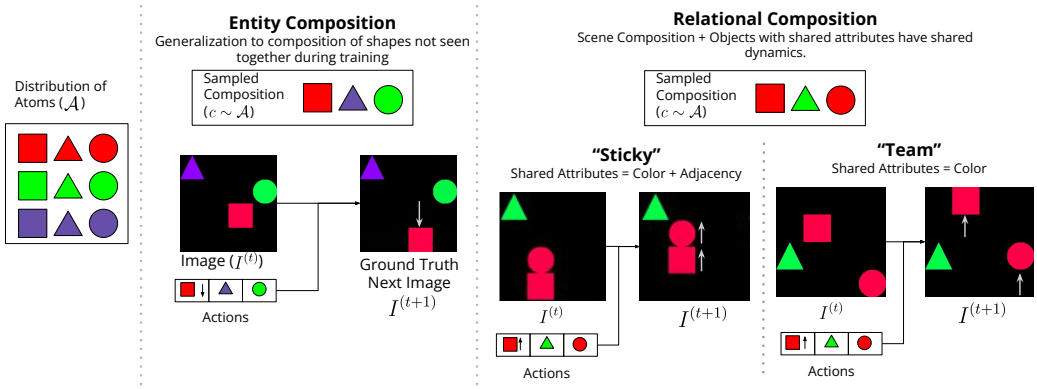

Figure 5: Overview of types of compositions studied. Entity composition (left) necessitates learning a world model that is equivariant to object replacement. Relational compositions (right) necessitates learning the properties of entity composition as well as additional constraints where objects with shared attributes also have shared dynamics. We study two instantiations of shared attributes sets: "Sticky" and "Team". Details on these instantiations are given in Appendix A.2.

A Appendix

The Appendix is divided into eight sections. Section A.1 explains how we leverage SAM to generate a set-structured representation, Section A.2 surveys the types of compositions we study in detail, Section A.3 introduces a faster algorithm used in implementation for module selection, Section A.4 outlines experiment details and, notably, presents justification for the Equivariant MRR metric employed to study encoding separation, Section A.5 presents details of how the downstream planning experiments were conducted, Section A.6 goes over our reasoning for selecting relevant benchmarks, Section A.7 details an ablation study with a "fully-symbolic" model, and Section A.8 showcases qualitative results on a randomly sampled subset of five-object state-action pairs.

A.1 Generating set-structured representations with Segment Anything

We prompt SAM (Kirillov et al., 2023) with the image (I) and a 8×8 grid of points. This yields 64×3 potential masks (as there are three channels). To ensure a set structured representation, we must ensure that (1) each mask captures a specific property of the image, (2) collectively, all masks describe the entire image. We ensure (1) by removing duplicate masks and (2) by evaluating all combinations of remaining slots and selecting the k tuple (where k is the number of slots) that, when summed, most closely matches the image. The resulting masks $\{M'_1, \dots M'_k\}$ are point-wise multiplied with the image to yield $\{I_1, \dots I_k\}$. Each masked image is passed through a finetuned Resnet to yield $\{S_1, \dots S_k\}$.

A.2 Types of Compositions

Entity Composition: Entity composition (Figure 5) necessitates learning a world model that is equivariant to object replacement. The dynamics of the environment depends on which objects are present in the scene.

Relational Composition: Relational composition (Figure 5) necessitates learning all the properties present in entity composition. Additionally, in relational composition, the composition is determined by constraints placed on observable attributes of individual objects. For instance, in Sticky block pushing (Fig 5), the scene is constrained so that two objects start out with the same color adjacent to each other; and an action on one object moves all objects of the same color with it. This gives the appearance of two objects being stuck to each other. At test time, the objects stuck together change. Sticky block pushing demonstrates compositionality constraints based on two attributes: position and color. In the team block pushing (Figure 5), we relax the adjacency constraint in the sticky block pushing domain. An action on any object also moves other objects of the same color. This

allows us to study whether the adjacency constraint places a larger burden on dynamics learning than the color constraint.

A.3 TRANSITION ALGORITHM

NPS (Goyal et al., 2021) necessitates selecting a primary slot (p) to be modified, a contextual slot (c), and a rule (r) to modify the primary slot in the presence of the contextual slot. The naive algorithm to compute this tuple has a runtime of $O(k^2 l)$ where k is the number of slots and l is the number of rules. However, in implementation, the selection of (r, p, c) can be reduced to a runtime of $O(kl + k)$ by *partial application* of the query-key attention. This is achieved by selecting the primary slot p and MLP_i, partially transforming S_p using a partial transition module $\text{MLP}_{(i,\texttt{left})}$, selecting the contextual slot c, and performing a final transformation of $\text{MLP}_{(i,\texttt{left})}(S_p)$ with S_c using $\text{MLP}_{(i,\texttt{right})}$. Algorithm 2 presents this faster algorithm.

Algorithm 2 Faster Transition Algorithm. This algorithm has a faster runtime than the one presented in the manuscript. The main difference is that the transition step is bifurcated into two parts, reducing the runtime of the selection from $O(k^2 l)$ to $O(kl + k)$ where k is the number of slots and l is the number of rules.

function TRANSITION(Key=$\{\overline{\Lambda}_1, \ldots \overline{\Lambda}_k\}$, Query=$\{\vec{R}_1, \ldots \vec{R}_l\}$, Value=$\{S_1, \ldots S_k\}$)
  $\mathbf{A}^\star \leftarrow$ **GumbelSoftmax**(**KQAttention**(key=$\{\overline{\Lambda}_1, \ldots \overline{\Lambda}_k\}$, query=$\{\vec{R}_1, \ldots \vec{R}_l\}$))
  $p, r \leftarrow$ **argmax**($\mathbf{A}^\star$, axis = `'all'`)
  $S^\star \leftarrow \text{MLP}_{r,\texttt{left}}(\text{concat}(S_p, \vec{R}_r))$
  $\mathbf{A}_2^\star \leftarrow$ **GumbelSoftmax**(**KQAttention**(key=$\{\overline{\Lambda}_1, \ldots \overline{\Lambda}_k\}$, query=$\{S^\star, S^\star, S^\star\}$))
  $c, \_ \leftarrow$ **argmax**($\mathbf{A}_2^\star$, axis = `'all'`)
  $S_2^\star \leftarrow \text{MLP}_{r,\texttt{right}}(\text{concat}(S_c, S^\star))$
  return $S_2^\star$

A.4 EVALUATION PROCEDURE

Dataset Generation: To generate each dataset, we first create a scene configuration file that prescribes the permissible shapes and colors for objects within a given dataset. The scene configuration ensures that $\mathcal{F}_\texttt{C}(\mathcal{D}_{train}) \cap \mathcal{F}_\texttt{C}(\mathcal{D}_{eval}) = \emptyset$. Next, we sample $\mathcal{D}_{train}$ and $\mathcal{D}_{eval}$ from the permissible scenes. Each dataset is a trajectory of state-action pairs, where the state is the image of the shape $3 \times 224 \times 224$ and the action is a vector, factorized by object ID (Object x North-East-South-West). Overall, we generate 3000 trajectories of length 32 each where the actions are sampled from a random uniform distribution. Our domain is equivalent to the observed weighted shapes setup studied in Ke et al. (2021) with a compositionality constraint where the weight of the objects depend only on the shapes.

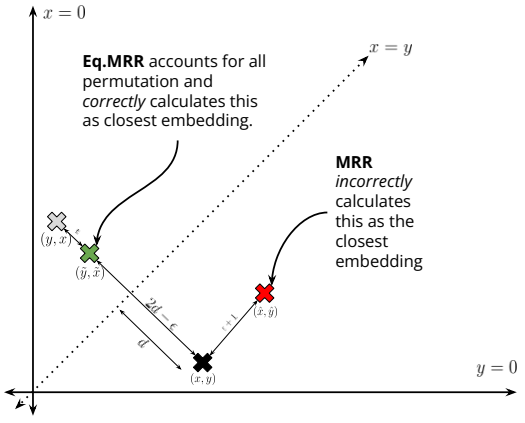

Figure 6: Intuition for shortcomings of MRR when number of slots $k = 2$ and $d_{slot} = 1$. The MRR metric incorrectly finds a point $(\hat{x}, \hat{y})$ that is $\epsilon + 1$ units away from (x, y) while Equivariant MRR considers all possible permutations and finds a point $(\tilde{y}, \tilde{x})$ that is ϵ units away from (y, x) and, in turn, closer to (x, y) than $(\hat{x}, \hat{y})$.

Baselines: We compare against past works in compositional world modeling with publically available codebases at the time of writing. For the block pushing domain, we compare against homomorphic world models (GNN) (Zhao et al., 2022) and an ablation of our model without symbols (ALIGNEDNPS). GNN uses a slot autoencoder, an action attention module and a graph neural network for modeling transitions. It requires a two-step training process: first

		3 objects			5 objects		
Dataset	**Model**	**MSE ↓**	**AE-MSE ↓**	**Eq.MRR ↑**	**MSE ↓**	**AE-MSE ↓**	**Eq.MRR ↑**
RC (Sticky)	COSMOS	**4.23E-03 +/- 1.49E-04**	**4.90E-04 +/- 1.03E-04**	**1.20E-01 +/- 2.05E-02**	**4.15E-03 +/- 3.21E-03**	**1.68E-03 +/- 1.48E-03**	**3.67E-01 +/- 7.73E-02**
	ALIGNEDNPS	1.14E-02 +/- 9.89E-04	7.72E-03 +/- 1.21E-03	8.01E-02 +/- 6.79E-02	6.07E-03 +/- 8.30E-04	2.47E-03 +/- 3.60E-04	3.62E-01 +/- 1.81E-02
	GNN	7.94E-03 +/- 5.47E-03	5.11E-03 +/- 4.94E-03	6.03E-04 +/- 1.02E-04	6.21E-03 +/- 1.26E-03	2.73E-03 +/- 1.27E-03	5.30E-04 +/- 5.15E-05
RC (Team)	COSMOS	**4.60E-03 +/- 2.32E-03**	**4.33E-04 +/- 1.58E-04**	1.04E-01 +/- 3.19E-02	**5.53E-03 +/- 1.95E-03**	1.86E-03 +/- 1.61E-03	2.86E-01 +/- 4.32E-02
	ALIGNEDNPS	1.24E-02 +/- 4.11E-04	8.36E-03 +/- 6.78E-04	**1.75E-01 +/- 2.68E-02**	9.64E-03 +/- 1.95E-04	3.12E-03 +/- 6.07E-04	**2.93E-01 +/- 2.02E-02**
	GNN	8.92E-03 +/- 6.05E-03	3.82E-03 +/- 3.64E-03	7.16E-04 +/- 1.09E-04	7.01E-03 +/- 9.81E-04	**1.62E-03 +/- 1.04E-03**	5.46E-04 +/- 1.37E-04
EC	COSMOS	**7.66E-04 +/- 4.08E-04**	**6.34E-05 +/- 2.01E-05**	2.99E-01 +/- 2.85E-02	**4.08E-04 +/- 4.68E-06**	**2.92E-06 +/- 6.34E-07**	3.03E-01 +/- 3.88E-02
	ALIGNEDNPS	3.51E-03 +/- 6.30E-04	2.69E-03 +/- 6.89E-05	2.97E-01 +/- 7.99E-02	2.45E-03 +/- 3.47E-04	1.22E-03 +/- 9.06E-04	3.19E-01 +/- 1.01E-01
	GNN	9.89E-03 +/- 5.77E-03	1.03E-02 +/- 5.44E-03	**5.50E-01 +/- 5.18E-01**	1.20E-02 +/- 1.13E-02	1.28E-02 +/- 1.08E-02	**5.25E-01 +/- 2.67E-01**

Table 2: Evaluation results on the 2D block pushing domain for entity composition (EC) and relational composition (RC) averaged across three seeds. This table includes standard deviation numbers as well. Our model (COSMOS) achieves best next-state reconstructions for all datasets.

warm starting the slot-autoencoder and then training the action attention model and GNN with an equivariant contrastive loss (Hungarian matching loss). ALIGNEDNPS uses a slot autoencoder for modeling perceptions and a NPS module (Goyal et al., 2021) for modeling transitions. The pipeline is trained end to end with contrastive loss. We use action attention with NPS as well. For both models, we weren't able to reproduce the results using the provided codebases due to issues in training robust perception models for large images ($3 \times 224 \times 224$). To ensure a fair comparison, and since both these methods are agnostic to the perception model, we opt to reimplement the core ideas for both these models and use the same fine-tuned perception model for all models.

Evaluation Procedure: We evaluate all models on a single 48 GB NVIDIA A40 GPU with a (maximum possible) batch size of 64 for 50 epochs for three random seeds. Contrastive learning necessitates a large batch size to ensure a diverse negative sampling set. As a result, the small batch size made contrastive learning challenging in our domain. To ensure a fair comparison, we report results for all models trained using reconstruction loss. We first train the slot autoencoder (ENTITYEXTRACTOR and SPATIALDECODER) until the model shows no training improvement for 5 epochs. This is sufficient to learn slot autoencoders with near-perfect state reconstructions. All transition models are initialized with the same slot-autoencoder and are optimized to minimize a mixture of the autoencoder reconstruction loss and the next-state reconstruction loss. For compositional world modeling, we are interested in two aspects of model performance: next-state predictions and separation between latent states. We evaluate next-state predictions on all models using the mean squared error (MSE) between the predicted next image and the ground truth next image in the experience buffer. We also measure the performance of the autoencoder on reconstructing the current state by calculating the slot-autoencoder mean squared error (AE-MSE). Generally, training world models improves the perception model's ability to reconstruct states as well. We also evaluate the separability of the learned latent encodings. This is done by measuring the L2 distance between the predicted next slot encodings and the ground-truth next slot encodings obtained from the encoder and using information theoretic measures such as mean reciprocal rank (MRR) to measure similarity. Notably, the MRR computation in previous work does not to account for the non-canonical order of slots, causing higher L2 distance and, consecutively, higher MRR scores when the target and predicted slots have different orderings. The core issue here is that MRR computation, as used in previous works, fixed the order of the slots before calculating L2 distance. This ignored $k! - 1$ possible orderings where a closer target encoding could be found. To rectify this, we propose a new metric, Equivariant MRR (Eq.MRR), which uses the minimum L2 distance among all permutations of slot encodings to calculate mean reciprocal rank. This metric ensures that the latent slot encodings are not penalized for having different slot orders. Figure 6 presents an illustration of the shortcomings of MRR on a simple example. This limitation is characteristic of algorithms which do not align the slots to a canonical slot ordering. In practice, we observe that the Equivariant MRR is always lower than or equal to the MRR.

A.5 DOWNSTREAM EVALUATION SETUP

Following Veerapaneni et al. (2020), we use a greedy planner that chooses the action that minimizes the Hungarian distance between the current and the goal state. These actions are applied $t - 1$ times over a trajectory of length t, with the output from the world model at the $(d - 1)$-th step becoming the state for step d in the trajectory. Due to this compounding nature, we see an increased divergence from the ground truth as we get deeper into the trajectory. At each step d in the trajectory,

Dataset	Only Symbols (MSE ↓)	Only Neural / AlignedNPS (MSE ↓)	COSMOS (MSE ↓)
3 Object RC - Sticky	1.36E-02	1.14E-02	**4.23E-03**
3 Object RC - Team	1.39E-02	1.24E-02	**4.60E-03**
3 Object EC	1.21E-02	3.51E-03	**7.66E-04**

Table 3: Evaluation results on the 2D block pushing domain for ablations of COSMOS.

the accuracy of the world model is evaluated as the L1 error of the difference between the current ground truth and predicted states in the form of their xy-coordinates. These xy-coordinates are initialized for each object to the origin and updated with every action taken by the corresponding rule. For example, after one step, if the ground truth moves an object to the east, but the planner chooses to move the same object to the west, then the distance between the two states would be 2. We run these experiments for the 500 trajectories of length $t = 32$ in our test dataset and average the scores at each trajectory depth. We showcase results in Figure 4. Our model (COSMOS) shows the most consistency and least deviation from the goal state in all datasets, which suggests that neurosymbolic grounding helps improve the downstream efficacy of world models.

A.6 DATASET COMPARISON

The focus of our paper is to demonstrate the first neuro-symbolic framework leveraging foundation models for compositional object-oriented world modelling, and we evaluate on the same benchmarks as existing work (Kipf et al., 2019; Goyal et al., 2021; Zhao et al., 2022). We chose our evaluation domain based on four properties: (1) object-oriented state and action space, (2) history-invariant dynamics, (3) action conditioned (plannable) dynamics, and (4) ease of generating new configurations (to evaluate entity and relational composition). We curate the following list of domains from related work to explain what properties are missing for each dataset.

Dataset and Relevant Works	Object Oriented State and Action Space	History Independent	Action Conditioned (Plannable)	Configurable
2D Block Pushing (Kipf et al., 2019; Goyal et al., 2021) (Zhao et al., 2022; Ke et al., 2021)	✓	✓	✓	✓
Physion (Wu et al., 2023)	✓			✓
Phyre (Wu et al., 2023)	✓			✓
Clevrer (Wu et al., 2023)	✓			
3DObj (Wu et al., 2023)	✓			✓
MineRL (Hafner et al., 2023)			✓	
Atari (Pong/Space invaders/Freeway) (Goyal et al., 2021)	✓			
Minigrid/BabyAI Mao et al. (2022)	✓		✓	✓

A.7 SYMBOLIC ABLATION OF COSMOS

ALIGNEDNPS serves as a "fully neural" ablation to demonstrate the effectiveness of our model. In this section, we detail another "fully symbolic" ablation of our model to demonstrate the need for a neurosymbolic approach. Specifically, we maintain the algorithm presented in 1 but modify the transition model to use the symbolic embedding to predict the next state. Specifically, line 9 changes to:

$$S_p \leftarrow S_p + \textsc{MlpBank}[r](\textbf{concat}(\Lambda_p,\ \Lambda_c,\ \vec{R}_r))$$

The results, detailed in Table 3, indicate that the "symbols-only" model significantly underperforms compared to COSMOS. We believe this is because the symbolic embedding is constructed by concatenating symbolic attributes, and the rule module is not aware of this structure. This causes the MLP to overfit to the attribute compositions seen at train time. COSMOS sidesteps this issue by using the symbolic embedding in the key-query attention module to select the relevant rule module, while allowing the real vector to learn local features useful for modeling action-conditioned transitions.

A.8 QUALITATIVE RESULTS

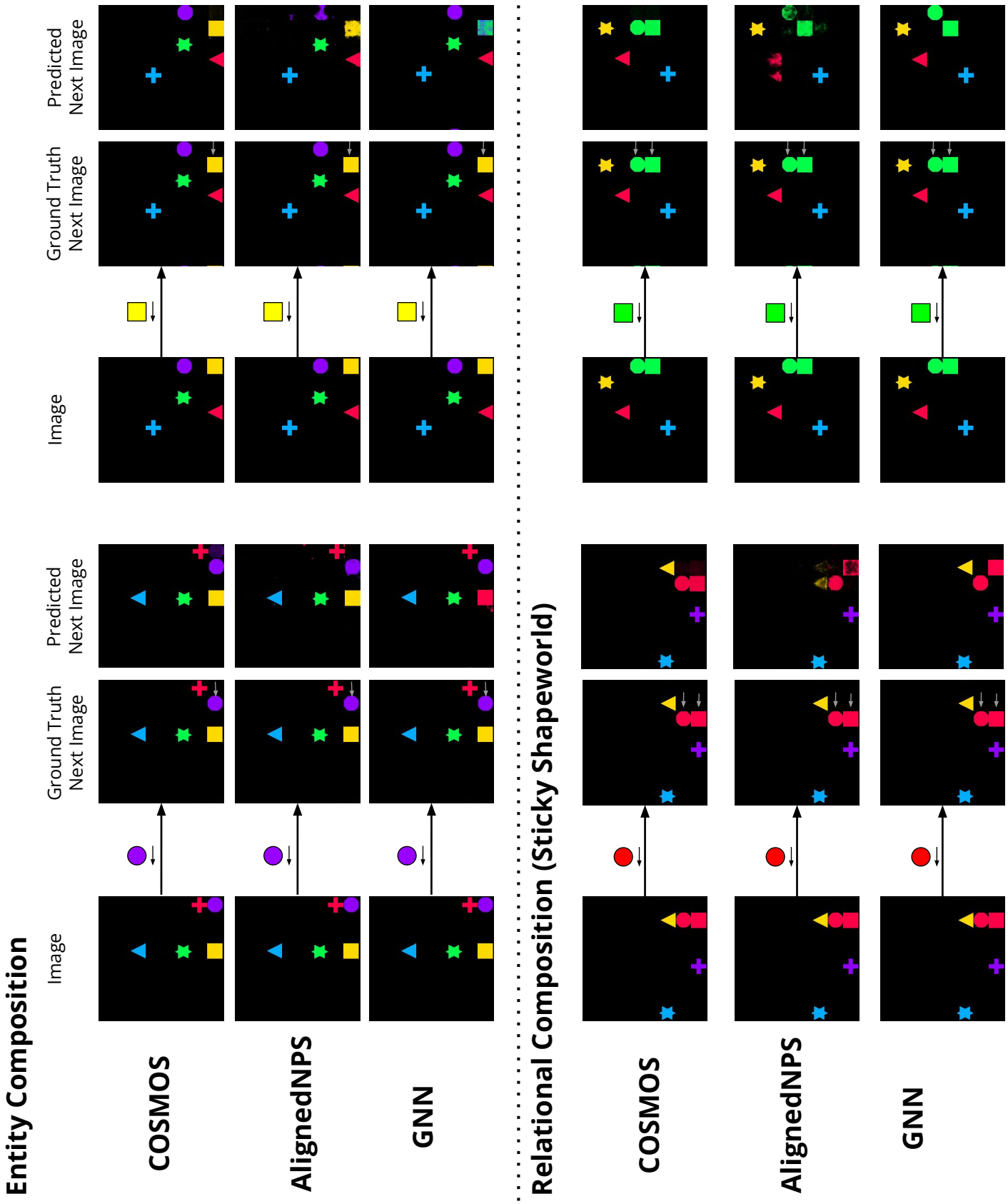

Figure 7: Qualitative outputs on randomly chosen state-action pairs for all baselines. We show two samples for each experiment and dataset type with 5 objects. Color is shown for illustrative purposes only; in implementation, the action conditioning does not carry any information about color.