Appendices

A. Model description: architectures

Here we describe the precise details of the architectures used in the main text.

We note that we did not optimise our results by tweaking architectures or hyperparameters in any systematic or substantial way. Rather, we simply picked sensible-looking values. We anticipate that better performance could be obtained by improving these decisions, but this is beyond the scope of this work.

A.1. Common elements.

Pre-processing. Both the character net and the mental state net consume trajectories, which are sequences of observed state/action pairs, $\tau_{ij}^{(obs)} = \{(x_t^{(obs)}, a_t^{(obs)})\}_{t=0}^{T}$, where i is the agent index, and j is the episode index. The observed states in our experiments, $x_t^{(obs)}$, are always tensors of shape $(11 \times 11 \times K)$, where K is the number of feature planes (comprising one feature plane for the walls, one for each object, and one for the agent). The observed actions, $a_t^{(obs)}$, are always vectors of length 5. We combine these data through a *spatialisation-concatenation* operation, whereby the actions are tiled over space into a $(11 \times 11 \times 5)$ tensor, and concatenated with the states to form a single tensor of shape $(11 \times 11 \times (K + 5))$.

Training. All ToMnets were trained with the Adam optimiser, with learning rate 10^{-4}, using batches of size 16. We trained the ToMnet for 40k minibatches for random agents (Section 3.1), and for 2M minibatches otherwise.

A.2. ToMnet for random agents (Section 3.1)

Data. For each species, $S(\alpha)$, we trained a single ToMnet, by first forming a training set by sampling 1000 agents from $\mathcal{S}(\alpha)$. For each of these agents, we then generated behavioural traces on randomly-generated POMDPs. We then trained a ToMnet to observe how randomly-sampled agents $\mathcal{A}_i \sim \mathcal{S}(\alpha)$ behave on a variable number of past episodes $N_{\text{past}} \sim U\{0, 10\}$. Each past episode was of length 1 (i.e. each trajectory consisted of a single state-action pair). When no past episodes were sampled for a given agent, the character embedding was set to $e_{\text{char}} = 0$.

Character net. Each trajectory τ_{ij} comprises a single state/action pair. We spatialise the action, and concatenate this with the state. This is passed into a 1-layer convnet, with 8 feature planes and ReLU nonlinearity. We then passed the sequence of these (indexed by j) into a convolutional LSTM, with the output passed through an average pooling layer, and a fully-connected layer to a 2D embedding space, to produce $e_{\text{char,i}}$. We obtained similar results with a wide range of different architectures.

Mental net. None.

Prediction net. In this experiment, we predict only next-step action (i.e. policy, $\hat{\pi}$) We spatialise $e_{\text{char,i}}$, and concatenate with the query state. This is passed to a 2-layer convnet, with 32 feature planes and ReLUs. This is followed by average pooling, then a fully-connected layer to logits in $\mathbb{R}^5$, followed by a softmax.

A.3. ToMnet for inferring goals (Section 3.2)

A.3.1. Experiment 1: single past MDP

Data. Character embedding formed from a single past episode, comprising a full trajectory on a single MDP. Query state is the initial state of a new MDP, so no mental state embedding required.

Character net. For the single trajectory τ_i in the past episode, the ToMnet forms the character embedding $e_{\text{char,i}}$ as follows. We pre-process the data from each time-step by spatialising the actions, $a_t^{(obs)}$, concatenating these with the respective states, $x_t^{(obs)}$, passing through a 5-layer resnet, with 32 channels, ReLU nonlinearities, and batch-norm, followed by average pooling. We pass the results through an LSTM with 64 channels, with a linear output to either a 2-dim or 8-dim $e_{\text{char,i}}$ (no substantial difference in results).

Mental net. None.

Prediction net. In this and subsequent experiments, we make three predictions: next-step action, which objects are consumed by the end of the episode, and successor representations. We use a shared torso for these predictions, from which separate heads branch off. For the prediction torso, we spatialise $e_{\text{char,i}}$, and concatenate with the query state; this is passed into a 5-layer resnet, with 32 channels, ReLU nonlinearities, and batch-norm.

Action prediction head. From the torso output: a 1-layer convnet with 32 channels and ReLUs, followed by average pooling, and a fully-connected layer to 5-dim logits, followed by a softmax. This gives the predicted policy, $\hat{\pi}$.

Consumption prediction head. From the torso output: a 1-layer convnet with 32 channels and ReLUs, followed by average pooling, and a fully-connected layer to 4-dims, followed by a sigmoid. This gives the respective Bernoulli probabilities that each of the four objects will be consumed by the end of the episode, $\hat{c}$.

Successor representation prediction head. From the torso output: a 1-layer convnet with 32 channels and ReLUs, then a 1-layer convnet with 3 channels, followed by a softmax over each channel independently. This gives the predicted normalised SRs for the three discount factors, $\gamma = 0.5, 0.9, 0.99$.

A.3.2. Experiment 2: many past MDPs, only a single snapshot each

Data. Character embedding is formed from many past episodes ($N_{\text{past}} \sim U\{0, 10\}$); however, we only use a snapshot of a single time point (i.e. a single state/action pair) from each past episode. Query state as for Experiment 1.

Character net. For each trajectory τ_{ij}, the character net adds a contribution $e_{\text{char},ij}$ to the character embedding $e_{\text{char,i}}$ as follows. We pre-process the single state/action pair in τ_{ij} as in Experiment 1, i.e. spatialising, concatenating, resnet, average pooling. There is no LSTM as there is only a single state/action pair; instead we use a fully-connected layer to form $e_{\text{char},ij} \in \mathbb{R}^2$. These are summed across the N_{past} past trajectories to form $e_{\text{char,i}}$.

Mental net. None.

Prediction net. As for Experiment 1.

A.3.3. Experiment 3: greedy agents

Same as Experiment 1, but with $N_{\text{past}} \in U\{0, 5\}$.

A.4. ToMnet for modelling deep RL agents (Section 3.3)

Data. Character embedding is formed from observing full trajectories of agents on $N_{\text{past}} = 4$ POMDPs. While the agents have partial observability, potentially with significant parts of the state masked, the observer sees the entire gridworld (albeit without any indication of the field of view of the agent). The current episode is split at a random time (drawn uniformly from $U\{0, T - 1\}$ where T is the length of the trajectory). The trajectory prior to the split forms the "recent trajectory", and is passed to the mental net. The state at the time of the split is used as the query state.

Character net. For each past trajectory τ_{ij}, the character net adds a contribution $e_{\text{char},ij}$ to the character embedding $e_{\text{char,i}}$ via the same architecture as in Experiment 1 described in Appendix A.3 above, with an 8-dim $e_{\text{char},ij}$. These are summed to form $e_{\text{char,i}}$.

Mental net. We pre-process each time step's state/action pair in the recent trajectory as follows: we spatialise the action, concatenate with the state, pass through a 5-layer resnet, with 32 channels, ReLU nonlinearities, and batch-norm. The results are fed into a convolutional LSTM with 32 channels. The LSTM output is also a 1-layer convnet with 32 channels, yielding a mental state embedding $e_{\text{mental},i} \in \mathbb{R}^{11\times 11\times 32}$. When the recent trajectory is empty (i.e. the query state is the initial state of the POMDP), $e_{\text{mental},i}$ is the zero vector.

Prediction net. As in Experiment 1 described in Appendix A.3. However, the prediction torso begins by spatialising $e_{\text{char,i}}$ and concatenating it with both $e_{\text{mental},i}$ and the query state. Also, as these agents act in gridworlds that include the subgoal object, the consumption prediction head outputs a 5-dim vector.

DVIB. For the Deep Variational Information Bottleneck experiments, we altered the architecture by making the character net output a posterior density, $q(e_{\text{char,i}})$, rather than a single latent $e_{\text{char,i}}$; likewise, for the mental net to produce $q(e_{\text{mental},i})$, rather than $e_{\text{mental},i}$. We parameterised both densities as Gaussians, with the respective nets outputting the mean and log diagonal of the covariance matrices, as in Kingma & Welling (2013). For the character net, we achieved this by doubling the dimensionality of the final fully-connected layer; for the mental net, we doubled the number of channels in the final convolutional layer. In both cases, we used fixed, isotropic Gaussian priors. For evaluating predictive performance after the bottleneck, we sampled both e_{char} and e_{mental}, propagating gradients back using the reparameterisation trick. For evaluating the bottleneck cost, we used the analytic KL for $q(e_{\text{char,i}})$, and the analytic KL for $q(e_{\text{mental},i})$ conditioned on the sampled value of $e_{\text{char,i}}$. We scaled the bottleneck costs by $\beta_{char} = \beta_{mental} = \beta$, annealing β quadratically from 0 to 0.01 over 500k steps.

A.5. ToMnet for false beliefs (Sections 3.4–3.5)

The ToMnet architecture was the same as described above in Appendix A.4. The experiments in Section 3.5 also included an additional belief prediction head to the prediction net.

Belief prediction head. For each object, this head outputs a 122-dim discrete distribution (the predicted belief that the object is in each of the 11×11 locations on the map, or whether the agent believes the object is absent altogether). From the torso output: a 1-layer convnet with 32 channels and ReLU, branching to (a) another 1-layer convnet with 5 channels for the logits for the predicted beliefs that each object is at the 11×11 locations on the map, as well as to (b) a fully-connected layer to 5-dims for the predicted beliefs that each object is absent. We unspatialise and concatenate the outputs of (a) and (b) in each of the 5 channels, and apply a softmax to each channel.

B. Loss function

Here we describe the components of the loss function used for training the ToMnet.

For each agent, $\mathcal{A}_i$, we sample past and current trajectories, and form predictions for the query POMDP at time t. Each prediction provides a contribution to the loss, described below. We average the respective losses across each of the agents in the minibatch, and give equal weighting to each loss component.

Action prediction. The negative log-likelihood of the true action taken by the agent under the predicted policy:

$$\mathcal{L}_{\text{action},i} = -\log \hat{\pi}(a_t^{(obs)} | x_t^{(obs)}, e_{\text{char,i}}, e_{\text{mental},i})$$

Consumption prediction. For each object, k, the negative log-likelihood that the object is/isn't consumed:

$$\mathcal{L}_{\text{consumption},i} = \sum_k -\log p_{c_k}(c_k | x_t^{(obs)}, e_{\text{char,i}}, e_{\text{mental},i})$$

Successor representation prediction. For each discount factor, γ, we define the agent's empirical successor representation as the normalised, discounted rollout from time t onwards, i.e.:

$$SR_\gamma(s) = \frac{1}{Z} \sum_{\Delta t=0}^{T-t} \gamma^{\Delta t} I(s_{t+\Delta t} = s)$$

where Z is the normalisation constant such that $\sum_s SR_\gamma(s) = 1$. The loss here is then the cross-entropy between the predicted successor representation and the empirical one:

$$\mathcal{L}_{\text{SR},i} = \sum_\gamma \sum_s -SR_\gamma(s) \log \widehat{SR}_\gamma(s)$$

Belief prediction. The agent's belief states for each object k is a discrete distribution over 122 dims (the 11×11 locations on the map, plus an additional dimension for an absent object), denoted $b_k(s)$. For each object, k, the loss is the cross-entropy between the ToMnet's predicted belief state and the agent's true belief state:

$$\mathcal{L}_{\text{belief},i} = \sum_k \sum_s -b_k(s) \log \hat{b}_k(s)$$

Deep Varational Information Bottleneck. In addition to these loss components, where DVIB was used, we included an additional term for the β-weighted KLs between posteriors and the priors

$$\begin{aligned}\mathcal{L}_{DVIB} &= \beta D_{KL}\left(q(e_{\text{char,i}}) || p(e_{\text{char}})\right) + \\ &\quad \beta D_{KL}\left(q(e_{\text{mental},i}) || p(e_{\text{mental}})\right)\end{aligned}$$

C. Gridworld details

The POMDPs $\mathcal{M}_j$ were all 11×11 gridworld mazes. Mazes in Sections 3.1–3.2 were sampled with between 0 and 4 random walls; mazes in Sections 3.3–3.5 were sampled with between 0 and 6 random walls. Walls were defined between two randomly-sampled endpoints, and could be diagonal.

Each $\mathcal{M}_j$ contained four terminal objects. These objects could be consumed by the agent walking on top of them. Consuming these objects ended an episode. If no terminal object was consumed after 31 steps (random and algorithmic agents; Sections 3.1–3.2) or 51 steps (deep RL agents; Sections 3.3–3.5), the episodes terminated automatically as a time-out. The sampled walls may trap the agent, and make it impossible for the agent to terminate the episode without timing out.

Deep RL agents (Sections 3.3–3.5) acted in gridworlds that contained an additional subgoal object. Consuming the subgoal did not terminate the episode.

Reward functions for the agents were as follows:

Random agents (Section 3.1.) No reward function.

Algorithmic agents (Section 3.2). For a given agent, the reward function over the four terminal objects was drawn randomly from a Dirichlet with concentration parameter 0.01. Each agent thus has a sparse preference for one object. Penalty for each move: 0.01. Penalty for walking into a wall: 0.05. Greedy agents' penalty for each move: 0.5. These agents planned their trajectories using value iteration, with a discount factor of 1. When multiple moves of equal value were available, these agents sampled from their best moves stochastically.

Deep RL agents (Sections 3.3–3.5). Penalty for each move: 0.005. Penalty for walking into a wall: 0.05. Penalty for ending an episode without consuming a terminal object: 1.

For each deep RL agent species (e.g. blind, stateless, 5×5, ...), we trained a number of canonical agents which received a reward of 1 for consuming the subgoal, and a reward of 1 for consuming a single preferred terminal object (e.g. the blue one). Consuming any other object yielded zero reward (though did terminate the episode). We artifically enlarged this population of trained agents by a factor of four, by inserting permutations into their observation functions, ω_i, that effectively permuted the object channels. For example, when we took a trained blue-object-preferring agent, and inserted a transformation that swapped the third object channel with the first object channel, this agent behaved as a pink-object-preferring agent.

D. Deep RL agent training and architecture

Deep RL agents were based on the UNREAL architecture (Jaderberg et al., 2017). These were trained with over 100M episode steps, using 16 CPU workers. We used the Adam optimiser with a learning rate of 10^{-5}, and BPTT, unrolling over the whole episode (50 steps). Policies were regularised with an entropy cost of 0.005 to encourage exploration.

We trained a total of 660 agents, spanning 33 random seeds $\times$ 5 fields of view $\times$ 2 architectures (feedforward/convolutional LSTM) $\times$ 2 depths (4 layer convnet or 2 layer convnet, both with 64 channels). We selected the top 20 agents per condition (out of 33 random seeds), by their average return. We randomly partitioned these sets into 10 training and 10 test agents per condition. With the reward permutations described above in Appendix C, this produced 40 training and 40 test agents per condition.

Observations. Agents received an observation at each time step of nine 11×11 feature planes – indicating, at each location, whether a square was empty, a wall, one of the five total objects, the agent, or currently unobservable.

Beliefs. We also trained agents with the auxiliary task of predicting the current locations of all objects in the map. To do this, we included an additional head to the Convolutional LSTMs, in addition to the policy (π_t) and baseline (V_t) heads. This head output a posterior for each object's location in the world, b_k (i.e. a set of five 122-dim discrete distributions, over the 11×11 maze size, including an additional dimension for a prediction that that the object is absent). For the belief head, we used a 3-layer convnet with 32 channels and ReLU nonlinearities, followed by a softmax. This added a term to the training loss: the cross entropy between the current belief state and the true current world state. The loss for the belief prediction was scaled by an additional hyperparameter, swept over the values 0.5, 2, and 5.

E. Additional results

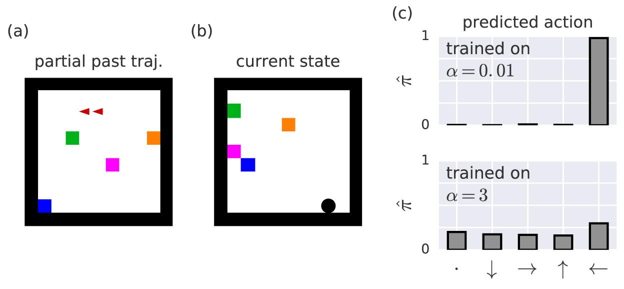

Figure A1. **Example gridworld in which a random agent acts.** **(a)** Example past episode. Coloured squares: objects. Red arrows: agent's positions and actions **(b)** Example query: a state from a new MDP. Black dot: agent position. **(c)-(d)** ToMnet's predictions when trained on species of agents with (a) near-deterministic policies, or (d) more stochastic policies.

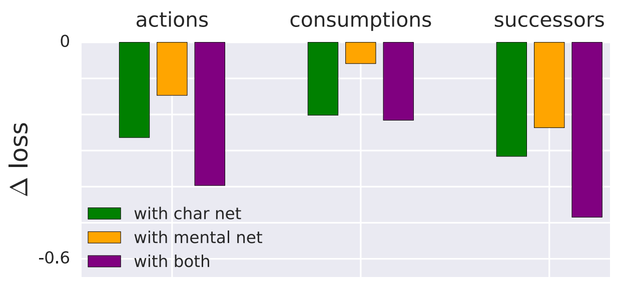

Figure A2. **Usefulness of ToMnet components for the three behavioural prediction targets, compared with a simple ToMnet with no character nor mental net.** Longer bars are better; including both character and mental nets is best. More details are given in Table A1.

Model	Train agents	Test agents
Action loss		
none	1.14	1.12
char net	0.84	0.86
+ shuffled e_{char}	1.61	1.62
mental net	0.83	0.98
+ shuffled e_{mental}	1.61	1.65
both	**0.72**	**0.73**
+ shuffled e_{char}	1.57	1.69
+ shuffled e_{mental}	1.16	1.20
+ shuffled both	1.99	2.02
Consumption loss		
none	0.34	0.36
char net	0.19	0.16
+ shuffled e_{char}	0.83	0.77
mental net	0.32	0.30
+ shuffled e_{mental}	0.43	0.43
both	**0.16**	**0.14**
+ shuffled e_{char}	0.82	0.78
+ shuffled e_{mental}	0.23	0.23
+ shuffled both	0.83	0.77
Successor loss		
none	2.48	2.53
char net	2.23	2.21
+ shuffled e_{char}	3.17	3.13
mental net	2.36	2.29
+ shuffled e_{mental}	2.92	2.80
both	**2.16**	**2.04**
+ shuffled e_{char}	3.27	3.19
+ shuffled e_{mental}	2.45	2.33
+ shuffled both	3.53	3.31

Table A1. **Full table of losses for the three predictions in Fig 6.** For each prediction, we report the loss obtained by a trained ToMnet that had no character or mental net, had just a character net, just a mental net, or both. For each model, we quantify the importance of the embeddings e_{char} and e_{mental} by measuring the loss when $e_{\text{char,i}}$ and $e_{\text{mental},i}$ are shuffled within a minibatch. The middle column shows the loss for the ToMnet's predictions on new samples of behaviour from the agents used in the trained set. The right column shows this for agents in the test set.

F. Additional notes

F.1. Hypersensitivity of 3×3 agents to swap events with swap distance 1

In Fig 11c, the policies of agents with 3×3 fields of view are seen to be considerably more sensitive to swap events that occur adjacent to the agent than the agents with 9×9 fields of view. Agents with 5×5 and 7×7 were similar to the 9×9 agents in their sensitivities.

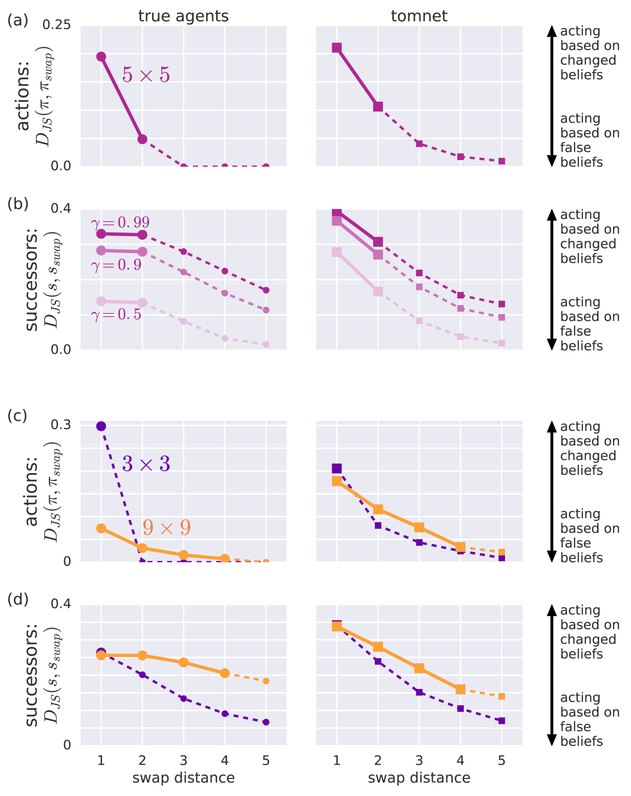

Figure A3. **Natural Sally-Anne test, using swap events within the distribution of POMDPs. (a)** For SRs of different discount factors (γ). D_{JS} measured between normalised SRs. **(b)** As for (a), but for a ToMnet trained on a range of agents with different fields of view. Showing only 3×3 and 9×9 results for clarity.

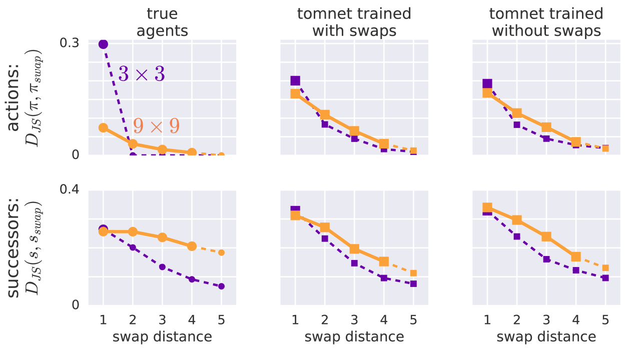

Figure A4. **ToMnet performance on the Natural Sally-Anne test does not depend on the ToMnet observing swap events during training.** The left two columns show the data presented in Fig 11 and Fig 12. The rightmost column shows the predictions of the ToMnet when it is trained on data from the same agents, but rolled out on POMDPs where the probability of swap events was $p = 0$ instead of $p = 0.1$.

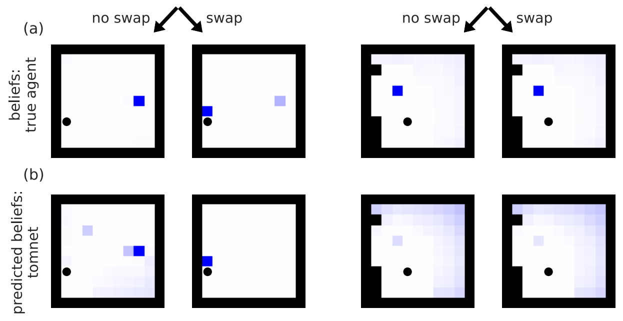

Figure A5. **Supervised prediction of beliefs. (a)** Belief states for the blue object's location ($b = p(\text{loc}|x_{0:t_{swap}})$) reported by the agent in the POMDPs shown in Fig 9 at the time of subgoal consumptions. Left two panels: the swap event occurred within the agent's field of view, so the agent's beliefs changed. Right two panels: the swap event was not within the agent's field of view, so its beliefs did not change. **(b)** Predictions $\hat{b}$ made by the ToMnet, given only the trajectory of states and actions. The ToMnet predicts that the observable swap event (left) leads to a change in belief state, whereas the unobservable swap event (right) does not.

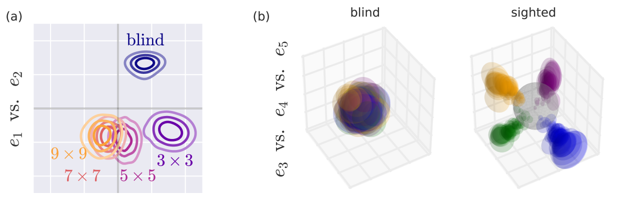

Figure A6. **Variational character embeddings of agents with different fields of view. (a)** First two dimensions of e_{char} represent field of view. Contours are shown of the marginal posteriors for each agent species. **(b)** Next three dimensions represent preferred objects. Volumes show the approximate marginal posteriors for agents preferring each of the four objects (colours). Blind agents (left) cannot express their preference through their overt behaviour; the ToMnet therefore reverts to the prior. Sighted agents (right) produce embeddings arranged in a roughly tetrahedral arrangement.

We did not perform a systematic analysis of the policy differences between these agents, but we speculate here as to the origin of this phenomenon. As we note in the main text, the agents were competent at their respective tasks, but not optimal. In particular, we noted that agents with larger fields of view were often sluggish to respond behaviourally to swap events. This is evident in the example shown on the left hand side of Fig 9. Here an agent with a 5×5 field of view does not respond to the sudden appearance of its preferred blue object above it by immediately moving upwards to consume it; its next-step policy does shift some probability mass to moving upwards, but only a small amount (Fig 9c). It strongly adjusts its policy on the following step though, producing rollouts that almost always return directly to the object (Fig 9d). We note that when a swap event occurs immediately next to an agent with a relatively large field of view (5×5 and greater), such an agent has the luxury of integrating information about the swap events over multiple timesteps, even if it navigates away from this location. In contrast, agents with 3×3 fields of view might take a single action that results in the swapped object disappearing altogether from their view. There thus might be greater pressure on these agents during learning to adjust their next-step actions in response to neighbouring swap events.

F.2. Use of Jensen-Shannon Divergence

In Sections 3.4–3.5, we used the Jensen-Shannon Divergence (D_{JS}) to measure the effect of swap events on agents' (and the ToMnet's predicted) behaviour (Figs 11-12). We wanted to use a standard metric for changes to all the predictions (policy, successors, and beliefs), and we found that the symmetry and stability of D_{JS} was most suited to this. We generally got similar results when using the KL-divergence, but we typically found more variance in these estimates: D_{KL} is highly sensitive the one of the distributions assigning little probability mass to one of the outcomes. This was particularly problematic when measuring changes in the successor representations and belief states, which were often very sparse. While it's possible to tame the the KL by adding a little uniform probability mass, this involves an arbitrary hyperparameter which we preferred to just avoid.