Supplementary Information, Logan & Cox (2023): Serial Order Depends on Item-Dependent and Item-Independent Contexts

Gordon D. Logan & Gregory E. Cox

Simulating CRU and SBAC

We simulated retrieval from CRU and SBAC memory matrices with CRU and SBAC retrieval processes with a Matlab program called *CRUsolway.m*. The program began by creating a set of generic CRU contexts for six-item lists. The first row represented the initial list cue, so it began with 1.0 in the first column and decreased by ρ for each successive column (Equation 1). Rows j = 2-6 began with β in column j and decreased by ρ for each successive column. The top left panel of Figure 1 presents a set of generic contexts with β = .6 and ρ = .8. SBAC matrix was produced by creating matrices of forward and backward associations and adding them together. The matrix of forward associations, $\boldsymbol{S}$, was created by multiplying the first row of the CRU matrix by β and copying the remaining elements in corresponding positions. The matrix of backward associations, $\boldsymbol{B}$, was created by multiplying each element in the forward matrix by w_b such that element *i,i-x* in $\boldsymbol{B}$ equaled w_b times element *i-x,I* in $\boldsymbol{F}$ (Equation 5). $\boldsymbol{F}$ and $\boldsymbol{B}$ were added to produce the SBAC matrix. The top right panel of Figure 1 presents a SBAC matrix constructed with β = .6 and w_b = .8.

We simulated the CRU retrieval process by creating a current context that began with the first column of the CRU matrix. We calculated the dot product between the current context and each of the stored contexts in the CRU matrix and we used the dot products as drift rates in a racing diffusion process. The first item to be retrieved was output as the retrieved item and added to the current context following the CRU/TCM updating equation (Equation 1). Retrieval continued until 7 items or the end-of-list element (? in the simulations) was retrieved.

We simulated SBAC retrieval using the first element (1,1 in the SBAC matrix) as the retrieval cue to initiate retrieval. This retrieval cue selected the top row of the SBAC matrix, and the elements in that row (association strengths) were used as drift rates in a racing diffusion process. The first item retrieved was output as the retrieved item and then used as the retrieval cue for the next item. Retrieval continued until 7 items or the end-of-list element was retrieved.

We simulated 100,000 trials for each combination of memory matrix and retrieval process (CRU-CRU, CRU-SBAC, SBAC-CRU, SBAC-SBAC), varying β (.5, .6, .8) and fixing w_b at .8. The output of the program was a confusion matrix representing the probability of recalling each item for each retrieval cue. The confusion matrices were converted to transposition gradients and presented graphically in the bottom panels of Figure 1.

The near-zero values for lag -1 in the transposition gradients was surprising and deserve comment. We analyzed the transitions when CRU contexts were used to cue retrieval from SBAC matrices. The top panel of Figure S1 presents CRU and SBAC contexts expressed as values of β and ρ, and the bottom panel plots dot products for each position as a function of the lag between the correct item and the retrieved item for $\beta = \{.5, .6, .7\}$. There is a dip in the dot products at lag -1 in each case: Lag -1 is lower than lag +1 and lag -2. The middle panel shows dot products between the CRU context for the fourth item (C4) and SBAC for columns 2, 3, 4, and 5 (S2-S5) expressed as values of β and ρ. The difference between lag + 1 (Dot(C4,S5)) and lag -1 (Dot(C4,S3)) is always positive because $(\beta\text{-}1)\rho^7 - (\beta\text{-}1)\rho^5 > 0$. The difference between lag -1 and lag -2 (Dot(C4,S2)) is positive as long as $\rho + w > 1.1$. The matrices in the top panel show why the lag -1 dot product is suppressed: The SBAC matrix contains zero in the location that has the highest value (β) in the CRU contexts.

The near-zero values for lag -1 transitions in the SBAC-SBAC simulations occur because lag -1 transitions following correct responses are repetitions whose retrieval would depend on self-associations. There are no self-associations in SBAC (see Figure 1).

Fitting CRU to Fill-In and In-Fill Data

We fit CRU to the whole report, serial recall, and copy typing tasks in Experiments 1 and 2 Logan (2021). A list of Matlab programs we used for fitting and analysis is presented in Table S1. We used the best-fitting parameters from the fits of the encoding decrease plus serial order decrease (ED-SOD) models for each subject. These models allowed the perceptual discriminability (g) and β parameters to decay exponentially across list position. We focused the fits on trials with fill-in and in-fill errors. We used current contexts built from Equation 1 with decaying β parameters for every letter in each string except the letter following the initial omission that resulted in a fill-in or in-fill error. The context that cued memory for those letters was a mixture of the current context calculated from the letter following the omission and the

initial context, using λ and λ' parameters to weight the list cue and the current context, respectively. λ and λ' were defined as in Equation 7 and combined as in Equation 6.

The fitting routine calculated the likelihood of each response using the dot products between the current context or the combination of the current context and the list context as drift rates (v) in a racing diffusion process with a fixed threshold of 200, following Logan (2021). We used the likelihood function from Equation 10 in Logan (2021):

$$f(t,i) = \theta_i(2\pi t^3)^{-1/2} \cdot exp\left[-\frac{1}{2t}(v_i t - \theta_i)^2\right] \times \prod_{j\neq i}^{N}[1 - \Phi(t^{-\frac{1}{2}}(v_j t - \theta_j))$$

$$-\exp(2v_j\theta_j)\Phi(-t^{-\frac{1}{2}}(v_j t - \theta_j))]$$

We summed the log of the likelihood for each letter and each string and used the Matlab function *fmincon* to minimize the (negative) log likelihood for each subject. The mean values of the best-fitting λ parameters and the log likelihoods for each experiment are presented in Table 1.

We calculated the predicted fill-in and in-fill error probabilities by simulating CRU with the ED-SOD models from Logan (2021) supplemented with the mixture of list context and current context vectors for retrievals following initial omissions that led to fill-in and in-fill letters. We simulated 1000 repetitions of the 576 trials for each subject and scored the data with the same routines we used to score the actual data. The results are presented in Figure 2.

Error Ratio in CRU

Consider a situation in which CRU is retrieving members from a list with no repeated items (ABCDE) and makes an omission (ABD). The extension of CRU that we consider assumes that the context for CRU's next retrieval is a mixture of the list context and the current item context. These two components are mixed according to parameter λ, which ranges between 0. Specifically, the cue vector is

$$\mathbf{c}_{\text{cue}} = \lambda\mathbf{c_1} + \lambda'\mathbf{c_i}$$

where $\boldsymbol{c_1}$ is the list context vector, $\boldsymbol{c_i}$ is the item context vector at output position i, and λ' is chosen such that $\boldsymbol{c}_{cue}$ is of unit length using Equation 7.

The retrieval strengths that drive CRU's racing diffusion process for recall are the dot products of $\boldsymbol{c}_{cue}$ with each of the stored contexts from the list. Let $\boldsymbol{m_j}$ denote the context stored

for the item studied in position j. Given how $\boldsymbol{c}_{cue}$ is created and the properties of the dot product, its dot product with a stored context $\mathbf{m_j}$ can be decomposed:

$$\mathbf{c}_{\text{cue}} \cdot \boldsymbol{m_j} = \lambda(\boldsymbol{c_1} \cdot \boldsymbol{m_j}) + \lambda'(\boldsymbol{c_i} \cdot \boldsymbol{m_j})$$

We will be interested in two such dot products: that with the context stored for the "fill-in" item ($\boldsymbol{m_{i-1}}$) and that for the "in-fill" item (position $\boldsymbol{m_{i+1}}$). Note that our subscripts embody the idea that because an omission was made at the previous output position $i-1$ and CRU is currently trying to decide what to produce at output position i, the fill-in item is "one position too late" (hence $i-1$) whereas the in-fill item is "one position too early" (hence $i+1$). In any event, what chiefly matters is the difference between them, because if $\boldsymbol{c}_{cue} \cdot \boldsymbol{m_{i-1}} - \boldsymbol{c}_{cue} \cdot \boldsymbol{m_{i+1}} > 0$, CRU will strongly favor a fill-in rather than an in-fill. Note, however, that because of the stochastic nature of CRU's responding, it is entirely possible for CRU to make a fill-in rather than in-fill response even if the above inequality does not hold.

The components of the inequality can now be arrayed:

$$\boldsymbol{c}_{cue} \cdot \boldsymbol{m_{i-1}} - \boldsymbol{c}_{cue} \cdot \boldsymbol{m_{i+1}} > 0$$

$$[\lambda(\boldsymbol{c_1} \cdot \boldsymbol{m_{i-1}}) + \lambda'(\boldsymbol{c_i} \cdot \boldsymbol{m_{i-1}})] - [\lambda(\boldsymbol{c_1} \cdot \boldsymbol{m_{i+1}}) + \lambda'(\boldsymbol{c_i} \cdot \boldsymbol{m_{i+1}})] > 0$$

$$\lambda[(\boldsymbol{c_1} \cdot \boldsymbol{m_{i-1}}) - (\boldsymbol{c_1} \cdot \boldsymbol{m_{i+1}})] + \lambda'[(\boldsymbol{c_i} \cdot \boldsymbol{m_{i-1}}) - (\boldsymbol{c_i} \cdot \boldsymbol{m_{i+1}})] > 0$$

The dot products with the list context $\mathbf{c_1}$ come directly from how CRU updates context:

$$\mathbf{c_1} \cdot \mathbf{m_{i-1}} = \rho^{i-2}, \qquad \mathbf{c_1} \cdot \mathbf{m_{i+1}} = \rho^{i}$$

As a result, their difference is

$$(\boldsymbol{c_1} \cdot \boldsymbol{m_{i-1}}) - (\boldsymbol{c_1} \cdot \boldsymbol{m_{i+1}}) = \rho^{i-2} - \rho^{i} = \rho^{i-2}(1-\rho^2)$$

The dot products with the current temporal context $\boldsymbol{c_i}$ are somewhat more involved, but can also be derived from CRU's context updating process:

$$\boldsymbol{c_i} \cdot \boldsymbol{m_{i-1}} = \beta^2 \sum_{k=1}^{i-1} \rho^{2k-1} + \rho^{2i-1}, \qquad \boldsymbol{c_i} \cdot \boldsymbol{m_{i+1}} = \beta^2 + \beta^2 \sum_{k=2}^{i} \rho^{2k-1} + \rho^{2i+1}$$

Their difference can then be found by cancelling the common terms in the summations:

$$\boldsymbol{c_i} \cdot \boldsymbol{m_{i-1}} - \boldsymbol{c_i} \cdot \boldsymbol{m_{i+1}} = \beta^2\rho + \rho^{2i-1} - \beta^2 - \beta^2\rho^{2i-1} - \rho^{2i+1}$$

$$= \beta^2(\rho - 1) + \rho^{2i-1}(1 - \beta^2) - \rho^{2i+1}$$

$$= (1 - \rho^2)(\rho - 1) + \rho^{2i-1}\rho^2 - \rho^{2i+1}$$

$$= (1 - \rho^2)(\rho - 1) + \rho^{2i+1} - \rho^{2i+1}$$

$$= (1 - \rho^2)(\rho - 1)$$

where we also made use of the fact that $\rho = \sqrt{1 - \beta^2}$ since the list is assumed not to contain any repetitions.

We can now rewrite the inequality:

$$\boldsymbol{c}_{cue} \cdot \boldsymbol{m_{i-1}} - \boldsymbol{c}_{cue} \cdot \boldsymbol{m_{i+1}} > 0$$

$$\lambda\rho^{i-2}(1 - \rho^2) + \lambda'(1 - \rho^2)(\rho - 1) > 0$$

$$\lambda\rho^{i-2} + \lambda'(\rho - 1) > 0$$

Recall that, for a list with no repetitions, λ' can be determined solely on the basis of the strength of the list context element, which is ρ^i:

$$\lambda' = \sqrt{1 + \lambda^2(\rho^{2i} - 1)} - \lambda\rho^i$$

Now the inequality becomes:

$$\lambda\rho^{i-2} + \left[\sqrt{1 + \lambda^2(\rho^{2i} - 1)} - \lambda\rho^i\right](\rho - 1) > 0$$

which, after some rearranging of terms, yields the minimum value of λ needed to guarantee the inequality:

$$\lambda > \frac{1 - \rho}{\sqrt{\rho^{2i}(\rho^{-4} + 2\rho^{-2} - 2\rho^{-1}) + (1 - \rho)^2}}$$

As can be seen, as serial position i increases, the denominator decreases (because $\rho < 1$, so taking ρ to a higher power results in a lower value). As a result, the minimum λ increases with serial position i, as illustrated in Figure S2.

We emphasize that this derivation, while instructive, is much simplified relative to the full set of processes involved in CRU. The inequality describes the conditions under which the *drift rate* for the fill-in response is guaranteed to be higher than the drift rate for the in-fill response at position i, but because responding is not deterministic it is entirely possible for CRU to produce more fill-ins than in-fills on average without necessarily exceeding the critical λ at each serial position. That said, as noted in the main text, it is likely that this is one reason why even the augmented CRU does not quite match the error levels observed in the data—a value of λ that is suitable for early serial positions may not be enough for later serial positions.

Simulating Prior List Intrusions with Caplan et al.

Caplan et al. (2022) proposed a mechanism that enabled a chaining model to produce protrusions without making runs of prior-list intrusions. This mechanism, based on a similar idea by Lewandowski & Murdock (1989), assumes that the cue for the next retrieval is not the item that was most recently retrieved, but the vector of item activations which drove the retrieval of that item. Caplan et al. (2022) showed that this mechanism would allow a chaining model to make protrusions without going "off the rails" and making a run of PLI's. The reason this works with a chaining model is because errors typically have lower activation than correct responses. Protrusions are only made when these low-activation options are chosen by chance. As a result, the correct response has the strongest activation in the subsequent retrieval cue even if it was not reported.

While Caplan et al.'s mechanism works in the chaining model they explored, it does not work in CRU. When CRU updates its cue context using the vector of drift rates rather than using a vector containing just the retrieved item, CRU has a tendency to skip to the end of the list. The reason why is illustrated in the numerical examples of each updating rule shown in Figure S3. In CRU, contexts associated with items from later in either the target list or the prior list have non-zero dot products with early cue contexts. By updating the cue context with the vector of dot products, the cue context comes to contain items from later in the list, even if they have not yet been reported. As recall proceeds, the cue context represents later items ever more strongly. Because these later items are only present in the contexts stored with later list items, CRU becomes ever more likely to report these later items prematurely.

More broadly, the fact that this mechanism has different effects in a chaining model relative to CRU highlights an important difference between CRU and compound chaining models like that explored by Caplan et al. In compound chaining models, different cues act separately to activate items for retrieval. As such, multiple cues act like a superposition of many individual cues. In contrast, CRU treats its cue context as a configural whole, with items activated to the extent that the *pattern* of context information in the cue matches that stored at study. CRU contexts are integral representations of the past that act in a configural manner to cue retrieval, whereas compound chaining models treat cues as separable.

Simulating Prior List Intrusions with CRUposition

We simulated prior list intrusions using a position coding version of CRU, CRUposition (Figure 5; Logan & Cox, 2021), that alternated with or ran in parallel with a "standard" item coding version of CRU. CRUposition contexts were created with the CRU/TCM updating equation (Equation 1) and each context was associated with a generic position code. We represented the position codes as consecutive numbers in Figure 5, but the position codes should not be interpreted as numbers. They should be interpreted as generic representations of an ordered series. When a list is presented, the items are associated with the generic position codes. The simulations represented two consecutive lists. Each list was associated with the same generic position codes but the strength of association varied between lists. The strength of association between position codes and the current list was always 1.0 (as were the strengths of associations between the generic contexts and the position codes). The strength of association between position codes and the prior list varied between 0 and 1, represented by the parameter *List1strength*. We also created a set of item-dependent contexts for the current list, associating the items in the current list with the contexts in which they appeared with strength = 1.0.

We ran two versions of the simulations. The first version alternated between CRUposition and CRU, using CRUposition with probability *plist* and CRU with probability 1-*plist* (program: *CRUposition5.m*). The second version ran CRUposition and CRU in parallel with probability *pposition* or CRU by itself with probability 1-*pposition* (program: *CRUposition4.m*). When CRUposition and CRU ran in parallel, we generated a set of drift rates for each model (from dot products between current context and stored context vectors) and we chose the response generated by the faster of the two models. We simulated 100000 trials with

each version, producing sequences of recalled items. We scored the simulated sequences with the same program we used to score the actual data. Transposition gradients for the current (top panel) and prior lists (middle panel) are presented in Figure 6. For those simulations, $\beta = .55$, *List1strength* = .90, and *pposition* varied from

To investigate the tradeoff between the probability of using position coding (*plist*) and the strength of prior-list associations (*List1strength*), we ran an initial set of simulations in which we varied *List1strength* from .8 to 1.0 fixing *pposition* = 1 and $\beta = .55$ (program: *CRUposition2.m*). The orange bars in the bottom panel of Figure 4 show that prior list intrusions increased as *List1strength* increased. We compared these values from a second set of simulations in which we fixed *List1strength1* at 1.0 and β at .55 and varied *pposition* by hand to find intrusion probabilities that matched those from the first set of simulations (programs: *CRUposition5.m* for alternation and *CRUposition4.m* for parallel). The results are plotted as blue bars in the bottom panel of Figure 6. The numbers above the bars represent the *pposition* values that produced the matching intrusion probabilities. There were 100000 trials in each simulation.

Simulating Grouping and Between Group Intrusions

We ran two sets of simulations to address grouping phenomena. One set implemented Osth and Hurlstone's (2022) modification of CRU that increased β at the beginning of each group such that $\beta_g = \beta + binc \times (1 - \beta)$, creating CRU contexts for a list of 9 items presented in groups of 3 (program: *CRUgrouplishOH.m*). A set of contexts with $\beta = .5$ and *binc* = .5 are presented in the top left panel of Figure 7. We compared recall with lists grouped in these ways with recall for a 9-item ungrouped list. Each simulation ran for 10,000 trials. The advantage of grouped over ungrouped lists is presented in Table 2 for combinations of β (.5, .6, .7) and *binc* (.5, .6, .7, .8, .9). The serial position curve and transposition gradient from the first set with $\beta =$ .5 and *binc* = .5 are presented in the second row of Figure 7. The simulations showed a grouping advantage in accuracy but no evidence of position-specific between-group intrusions.

The second set of simulations implemented hierarchical position coding in CRU (program: *CRUgrouplisthierarchy.m*). We generated four sets of contexts using the CRU updating equation with $\beta = .4, .5, .55, .6$, and .7 (see Figure 7, top right panel). One set of contexts represented the three groups. The other three sets represented the three items within each group. Retrieval began by choosing a group from the set of group contexts and then

choosing items from the item contexts within the chosen group. All nine items competed at retrieval, but their contexts were weighted by the similarity of the group contexts, ρ^{x-1}, where $x = \{0, 1, 2\}$ is the lag between the selected group and the competing group. When three items from the group had been retrieved, the next group was retrieved from the higher-level set of contexts representing the groups. Retrieval proceeded until three items had been chosen from each of the three groups. We simulated 10,000 trials for each value of β. Simulations with different β values produced very similar results. Overall, recall accuracy increased with β and the probability of between-group intrusions decreased, but all values of β produced scalloped serial position curves and spikes in the transposition gradients at lags 3 and 6, indicating between-group intrusions. The results for $\beta = .5$ are presented in the two right panels in the second row of Figure 7.

The mixtures of the Osth and Hurlstone (2022) version of CRU and our hierarchical position code version of CRU, plotted in the bottom panel of Figure 7, were obtained by multiplying the position and lag effects from the Osth and Hurlstone version by p and the position and lag effects from the hierarchical version by $1 - p$.

Table S1

Programs and data used to fit in-fill and fill-in data from Logan (2021).

	Experiment 1:
Data:	`RepRecTyp_Length_Data.xlsx`
Programs:	`CRUfitPMDecayFILLININFILLr2.m`
	`CRUcalcPMDecayFILLININFILL3.m`
	`makewordsserialsimDECAY.m`
	`makewordsERRLIKEserialDecay.m`
	`WaldlikelihoodNEW.m`
	`Shellfillininfill.m`
	`typeNwordsCRUpmDECAYPMfillininfill2.m`
	`ranschscore3SsSIMFILLININFILL(simlistout).m`
	`ranschscore3SsFILLININFILL.m`
	Experiment 2:
Data:	`GordonRanschburg_Exp1B_Data.xlsx`
Programs:	`CRUfitPMDecayFILLININFILLx1Br.m`
	`CRUcalcPMDecayFILLININFILLx1B2.m`
	`makewordsserialsimDECAY.m`
	`makewordsERRLIKEserialDecay.m`
	`WaldlikelihoodNEW.m`
	`shellfillininfill2.m`
	`typeNwordsCRUpmDECAYPMfillininfill3.m`
	`ranschscore2ASsSIMFILLININFILL(simlistout).m`
	`ranschscore2ASsFILLININFILL.m`

Figure S1: CRU and SBAC Matrices and Dot Products

CRU Matrix:

1	ρ	ρ^2	ρ^3	ρ^4	ρ^5
	β	$\beta\rho$	$\beta\rho^2$	$\beta\rho^3$	$\beta\rho^3$
		β	$\beta\rho$	$\beta\rho^2$	$\beta\rho^3$
			β	$\beta\rho$	$\beta\rho^2$
				β	$\beta\rho$
					β

SBAC Matrix:

β	$\beta\rho$	$\beta\rho^2$	$\beta\rho^3$	$\beta\rho^4$	$\beta\rho^5$
0	β	$\beta\rho$	$\beta\rho^2$	$\beta\rho^3$	$\beta\rho^4$
$w\beta p$	0	β	$\beta\rho$	$\beta\rho^2$	$\beta\rho^3$
$w\beta p^2$	$w\beta p$	0	β	$\beta\rho$	$\beta\rho^2$
$w\beta p^3$	$w\beta p^2$	$w\beta p$	0	β	$\beta\rho$
$w\beta p^4$	$w\beta p^3$	$w\beta p^2$	$w\beta p$	0	β

Dot Products with C4:

$\text{dot(C4,S2)} = \rho^3\beta\rho + \beta\rho^2\beta + \beta w\beta\rho = \rho^2 + (\beta-1)\rho^4 + \rho w(1-\rho^2)$

$\text{dot(C4,S3)} = \rho^3\beta\rho^2 + \beta\rho^2\beta\rho + \beta\rho\beta = \rho + (\beta-1)\rho^5$

$\text{dot(C4,S4)} = \rho^3\beta\rho^3 + \beta\rho^2\beta\rho^2 + \beta\rho\beta\rho + \beta^2 = 1 + (\beta-1)\rho^6$

$\text{dot(C4,S5)} = \rho^3\beta\rho^4 + \beta\rho^2\beta\rho^3 + \beta\rho\beta\rho^2 + \beta^2\rho = \rho + (\beta\text{-}1)\rho^7$

Dot Products x Lag:

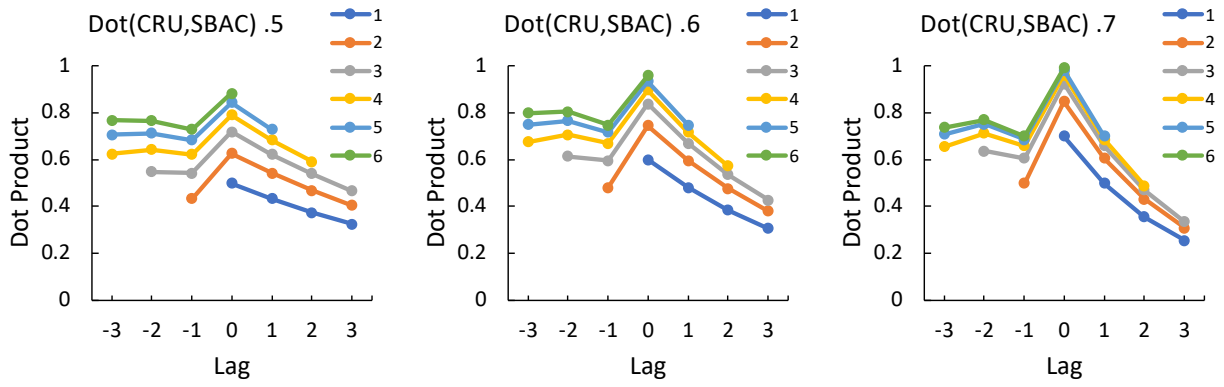
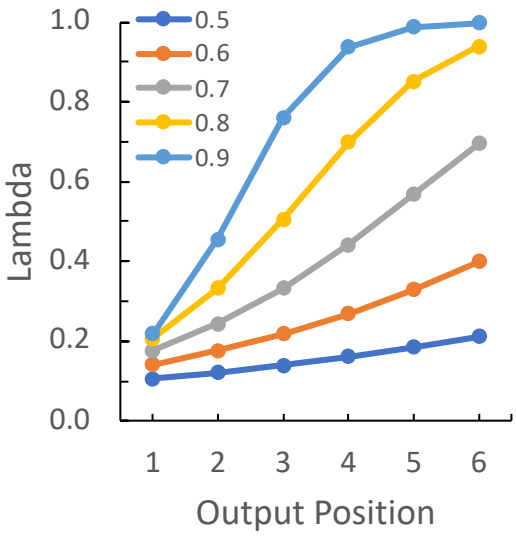
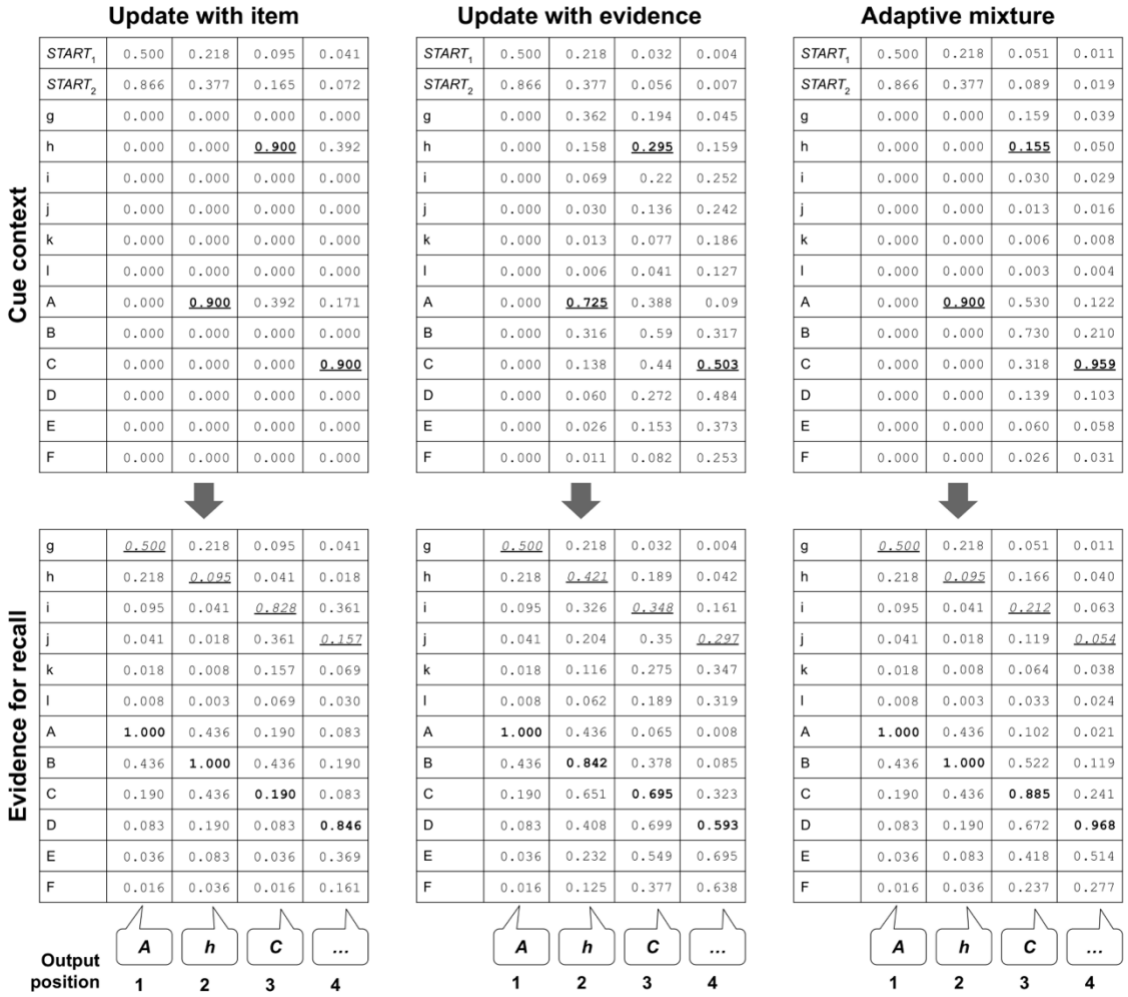


Figure S2: List Context and Output Position



Note: The minimum list context weight λ needed for CRU to strongly favor a fill-in over an in-fill response increases with both output position and with context updating parameter β.

Figure S3: Numerical examples of three variants of context updating during retrieval



Note: In each example, two lists of six items each were studied (*ghijkl* followed by *ABCDEF*) and second list is the target for recall. Two context elements represent the start of each list (*START*1 and *START*2) and are set such that they have partial similarity (s = 0.5). The top matrices represent the context used as a retrieval cue at the first four output positions, with the underlined and bolded entries representing those corresponding to the item that had been reported on the previous output position. The bottom matrices represent the dot-products between the cue context and the stored contexts for each of the 12 studied items; these dot-product similarities are the evidence for recall because they set the rate at which each item races to be retrieved. In the bottom matrices, bold entries indicate the items that would be correct responses and underlined and italicized entries indicate the items that would be protrusions. In each example, CRU reports A, h, and C in the first three output positions.