, as there is no μ-hypothesis to predict it. This mechanism therefore provides a continuing source of new signs to drive the learning process indefinitely.

Specialisation (differentiation)

8.3 for all h, such that $h \in \mathcal{H}^*$ **AND** `hypo_maturity`(h) > Ψ
AND `hypo_prob`(h) > θ **AND** `hypo_prob`(h) < Θ

8.3.1 `s1` ← $\mathcal{S}$(`s1`(h) + $x^{@-t}$) [differentiate s1]

8.3.2 `r1` ← `r1`(h) [copy action]

8.3.3 `s2` ← `s2`(h) [copy s2]

8.3.4 $\mathcal{H}$ ← $\mathcal{H}$ + $\mathcal{H}$(`s1`,`r1`,`s2`$^{@+t}$) [install new μ-hypothesis]

8.3.5 $\mathcal{S}$ ← $\mathcal{S}$ + `s1` [install new sign in $\mathcal{S}$]

8.3.6 `rebuildpolicynet` ← `rebuildpolicynet` + Δ

Figure 4-16: Step Eight, Hypothesis Management - Specialisation

Step 8.4 (figure 4-17) defines the criteria for μ-hypothesis deletion. μ-Hypotheses that persistently fail to make effective predictions may be removed. The degree of maturity should be high and the corroboration measures should indicate that the μ-hypothesis has little or no predictive value. μ-Hypotheses are deleted by simply removing them from the Hypothesis List (**Step 8.6**).

Deletion (forgetting) under competition

initialise $\mathcal{H}^{\#} \leftarrow \{\}$;

8.4 for all h, such that $h \in \mathcal{H}^{*}$ **AND** `hypo_maturity`(h) > Ψ

AND `hypo_prob`(h) < Θ

8.4.1 $\mathcal{H}^{\#} \leftarrow \mathcal{H}^{\#} + h$ [build candidate list]

8.5 $h^{delete} \leftarrow$ min(`hypo_prob`($\mathcal{H}^{\#}$)) [select a deletion candidate]

8.6 $\mathcal{H} \leftarrow \mathcal{H} - h^{delete}$ [update Hypothesis List]

8.7 `rebuildpolicynet` ← `rebuildpolicynet` + Δ

Figure 4-17: Step Eight, Hypothesis Management - Forgetting

4.13. Implementation

The SRS/E algorithm is implemented in Microsoft *Visual C++* and runs as a text only Window under *Microsoft Windows* v3.1 or Windows 95. Each of the major lists and their associated functions are defined as object classes. The use of the term "list" here does not imply the use of a list processing language such as *LISP*. Elements of these Lists are allocated and reallocated dynamically, typically stored and indexed as array members. In the interests of efficiency this implementation eschews conventional object oriented message passing in favour of cross-class access functions.

4.14. SRS/E - A Computer Based Expectancy Model

In this chapter the *Dynamic Expectancy Model* developed in chapter three of this thesis has been translated into a single algorithm, SRS/E. MacCorquodale and Meehl (1953) recognised that their expectancy theory postulates were "*incomplete, tentative and nonsufficient*". Becker's JCM was only presented as a proposal for implementation. Mott achieved a substantive implementation of ALP, but was heavily constrained by the timesharing technology available at the time, and by the generally impoverished nature of the robot interface he employed. Drescher provides scant indication of the results for his claimed implementation, beyond an indication of the extensive computational resources required to sustain the marginal attribution process.

The SRS/E algorithm, and its implementation, stands as a "proof by existence", a working model created from the postulates presented in chapter three. The SRS/E algorithm claims to be "sufficient" in this respect, and as an implementation at least one step beyond "tentative". Each of the postulates contributes a small component part of the whole. The processes are less tightly coupled than, say, Watkins' *Q*-learning; a repetitive application of a simple reinforcement transfer rule. More tightly coupled than, say, the idea implicit in Minsky's (1985) notion of a "society of mind". The relatively large number of Dynamic Expectancy Model postulates, and so algorithmic steps, reflects the apparent need to construct a balanced and functional mechanism; in much the same manner as an automobile design requires many coupled systems to achieve an acceptable level of usability, safety, reliability, maintainability and performance. It may be that further work will demonstrate that the system is still overspecified, and elements may be deleted without affecting overall functionality.

Yet SRS/E does not claim completeness. There is still a substantial "back-catalogue" of published research describing a huge range of phenomena that must eventually be explained or incorporated into a larger single model of the animat. In keeping with an idea that evolution adds capabilities to the best of previous generations and proto-typical species it seems inevitable that extra postulates, rather than simplification, will be found necessary.

The next chapter describes an experimental environment to investigate the properties of the SRS/E algorithm as implemented.