which simply tabulates the Euclidean distance between firing rate patterns for each pair of conditions. This dissimilarity matrix is then centered, diagonalized, and projected along the first 3 eigenvectors rescaled by the squared root of the corresponding eigenvalues, in accordance with the Classical Multidimensional Scaling algorithm (Borg & Groenen 2003).

SUPPLEMENTAL VIDEOS

Supplemental Video 1, related to Fig. 3: Multi-dimensional scaling representation for HPC

Supplemental Video 2, related to Fig. 3: Multi-dimensional scaling representation for DLPFC

Supplemental Video 3, related to Fig. 3: Multi-dimensional scaling representation for ACC

Methods S1 Simulations of the parity/magnitude task: dependence on hyperparameters (Related to Figure 7)

The simple simulations described in the main text lead to representations that exhibit a geometry which is very similar to the one observed in the data. In particular, we observe high CCGP for multiple variables and high SD. How do these results depend on the parameters of the network? The results of extensive simulations will be reported in a different publication. Here we summarize some of the results:

Dependence on the number of neurons. We varied the number of neurons in the second layer, ranging from 2 to 100. The observed geometry is qualitatively and quantitatively similar to the one reported in the main text, with one exception: for a small number of neurons the SD is much lower, as expected. CCGP is still relatively high even for the extreme case of 2 neurons.

Dependence on the number of layers. The two hidden layers we used in the article are not necessary. One layer is sufficient to get high CCGP for both parity and magnitude and high SD.

Dependence on the number of variables. We added to the output two units that encode the third possible abstract dichotomy. The geometry after training has high CCGP for all 3 variables, and SD is high as well. Scaling beyond 3 variables requires a different dataset (with at least 16 classes), and it remains to be studied.

Dependence on the output patterns. In the simulations presented, we have two units for each dichotomy (2 for parity and 2 for magnitude). One unit per dichotomy leads to the same geometry. Instead, when we specified a categorical output (8 units, one for each individual digit), CCGP is not different from chance. Also when each output unit encodes one of the four combinations of parity and magnitude, CCGP is not different from chance.

Dependence on the number of training epochs. We studied how CCGP, SD and another measure of dimensionality (participation ratio, PR) change during training (1000 training epochs in total). CCGP starts low, and it reaches a maximum after 50 epochs, and then it slightly decreases. SD starts relatively high, as expected for initial random conditions, and it grows monotonically up until 10 epochs, where it saturates. By contrast, PR exhibits non-monotonic behavior: it first grows, and then after 100 epochs, it decreases. This confirms our intuition that SD can be different from other more direct measures of dimensionality, like PR.

Methods S2 Clustering index as a measure of abstraction (Related to Figure 2)

A simple way to achieve a neural representation in an abstract format involves clustering together the activity patterns corresponding to the conditions on either side of a certain dichotomy that define a binary variable. For example, if the spike count patterns in a certain brain area were equal for all four stimuli in context one, and similarly coincided (at a different value) for context two, this area would represent context in an abstract format, but it would be at the expense of not encoding any information about stimulus identity, operant action or reward value.

We can assess the degree of clustering by comparing the average distance between the mean neural activity patterns corresponding to the conditions on the same side of a dichotomy (within or intra-group) to the average distance between those on opposite sides of the dichotomy (between or inter-group). For balanced (4 vs. 4) dichotomies of the eight experimental conditions (context, value and action of the previous trial), there are 16 inter-group distances, and 12 intra-group distances (six on each side of the dichotomy) that contribute. We define the ratio of the average between group distance to the average within group distance as the **abstraction index**, which measures *the degree of clustering of a set of neural activity patterns associated with a certain dichotomy*. In the absence of any particular geometric structure (such as clustering), we would expect these two average distances to be equal, resulting in an abstraction index of one, while clustering would lead to values larger than one.

Fig. S4d shows the abstraction index for the context variable computed from the measured neural activity patterns in HPC, DLPFC, and ACC. As above for decoding, we retain only correct trials without a contextual frame that didn't occur within 5 trials of a context switch for this analysis. We z-score the overall activity distribution (across all task conditions) of each neuron before computing the mean activity pattern of each condition, and use a simple Euclidean distance metric (we verified that employing a Mahalanobis distance metric instead of the Euclidean distance metric yields similar results).

Clustering can also identify an abstract variable (as verified by CCGP) when the geometry of a neural representation differs from that shown in Figure 2b. For example, when the points of 4 different conditions are arranged in a square, as in Figures 2c,d, the inter-class distances are slightly larger than the intra-class distances (the diagonals connecting opposite points are larger than the length of the side of the square). The geometry is thereby clearly less clustered than in the case discussed in Figure 2b, but nonetheless the analysis could lead to the correct conclusion that the representation is abstract. However, the abstraction index, though larger than 1, would be significantly smaller than in the case of clustering illustrated in Figure 2b, and we do not know how robust this measure would be to noise.

We emphasize that although clustering can identify variables in an abstract format, there are several situations in which a clustering analysis fails to identify abstract variables, even though these variables are identified by CCGP. For example, in the geometry in Figure S4e, a cube is squeezed along the vertical direction. Here context is represented in an abstract format as we define it because CCGP would be close to 1 when a decoder is trained on 6 conditions, 3 per context, and tested on the remaining ones, so long as noise is not too large. However, an analysis of clustering would not identify context as being represented in an abstract format. This is because when the vertical side of the rectangular cuboid is d and the sides of the squares are 1, then for any $d < 0.634$ the average intra-group distance is larger than the average inter-group distance. In these cases, the abstraction index is less than 1. The geometry depicted in Figure S4e is not an unlikely situation because different abstract variables are typically encoded with different strengths (in the referenced figure, action and value are represented more strongly than context).

Methods S3 Relation between CCGP and decoding performance in classification tasks (Related to Figure 2)

We motivate our CCGP measure of abstraction operationally: abstraction of one variable is quantified in terms of performance at predicting the value of the variable in conditions not used for training. In particular, we measure CCGP of the neural response vector representation of context (or any other variable corresponding to a dichotomy) by training a weak classifier (a linear one) to decode context from response vectors in a subset of training experimental conditions, and then testing the trained classifier on a held out subset of experimental conditions (which is analogous to ”novel” conditions to the decoder). Intuitively, the accuracy of the classifier in predicting context on the held out experimental conditions quantifies how "consistently" context is being encoded across conditions. So, for instance CCGP would be very low for response patterns in generic random position. Indeed, it would be straight-forward to train a linear classifier with high accuracy (in a low noise regime), but the same classifier would perform at chance level on the response patterns recorded on held out conditions.

This should also clarify that, despite the partial homonymy, CCGP is not a synonym of generalization performance, the quantity that is the central object of statistical learning theory. In fact, generalization performance typically refers to testing the classifier on new data that is sampled from the SAME conditions used at training, while CCGP (it is important to stress) tests the classifier on new data from conditions DISTINCT from those used for training.

Analogously, quantities that are the purview of statistical learning theory and the empirical risk minimization approach in particular might seem related to our measure of abstraction to a deceptively high degree. For instance, the VC-dimension of a class of, say, maximum margin classifiers provides a PAC bound on their generalization performance as a function of sample complexity and training performance. So, it might seem superficially related to CCGP. What is important to keep in mind, however, is that we are not operating under the classical assumptions of statistical learning theory, as we don't have the same goals. In particular, we're are not interested in assuming that the held out response patterns are sampled from the same distribution as the training patterns (the so-called sampling hypothesis). On the contrary, we are assuming that the held out response patterns are being generated from unsampled experimental conditions, previously unseen by the classifier. And what we care about is measuring classification accuracy despite the covariate shift resulting from being presented with new experimental conditions. In other words, as profusely explained in the paper, we view abstraction as being supported by whatever geometrical property will guarantee generalization in previously unseen experimental conditions. Accordingly, if one really wanted to find a parallel between CCGP in our experiments and generalization performance within a statistical learning framework, it would be in the specific setting of domain adaptation, which investigates the classification performance in a target domain of classifiers trained in a different source domain. This is a theoretical connection that might be worth exploring in future work.

Methods S4 PCA Dimensionality of the neural representations (Related to Figures 3 and 5)

We utilize a technique developed in (Machens et al. 2010) to estimate a lower bound for the dimensionality of the neural response vectors in a specific time bin during a task. Notice that this measure of dimensionality is different from the shattering dimensionality (SD) that we discussed extensively in the main text. The measure describes more directly the actual geometry of the representations and it is similar to the PCA dimensionality. By contrast, the shattering dimensionality is related to the ability of a linear decoder to classify many different dichotomies. It is possible for the PCA dimensionality to be low while the SD is high. Similar to our other analyses, to compute the PCA dimensionality we build average firing rate patterns for all recorded neurons by averaging spike counts sorted according to task conditions indexing the trial where the activity is recorded (current trial) or the previous trial. The spike counts are z-scored and averaged in 500 ms time bins displaced by 50 ms throughout the trial. We then apply the method presented in (Machens et al. 2010) on the obtained average firing rate activity patterns independently within each 500 ms time bin. This procedure allows us to bound the number of linear components of the average firing rate patterns that are due to finite sampling of the noise, therefore providing an estimate of their dimensionality.

Figure S3c shows the result of this analysis for all neurons recorded in HPC, DLPFC and ACC for which we had at least 15 trials per condition. For average firing rate patterns obtained by sorting spike counts according to the 8 conditions of the current trial (continuous lines), dimensionality peaks at its maximum possible value shortly after the presentation of the image for all three areas. The dimensionality for firing rate patterns obtained by sorting the activity according to the condition of the previous trial remains around 5 throughout the trial, which is close to the value to which dimensionality in the current trial decays towards the end of the trial.

Methods S5 The trade off between dimensionality and our measures of abstraction (Related to Figure 2)

A high degree of abstraction does not necessarily entail low shattering dimensionality. At first sight, it might seem that the opposite is true. In fact, the class of neural geometries we denote as abstract require neurons to (at least approximately) exhibit linear mixed selectivity (Rigotti et al. 2013; Fusi et al. 2016), which would imply that these neural representations have low dimensionality. This is because the dimensionality of such a geometry would be equal to the number of the (in our case binary) abstract variables involved (not counting the possible offset from the origin of the neural activity space). This dimensionality is small compared to the total number of conditions (minus one), which corresponds to the maximal possible dimensionality achieved if all neural response vectors are linearly independent. In the idealized case in which all abstract variables have a parallelism score of one, the representations of the different conditions coincide with the vertices of a parallelotope, and it is easy to see that the dimensionality equals the number of variables, which is much lower than the maximal dimensionality.

It has been argued (Rigotti et al. 2013; Fusi et al. 2016) that high-dimensional neural representations are often important for flexible computation, because a downstream area may in principle have to read out an arbitrary dichotomy of the different conditions (i.e., an arbitrary binary function) to solve different tasks. This can be achieved using simple linear classifiers (used to model the type of computation that a readout neuron may be able to implement directly) only if the dimensionality of the neural representation is maximal. This desire for flexible computations afforded by high dimensionality seems to be in direct opposition to the low dimensionality implied by the abstract neural geometries that allow for cross-condition generalization.

However, there is in fact a large class of geometries that combine close to maximal dimensionality with excellent generalization properties for a number of abstract variables (which form a preferred subset of all dichotomies). Maximal dimensionality implies decodability by linear classifiers of almost all dichotomies. We refer to this classifier-based measure of dimensionality as the 'shattering dimensionality'. This quantity is similar to the one introduced in (Rigotti et al. 2013), where it was measured to be maximal in neural representations in monkey pre-frontal cortex. The geometries with high dimensionality and excellent generalization don't have unit parallelism scores for the abstract variables (they are not exactly factorizable). A simple way to construct examples of such geometries is to start from a simple factorizable case, namely a cuboid, and then distort it to reduce the parallelism score.

We illustrate this cuboid geometry in the case of eight conditions. Here there are at most three completely abstract variables, as in our experimental data. We can generate an artificial data set with the desired properties by arranging the eight conditions initially at the corners of a cube (with coordinates plus/minus one), embedding this cube in a high-dimensional space by padding their coordinate vectors with zeros (here we use $N = 100$ dimensions, so we append 97 zeros), and acting on them with a random (100-dimensional) rotation to introduce linear mixed selectivity. We then distort the cube by moving the cluster center of each condition in a random direction - chosen independently and isotropically - by a fixed distance, which parameterizes the magnitude of the distortion. This operation reduces the parallelism score for the three initially perfectly abstract variables, which correspond to the three principal axes of the original cube, to values less than one. We sample an artificial data set of 1,000 data points per condition by assuming an isotropic Gaussian noise distribution around each cluster center.

On this data set we can run the same analyses as on our experimental data. In particular, we compute the parallelism score and the cross-condition generalization performance averaged over the three preferred (potentially abstract) dichotomies, with the results of these analyses shown in Fig. S2. In addition to these quantities, we compute the average decoding performance across all 35 balanced dichotomies, which defines the shattering dimensionality (SD). Since the upper bound on SD is one, shattering dimensionality is essentially a normalized measure of dimensionality as described by (Rigotti et al. 2013). For small noise, the mean parallelism score of the three abstract variables starts very close to one and from there decreases monotonically as a function of the magnitude of the distortion. The same is true for their mean CCGP, but its decline is much more gradual and almost imperceptible for small distortions. This means that for intermediate values of the displacement magnitude (of order one), we still see excellent generalization properties across conditions, and the three preferred variables are still in an abstract format.

In contrast, the SD increases as a function of the magnitude of the distortion, due to the increasing dimensionality of the representation. Balanced dichotomies include the most difficult (least linearly separable) binary functions of a given set of conditions. If all these dichotomies can be decoded by linear classifiers, the dimensionality of the neural representation will be maximal. For small values of the displacement magnitude (for which the neural geometry is three-dimensional), some dichotomies are clearly not linearly separable. Therefore, SD is initially substantially less than one, but it steadily increases with the degree of distortion of the cube. Crucially, it reaches its plateau close to the maximal value of one before the CCGP drops substantially below one. This indicates that there is a parameter regime in which this type of neural geometry exhibits almost maximal dimensionality, which enables flexible computation, but at the same time the geometry also enables abstraction (in the form of excellent CCGP) for the three preferred variables. Therefore, we can conclude that these two favorable properties (high dimensionality and high CCGP) are not mutually exclusive.

In the case of larger noise, the PS and CCGP start out at values substantially smaller than one already for zero distortion. Although the qualitative trends of all the quantities discussed remain the same, the tradeoff between SD and CCGP (i.e., between flexible computation and abstraction) is much more gradual under these circumstances.

Methods S6 Selectivity and abstraction in a general linear model of neuronal responses (Related to Figure 2)

We model the selectivity of a neuron to the various task variables in a given time bin as FR with the following linear regression model (see also Supplemental Material of (Rigotti et al. 2013)):

$$\mathrm{FR} = \beta_0 + \sum_{i=1,2} \beta_i^{ctxt} \left[\mathrm{ctxt} = i \right] + \sum_{j=\mathrm{A,B,C,D}} \beta_j^{cue} \left[\mathrm{cue} = j \right] + \eta, \qquad (1)$$

where the terms in square brackets represent a design matrix whose elements indicize the type of trial according to context type and cue identity. The coefficient β_0 is a bias and η represents a Gaussian noise term. Fitting the model described by equation (1) to the firing rate of a neuron is related to performing a 2-way ANOVA, where the two factors correspond to context type, and cue identity. The β coefficients that are determined by the fit quantify the selectivity to their corresponding aspects. Since equation (1) does not include any interaction term among the factors, the model is linear. Because of this, the corresponding neuron would be denoted as a *pure selectivity neuron* neuron if only β's corresponding to one factor are significantly non-zero, or it would be denoted as a *linear mixed selectivity neuron* in case at least one β for each factor is significantly non-zero (see (Rigotti et al. 2013)).

The model (1) can be complemented with additional factors aimed at fitting the deviation of the activity FR from a linear model. These are typically included as 1-factor interaction terms:

$$\mathrm{FR}_{\mathrm{nlin}} = \mathrm{FR} + \sum_{(i,j)=(1,A),\ldots,(2,D)} \beta_{ij}^{ctxt,cue} \left[\mathrm{ctxt} = i \right] \cdot \left[\mathrm{cue} = j \right]. \qquad (2)$$

Whenever any coefficient of an interaction term is significantly different from zero, the neuron is denoted as a *nonlinear mixed selectivity neuron*. Nonlinear interaction terms increase the dimensionality of the neural response patterns, which benefit the downstream linear decoding(Rigotti et al. 2013). However, increasing dimensionality is generally at odds with abstraction as discussed in Section Methods S5.

Linear mixed selectivity on the other hand does not increase dimensionality beyond the dimensionality of a population of pure selectivity neurons that collectively display all the factors explaining the activity of the pure selectivity neurons (see (Rigotti et al. 2013)).

Methods S7 Recorded neurons are not highly specialized (Related to Figure 4)

An ensemble of neurons could in principle encode multiple abstract variables simply by assigning each neuron to be tuned to precisely one of these variables. In this case, the ensemble can be divided into a number of subpopulations each of which exhibits pure selectivity for one of the abstract variables. Geometrically, this situation is similar to the one depicted in Fig. 2c, when the square is aligned to the coordinate axes. Fig. 2c shows the situation in which neurons exhibit mixed selectivity. This rotated representation would show the same generalization properties. In the data, we do not observe many pure selectivity neurons. If neurons were highly specialized for particular variables, training a linear classifier to decode that variable should lead to very large readout weights from the associated specialized subpopulation, but very small readout weights from other neurons (which might specialize on encoding other variables). Therefore, in a scatter plot of the (absolute values) of the readout weights for different classifiers we would expect specialized neurons to fall close to the axes (with large weights for the preferred variable, but small ones for any others). If this situation occurred for a large number of neurons, we might expect a negative correlation between the absolute values of the decoding weights for different variables. However, in fact the weights do not cluster close to the axes, as shown in Fig. S6. The correlation coefficients of their absolute values are positive. We conclude that highly specialized neurons are not particularly common in the neural ensembles we recorded, i.e. pure selectivity appears to be no more likely than coding properties corresponding to a random linear combination of the task-relevant variables.

For each neuron we can consider the three-dimensional space of the readout weights for the three variables (components of three different unit-norm weight vectors in the space of neural activities). Clearly neurons with large readout weights for the linear classifier trained to read out a particular variable are important for decoding that variable. Some neurons have larger total readout weights than others (e.g. we can consider the sum of squares of the three weight components, corresponding to the squared radius in the three-dimensional space) and are therefore more useful for decoding overall than others which have only small readout weights.

We can also rank neurons according to the angle of their readout weight vector from the axis associated with one of the variables in the three-dimensional space. This quantifies the degree of specialization of the neuron for the chosen variable. We call the absolute value of the cosine of this angle the pure selectivity index. We can now ask whether neurons with a large pure selectivity index are particularly important for generalization, as quantified by the cross-condition generalization performance (CCGP). This can be tested by successively removing neurons with the largest pure selectivity indices from the data and performing the CCGP analysis on the remaining population of (increasingly) mixed selectivity neurons. The results of this ablation analysis are shown in Fig. S6b, in which we plot the decay of the CCGP with the number of neurons removed. It demonstrates that while pure selectivity neurons are important for generalization (as expected in a pseudo-simultaneous population of neurons with re-sampled trial-to-trial variability, in which the principal axes of the noise clouds are aligned with the neural axes), they are not more important than neurons with overall large decoding weights which typically have mixed selectivity.

We can perform the same analyses also for the neural network model of Fig. 7. We focus on the neural representations obtained from the second hidden layer of the trained network. Fig. S6c shows a scatter plot of the decoding weights for the parity and magnitude variables. As in the neural recordings, many neurons exhibit mixed selectivity. We can again successively ablate neurons with the largest pure selectivity indices. Unlike for the case of the neural data, this only has a small effect on CCGP (see Fig. S6c). Presumably this is because here the neural activities are simultaneously ‘recorded’, and therefore the decoder can use information contained in the correlations between different neurons in these representations. Unlike the case of resampled noise in a pseudo-simultaneous population, the principal axes of the noise clouds are not necessarily aligned with the neural axes. For such a simultaneously observed representation there is no reason why pure selectivity neurons should contribute preferentially to the generalization ability of the network.

Methods S8 Deep neural network models of task performance (Related to Figure 7)

The study of performance-optimized neural networks is becoming increasingly common practice in computational neuroscience. This technique consists in training an artificial neural network model on the simulated version of a behavioral task of interest, and examining the neural activations generated by the neural network during or after learning the task. Following such practice, we train neural networks to perform a simulated version of our task under two learning paradigms and we show that after learning the neural representations are similar to those recorded in the experiment. In particular, we show that the learning algorithms that we use both lead to neural representations in which the hidden variable context is in an abstract format, and multiple abstract task-relevant variables are simultaneously represented. We assume that the inputs provided to the neural network are the following observable quantities: the stimulus, action and outcome in the previous trial, and the stimulus in the current trial. It is therefore intriguing that through the process of learning to execute the context-dependent task, the network learns to internally represent context (a latent variable not explicitly represented in the task), and moreover that this representation is in an abstract format as quantified by our metrics. The functional role of this abstract context representation is to modulate the internal representation of the current stimulus, as demonstrated by the fact that the trained network selects the correct action in response to a stimulus in each given context.

As a first general learning paradigm we investigate *Reinforcement Learning*: our neural network model simulates the behavior of the animals engaged in the task and is trained by trial and error to maximize its performance on the simulated task. We then take a drastic simplification step and model the interaction between the animal and its environment during the task as a *supervised learning* paradigm: our network model is essentially trained to output the correct (assumed to be known) behavioral response and predict the resulting outcome, given the current stimulus, and the sequence of stimulus, correct response and outcome in the previous trial. As we will see, this simplification still preserves the main features of the activity patterns generated by the more complex Reinforcement Learning model, while at the same time providing us with direct control of the statistics of the inputs presented at training (which are no longer dictated by the behavior of the model itself), thereby enabling us to systematically study their effect on the abstraction of the variables emerging during learning of the task.

Reinforcement Learning model. We simulate a sequential version of the task that is used as an environment to train by Reinforcement Learning a decision policy (an agent) parametrized as a multi-layer network with two hidden layers (whose size is indicated in Figure S7c), three outputs (one for each of the actions available to the agent), and ReLU activations.

In order to simplify the modeling, we decided to explicitly encode as external inputs all the information necessary to determine the current context. In particular, in addition to the current stimulus, we also provide to the network the previous stimulus, the previous action, and the resulting previous outcome. Previous stimulus, action and outcome are necessary and sufficient to determine the current context (assuming there was no switch between the previous and current trial). One might postulate that these inputs are internally maintained through, for instance, some recurrent synaptic circuit. However, representing the inputs explicitly does not qualitatively affect the model, besides allowing us to eschew a recurrent architecture in favor of a simpler feed-forward one. Notice, moreover, that even if the model were to directly maintain an encoding of context, information about the outcome in the previous trial (together with previous stimulus and action) is essential in order to correctly predict the correct action after a contextual switch. As we will detail in the next paragraphs, the result of this modeling exercise is that, indeed, after training, our model internally encodes the current context when previous stimulus, action and outcome are supplied to the network. Moreover, context is being represented in an "abstract format" (high CCGP and parallelism score), and this modulates the hidden layer representation of the current stimulus to result in the selection of the correct context-dependent action.

In constructing the model, all observable inputs (information observable in the environment) are encoded and presented to the network as binary input vectors with added Gaussian noise sampled independently for every trial. Figure S7a details the construction of these input vectors. At the beginning of a simulation we randomly sample 32-dimensional binary vectors encoding each of the values of all features appearing in the task (previous stimulus, previous action, previous outcome and current stimulus). An input representing a given task condition is then composed by stacking the binary vectors corresponding to the features describing the condition, and adding centered Gaussian noise with amplitude of 0.4.

An episode in the environment consists of two steps. In a first step corresponding to the pre-stimulus interval in the task, the agent observes the stimulus, action and outcome of the previous trial. The correct action at this step is "Wait", upon which the agent receives a reward of 0. If the agent mistakenly selects "Hold" or "Release", it receives a negative reward of -1 and the episode is terminated. In the second step of each episode, in addition to the observables presented in the first step, the agent observes the stimulus of the current trial (Figure S7a). At this point the agent has to select the correct action between "Hold" and "Release", which depends on the currently active context and the current stimulus identity. Upon selection of the correct action, the agent receives a reward of +1 for a rewarded trial, or a reward of 0 in the case of an unrewarded trial. Again, if the agent selects the wrong action, it receives a negative reward of -1. At this point the episode terminates, and a new one is initiated whose context is chosen to be identical to the one of the previous trial with a probability of 98%. Notice that information about the context identity is never made apparent in the observations, but is only reflected in the determination of the correct action, given the current stimulus. As a consequence, assuming that the context hasn't changed between the previous and current episode, the current context can be inferred from the triplet of previous stimulus, previous action and previous outcome, since these uniquely determine the correct action in the previous episode and therefore its context.

The particular choice of dividing each episode into two steps overlapping two adjacent trials was dictated by the requirement of being able to analyze a trial epoch corresponding to the pre-stimulus interval analyzed in the recorded data. Moreover, explicitly providing information about the previous trial in the input to the network allows us to eschew complicated mechanisms to preserve information across time steps, such as for example having to resort to a recurrent neural network model. Future studies will consider these more complicated mechanisms. The environment modeling choices just described thereby allow us to model the agent with a very simple feed-forward architecture, as depicted in Figure S7a. The only slightly unconventional element in our policy network architecture is the addition of a *reconstruction loss* (Figure S7a, lower right). This consists in a weighted term added to the optimization cost that is proportional to the squared difference between the input to the networks and a linear transformation of the activations of the second hidden layer. The role of this additional cost is to allow us to tune the amount of information that the hidden layer representations encode about the inputs. Practically speaking, this provides a mechanism to counteract a tendency observed in our simulations: larger numbers of training epochs and increasing depth tend to be associated with the representations of previous outcome and action variables in a less abstract format (as measured by CCGP and PS). This effect is analyzed more quantitatively using the supervised learning model in the next section.

The policy network described in the previous paragraph is trained to select the correct action with Deep Q-learning(Mnih et al. 2015). In particular, we use a frozen copy of the policy network that is being updated only every 10 episodes to estimate the Q-values corresponding to an observation-actions-reward triplet randomly sampled from a replay memory buffer of 5000 steps in the environment. After every step, a plastic copy of the policy network is being trained to fit the Q-values estimated from the frozen policy network by using one step of adam-SGD(Kingma & Ba 2014) (with the default parameters in PyTorch (Paszke et al. 2017), aside from a learning rate of 0.005 with a decay factor of 0.999) and mini-batch of size 100. The use of a slowly updated frozen copy of the policy network to estimate the Q-values is a technique that had been proposed in (Mnih et al. 2015), and it helps to stabilize the learning procedure. Moreover, as customary to promote exploration in Reinforcement Learning, actions are selected using an ϵ-greedy algorithm, i.e., they are selected based on the largest Q-value with a probability $1-\epsilon$, and selected uniformly at random with probability ϵ. We set $\epsilon = 0.9$ at the beginning of learning and decrease it exponentially to 0.025 with a time constant of 500 steps. Finally, we use a discount factor of $\gamma = 0.99$.

Figure S7d shows the average learning curve for a population of 100 policy networks initialized at random and trained on random instantiations of the environment described in the previous paragraphs. The blue curve shows the fraction of correct terminal actions within blocks of 50 episodes, averaged across 100 networks, as a function of episode.

The results of analyzing the activity of the Reinforcement Learning model are presented in Figure S7e. All analyses are conducted during the first step in the simulated tasks, which emulates the pre-stimulus interval in the experimental task, on a population of 100 randomly initialized and independently trained networks. The figure shows the results of the analyses averaged over the population of models, both before (grey data points) and after training (orange data points). We compute decoding accuracy (first row of panels), PS (second row) and CCGP (third row) for context (first column), value (second column) and action (third column).

The first column of Figure S7e shows that the *context* cannot be decoded with above-chance accuracy (50%) in the input but can already be decoded with high accuracy starting from the randomly initialized first hidden layer (gray datapoints). This is consistent with previous results demonstrating that large enough networks of randomly connected neurons with non-linear activations generate linearly separable patterns with high-probability (see (Rigotti et al. 2010; Barak et al. 2013; Lindsay et al. 2017)).

The second row of Figure S7e shows that, despite being decodable with high accuracy in both hidden layers, context only becomes appreciably abstract in layer L2, where on average across the models CCGP becomes significantly higher than chance (i.e., 50%). Correspondingly, the average PS in L2 is fairly high at around 0.25 (Figure S7e, first plot in the second row). Moreover, in both hidden layers of the trained models, PS and CCGP for context are significantly higher than in untrained models. In this respect, it is worth emphasizing that for the variable context, the CCGP observed in randomly initialized models is consistently and considerably below chance level. This reflects the anti-correlation structure induced by the contextual character of the task: stimuli have to be associated to opposite context labels in the two contexts, which will naturally induce a linear decoder trained on a subset of conditions to predict the wrong context for the untrained conditions. This structure is also reflected in the parallelism score which tends to be negative for untrained models.

As for the previous value and action variables, they maintain relatively high CCGP and PS after training, but there was a general tendency for representations to become progressively less abstract as a function of training and depth of the layer. This presumably reflects the fact that value and chosen action in the previous trial are not features that are directly predictive of the current value and action, although they can be utilized to compute the current context and therefore provide information that could allow an agent infer the current value and action.

Besides comparing the PS and CCGP of the trained models with those of the untrained models as baseline, we can quantify their statistical significance using the more stringent statistical tests developed to analyze the experimental data. In particular, we can use the geometric random model as a null-distribution to establish the statistical significance of CCGP and use a shuffle test as a null-distribution for the PS (as in Figure 3). Accordingly, Figs. S7f,g show beeswarm plots of the probability that the CCGP and PS of each of the trained Reinforcement Learning models is above the distributions given by the geometric random model and a shuffle test, respectively, for all variables in the task individually (context, previous value, and previous action) and all variables simultaneously for the same model (All). These probabilities are empirically estimated over 100 trained models and 1000 samples of the null-distributions per trained model. Figs. S7f,g also display the mean of the null-distributions (dotted lines) and the 95th percentile (dashed line). Above each plot, the fraction of trained models whose PS and CCGP is above the 95th percentile of the null-distribution is depicted. Therefore, particularly in layer L2, a considerable fraction of trained models display a PS and CCGP above the 95th percentile of the null-distributions. For instance, in L2 40% of the trained models have a PS that is higher than 95% of the shuffle null-distribution simultaneously for all variables, while 11% have a CCGP above 95% of the null-distribution obtained with the random model.

Supervised Learning neural network model. Besides investigating a Reinforcement Learning model of the task, we also devised simulations in a much simpler open-loop learning paradigm: supervised learning. In this case, the network is trained to associate inputs corresponding to the stimuli presented to the monkeys with outputs corresponding to the correct action and expected outcome. Given a task condition, the inputs are generated in the same way as for the Reinforcement Learning model. At the beginning of a simulation we randomly sample 32-dimensional binary vectors encoding each of the values of all features appearing in the task (previous stimulus, previous action, previous outcome, current stimulus, correct action and predicted outcome). An input representing a given task condition is composed by stacking the binary vectors corresponding to the features describing the condition, and adding centered Gaussian noise with amplitude of 0.4. A corresponding output is built by stacking the binary vectors encoding the correct action and predicted outcome in the current condition (see Figure S8a). As for the Reinforcement Learning case, we simulate two task epochs for each trial type: a pre-stimulus epoch defined by the previous outcome, previous action and previous stimulus (the activity of the input neurons encoding the current stimulus is set to zero), and a second epoch where also the current stimulus is presented. For inputs corresponding to a pre-stimulus epoch the network is trained to output a ‘null’ action by activating a third action neuron, and a ‘null’ outcome by activating a third outcome neuron.

Training proceeds in epochs of 128 mini-batches of input-output pairs generated from the first step in the trial (where inputs are the stimulus, action, and outcome in the previous trial, while outputs are the null action and outcome) and the second step in the trial (where inputs are the same as in the first step in addition to the stimulus in the current trial, and the outputs are the correct action and predicted output), randomly intermixed. A fraction of 0.25 inputs-output pairs are generated so as to encode switch trials in the task, i.e. error trials that are following a contextual switch. The implicit assumption here is that the agent does not make mistakes, unless there is a contextual switch, which means that switch trials will be characterized by an outcome indicating an error (see Figure S8a, two rightmost input-output pairs). Errors are encoded with outcome neurons set to a fixed random binary pattern corresponding to a negative outcome in the previous trial, while the target output patterns are chosen so as to correspond to the opposite context compared to the one that was (incorrectly) followed in the previous step.

We train a population of 100 randomly initialized networks on the supervised learning version of the task. Figure S8c shows the resulting learning curves as a function of training epochs in terms of the average accuracy across the population at correctly predicting the null action and outcome after the first step, and correctly predicting the correct action and outcome after the second step, respectively (error bars denote the standard deviation of the accuracy across the population of trained networks).

The main results are analogous to what we observed for the Reinforcement Learning case. Figure S8d reveals that the *context* again cannot be decoded in the input but can already be decoded with high accuracy starting from the randomly initialized first hidden layer. Context becomes abstract in layer L2, where on average CCGP (across the population of models) becomes significantly higher than chance. Correspondingly, the average PS in L2 is fairly high, hovering around 0.25. In both hidden layers of the trained models, the PS and CCGP for context are significantly higher than in the untrained models.

After training, the CCGP and PS for the previous outcome and action variables are very close to the untrained models. However, as we noticed earlier, we note a slight decrease in abstraction of value and action that seems to correlate with an increase in the abstraction of context. We will study this effect in Figure S8h.

As in the case of the Reinforcement Learning model, we compare the PS and CCGP of the trained models to the null-distributions obtained from a shuffle of the data and the geometric random model, respectively. Figure S8e,f show beeswarm plots of the probability that the CCGP and parallelism score of each of the trained supervised learning models is above the distributions given by the geometric random model and a shuffle test, respectively, for all variables in the task individually (context, previous value, and previous action) and all variables simultaneously for the same model (All). These probabilities are empirically estimated over 100 trained models and 1000 samples of the null-distributions per trained model. Figure S8e,f are laid out similarly to Figs. S7f,g. As in the Reinforcement Learning case, a considerable fraction of trained models display a PS and CCGP above the 95th percentile of the null-distributions, particularly in hidden layer L2. For instance, in L2 91% of the trained models have parallelism score that is higher than 95% of the shuffle null-distribution simultaneously for all three variables, while 40% have CCGP above 95% of the null-distribution obtain with the random model.

Next we set out to study the effect that we noticed whereby abstraction of value seems to be negatively correlated with abstraction of context. First, we hypothesize that abstraction of value might be positively correlated with the fraction of switch trials, since maintaining information about the previous outcome (i.e., the value) is crucial to identify a contextual switch. Figure S8g shows scatter plots of the PS and CCGP for value in the hidden layer L2 of 100 supervised learning models trained as in the previous paragraphs but with different fractions of switch trials (between 0 and 0.5), revealing that indeed both quantities are positively correlated with the fraction of switch trials.

Next we utilize the fraction of switch trials as an independent variable to control PS and CCGP of value, and quantify the correlation between the observed PS and CCGP for value and context. The first panel of Figure S8h reveals a statistically significant negative correlation -0.34 between the PS for value and the PS for context in hidden layer L2 of the supervised learning model. Analogously, the second panel of Figure S8h shows that the logit transform of the CCGP for context is also negatively correlated with the CCGP for value (we used the logit transform, because CCGP for context tends to saturate close to one for low values of CCGP for value). These results quantify the effect that we had qualitatively alluded to earlier in the Reinforcement Learning model that caused value to be represented in a less abstract format when context is represented in a more abstract format, and which prompted us to introduce the reconstruction loss term.

LEGENDS OF SUPPLEMENTAL FIGURES

Figure S1: Schemes that illustrates how CCGP (a) and parallelism score (b,c) are computed. Related to Figure 2.a. Scheme that explains how CCGP is computed for all the dichotomies in an experiment with 4 conditions (2 stimuli in each of the two contexts). The example has the same geometry as the representation in Figures 2c,d (value and context are simultaneously abstract). Each dichotomy corresponds to a different way of dividing the 4 conditions in two groups of two (different colored clouds for each group). In this example, all possible dichotomies correspond to variables that have a name (”context”, ”value” and ”stimulus”). However, this is not necessarily true in other situations. For each dichotomy, CCGP is computed by training a linear decoder on a subset of conditions. These are the only conditions highlighted (by shaded colored circles) in the figure. The decoder is then tested on the remaining (non-shaded) conditions. For each dichotomy there are 4 possible ways of choosing 2 conditions, one from each side of the dichotomy, as illustrated in the figure in the row next to each dichotomy. The final CCGP is obtained by computing the average test performance over all the different ways of choosing two training conditions. For this geometry, CCGP is 0.75 for the dichotomies context and value, and 0 for the dichotomy stimulus. These values are obtained under the assumption that the

noise is isotropic and not too large. For context and value CCGP is equal to one only for two choices of the conditions used for training. This is a peculiarity of this simple case which contains only 4 points in the firing rate space. By contrast, consider the case of 8 points arranged on a cube. Here the CCGP for the analogous dichotomy (i.e., one that separates the 4 points of one face from the 4 points of the opposite face) is equal to 1 for all possible choices of 6 training conditions (3 conditions per face). Dichotomy 3 appears not to be linearly decodable (and thus neither traditional decoding nor CCGP will have performance above chance). However, this scheme is highly simplified, and in the real data we observed variables that are decodable and not in an abstract format (e.g. action in the hippocampus in the interval preceding stimulus appearance, and context in DLPFC during the interval following stimulus presentation). One can visualize a situation in which a variable is decodable but not in an abstract format by introducing a third dimension to the scheme shown for Dichotomy 3. This scheme places the 4 points on a squeezed cube, with the two red points on one face and the two blue points on the other face. If the distance between these two faces is small compared to the other distances, then the CCGP will be approximately the same as shown in the Figure for all 3 dichotomies, but now Dichotomy 3 would be decodable at above chance levels. **b,c.** Scheme that explains the Parallelism Score (PS). In the two panels we show the firing rate space of two neurons. These neural representations are similar to those in Figures 2d, which allow for cross-condition generalization for both context and value. b. Training a linear classifier to decode context on the two rewarded conditions leads to the magenta separating hyperplane, which is defined by a weight vector orthogonal to it. Similarly, training on the unrewarded conditions leads to the dark purple hyperplane and and its associated orthogonal weight vector. If these two weight vectors are close to parallel, the corresponding classifiers are more likely to generalize to the conditions not used for training. The parallelism score (PS) is defined as the cosine of the angle between these coding vectors, maximized over all possible ways of pairing up the conditions (see Methods for details). c. Same as e but for the variable value.

Figure S2: Trade-off between CCGP/PS and shattering dimensionality. Related to Figure 2. Analysis of an artificial data set generated by randomly embedding a (three-dimensional) cube in a 100-dimensional space, displacing its corners in independent random directions by a certain distance (displacement magnitude), and then sampling data points corresponding to the eight experimental conditions from isotropic Gaussian distributions around the cluster centers obtained from this distortion procedure. The shattering dimensionality (SD, blue) is plotted across all 35 balanced dichotomies, as well as the mean PS (red) and CCGP (yellow) of the three potentially abstract variables for low (left) and high noise (right), with noise sampled as i.i.d. unit Gaussian vectors multiplied by overall coefficients 0.2 and 1.0, respectively. For low noise, both SD and CCGP can simultaneously be close to one, indicating maximal dimensionality and the presence of three abstract variables for these representations. In the case of high noise, we observe a smooth tradeoff between CCGP (abstraction) and SD (dimensionality).

Figure S3: Single neuron activity, task relevant variables decoding and dimensionality. Related to Figure 4. a. The activity of all neurons recorded for at least ten trials per condition after excluding: 1) those trials performed incorrectly; 2) those trials in which a contextual frame was shown either for the current or the previous trial; and 3) those trials that occurred less than five trials after a context switch). Z-scored firing rates are calculated in the 900 ms time window that starts 800ms before stimulus onset. Each row represents the activity of an individual neuron. Different columns correspond to different trial conditions (i.e., the stimulus-action-outcome sequence preceding the interval). Neurons are ordered such that for adjacent rows, the eight-dimensional vectors of z-scored activities point in similar directions. The responses are very diverse in all three brain areas. b. Population level encoding of task-related variables. Performance of a linear decoder plotted a function of time relative to image onset for classifying a task-relevant variable. 1) Context on the current trial. 2) Reinforcement outcome on current (left) and prior (right) trials. 3) Operant action on current (left) and prior (right) trials. The decoding performance was computed in a 500-ms sliding window stepped every 50 ms across the trial for the three brain areas separately (blue, HPC; red, ACC; green, DLPFC). Shaded areas around chance level (dotted line) indicate two-sided 95%-confidence intervals calculated with a permutation test obtained by randomly shuffling

trials (1,000 repetitions). The image is displayed on the screen from time 0 to 0.5 sec. Analyses were run only on correct trials at least 5 trials after a context switch. c. Dimensionality of the average firing rate activity patterns as a function of time throughout the trial. The left panel illustrates the result of the analysis developed in (Machens et al. 2010) on HPC, the central panel refers to DLPFC, and the right panel to ACC. Continuous lines refer to the analysis carried out on average firing rate patterns obtained by averaging spike counts according to the task conditions of the trial that was being recorded (current trial). The dashed line shows the same but for conditions defined by the previous trial. The lines indicate the number of principal firing rate components that are larger than all noise components, averaged over 1000 re-samplings of the noise covariance matrix (see (Machens et al. 2010)). The shadings indicate the 95% confidence intervals estimated using the method for quantifying uncertainty around the mean of bounded random variables presented in (Chen 2008).

Figure S4: a,b The geometry of the recorded neural representations. c,d,e: Abstraction by clustering: geometry and measures of the degree of abstraction. Related to Figure 3. a. Multi-dimensional scaling plots (using Euclidean distances on z-scored spike count data in the 900 ms time window that starts 800ms before stimulus onset) showing the dimensionality-reduced firing rates for different experimental conditions in the three brain areas recorded from: HPC, DLPFC and ACC. The labels refer to value ($+/-$) and operant action (R/H) corresponding to the previous trial. Yellow rings are rewarded conditions. While there is a fairly clean separation between the two context sub-spaces, other variables are encoded as well and the representations are not strongly clustered. Note that the context sub-spaces appear to be approximately two-dimensional (i.e., of lower dimensionality than expected for four points in random positions). See also the videos in the Supplemental Material. b. The MDS plots of panel (a) are rotated to highlight the dependence on the action of the previous trial (H=Hold, R=Release; in boldface). In DLPFC and ACC the points corresponding to the conditions with the same action are on the same side of the plot. In these areas, action is decodable and in an abstract format. In HPC all the 8 points are distinct, and action is decodable. However, it is not in an abstract format. Indeed, the H and R points are along two almost orthogonal diagonals. Notice that value ($+/-$ symbols) and context (red/blue; less clear from this perspective) are in an abstract format in all three brain areas. c. Schematic of firing rate space in the case of clustering abstraction. Due to the clustering, the average within-context distance is shorter than the mean between-context distance. d. Abstraction index for the context dichotomy (ratio of average between-context distance to average within-context distance using a simple Euclidean metric) for the z-scored neural firing rates recorded from HPC, DLPFC and ACC, averaged over a time window of -800ms to 100ms relative to stimulus onset. The error bars are plus/minus two standard deviations around chance level (unit abstraction index), obtained from a shuffle of the data. Notice that the abstraction index based on clustering for DLPFC and ACC is barely different from chance. e. An example in which the clustering analysis fails to detect that context is in an abstract format. The firing rate space is represented as in panel b. The points are arranged in a rectangular cuboid (a cube squeezed along the vertical direction), and the distance between the two squares that represent the two contexts is *d*. This distance is smaller than the sides of the two squares. When *d* is sufficiently small, the abstraction index is one or less than one, despite the fact that context is in an abstract format according to the CCGP (see the separating hyperplane which illustrates one linear classifier that clearly allows for cross-condition generalization). This type of geometry (the squeezed cube) may be observed whenever variables are encoded with different strengths (see also text in this section).

Figure S5: Decoding accuracy, CCGP and parallelism score (PS) in the three recorded brain areas in Monkey F and Monkey R separately. Related to Figures 3 and 5. a-b,e-f: CCGP, decoding accuracy and PS for the variables that correspond to all 35 possible dichotomies shown separately for each brain area in a 900 ms time epoch beginning 800 ms before image presentation. a-b is for monkey F and e-f for monkey R. The points corresponding to the context, value and action of the previous trial are highlighted using circles of different colors. c-d,g-h. CCGP, decoding accuracy and PS for all 35 dichotomies in the time interval from 100ms to 1000ms after stimulus onset. c-d for monkey F and g-h for monkey R. For all panels, error bars are $\pm$ two standard deviations around chance level as obtained from

a geometric random model (CCGP) or from a shuffle of the data (decoding accuracy and PS). Almost all dichotomies can be decoded with accuracy > 0.5 in all 3 brain areas, and the average decoding performance across the 35 dichotomies (shattering dimensionality) is significantly greater than chance ($p < 0.01$, permutation test). Nonetheless, multiple variables are represented in an abstract format simultaneously, with CCGP well above the confidence bounds around chance levels.

Overall, although the decoding performance is weaker in Monkey R than Monkey F, perhaps in part accounted for by the smaller sample size, the basic results concerning how CCGP reveals that multiple variables are represented in an abstract format simultaneously.

Figure S6: Neurons are rarely specialized and the ability to cross generalize does not rely mainly on the few specialized neurons that are observed. Related to Figure 4. a. Two-dimensional scatter plots of the absolute values of the (normalized) decoding weights for the three task-relevant variables. The three columns (from left to right) correspond to HPC, DLPFC, and ACC. The three rows show the magnitudes of the weights for pairs of variables plotted against each other: context vs. value (top), value vs. action (middle), and action vs. context (bottom). The inset in each scatter plot shows a histogram of the weight counts as a function of the angle from the vertical axis (in radians). These distributions are approximately uniform, and therefore pure selectivity neurons (whose weights would fall close to one of the axes in the scatter plots) are not prevalent. Similar distributions have been observed in the rodent hippocampus (Stefanini et al. 2020).

b. CCGP as a function of the number of ablated neurons for the HPC (left), DLPFC (middle), and ACC (right). The solid lines show the decay of CCGP if we successively remove the neurons with the largest pure selectivity indices for context (blue), value (red) or action (yellow). The dashed lines show the decline of the CCGP for the same three variables if we instead ablate neurons with the largest sum of squares of their three decoding weights (i.e., those with the radial position furthest from the origin in their three-dimensional weight space), independent of their pure selectivity indices. The two sets of curves are rather close to each other, and thus these two sets of ablated neurons are of similar importance for CCGP. (For HPC, the CCGP of the action variable is always below chance level for both curves; not shown). This is similar to what has been observed in simulations of deep networks (Morcos et al. 2018).

c. Specialization and ablation analyses of second hidden layer in the neural network of Fig. 7. Left: Two-dimensional scatter plot of the absolute values of the (normalized) decoding weights for the parity and magnitude dichotomies, as in panel a. Right: CCGP when training on three digits from either side of the magnitude (red) and parity (blue) dichotomies and testing on the fourth one as a function of the number of ablated neurons. We ablate the neurons with the largest pure selectivity indices (solid lines), or the ones with the largest sum of squared decoding weights (dashed lines), as in panel b. Note that even though we don't z-score the neural responses for computations of the CCGP, or for any of the analyses shown in Fig. 7, the decoding weights shown here (and used to select the ablated neurons) are taken from classifiers trained on z-scored responses so that the weights reflect the relative importance of different neurons for decoding.

Figure S7: Reinforcement Learning (RL) model trained with Deep Q-learning (DQN) on a simulated version of the serial reversal-learning task. Related to Figure 7. a., b. An episode in the simulated environment is composed of two sequential steps, each corresponding to one trial. In the first step, corresponding to the pre-stimulus interval in the trial, the network receives as observations the stimulus, action and outcome of the previous trial, at which point it is required to issue the action "Wait" to initiate a new trial. In the second step of the episode the network receives the current stimulus (in addition to all observations presented in the first step) and is required to issue the correct action in the task ("Hold" or "Release"). If at any step the agent issues an incorrect action, it receives a reward of -1 and the episode terminates. Otherwise, it receives a reward of 0 after correctly issuing "Wait" at the end of the first step, and for trials in which the agent issues the correct action, it receive a reward corresponding to the value of the stimulus in the task at the end of the second step (+1 or 0 depending upon whether it is a

rewarded trial type). c. The agent is parametrized as a two-hidden layers neural network trained with DQN. We add a reconstruction loss to the optimization loss (indicated in blue in the lower right corner of the panel) that forces the network to find representations in the L2 hidden layer that can linearly reconstruct the inputs (i.e., the reconstruction loss is the mean squared error between the inputs and a linear transformation of the L2 activations). d. Learning curves for the DQN model on the serial-reversal task. The blue line is the fraction of correct actions averaged over 100 random initializations of the network and blocks of 50 episodes. The gray lines show the performance of 10 randomly chosen individual networks (also in blocks of 50 episodes). e. Analysis of the activity generated by the Reinforcement Learning model: *decoding accuracy* (first row), *parallelism score* (second row) and *cross-condition generalization performance* (third row) during the simulated first step (pre-stimulus epoch) in the task, for the variables *context* (first column), *previous value* (second column) and *previous action* (third column). In each panel we plot one of the quantities of interest as a function of the layer where it is measured in the architecture: inputs, first hidden layer (L1) and second hidden layer (L2) (see Panel c). The plots show the mean quantity of interest across 100 randomly initialized and independently trained networks, before (grey data points) and after training (orange data points). Error bars indicate standard deviations computed over the same distribution of 100 networks. f,g. Beeswarm plots of the probability of PS and CCGP of trained Q-learning models in layers 1 (f) and 2 (g) to be above the null-distributions given by the geometric random model and a shuffle test of the data (empirically estimated with 1000 samples per model), respectively (see Figure 3). Every dot in the graph represents a trained model. The top panels of f,g show the probability that the Parallelism Score of each model for every variable individually (Context, Value and Action) and all variables simultaneously (All) is above the null-distribution given by a shuffle of the data in layer L1 and L2. The bottom panels are laid out similarly, but show the probability that the CCGP of each model is above the null-distribution given by the geometric random model. The dotted lines correspond to the mean of the null-distribution, and the dashed lines correspond to the 95th percentile of the null-distribution. The numbers written in red above each plot report the fraction of models that are above the 95th percentile threshold.

Figure S8: Supervised learning model. Related to Figure 7 a. The supervised learning model is trained on two types of input-output combinations, corresponding to sequential steps in a trial. The inputs for the first step (the pre-stimulus interval in the trial) are composed of the stimulus, the action and the outcome (reward) in the previous trial. The target output for the first step is a 'null' action and outcome value (corresponding to a third null action and a third null outcome neuron). Inputs corresponding to the second step encode the current stimulus (along with the features that were already presented in the first step), and the second step output encodes the correct action (Hold or Release) and outcome (Reward or No reward) in the trial. Besides being distinguished by whether they correspond to the first step or second step in the trial, generated inputs can be distinguished depending on whether or not they correspond to a switch trial. Inputs generated from non-switch trials (see first and second input-output combinations in the panel) define the correct action and subsequent outcome, under the assumption that the current context is the same as in the previous trial (encoded in the input). Inputs generated from switch trials (the first trial after a context switch) are characterized by outcome features encoding an error in the previous trial (see second and third input-output combinations in the panel), implying that the observed stimulus-action combination is incorrect and counterfactually defining the currently correct context and action. b. The neural network is a multi-layer network with two hidden layers (number of unites indicated in the figure) trained with backpropagation in a supervised way to output the correct action and outcome. c. Learning curves for the (simultaneously learned) action and value prediction tasks at the end of the second step (corresponding to the response and value prediction of the animal) as a function of training episodes. Every epoch consists of 128 mini-batches of 100 noisy versions of the inputs just described. Data points are the means and standard deviations across 100 distinct random initializations of the network. d. Activity analysis: *decoding accuracy* (first row), *parallelism score* (second row) and *cross-condition generalization (CCG) performance* (third row) during the simulated first step (pre-stimulus epoch) in the task, for the variables *context* (first column), *previous outcome* (second column) and *previous action* (third column). In each

panel we plot one of the quantities of interest as a function of the layer where it is measured in the architecture: inputs, first hidden layer (L1) and second hidden layer (L2). Each point in the plots represents the mean quantity of interest across 100 randomly initialized and independently trained networks, before training (grey data points) and after training (orange data points). Error bars indicate standard deviations computed over the same distribution of 100 networks. The lines connect points computed within the same training condition (before training or after training) in adjacent layers in the architecture. e.,f. Beeswarm plots of the probability of PS and CCGP of trained supervised learning models in layers 1 (e) and 2 (f) to be above the null-distributions given by the geometric random model and a shuffle test of the data (empirically estimated with 1000 samples per model), respectively (see Figure 3). Every dot in the graph represents a trained model. The top panels of e,f show the probability that the Parallelism Score of each model for every variable individually (Context, Value and Action) and all variables simultaneously (All) is above the null-distribution given by a shuffle of the data in layer L1 and L2. The bottom panels are laid out similarly, but show the probability that the CCGP of each model is above the null-distribution given by the geometric random model. The dotted lines corresponds to the mean of the null-distribution, and the dashed line corresponds to the 95th percentile of the null-distribution. The numbers written in red above each plot report the fraction of models that are above the 95th percentile threshold. g. Correlation between abstraction of value and fraction of switch trials in hidden layer L2 of the supervised learning model. Both PS and CCGP for value are positively correlated with the fraction of switch trials with statistically significant Pearson correlation coefficients ρ (t-test). Notice that this model makes mistakes only at the switches. h. Anti-correlation between abstraction of value and abstraction of context in hidden layer L2 of the supervised learning model. Both PS and CCGP for value are negatively correlated with respectively the PS and CCGP for context. Since CCGP for context tends to saturate close to one for low CCGP for value, we applied a logit transform to the CCGP for context, before computing the Pearson correlation coefficients.

LEGENDS OF SUPPLEMENTAL TABLES

Table T1: Condition-specific cross-validation used for analyses of error trials. Related to Figure 6. This figure illustrates the method used for inclusion of neurons and conditions for the analysis of error trials shown in Fig. 6. A fictitious dataset is shown for 9 neurons in 4 task conditions. The table lists the number of error trials for each of the 4 conditions, which corresponds to the held-out error trials used for cross-validation for each neuron in each condition. The red cross indicates conditions for which not enough error trials were recorded for a given neuron. If the criterion for including a neuron in this analysis specified that enough error trials had to be recorded for all conditions simultaneously, then all but the first two neurons (highlighted in blue) in the dataset would be excluded. On the other hand, if we take an approach that cross-validates each condition separately, then for each condition we can include (different but overlapping) neural populations totaling 7 neurons. To cross-validate a decoder on a given condition, we generate training patterns for all conditions using only the neurons with enough trials to be included in the cross-validation condition. These patterns are then used for training, and the resulting decoder will be cross-validated on the held-out trials in the cross-validation condition (which is composed of error trials).

Table T2: CCGP and PS for every dichotomy. Related to Figs. 3 and 5. Dichotomies corresponding to context, action, value, and various combinations of pairs of stimuli are labeled. The condition numbers 1 to 8 refer to the stimulus/action/reward combinations $\left(A_R^+, B_H^+, C_R^-, D_H^-, B_R^+, C_H^+, D_R^-, A_H^-\right)$, where A-D indicate the stimulus identities, +/- indicates reward/no reward for correct actions, and H/R indicates hold or release operant action. Bold indicates CCGP or PS values greater than two standard deviations above chance level.