

## Re-entrant PP

### Abstract

A re-entrant scheme is proposed to give the PURR-PUSS learning system greater sequential strength at low memory cost. An example of its operation in learning Nursery Rhymes provides evidence of the feasibility of the scheme.

### Introduction

The PURR-PUSS system, which will be abbreviated to PP, is a brain for a robot and a theory of human learning. PP was described in detail in Andrae(1998). It is a working system that can be implemented in a standard computer, but is designed for parallel operation in dedicated hardware.

The structure of PP originated from a desire to produce a learning system with “sequential strength”, largely because of the demands of language learning. This was formulated as the problem of learning a finite state machine, and later a Turing Machine. It was also influenced strongly by Newell & Simon’s (1972) production system theory of human problem solving. PP fits comfortably in the framework of Reinforcement Learning (Sutton and Barto, 1998), but is less mathematically tractable because of a novelty mechanism. Here, I will use the fourth source of the PP structure, a simplified view of the cerebral cortex of the human brain, because it points strongly to the need for re-entrant features, as argued by Gerald Edelman since 1978 (Edelman & Mountcastle, 1978; Edelman, 1992; Edelman & Tononi, 2000).

I assume that the cortex consists of a large number of (say 100) distinct areas, “cortical areas”, to which bundles of “afferent” (input) neurons converge and from which other bundles of “efferent” (output) neurons leave. The efferent neurons are assumed to convey a conjunction (AND) of the activity of the afferent neurons. Cortical areas differ in the sources of their afferent neurons, which may be outside the cortex (e.g. sensors or other parts of the brain, such as reward centres) or other cortical areas. These assumptions are made specific in the PP structure by defining cortical areas, “event-types” and “events”. It is assumed that each neuron carries an event of the type appropriate to its bundle.

Time advances in steps, the current step being “now”. Bundles of neurons can be delayed by none, one or more steps, so the minimal definition of a cortical area (CA) looks like this with afferent neuron bundles to the left of the “>>” sign and efferent neuron bundles to the right of it:

[CA: event-type/delay, event-type/delay, ... >> event-type(s)]

As an illustrative example, imagine that a robot sings nursery rhymes described by two event-types, “note” and “beat”. By “note” I mean a note of a musical scale such as middle C, D, E, .., while beat includes note-length (crotchet, quaver etc) and stress marking three-time (e.g. waltz time) or four-time (e.g. march time). Our imaginary robot might have its first cortical area, CA-1, defined by

{CA-1: note-3, beat-3, note-2, beat-2, note-1, beat-1 >> note, beat}

In words, CA-1 says that the current note and beat “can be obtained from” or “can be predicted by” the conjunction of (AND) the note and beat in the previous step, the note and beat in the step before that, and the note and beat in the step before that. The “-1” in “note-1” and “beat-1” should be interpreted as meaning one step before the current step. Similarly, “-2” and “-3” mean 2 and 3 steps before the current step. The six event-types to the left of the >> sign describe the “context” of the CA.

The nursery rhyme “Baa, Baa, Black Sheep” begins

		Abbreviation	Word
Step 1.	note C, crotchet, stressed;	CcrS	Baa
Step 2.	note C, crotchet, unstressed;	Ccr	Baa
Step 3.	note G, crotchet, unstressed;	Gcr	Black
Step 4.	note G, crotchet, unstressed;	Gcr	Sheep
Step 5.	note A, quaver, stressed;	AquS	Have
Step 6.	note B, quaver, unstressed;	Bqu	You
Step 7.	note C (octave), quaver, unstressed;	C'qu	An-
Step 8.	note A, quaver, unstressed;	Aqu	-y
...	... ..	...	...

Notice that each step is a new note, so the steps are not of equal duration. This is just a choice made here for simplicity. After step 4 we can store in the memory of the robot brain the first instance of a CA-1, using the abbreviations listed in the 3<sup>rd</sup> column:

[1CA-1: CcrS Ccr Gcr >> Gcr]

If the sequence “CcrS Ccr Gcr” reappears later on, then the “association” 1CA-1 will suggest that the next event is going to be “Gcr”. However, if on that occasion the next event is “Fqu”, 1CA-1 will acquire a second “associated event”:

[1CA-1: CcrS Ccr Gcr >> Gcr, Fqu]

and from then on this instance of a CA will predict the alternatives “Gcr” and “Fqu”.

Another addition to the terminology leads to more compact descriptions. An instance of the context of a CA will be called a “node”. This will enable me to say the 1CA-1 node “CcrS Ccr Gcr” has 2 associated events. In the full PP system, a variety of flags, numbers and pointers are attached to nodes, but these are not needed here.

My book *Associated Learning for a Robot Intelligence* (Andreae, 1998) gave many examples of the power and versatility of small groups of Cortical Areas, including the mimicking of a Universal Turing Machine. One of the ideas that was played with there, but not usefully, was the extension of the notion of “event” to “node event”, that is an event that is itself the context or node of a CA. (By the way, CAs were called templates in that book.). This idea was called “Contexts of contexts” but I failed to see how to make it work until February of this year (2005). Before that I was taking the context events from different CAs, which had the effect of broadening the context. This was of no advantage because it could be achieved by using a larger CA in the first place. However, if the node event in a CA is taken to be the previous instance of the **same** CA, we produce a CA of too much sequential strength! Indeed, if there was memory space for 10,000 nodes of the CA it would generate a sequence of that length before “forgetting” started to break up the sequence. Also, it would be useless because it couldn’t use repetitions of sequences for predicting future behaviour. So far, I have used two ways of limiting the sequential strength of one of these “re-entrant” CAs. (Re-entrant because a CA contains a previous instance of itself.) The first way is by “hashing” the number of the CA node to a much smaller number, and the second is by “zeroing” the number when there is a “pause” situation. The next section reports an example of these two processes being used together.

Hashing is a random, but fixed, transform which generates a number in a specified range for each node (instance of the context) of a CA. The method I will use is the division of the node number by a prime number, taking the remainder. (In other words, expressing the node modulo the prime.) A big diagnostic advantage of this simple method of hashing is that it is easy to check results.

In the example described in the next section, two CAs are used, each with a different prime, 11 for the first and 13 for the second. The amount of memory used by several re-entrant CAs will be proportional to the **sum** of the primes, while the resolution (inverse probability of ambiguity) of the combined CAs will be proportional to the **product** of the primes. This means that there is an enormous advantage in having several CAs operating in conjunction, a discovery which promises to make the re-entrant CAs, and therefore re-entrant PP, significantly more powerful than ordinary CAs and PP.

### Nursery Rhymes Example

It is a remarkable fact that young children can learn a large number of nursery rhymes without confusing them, even though the tunes have a considerable amount of similarity. Consider the 71 tunes in a book called *My First Sing-A-Song Book* (Lloyd, 1966) as a test for a robot brain capable of learning sequences. To avoid irrelevant complexity (and to avoid giving PP extra clues!), each of the tunes has been transposed to the key of C and the notes have been coded for pitch, stress and duration. In each tune, the length of the main beat is 8, so in 3-time there will be the equivalent of three length-8 beats, while in

4-time, there will be the equivalent of four length-8 beats. The tunes are presented in random order to the robot and between each pair of tunes there is a sequence of pauses during which the robot can take over the “singing”. We look at one of the attempts of the PP robot to reproduce the tunes.

To remove any effects of human interference, the set-up for the experiment has two robots, robot A having the learning brain and robot B being programmed to behave in a certain way which includes the singing of songs. I will frequently refer to robot A as PP. Both robots are programmed to wait (do pause actions) while the other robot sings until the other robot signals that it is finished. The default action for PP, when it can no longer choose a singing action, is a “hand-over pause”.

PP is given 2 cortical areas, SICA and SICB, defined as follows:

{SICA: note&beat-2 note&beat-1 SICA-1 >> note&beat}          hashing prime = 11

{SICB: note&beat-2 note&beat-1 SICB-1 >> note&beat}          hashing prime = 13

Note. These definitions are exactly equivalent to the CAs used in the simulation, but omit complications arising from the fact that the system was developed for a more complex interaction of robots. Specifically, I will be describing the system as though only robot actions are involved, while in the actual simulation stimuli were used as well. It can be seen that here the note and beat are combined in a single event-type note&beat, instead of keeping them separate as in CA-1.

A simulation was run for 8,000 steps and then the memory of PP was dumped to give us a record of all the CA nodes (instances) stored. During the simulation, as a result of the random choice of song, 47 of the 71 songs were sung by robot B, a few as many as 3 times. On looking at the behaviour of robot A, I found a sequence of 93 notes starting at step 7813 during which robot A reproduced song number 14, which was OLD KING COLE. Let me call this sequence List 8. I could have chosen many other sequences, but this was the one nearest the dump. The coding used for a note&beat is given in Appendix-1.

Robot B sang OLD KING COLE from step 7719 to step 7811 and this was the first time, so it was novel to PP. PP starts on step 7813 with the introductory pauses given by robot B at the beginning of 3-time and 4-time songs. On step 7817, PP chooses the first note of song 14 from 18 possibles. It then proceeds to sing song 14 accurately, with an unambiguous choice of note on all but 6 of the remaining 87 notes. In each of the six cases, it chooses the appropriate note because of the increased worth of novel events.

Note. In the following, I will refer to the first two event-types of SICA and SICB, namely note&beat-2 and note&beat-1, as the “fixed context”, the remaining re-entrant event-type as the “node event-type”, SICAnode or SICBnode.

There are a number of ways of showing the effect of the node events. If the re-entrant node events had not been included in SICA and SICB, all 331 2-note matches between List 8 and the other 46 songs heard by PP would offer a possible transfer from song 14 to another song. This is convincing evidence that the re-entrant events increase the sequential strength of the CAs beyond the 2 events of the fixed context. That there are 22 3-note matches between List 8 and the other 46 songs demonstrates a sequential strength greater than 3 notes. The work being done by the node events is even more clear when we see the details (Figure 1) of how they deal with the 2-note match between (for example) notes 16 and 17 of List 8 and 35 places in the other songs having the same pair of notes. The probability of two entries in Figure 1 having the same pair of node events is only  $(1/11)*(1/13)$  or  $1/143$ , but more entries in the 36 rows of Figure 1 have the same pairs of node events than we should expect. To explain this, we need to return to the “zeroing” of nodes, which was mentioned briefly in the Introduction.

We would like identical sentences, behaviours, procedures and other complete sequences to be recognized as the same when they recur. For this to happen, there needs to be a resetting of parameters in the pause before a sentence, behaviour or procedure begins. This is achieved in Re-entrant PP by “zeroing” the node events during a pause. The rule, which is carried out automatically by PP is that if the last two events are pauses, the node events (SICAnode and SICBnode) being stored are each given the value “2”. This value is allowed because the hashed values have 3 added to them, leaving the value “2” for this purpose and the values “0” and “1” for general housekeeping. All of the six entries in Figure 1 with the node events 4 and 5 come from the beginning of songs just after zero node events have been stored. There has not been time for the various sequences to be distinguished, as they all begin with the same two note&beats.

Two of the entries in Figure 1 with node events 9 and 8, and two with node events 7 and 6 are from adjacent segments of the same 29-note sequence in song number 27, ON THE BRIDGE, which is repeated as a chorus. It is only by chance that the two occurrences of this sequence are not distinguished by the node events, but non-re-entrant CAs would be unable to distinguish them without having a context of length 29 notes! Two of the other three entries with these same pairs of node events are from the same repetitive song in which the context in question occurs 12 times. Anyway, if we assume that the 36 entries in the table are independent of each other, then we can use the Binomial Distribution to show the probability of clashes. This is done in the Appendix-2, where it is shown that there is a 78% probability that the set {9,13} for SICAnode and SICBnode of note 18 in List 8 will not be seen among the 35 other sets, and it is not. It is also shown that there are likely to be about 3 pairs of associations among the 35 that share the same sets. These predictions assume random assignment of the sets, so the associations with set {4,5} can be disregarded because they are controlled by the zeroing process. There is certainly correlation between the two sets {7,6} and {9,8} because they occur in song 27, so the 9 associations sharing 3 sets {7,6}, {8,6} and {9,8} are not too far away from the prediction of 3 pairs sharing the same sets.

---

---

Note/Song,	Note&Beat (coded)	SICAnode	SICBnode
18/List 8:	1002D0	9	13
06/7	80248	4	5
19/7	80248	3	6
33/7	80248	12	7
38/7	100250	5	6
53/7	80248	7	6
04/8	480408	4	5
33/8	480408	7	14
08/21	80248	6	14
16/21	80248	8	6
59/25	100550	6	11
04/27	480248	4	5
06/27	4802C8	9	8
19/27	480248	7	6
21/27	4802C8	9	8
34/27	480248	12	10
36/27	5002D0	3	12
41/27	80248	5	14
43/27	5002D0	6	15
47/27	480248	8	6
49/27	4802C8	9	8
62/27	480248	7	6
64/27	4802C8	9	8
29/29	4802C8	13	11
05/37	80248	4	5
12/39	4802C8	7	4
06/60	80248	4	5
18/60	80248	11	9
43/60	80248	9	4
06/65	802C8	4	5
14/65	802C8	12	11
21/65	802C8	5	3
06/69	480108	12	5
10/69	5002D0	5	9
22/69	480108	6	8
26/69	880008	9	12

For note 18 of List 8, the fixed context (coded) is "480248 80248" or "CcrS Ccr". These two note&beats occur just before each note of a song given in the lefthand column. Node events used to store the note&beat following this context are shown in columns 3 and 4. The node event values are "modulo prime + 3", so the SICAnode values range from 3 to 13 for prime = 11, and the SICBnode values from 3 to 15 for prime = 13.

**Figure 1**

The song number 14, OLD KING COLE, which the re-entrant-PP reproduces, has a sequence of 18 notes that are repeated later in the song. These two occurrences of the

sequence were distinguished in the reproduction, something that a standard PP couldn't do without having CAs covering the length of the sequence.

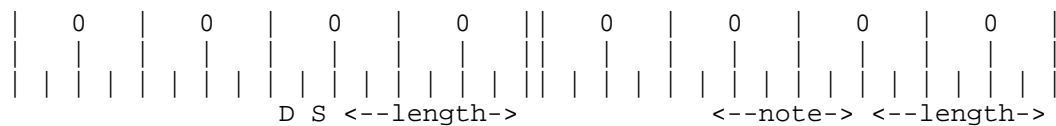
If the CA with a fixed context of 2 events takes M units of memory, then each additional value of node event will require another M units of memory. That is, roughly, the memory required for a CA is proportional to the number of values available to the node event. Using the prime method, the memory required for the CA will be M x prime. With 2 CAs, the memory will be M x prime1 + M x prime2. In general, the memory required by several CAs will be proportional to the **sum** of the prime numbers. However, we have already seen that the ability of the CAs to distinguish sequences depends on the **product** of the primes, 143 in the case of the 2 primes 11 and 13. It follows, that it is preferable to use several CAs with small hashing primes, rather than one CA with a large hashing prime. Of course, we must ensure that all decisions by the CAs are unanimous.

## References

- Andrae, J.H. (1998): *Associative Learning for a Robot Intelligence*. Imperial College Press.  
 Edelman, G.M. & Mountcastle, V.B. (1978): *The Mindful Brain*. MIT Press.  
 Edelman, G.M. (1992): *Bright Air, Brilliant Fire*. Penguin Books.  
 Edelman, G.M. & Tononi, G. (2000): *A Universe of Consciousness*. Basic Books.  
 Lloyd, N. (1966): *My First Sing-A-Song Book*. London: Paul Hamlyn.  
 Newell, A. & Simon, H.A.(1972): *Human Problem Solving*. Prentice-Hall.  
 Sutton, R.S. & Barto, A.G. (1998): *Reinforcement Learning*. MIT Press.

## Appendix-1

32 bit coding of note&beats:



(D=Intonation drop to hand over speech to other robot.  
 S=Stress on note. Note. Length of note appears twice.)

Notes:	E	1	Octave up:	13	2 octaves up:	25
	F	2		14		26
	F#, Gb	3		15		27
	G	4		16		28
	G#, Ab	5		17		29
	A	6		18		30
	A#, Bb	7		19		31
	B	8		20		
	C	9		21		
	C#, Db	10		22		
	D	11		23		
	D#, Eb	12		24		

## Appendix-2

Let the product of the primes for the cluster nodes be  $Z$ , so that the probability of an association having the same cluster nodes as a given association is  $p = 1/Z$  and the probability of an association having at least one different cluster node is  $1-p = (Z-1)/Z$ .

In the situation we are considering, PP has selected an association which happens to have a particular fixed context  $C$  and a particular set  $S$  of cluster nodes. There happen to be  $M$  other associations in PP's memory with the same fixed context  $C$ .

Assume  $Z > M$ .

Two questions need answering:

Q1. What are the probabilities that 0, 1, 2, ... of the other  $M$  associations in memory have the same set  $S$  of cluster nodes as that of the association selected by PP in List 8?

Q2. What are the probabilities that 0, 1, 2, ... pairs of the other  $M$  associations have the same sets of cluster nodes as each other?

### Deriving an answer to Question Q1

Take each of the  $M$  associations in turn. The probability that the first has a different set of cluster nodes from  $S$  is  $(1-p)$ . The same is true of the next since all  $Z$  sets are available to it. The same will be true for each of the  $M$  associations.

Therefore, the probability that none of the other  $M$  associations have the same set  $S$  of cluster nodes is  $(1-p)^M = (Z-1)^M/Z^M$ .

The probability that exactly 1 of the other  $M$  associations has the same set  $S$  of cluster nodes is  $Mp(1-p)^{M-1}$ , since there are  $M$  ways that there can be one.

The probability that exactly 2 of the other  $M$  associations have the same set  $S$  of cluster nodes is  $(M(M-1)/2).p^2.(1-p)^{(M-2)}$ , since there are  $(M(M-1)/2)$  ways that there can be 2.

Carrying on, the probability that exactly  $m$  of the other  $M$  associations have the same set  $S$  of cluster nodes is  ${}^M C_m . p^m . (1-p)^{(M-m)}$ , since there are  ${}^M C_m$  ways that  $m$  different combinations of associations can be chosen.

The above expressions are terms of the Binomial Expansion  $(p + 1 - p)^M$ , which must add up to unity as we would expect for all the ways that the  $M$  associations might have or not have the same set  $S$  of cluster nodes.

### Deriving an answer to Question Q2

This problem is the same as the problem of randomly scattering  $M$  objects onto  $Z$  places and finding how many places have 0, 1, 2, ... objects in them. Each of the  $M$  associations can “land” on any of the  $Z$  places, each of which has a different set of cluster nodes.

The probability of a place having no object in it after  $M$  placings is  $(1-p)^M = (Z-1)^M/Z^M$ . This expression will, therefore, give the proportion of places with no object. Since there are  $Z$  places, the expected number of places with no object will be

$$Z \cdot (1-p)^M = Z \cdot (Z-1)^M / Z^M.$$

The probability of a place having 1 object in it is  $p(1-p)^{M-1}$ . Since there are  $M$  placings, the object may be placed by any one of these, giving a probability of 1-object places equal to  $M \cdot p(1-p)^{M-1}$ . The expected number of places with 1 object will then be

$$Z \cdot M \cdot p(1-p)^{M-1}.$$

The probability of a place having 2 objects in it is  $p^2 \cdot (1-p)^{(M-2)}$ , but there are  $(M(M-1)/2)$  pairs of placings, so the probability of 2-object places is  $(M(M-1)/2) \cdot p^2 \cdot (1-p)^{(M-2)}$ . The expected number of places with 2 objects will be

$$Z \cdot (M(M-1)/2) \cdot p^2 \cdot (1-p)^{(M-2)}.$$

The probability of a place having  $m$  objects in it is  $p^m \cdot (1-p)^{(M-m)}$ , and there are  ${}^M C_m$  sets of  $m$  placings, so the probability of  $m$ -object places is  ${}^M C_m \cdot p^m \cdot (1-p)^{(M-m)}$ . The expected number of places with  $m$  objects will be

$$Z \cdot {}^M C_m \cdot p^m \cdot (1-p)^{(M-m)}.$$

The above expressions are, again, terms of the Binomial Expansion  $(p + 1 - p)^M$ , but multiplied by  $Z$ . They must add up to  $Z$ , as we would expect for the expected number of places that have all the possible numbers of  $M$  associations on them.

Note that the answer to Question 2 requires a different interpretation of the Binomial Expansion to that appropriate for Question 2. The first term of the expansion gives the number of sets of cluster nodes which do not belong to any of the  $M$  associations. The second term of the expansion (for the placing of one object) gives the number of sets of cluster nodes with only one association, so this is the number of associations with unique sets of cluster nodes. The third term gives the number of pairs of associations with identical sets of cluster nodes.

### Example from Paper.

$Z =$  product of primes =  $11 \times 13 = 143$ .

$M =$  number of other associations = 35.

First few terms of Binomial Distribution and cumulative probability:

Sums first  $m$  terms of  $n C_y p^y (1-p)^{(n-y)}$  with  $y=0$  to  $y=m$

$p$  is  $1/\text{Product of Primes}$ ,  $n$  is number of entries,  $y$  is number of matches.

Product of Primes = 143,  $p = 1/143 = 0.006993007$ ,  $n = \text{Entries} = 35$

matches = 0, term = 0.7822235, Cumulative Prob= 0.7822235  
matches = 1, term = 0.1928016, Cumulative Prob= 0.9750251  
matches = 2, term = 0.02308187, Cumulative Prob= 0.9981070  
matches = 3, term = 0.001788033, Cumulative Prob= 0.9998950  
matches = 4, term = 0.0001007342, Cumulative Prob= 0.9999957

Consider the first question:

Q1. What are the probabilities that 0, 1, 2, ... of the other M associations have the same set S of cluster nodes?

There is a 78.2% probability that none of the 35 entries will have the same set of cluster nodes as PP. In fact, none of them did.

Consider the second question:

Q2. What are the probabilities that 0, 1, 2, ... pairs of the other M associations have the same sets of cluster nodes as each other?

We need to be careful how we use the distribution for this question.

The second term multiplied by the product of primes gives the expected number of associations with unique cluster nodes (SICAnode and SICBnode):

$$143 \times 0.1928 = 27.57$$

The third term multiplied by the product of primes gives the expected number of cluster node sets with pairs of associations:  $143 \times 0.02308 = 3.30$  Since there are two associations in each place, we have 6.6 associations.

The fourth term multiplied by the product of primes gives the expected number of cluster node sets with triples of associations:  $143 \times 0.001788 = 0.256$  , Since there are three associations to each place, we have 0.768 associations.

The fifth term multiplied by the product of primes gives the expected number of cluster node sets with quadruples of associations:  $143 \times 0.0001007 = 0.0144$  , Since there are four associations to each place, we have 0.057 associations.

These add up to  $27.57 + 6.6 + 0.768 + 0.057 = 34.995$ , which is very close to 35, even without more of the very small terms.

The number of sets of cluster nodes with no association should be

$$143 \times 0.782 = 111.858$$

Now the number of sets of cluster nodes with associations is  $27.57 + 3.3 + 0.256 + 0.0144 = 31.14$

Adding these together, we get  $111.858 + 31.14 = 142.998$ , which is near enough to 143.

In summary, we can predict that if the sets of cluster nodes of the associations are independent of each other, there will probably be no match with the set of the association chosen by PP. There will be about 3 pairs of associations of the M which have matching sets of cluster nodes.

---

---