Supporting Text

The architecture of the full prefrontal cortex (PFC) model is shown in Fig. 1. All models contained the same input/output layers (two Stimulus locations, Task and Dimension Cue inputs, and Response output). The full model has three hidden layers (Task and Cue Hidden, each 16 units, and main Hidden layer with 83 units), which serve to encode the relevant stimuli and map them onto the appropriate responses. These are thought to correspond to association areas in the posterior cortex. The Task and Cue hidden layers enable the network to develop internal representations of these inputs that are influenced by the other layers of the network, in particular the PFC. For example, the PFC representation of a stimulus dimension can activate the Cue hidden layer with a corresponding representation during the uninstructed task (when there is no dimension cue input), and this helps to support the network in focusing on the appropriate dimension, through weights that were trained on the instructed cases when a dimension cue is present. We think of this as analogous to the process of subvocalization (inner speech), where people generate internal verbal representations to support task performance (*1–3*). As described, the PFC and associated adaptive gating (AG) unit support robust active maintenance that is also capable of rapid updating. Note that the PFC layer (30 units) had full self connectivity (each unit connected to all other units in the layer), which supported maintenance in conjunction with intrinsic bistability as described in detail below. The PFC layer projects to the hidden layer (posterior cortex) and the Response layer, to enable the maintained rule information to influence both internal processing and overt responding. This is also in accord with anatomical data (*4,5*).

As noted in Fig. 1, all of the synaptic weights in the model were trained, with those into the AG unit being trained through the temporal differences (TD) mechanism, and the rest being trained via the standard Leabra learning mechanisms as described in detail below.

We tested a number of variations on this basic architecture to determine the importance of different features. In all cases, the total number of internal (hidden + PFC) units was the same, to rule out changes in overall representational power as a confounding factor. In all cases, we found that these alternative architectures generalized significantly worse than the full PFC model (Fig. 1b). This is despite the fact that all network architectures were capable of learning the training patterns to a similarly high level of performance.

PFC No Gate

This variation was identical to the full PFC model, except it lacked the AG unit. The PFC layer was still capable of maintenance through its recurrent excitatory connections, but this maintenance was not subject to AG signals in response to differences between predicted and actual rewards (as computed by the AG unit). Because the bistable maintenance currents of the Full PFC model require some kind of modulatory mechanism to turn them on and off (i.e., the adaptive gating mechanism), they were also not present in this version of the model.

SRN-PFC

The simple recurrent network (SRN) architecture (*6*) is perhaps the most widely used mechanism in cognitive models for simulating the maintenance of contextual information for temporally extended tasks. The SRN uses a context layer that is updated from an associated hidden layer in an alternating, out-of-phase manner. That is, after a trial of processing, the hidden layer activations are copied to the context layer, which provides input to the hidden layer on the next trial. In this way, processing on the next trial can be contextualized by information present on the previous trial. We have noted that this regular context updating represents a simple form of a gating mechanism, where gating occurs on every trial (*7*). To determine how effective this form of gating is compared to the adaptive gating provided by the AG unit, the SRN-PFC model provides an SRN context layer for the PFC layer (Fig. 5).

SRN (No PFC)

The SRN architecture is usually implemented so that the SRN context layer copies a single hidden layer. In many respects, the SRN context layer can be considered a simple version of the PFC layer in the original full PFC model. Therefore, we tested a more standard SRN architecture with a single Hidden layer having the same number of units present in all the hidden and PFC layers in the original full PFC model (145 units), and an SRN context layer that copies this Hidden layer (Fig. 6). One might consider trying an SRN with the AG mechanism of the full PFC model. Given that the SRN mechanism provides suitable robust maintenance abilities, we would argue that this is essentially equivalent to the full PFC model in the first place. In short, the AG is the most important feature of the full PFC model (as the PFC No Gate model demonstrated), and the details of the robust maintenance mechanism are less important.

Posterior-Cortex Model and Variations

The final set of architectures consisted of a single large (145 units) hidden layer, representing posterior cortical areas, mediating the input-output mapping. This represents a standard three-layer architecture as commonly used in cognitive modeling. In its most basic form, the hidden layer has no capability of maintenance whatsoever. We also explored two variations having either self-recurrent (each hidden unit connected to itself) or full-recurrent connectivity (each hidden unit connected to all other hidden units).

Given that generalization is of primary concern in these networks, and that in some cases larger networks have been shown to generalize worse than smaller ones (i.e., due to overfitting from too many degrees of freedom), one might want to also see results from networks with smaller hidden layers. These networks were run and performed worse than those with larger hidden layers. This is consistent with the fact that the present task is completely deterministic and has no opportunities for overfitting to represent noise. In such conditions, larger networks are better, because the larger sample of hidden unit activities reduces the chances that the idiosyncrasies of individual units dominate the overall responding (i.e., a law of large numbers effect). See ref. 8 for further discussion, results, and analysis.

Training and Testing

Task Descriptions

All models were trained on four types of tasks, with each task type having instructed and uninstructed versions, for a total of eight different task conditions. Each task involved performing a simple operation on one or two multifeatured stimulus inputs (Fig. 8), such as naming one stimulus feature (NF), matching features across two stimuli (MF), and comparing the relative magnitude of features across two stimuli (smaller, SF or larger, LF). Each stimulus was composed by selecting one of four feature values along five different dimensions, which we think of as corresponding to simple perceptual dimensions such as number, size, color, shape, and texture (e.g., as in the cards used in the WCST task). For convenience, the five dimensions were abstractly labeled with the letters A-E, and the features numbered 1-4 (e.g., A2 refers to the second feature in dimension A). No attempt was made to impose any real-world semantic constraints on these stimuli. There are a total of $4^5 = 1,024$ different stimuli. Stimuli could be presented in two different locations in the input, meaning that there are more than one million distinct stimulus input patterns $(1,024 \times 1,024)$.

The Response output layer contained the same stimulus representation as the input, with an additional "No" output response.

In each task, one of the five dimensions is relevant at any given time. The relevant dimension can either be explicitly specified in the Dimension Cue input (instructed task) or it must be guessed by the network through trial and error (uninstructed task). Although the uninstructed case may be more ecologically prevalent, and we have shown that the full PFC model can learn dimensional representations with a purely uninstructed environment (*8*), it was important to include the instructed conditions for the control models without a PFC. These models cannot switch rapidly to find the relevant task dimension and thus suffer more from the uninstructed case.

The task type is always specified via the Task input layer. For example, in the NF-I (naming feature-instructed) task, the single input stimulus (presented in either input location) might correspond to three large, rough, blue squares, and if the relevant dimension in the Dimension Cue is size, the correct output would be "large." In the other three task types, the input display consists of two stimuli (in the left and right input locations), and these stimuli are compared along the relevant dimension. Consistent with the NF task, the output for these tasks is either the name of a feature or "No." For example, if the left stimulus and relevant dimension (size) are as in the prior example, and the right stimulus is medium sized, then the correct output for MF (matching) is "No," SF (smaller) is "medium," and LF (larger) is "large."

Training and Testing Parameters

Each network was trained 10 times with random initial weights for 100 epochs consisting of 2,000 individual training trials per epoch. Testing for crosstask generalization performance occurred every 5 epochs throughout training, and the best level of generalization performance for each training run was recorded. This procedure eliminated possible confounds associated with differential amounts of feature exposure per epoch, etc, depending on the different training conditions. The presence of the uninstructed versions of tasks required that relevant dimensions be organized into contiguous blocks of trials where a given dimension was always relevant. The size and ordering of these blocks were systematically varied in different "training protocols" (described below). Testing was always performed on the instructed tasks, so that the differential memory abilities of the various model architectures would not confound the results; no memory of the relevant dimension is required in the instructed version.

Crosstask generalization testing requires that certain stimuli not be experienced on a given task during training; these stimuli are then used during testing. Specifically, features B3, C2, D3, and E2 were always excluded from training on all the instructed tasks, and testing consisted only of instructed task cases where one of these nontrained stimuli was the relevant stimulus. This means that the instructed tasks were trained on $\frac{324}{1{,}024} = 31.64\%$ of the full stimulus space ($324 = 4 \times 3 \times 3 \times 3 \times 3$). The uninstructed versions of each of the four task types were trained with one omitted stimulus feature. Specifically, the B3 stimulus was never seen during the NF-U task, the C2 stimulus was omitted for the MF-U, etc. Generalization therefore required the learning on uninstructed tasks to transfer to instructed tasks. Although some of the generalization tested was within the same task type between uninstructed and instructed (e.g., training on NF-U C2 could promote generalization of NF-I on C2), these task types were actually quite different from the network's perspective. Specifically, the critical relevant dimension must come from two different sources between the instructed and uninstructed versions of the tasks, and this means that different representations and weights are involved. The ability to produce an appropriate response across both cases requires that the internal representation of the relevant dimension be consistent, and consistently used, across both cases. This generalization of dimensional representation across tasks was exactly what we sought to test, and it applies equally well to generalization across instructed and uninstructed as to generalization across naming and matching.

Two training/testing conditions were run; one in which all four task types were used, and another in which only two out of the four were used for any given network, with results averaged across all possible combinations of such task pairs. Due to the structuring of the omitted features, these two conditions did not differ in the range of features trained, but only in the extent to which a given feature appeared across multiple tasks. Thus, the differences in generalization between these two conditions reveal the benefits of having processed a given feature in four vs. two different task contexts.

Blocked Organization of Dimensions and Tasks

Finally, the sequencing of tasks and relevant dimensions was organized in a hierarchically blocked fashion, with the outermost block consisting of a loop through all instructed tasks followed by all uninstructed tasks, with each task being performed for a block of 25 input/output trials. The relevant dimension was switched after every two of the outermost blocks. This relatively low rate of switching allows the networks without adaptive gating mechanisms plenty of time to adapt to the relevant dimension. In other preliminary experiments, we tested a variety of blocking strategies, including: (i) interleaved instructed and uninstructed tasks within a 50 trial block, followed by a dimension switch; (ii) the standard protocol but with uninstructed cases first; (iii) standard but with inner-block sizes of 50 with 1 outer-block per dimension switch; (iv) like iii but with uninstructed first; (v) loop through all instructed tasks with 50 trials per relevant dimension, followed by a loop through all uninstructed tasks with 50 trials per dimension; and (vi) like v but with uninstructed first. The impact of these manipulations never made more than a 10% impact on performance in tests run on the generalization between the U and I versions of the NF task (e.g., the Full PFC model performance varied between 13.6% and 23.3% generalization error across these versions).

PFC Neuropsychological Data

As noted, we argue that our PFC layer in the model corresponds to the dorsolateral PFC area (DLPFC) in the human brain; establishing this relationship is an important prerequisite for understanding the relationship between damage to this PFC layer and the neuropsychological data on the Stroop and Wisconsin Card Sort Task (WCST) tasks. In general, we think that the specialized neural mechanisms in our model are characteristic of the entire PFC, with different PFC areas having different types of representations (*9,10*). In particular, we have argued that more dorsal lateral PFC areas (e.g., DLPFC) encode relatively more abstract representations (including things like stimulus dimensions), whereas more posterior and inferior lateral areas encode more specific representations of stimulus features (*10*). This distinction makes sense of data showing dissociated effects of lesions to these PFC areas on the intradimensional/extradimensional (ID/ED) version of the WCST task (*11*). Thus, the abstract dimensional representations that developed in the present model are consistent with a locus in DLPFC. As we show in the main paper, this DLPFC locus is consistent with relevant neuropsychological data on the Stroop and WCST tasks from Stuss et al. (*12,13*).

In the Stroop task, the Stuss *et al.* 2001 paper (*12*) shows that DLPFC damage produces an overall deficit in color naming, while superior-medial (SMPFC) damage produces a selective deficit on the incongruent trials. The model correctly produced an overall color-naming impairment associated with DLPFC damage (Fig. 4b). Note that we used the left frontal (LF) group from Stuss *et al.* (*12*) as the point of comparison, because this group had 75% (6/8) patients with dorsolateral damage, whereas the right frontal (RF) group only had 57% DLPFC (8/14). These patient data are consistent with the presence of abstract dimensional representations (e.g., representing color as an abstract dimension) that developed in the PFC of our model. These color-dimension representations provide top-down support to color-processing areas in the posterior cortex, and thus damaging these representations produces color-naming impairments. This is also consistent with a recent update (*14*) of the original Cohen *et al.* Stroop model (*15*). This model includes a domain-general PFC representation for color, in addition to the more specific color-naming representation central to the original model. The DLPFC is postulated to encode the domain-general color representation (equivalent to the abstract dimensional representations learned in our model), while the SMPFC encodes the color-naming specific representations. These modifications to the original Stroop model were inspired by fMRI data from Banich and others (*16,17*), providing converging evidence for the present interpretation of DLPFC contributions to the Stroop task.

In the WCST task, the 2000 Stuss *et al.* paper (*13*) shows that damage to DLPFC areas reliably produces the classic perseveration effects that we show in the main paper (Fig. 4c). In contrast, damage to inferior-medial (IM) areas (and posterior cortical areas), when the proper controls are applied, does not produce these effects. This may help to clarify some of the controversy about whether frontal damage reliably produces perseveration effects (*18,19*); it depends on which area of PFC is damaged. This DLPFC locus for perseverative effects is consistent with our interpretation of the model as representing DLPFC. Furthermore, it is consistent with the idea that DLPFC contains abstract dimensional representations, as we argued in an earlier paper (*10*). This paper presents a model of the data from Dias *et al.* (*11*) showing dissociated effects of lesions to dorsal and inferior PFC areas on the intradimensional/extradimensional (ID/ED) version of the WCST task. Specifically, this paper showed that imposing abstract dimensional representations in the simulated DLPFC of our model, and more specific featural representations in the simulated IM areas, reproduced the observed double-dissociation between ID and ED rule changes. The key point is that extradimensional rule changes are facilitated by the presence of abstract dimensional representations in DLPFC, such that lesions to this area produce selective impairments in this form of rule change.

The Stuss *et al.* WCST studies (*13*) also reported results from two different instructional conditions beyond the standard case (no instructions, denoted 128). The second condition (64A) involved informing the subjects about the possible dimensions on which they could sort the cards. We simulated this by activating all of the dimension cue inputs to the model, which provides additional excitation to those stimulus dimension representations in the network. However, given the lack of effect of this manipulation in both DLPFC patients and the model, we conclude that the bottom-up stimulus activation from the sample cards themselves likely provides sufficient activation of the relevant dimensions. The third condition (64B) involved informing participants when the rule changed (but not what the new rule is). This was simulated by reinitializing the PFC representations in the model when the rule changed. Again, this did not have substantial effects on either the DLPFC patients or the model. Furthermore, these instructional conditions were confounded by sequential order effects in the empirical studies, so the slight improvements seen with these manipulations could possibly be artifactual.

Other Forms of Cognitive Flexibility

Our primary focus in this work has been on one of the factors that contributes to the flexibility of cognitive control: the ability to generalize task performance to novel environments, and to switch task representations dynamically in response to changing demands (as in the WCST task). However, there is another equally important factor in the flexibility of control: the ability to generate entirely new behaviors. Our model does not directly address this capacity for generativity. Although the model was tested on its ability to perform tasks on new combinations of features, it was never asked to perform entirely new tasks. One might contend that the same is true for humans; the flexibility of cognitive control has its limits. People cannot actually perform entirely new tasks without training, at least not with any alacrity (e.g., one cannot play complex card games such as bridge very well when first learning them). Nevertheless, people can recombine familiar behaviors in novel ways. Such recombination, when built upon a rich vocabulary of primitives, may support a broad range of behavior and thus contribute to generativity.

We believe that the PFC representations learned by our model provide a rudimentary example of the type of task or rule "vocabulary" that can support recombination, and thus constitute an important step toward understanding the neural mechanisms underlying generativity. In particular, the discrete, orthogonal nature of these representations should facilitate the formation of novel combinations. As a result, the mechanisms and processes that we have identified as critical to generalization may also set the stage for understanding generativity. Generativity also highlights the centrality of search processes to find and activate the appropriate combination of representations for a given task, a point that has long been recognized by symbolic models of cognition (*20*). Such models have explored a range of search algorithms and heuristics that vary in complexity and power (e.g., grid search, hill-climbing, means-ends, etc.). Our model implemented a relatively simple form of search based on stabilizing and destabilizing PFC representations as a function of task performance, which amounts to random sampling with delayed replacement. This appeared to be sufficient for the performance of the simple tasks addressed, consistent with previous results (*10, 21*). An important focus of future research will be to identify neural mechanisms that implement more sophisticated forms of search. This is likely to be necessary to account for human performance in more complex problem solving domains, in which the PFC plays a critical role (*22*).

Representation vs. Process

It should be clear that our model of PFC function relies heavily on the nature of its representations, which contrasts with other approaches that focus instead on processes (e.g., manipulation, inhibition, and planning). Although it is often difficult to clearly distinguish these approaches, it has been argued that representational accounts provide a better fit with the body of existing data regarding how the brain stores and processes information (*23*). We concur with this view, and add that neural network models provide a particularly clear mechanistic perspective on these issues. Nevertheless, there may be important differences in the types of representations developed in our model compared to others (*23*), that we hope to explore in future modeling and empirical work.

Model Equations

All of the models were implemented by using the Leabra framework, which incorporates a number of standard neural network mechanisms (e.g., point-neuron biophysical activation function, bidirectional excitatory connectivity, inhibitory competition, and Hebbian and error-driven learning) (*24, 25*). This framework has been used to simulate over 40 different models (*25*), and a number of other research models, including some that are closely related to that used here (*8, 10*). Thus, models can be viewed as instantiations of a systematic modeling framework using standardized mechanisms, instead of constructing new mechanisms for each model. In keeping with this philosophy, default parameters and mechanisms were used in this model. All models and supplementary code can be obtained by email to oreilly@psych.colorado.edu.

Pseudocode

The pseudocode for Leabra is given here, showing exactly how the pieces of the algorithm described in more detail in the subsequent sections fit together.

```
Outer loop: Iterate over events (trials) within an epoch. For each event:

1. Iterate over minus and plus phases of settling for each event.

   (a) At start of settling, for all units:
```

        i. Initialize all state variables (activation, v_m, etc).
        ii. Apply external patterns (clamp input in minus, input and output in plus).
    (b) During each cycle of settling, for all non-clamped units:
        i. Compute excitatory netinput ($g_e(t)$ or η_j, Eq. 2).
        ii. Compute kWTA inhibition for each layer, based on g_i^{Θ} (Eq. 6):
            A. Sort units into two groups based on g_i^{Θ}: top k and remaining $k+1$ to n.
            B. If basic, find k and $k+1th$ highest; if avg-based, compute avg of $1 \rightarrow k$ & $k+1 \rightarrow n$.
            C. Set inhibitory conductance g_i from g_k^{Θ} and g_{k+1}^{Θ} (Eq. 5).
        iii. Compute point-neuron activation combining excitatory input and inhibition (Eq. 1).
    (c) After settling, for all units:
        i. Record final settling activations as either minus or plus phase (y_j^- or y_j^+).

2. After both phases update the weights (based on linear current weight values), for all connections:
    (a) Compute error-driven weight changes (Eq. 7) with soft weight bounding (Eq. 8).
    (b) Compute Hebbian weight changes from plus-phase activations (Eq. 9).
    (c) Compute net weight change as weighted sum of error-driven and Hebbian (Eq. 10).
    (d) Increment the weights according to net weight change.

Point Neuron Activation Function

Leabra uses a *point neuron* activation function that models the electrophysiological properties of real neurons, while simplifying their geometry to a single point. This function is nearly as simple computationally as the standard sigmoidal activation function, but the more biologically based implementation makes it considerably easier to model inhibitory competition, as described below. Further, using this function enables cognitive models to be more easily related to more physiologically detailed simulations, thereby facilitating bridge-building between biology and cognition.

The membrane potential V_m is updated as a function of ionic conductances g with reversal (driving) potentials E as follows:

$$\Delta V_m(t) = \tau \sum_c g_c(t) \overline{g_c} (E_c - V_m(t)), \tag{1}$$

with three channels (c) corresponding to: e excitatory input, l leak current, and i inhibitory input. Following electrophysiological convention, the overall conductance is decomposed into a time-varying component $g_c(t)$ computed as a function of the dynamic state of the network, and a constant $\overline{g_c}$ that controls the relative influence of the different conductances.

The excitatory net input/conductance $g_e(t)$ or η_j is computed as the proportion of open excitatory channels as a function of sending activations times the weight values:

$$\eta_j = g_e(t) = \langle x_i w_{ij} \rangle = \frac{1}{n} \sum_i x_i w_{ij}. \tag{2}$$

The inhibitory conductance is computed via the k-Winners-Take-All (kWTA) function described in the next section, and leak is a constant.

Activation communicated to other cells (y_j) is a thresholded (Θ) sigmoidal function of the membrane potential with gain parameter γ:

$$y_j(t) = \frac{1}{\left(1 + \frac{1}{\gamma [V_m(t) - \Theta]_+}\right)}, \tag{3}$$

where $[x]_+$ is a threshold function that returns 0 if $x < 0$ and x if $X > 0$. Note that if it returns 0, we assume $y_j(t) = 0$, to avoid dividing by 0. As it is, this function has a very sharp threshold, which interferes with graded learning learning mechanisms (e.g., gradient descent). To produce a less discontinuous deterministic function with a softer threshold, the function is convolved with a Gaussian noise kernel ($\mu = 0$, $\sigma = .005$), which reflects the intrinsic processing noise of biological neurons:

$$y_j^*(x) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-z^2/(2\sigma^2)} y_j(z - x) dz, \tag{4}$$

where x represents the $[V_m(t) - \Theta]_+$ value, and $y_j^*(x)$ is the noise-convolved activation for that value. In the simulation, this function is implemented by using a numerical lookup table.

kWTA Inhibition

Leabra uses a kWTA function to achieve inhibitory competition among units within a layer (area). The kWTA function computes a uniform level of inhibitory current for all units in the layer, such that the $k+1$th most excited unit within a layer is below its firing threshold, whereas the kth is above threshold. Activation dynamics similar to those produced by the kWTA function have been shown to result from simulated inhibitory interneurons that project both feedforward and feedback inhibition (*25*). Thus, although the kWTA function is somewhat biologically implausible in its implementation (e.g., requiring global information about activation states and using sorting mechanisms), it provides a computationally effective approximation to biologically plausible inhibitory dynamics.

kWTA is computed via a uniform level of inhibitory current for all units in the layer, as follows:

$$g_i = g_{k+1}^{\Theta} + q(g_k^{\Theta} - g_{k+1}^{\Theta}), \tag{5}$$

where $0 < q < 1$ (0.25 default used here) is a parameter for setting the inhibition between the upper bound of g_k^{Θ} and the lower bound of g_{k+1}^{Θ}. These boundary inhibition values are computed as a function of the level of inhibition necessary to keep a unit right at threshold:

$$g_i^{\Theta} = \frac{g_e^* \bar{g}_e (E_e - \Theta) + g_l \bar{g}_l (E_l - \Theta)}{\Theta - E_i}, \tag{6}$$

where g_e^* is the excitatory net input without the bias weight contribution — this allows the bias weights to override the kWTA constraint.

Hebbian and Error-Driven Learning

For learning, Leabra uses a combination of error-driven and Hebbian learning. The error-driven component computes essentially the same error derivatives as the widely used error backpropagation learning mechanism but does so by using bidirectional activation flow instead of biologically implausible error backpropagation. Specifically, we use the symmetric midpoint version of the GeneRec algorithm (*26*), which is functionally equivalent to the deterministic Boltzmann machine and contrastive Hebbian learning (CHL). The network settles in two phases, an expectation (minus) phase where the network's actual output is produced, and an outcome (plus) phase where the target output is experienced. The simple difference of the pre and postsynaptic activation product across these two phases produces the learning gradient:

$$\Delta_{err} w_{ij} = (x_i^+ y_j^+) - (x_i^- y_j^-), \tag{7}$$

which is subject to a soft-weight bounding to keep within the $0 - 1$ range:

$$\Delta_{sberr} w_{ij} = [\Delta_{err}]_+ (1 - w_{ij}) + [\Delta_{err}]_- w_{ij}. \tag{8}$$

For Hebbian learning, Leabra uses essentially the same learning rule used in competitive learning or mixtures-of-Gaussians, which can be seen as a variant of the Oja normalization. This learning rule extracts the principal component of variance across the input units over time and is thus important for allowing the units to encode the statistical structure of their environment. The equation for the Hebbian weight change is:

$$\Delta_{hebb} w_{ij} = x_i^+ y_j^+ - y_j^+ w_{ij} = y_j^+ (x_i^+ - w_{ij}). \tag{9}$$

The two terms are then combined additively with a normalized mixing constant k_{hebb}:

$$\Delta w_{ij} = \epsilon[k_{hebb}(\Delta_{hebb}) + (1 - k_{hebb})(\Delta_{sberr})]. \tag{10}$$

Temporal Differences and Adaptive Critic Gating Mechanisms

The adaptive gating mechanism was based on an adaptive critic unit (AG) that updated its activations according to the temporal-differences (TD) algorithm. The specific mechanism and its motivations are discussed in detail in ref. 9. The basic idea is that the AG unit computes expected reward values, with reward delivered as a function of network performance. When the network performs better than expected, the consequent positive "delta" in reward expectation leads to the activation of persistent intrinsic excitatory currents in PFC neurons. This stabilizes the PFC representations, enabling them to maintain information robustly over time. When the network performs worse than expected, the negative delta leads to decrements in these PFC maintenance currents, which destabilizes the representations and allows new PFC activity patterns to emerge. Thus, when the task rule changes, the resulting performance errors cause the AG to destabilize the PFC representations, allowing a more task-appropriate representation to become active. When such a representation does become active, the resulting positive feedback stabilizes these representations.

The AG unit was implemented in Leabra using plus-minus phase states that correspond to the expected reward at the previous time step (minus) and the current time step (plus). The difference between these two states is the TD error δ, which is essentially equivalent to the more standard kinds of error signals computed by the error-driven learning component of Leabra, except that it represents an error of prediction over time, instead of an instantaneous error in the network output. This δ value then modulates the strength of an excitatory ionic current (labeled m here) that helps to maintain PFC activations:

$$g_m(t-1) = 0 \text{ if } |\delta(t)| > r \tag{11}$$

$$g_m(t)_j = g_m(t-1) + \delta(t) y_j(t). \tag{12}$$

Thus, a positive δ increases this maintenance current for active units, and a negative δ decreases it. Furthermore, if δ is sufficiently large in magnitude (greater than the reset threshold $r = 0.5$), it resets any existing currents.

To prevent gating from occurring for occasional errors, we included a fault-tolerant mechanism where two errors in a row had to be made before the AG unit was delivered a 0 reward value. Furthermore, we included rapidly adapting and decaying negative-only bias weights, which serve to inhibit prior task representations (again see ref. 9 for more detailed motivations and explorations of this mechanism). These bias weights in the PFC layer learn on the plus-minus phase difference in unit activations, as normal bias weights do, but they learn only negative values, with a learning rate of 0.2 (compared to the standard 0.01 for the remainder of the network). These biases decay with a time constant of 0.5 (i.e., 0.5 of the existing weight value is subtracted on each trial).

1. Emerson, M. J & Miyake, A. (2003) *Journal of Memory and Language* **48**, 148–168.

2. Luria, A. R. (1961) *The role of speech in the regulation of normal and abnormal behavior*. (Liveright Publishing Corporation, New York).

3. Vygotsky, L. (1934/1986) *Thought and Language*. (MIT Press (Original work published in 1934), Cambridge, MA).

4. Barbas, H & Pandya, D. N. (1989) *Journal of Comparative Neurology* **286**, 353–375.

5. Fuster, J. M. (1997) *The Prefrontal Cortex: Anatomy, Physiology and Neuropsychology of the Frontal Lobe, 3rd Edition.* (Lippincott-Raven, New York).

6. Elman, J. L. (1990) *Cognitive Science* **14**, 179–211.

7. O'Reilly, R. C & Frank, M. J. (submitted).

8. Rougier, N. P & O'Reilly, R. C. (2002) *Cognitive Science* **26**, 503–520.

9. O'Reilly, R. C, Braver, T. S, & Cohen, J. D. (1999) in *Models of Working Memory: Mechanisms of Active Maintenance and Executive Control.*, eds. Miyake, A & Shah, P. (Cambridge University Press, New York), pp. 375–411.

10. O'Reilly, R. C, Noelle, D, Braver, T. S, & Cohen, J. D. (2002) *Cerebral Cortex* **12**, 246–257.

11. Dias, R, Robbins, T. W, & Roberts, A. C. (1997) *Journal of Neuroscience* **17**, 9285–9297.

12. Stuss, D. T, Floden, D, Alexander, M. P, Levine, B, & Katz, D. (2001) *Neuropsychologia* **39**, 771–786.

13. Stuss, D. T, Levine, B, Alexander, M. P, Hong, J, Palumbo, C, Hamer, L, Murphy, K. J, & Izukawa, D. (2000) *Neuropsychologia* **38**, 388–402.

14. Herd, S. A, Banich, M. T, & O'Reilly, R. C. (in press) *Journal of Cognitive Neuroscience*.

15. Cohen, J. D, Dunbar, K, & McClelland, J. L. (1990) *Psychological Review* **97**, 332–361.

16. Banich, M. T, Milham, M. P, Atchley, R, Cohen, N. J, Webb, A, Wszalek, T, Kramer, A. F, Liang, Z. P, Barad, V, Gullett, D, Shah, C, & Brown, C. (2000) *Cognitive Brain Research* **10**, 1–9.

17. Banich, M. T, Milham, M. P, Atchley, R, Cohen, N. J, Webb, A, Wszalek, T, Kramer, A. F, Liang, Z. P, Wright, A, Shenker, J, & Magin, R. (2000) *Journal of Cognitive Neuroscience* **12**, 988–1000.

18. Mountain, M. A & Snow, W. G. (1993) *The Clinical Neuropsychologist* **7**, 108–118.

19. Reitan, R. M & Wolfson, D. (1994) *Neuropsychology Review* **4**, 161–198.

20. Newell, A & Simon, H. A. (1972) *Human Problem Solving.* (Prentice-Hall, Englewood Cliffs, NJ).

21. Dehaene, S & Changeux, J. P. (1991) *Cerebral Cortex* **1**, 62–79.

22. Baker, S. C, Rogers, R. D, Owen, A. M, Frith, C. D, Dolan, R. J, Frackowiak, R. S. J, & Robbins, T. W. (1996) *Neuropsychologia* **34**, 515.

23. Wood, J. N & Grafman, J. (2003) *Nature Reviews Neuroscience* **4**, 139–147.

24. O'Reilly, R. C. (1998) *Trends in Cognitive Sciences* **2**, 455–462.

25. O'Reilly, R. C & Munakata, Y. (2000) *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*. (MIT Press, Cambridge, MA).

26. O'Reilly, R. C. (1996) *Neural Computation* **8**, 895–938.