9. Appendix One

This appendix collates all the stages in a single execution cycle of the SRS/E algorithm, as previously described in Chapter 4.

1.0 Gather Tokens and Update Sign-list

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

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

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

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

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

1.1.2.1 $I \leftarrow I + i$ [append i to I]

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

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

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]

2.0 Evaluate μ-Experiments on Basis of Prior Prediction

Initialise $S^{pred} \leftarrow \{\}$;

2.1 for every p ($p \in P$), such that `predicted_time`(p) = `now`, do

2.1.1 if `predicted_sign`(p) $\in S^*$ [prediction succeeds]

2.1.1.1 Update `predicting_hypo`(p) [according to α, eqn. 4-11]

2.1.1.2 $S^{pred} \leftarrow S^{pred}$ + `predicted_sign`(p)

2.1.2 if `predicted_sign`(p) $\notin S^*$ [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 $P \leftarrow P - p$ [remove spent prediction]

2.2 $S^{unexpected} \leftarrow S^* - S^{pred}$ [record unpredicted signs]

3.0 Select Innate Action and Set Goals

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) ∈ 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) ∈ S^*

3.5.1 $G \leftarrow G + b$ [build Goal List]

3.6 $G \leftarrow$ order(`goal_priority`(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

4.0 Build (re-build) Dynamic Policy Map (`Hypo::BuildPolicyNet()`)

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

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

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

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

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

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

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

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

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

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

4.4.5 if(`s1`(h) ∈ 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) ∈ $\mathcal{S}^{v=vn}$ [expand each sub-goal]

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

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

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

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

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

4.7.4 if(s1(h) ∈ $\mathcal{S}^{*}$)

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

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

4.8 return to step 4.5 [expand next valence level]

5.0 Select Valenced Action (Hypo::SelectValencedAction()**)**

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 $\mathcal{H}^{\#£}$ ← $\mathcal{H}^{£}$ ∩ (s1(h) ∈ $\mathcal{S}^{*}$) [candidate active signs]

5.5 h ← min($\mathcal{H}^{\#£}$) [select least policy cost]

5.6 valenced_action ← r1(h)

5.7 if(policy_value(h) ≤ VBP) [break-point reached?]

candidate_action ← valenced_action [no, use valenced action]

5.8 if(policy_value(h) > Ω) [goal cancellation level?]

5.8.1 $\mathcal{G}$ ← $\mathcal{G}$ - g^{1} [so cancel top-goal]

6.0 Perform Action

6.1 DoAction(candidate_action) [reify candidate action]

6.2 $\mathcal{R}^{*}$ ← candidate_action [record in trace]

7.0 Conduct μ-Experiments (Hypo::EvaluateHypotheses()**)**

initialise $\mathcal{H}^{*}$ ← {};

7.1 for all h, such that s1(h) ∈ $\mathcal{S}^{*}$ **AND** r1(h) ∈ $\mathcal{R}^{*}$

7.1.1 $\mathcal{H}^{*}$ ← $\mathcal{H}^{*}$ + h [record activation]

7.1.2 $\mathcal{P}$ ← $\mathcal{P}$ + $\mathcal{P}$(h, s2(h), now + *t*) [make prediction]

8.0 Hypothesis Management (`Hypo::NewHypo()`)

Creation on the basis of novelty

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

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

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

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

8.1.4 `s2` ← 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 $S^{new} \leftarrow S^{new} - s^{new}$

Creation on the basis of unpredicted event

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

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

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

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

8.2.4 `s2` ← $s^{unexpected}$

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

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

8.3.5 $S \leftarrow S$ + `s1` [install new sign in S]

8.3.6 `rebuildpolicynet` ← `rebuildpolicynet` + Δ

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

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

8.7 `rebuildpolicynet` ← `rebuildpolicynet` + Δ

9.0 Return to step 1