Appendix for "Learning Dynamic Attribute-Factored World Models for Efficient Multi-object Reinforcement Learning"

A Detailed Discussion and Comparison with Related Work

In this section we discuss the related work. We first shortly discuss related work in terms of our modelling assumptions and then provide a comparison with other object-centric RL methods for compositional generalization.

A.1 Factored, Relational and Object-oriented MDPs

Our modelling assumptions, formalized as a Dynamic Attribute FacTored Markov Decision Process (DAFT-MDP) in Definition 1, are related to the literature on factored and object-oriented (PO)MDPs. In particular DAFT-MDPs are an first order extension with class template graphs, interaction patterns and interaction graphs of factored (PO)MDPs [18–23]. In particular they define a family of factored POMDPs, in which the objects $\mathcal{O}$ can vary across environments in numbers, types and latent parameters. This extension takes inspiration from Relational MDPs [24–26] and their literature, especially Object-Oriented (PO)MDPs [27–29], which also define a family of MDPs that vary in terms of objects and types. Most of these methods focus on discrete states and often define relations or states as first-order predicates, with the exception of [28], which propose a physics-based approach that circumvents these issues, allowing for more realistic robotic applications.

As opposed to these works, we also consider continuous state spaces and propose a more fine-grained factorization of the transition and reward at the object *attribute* level, based on estimating class template graphs and interaction pattern graphs. Additionally, we consider object-specific latent parameters that modulate the dynamics of each object and use *dynamic* graphs to account for the sparsity of interactions between objects and between the agent and an object.

A.2 Compositional Generalization with Object-centric RL Approaches

Table A1 provides a comparison of the object-centric RL methods discussed in Section 4. The criteria for comparison include the extent of factorization in the dynamics, the interaction modeling, and the model's ability to adapt to changes in the environment with changing latent factors.

Methods	**Factored Dynamics**	**Interaction Modeling**	**Generalize to Changing Latent Factors**
GNN [4]	Object	Fully-connected	✗
Self-attention [15]	Object	Fully-connected	✗
Deep Sets [15]	Object	Fully-connected	✗
LRN [7]	Object	Fully-connected	✗
COBRA [9]	Object	✗	✗
FWM [16]	Object	Fully-connected	✗
SMORL [5]	Object	✗	✗
STOVE [8]	Object & pre-determined attributes [a]	Fully-connected	✗
SRICS [6]	Object	Sparse & dynamic (object-level)	✗
NCS [12]	Object & action/dynamics-relevant	Fully-connected	✗
STEDIE [17]	Object & interaction-relevant[b]	Fully-connected (object-level), factored (interaction-relevant)	✗
DAFT-RL	Object & attribute	Sparse, dynamic & factored (object-level & attribute-level)	✓

[a] STOVE focuses specifically on these attributes: (position, velocity, size, others).

[b] STEDIE learns which attributes interact during object interactions, similar to our interaction pattern graphs.

Table A1: Comparison of different object-oriented RL methods for compositional generalization. A check denotes that a method has an attribute, whereas a cross denotes the opposite.

In the context of factored dynamics, most of these related works take into consideration the object-factored states. Approaches such as STEDIE [17], NCS [12], and STOVE [8] go a step further to disentangle the states related to interaction, dynamics, or actions from other state variables, such as color, as mentioned in the Table. However, our model aims to offer a more comprehensive approach to factored dynamics, providing an attribute-level factorization for each object. For example this can include factorizing the transition function of dynamics-related factors in terms of position, velocity and mass, and factorizing action-irrelevant factors like color and shape. In terms of interaction modeling, many approaches assume fully-connected interaction patterns, where interactions occur between every pair of objects within a given scene. However, in many real-world scenarios, this is often inaccurate as the object interaction usually happens in a sparse and dynamic manner. An exception is SRICS [6], which takes into account dynamic interaction modeling. This model emphasizes the sparsity of interactions among objects in a scene, with a dynamically changing structure at the object level. In contrast, STEDIE [17] employs learned factored dynamics for each object to model interactions, assuming that only interaction-irrelevant objects will have an impact on others. Nonetheless, from the perspective of the object level, the pattern of interaction in STEDIE is still fully connected. Our method diverges from these approaches by integrating both dynamic and sparse interaction patterns at the object level, and factored interactions at the attribute level between pairs of objects. We provide a more fine-grained and realistic model of interactions as from the object level, the interaction pattern is dynamic and sparse, and for each object pair, the interaction only happens among some specific attributes. Finally, DAFT-RL is the only framework that considers the latent changing factors in the environment, which is also a realistic setting where the agent sometimes cannot observe all essential attribute variables in the system but these unobserved factors do have effects on the dynamics or reward.

B Full Example with Summary of Notations

Notation	Definition
$\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_k\}$	Set of object classes
$\{C_j.s_1, C_j.s_2, \dots, C_j.s_n\}$	Class attributes for class C_j
$C_j.\boldsymbol{\theta}$	Latent constant parameters for class C_j
$\mathcal{O} = \{o_1, \dots, o_m\}$	Set of objects in a domain
$C(i)$	Class of object o_i, in other words C_j such that $o_i \in C_j$
$\{o_i.s_1, o_i.s_2, \dots, o_i.s_n\}$	Object attributes for object o_i with semantics defined by the class $C(i)$
$o_i.\boldsymbol{\theta}$	Latent constant parameters of object i
$\mathbf{x}^t$	Pixel observation at time step t
$\mathbf{x}_i^t$	Pixel observation of object-i at time step t
$\mathbf{o}_i^t = \{o_i.s_1^t, o_i.s_2^t, \dots, o_i.s_n^t\}$	Symbolic state of object o_i at time step t
$\mathbf{s}^t = \{\mathbf{o}_1^t, \dots, \mathbf{o}_m^t\}$	Observable state at time step t for environment with objects
$\boldsymbol{\theta} = \{o_1.\boldsymbol{\theta}, \dots, o_m.\boldsymbol{\theta}\}$	Latent constant parameters for environment with $\mathcal{O}$ objects
$\mathbf{a}_i^t$	Action on object o_i at time step t
$\mathbf{a}^t = \{\mathbf{a}_1^t, \dots, \mathbf{a}_m^t\}$	Action at time step t for environment with $\mathcal{O}$ objects
r_i^t	Reward on object o_i at time step t
$r^t = \Sigma_i r_i^t$	Reward at time step t
$\mathcal{G}_{C_j}$	Class template graph for object with class C_j
$\mathcal{G}_{C_i,C_j}$	Interaction pattern graph for objects with class C_i and C_j
$\boldsymbol{\alpha}^t$	Action selector at time step t
$\mathcal{G}_{o_i}^t$	Instantiation of the class template graph for o_i with class C_j at time t
$\mathcal{G}_{\text{inter}}^t$	Dynamic object interaction graph at time step t

Table A2: Summary of notations in this paper.

In this section, we first provide a summary of the notation in Table A2. Then we show an example of the environment described in the main paper, and how the learned graphs are connected in a single ground graphical model, as described in Figure A1.

In our example, we consider two classes $\mathcal{C} = \{C_1, C_2\}$, which represent the classes of boxes and switches, respectively. We represent these two classes with cubes and cylinders in Figure A1.

For the class of boxes C_1, represented in Figure A1A as a pink cube, we consider the attributes $\{C_1.s_1, C_1.s_2, C_1.s_3\}$ to represent the color, velocity and position of an abstract box in two consecutive timesteps t and $t+1$. All of the relationship between the attributes are described in the class template graph $\mathcal{G}_{C_1}$. The edge $C_1.s_1^t \rightarrow C_1.s_1^{t+1}$ represents the fact that the color at timestep $t+1$ is only influenced by the color at timestep t (in this case being constant). The edge $C_1.s_2^t \rightarrow C_1.s_2^{t+1}$ represents the fact that the velocity at timestep $t+1$ is influenced by the velocity at the previous timestep t. The edge $C_1.s_2^t \rightarrow C_1.s_3^{t+1}$ means that velocity can influence position in the next timestep,

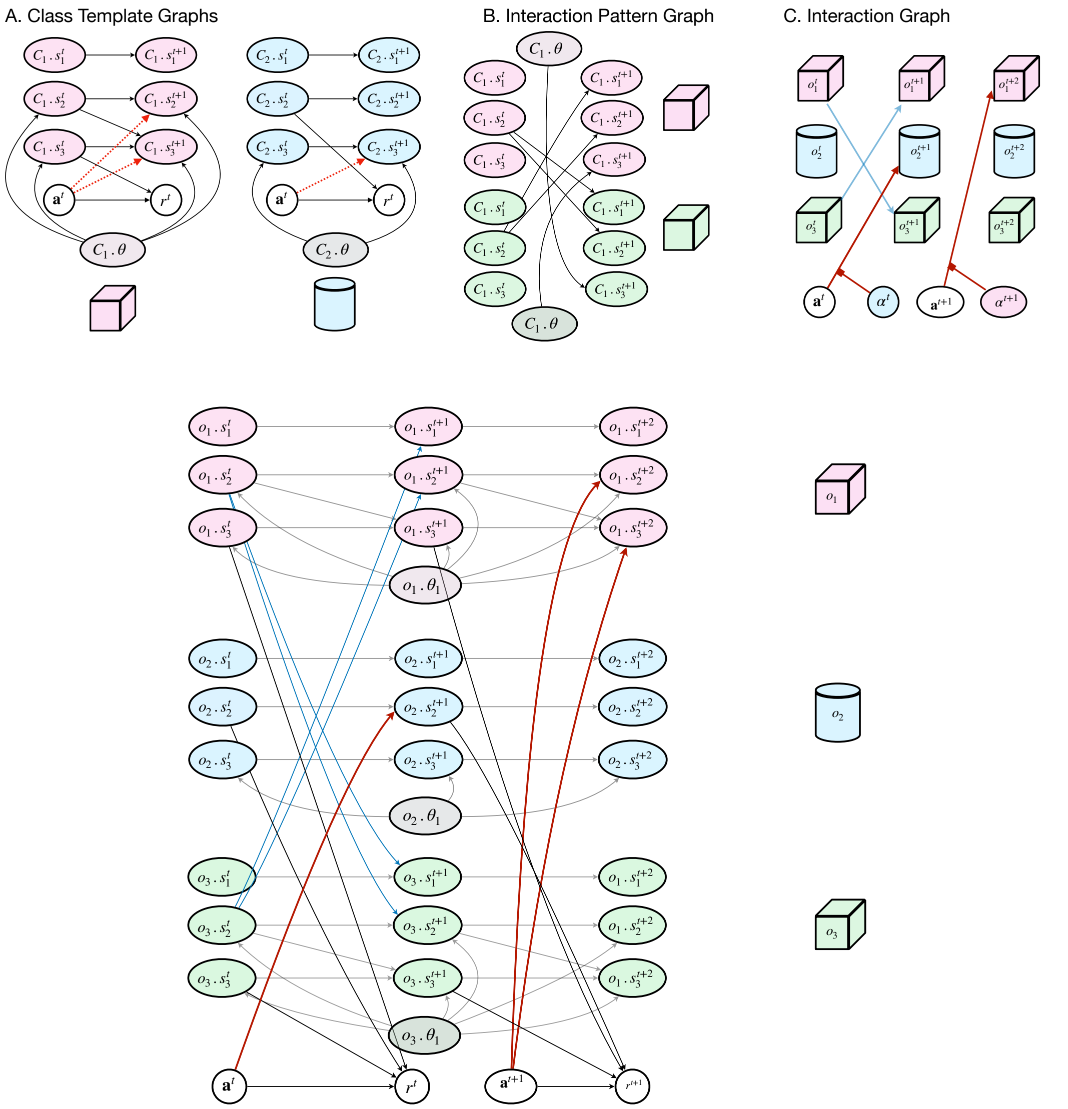

Figure A1: The graphical representation of DAFT-MDP. Fig A1.A represents the class template graphs for boxes and switches. Fig A1.B represents the interaction pattern graphs between two boxes. Fig A1.C represents the dynamic interaction graph, that at each timestep predicts which objects interact with each other and with the agent. The graph on the bottom shows an example of the instantiation of the complete ground graphical model for all of the objects in the environment from Fig A1.C, i.e. a pink box, a blue switch and a green box at time steps t, $t+1$ and $t+2$. The red lines describe the interactions of the agent with the objects (which follow the dotted lines in the class template graphs for each object class in Fig A1.A). The blue lines represent the interactions between objects, which follow the interaction patterns described in Fig A1.B.

which is also influenced by the previous position $C_1.s_3^{t+1}$. In this case the agent can optionally act (the dashed lines from $\mathbf{a}^t$) and influence the velocity $C_1.s_2^{t+1}$ and position $C_1.s_3^{t+1}$. Finally the latent constant parameters for the abstract box $C_1.\theta$, in this case representing friction, influence the velocity $C_1.s_2^{t+1}$ and position $C_1.s_3^{t+1}$ at each timestep $t+1$ and modulate the effect of the action on the position and velocity. The reward r^t only depends on the position $C_1.s_3^t$.

For the class of switches C_2, represented in Figure A1A as a blue cylinder, we consider the attributes $\{C_2.s_1, C_2.s_2, C_2.s_3\}$ to represent the color, angle and position of an abstract switch in two consecutive timesteps t and $t+1$. All of the relationship between the attributes are described in the class template graph $\mathcal{G}_{C_2}$. Similarly to boxes, the color $C_2.s_1$ does not influence anything and is only influenced by its previous value. In this example, we consider that none of the attributes influences each other, but that the reward r^t is only a function of the angle $C_2.s_2^t$. Additionally, the latent constant parameters $C_2.\theta$ only influence the position $C_2.s_3^t$, as can potentially the action $\mathbf{a}^t$ (represented by the dashed red lines).

In Figure A1B we show the interaction pattern graph for boxes $\mathcal{G}_{C_1,C_1}$ that represents the way that attributes of two objects of the class box C_1 interact with each other. In the figure, we use pink to represent the first object and green to represent the second object. Specifically, in this case, the interaction between two boxes means that the velocity of an object influences the velocity of another object. Similarly, the velocity of an object influences the color of the other object. Additionally, the latent constant parameters of an object influence the position of the other object.

In Figure A1C we consider a specific environment, specified by the objects $\mathcal{O} = \{o_1, o_2, o_3\}$, where o_1 and o_3 are boxes, while o_2 is a switch. We show an unrolled version of the dynamic interaction graph $\mathcal{G}_{\text{inter}}$ for three consecutive timesteps for these objects. At timestep t there is an interaction between the two boxes o_1 and o_3 (represented by blue lines), and the action $\mathbf{a}^t$ binds to the switch o_2, as selected by the action selector $\boldsymbol{\alpha}^t$. The interactions between the two objects are instantiated following the interaction pattern graphs in Figure A1B. The action binding instead activates the red dashed lines in the class template graph in Figure A1A. In the next timestep $t+1$ there are no interactions between objects, but the action is now bound to object o_1.

In the bottom graph in Figure A1, we show how the three types of graphs are combined in this environment for these three consecutive timesteps. In particular, it can be seen that for each object, its attributes follow the same pattern as described by the class template graphs $\mathcal{G}_{C_1}$ and $\mathcal{G}_{C_2}$ (in grey). The interactions between the two boxes in timestep t get instantiated following the interaction pattern graphs $\mathcal{G}_{C_1,C_1}$ (in light blue). The action binding specifies which object is affected by the action $\mathbf{a}^t$ at each timestep, but the way the action affects the attributes of an object is based on its class template graph. This graph represents the complete graphical model representation for these three timesteps, that describes how the transition and reward functions factorize in these three timesteps.

As can be seen from the example, to learn the factorization of the transition and reward functions in a new environment, we can reuse the class template graphs and the interaction pattern graphs across different domains with the same type of objects. We only need to relearn the dynamic interaction graph, which is also the only graph that is dynamic (i.e. the edges do not repeat across every couple of consecutive timesteps).

C Architecture and Implementation Details

In this section, we describe the architecture of DAFT-RL in more detail. Figure A2 summarizes the pipeline of learning the DAFT-RL framework, which uses the notation summarized in Table A2. We first provide a high-level illustration of the framework, and then provide the details in the following subsections. In the offline model estimation, the DAFT-MDP is learned through a two-step process, involving the estimation of both the class template graphs (Step 1), the interaction pattern graphs and dynamic interaction graphs (Step 2). These graphs capture the relationships and dependencies among objects from the attribute level in the environment. Once the DAFT-MDP is learned, policy learning is performed using trajectory imagination and planning methods (Step 3). During the adaptation phase, the dynamic graphs are inferred for the new domain (Step 4). This inference step allows for the transfer of the previously trained policy to the new domain without extensive retraining. By adapting the dynamic graphs to the new domain, the policy can be directly deployed and applied to the new environment. In the following, we define the graph structures and losses for each of these steps.

C.1 Binary Matrices in DAFT-RL

We first introduce the definitions of all structures that are used in the loss functions in DAFT-RL:

Binary matrices in class template graphs. As described in the previous sections, we learn the class template graphs $\mathcal{G}_{C_j}$ for each object class C_j. This graph is assumed to be constant in time. In practice, these graphs are learned as binary adjacency matrices. To help express the loss functions in a factorized way, we define the following binary matrices.

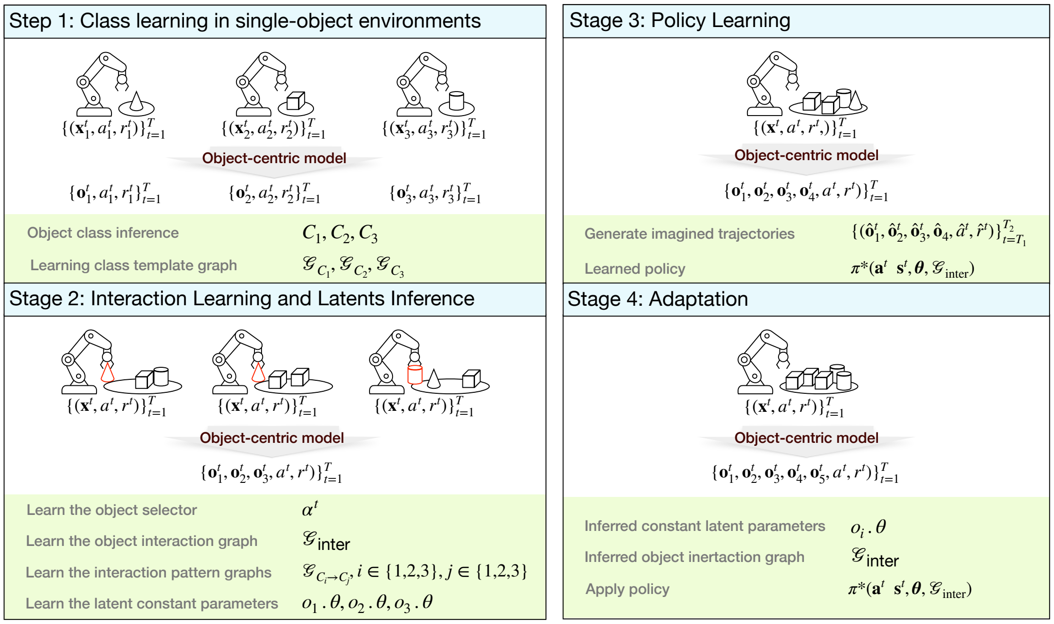
Step 1: Class learning in single-object environments
$\{(\mathbf{x}_1^t, a_1^t, r_1^t)\}_{t=1}^T$
$\{(\mathbf{x}_2^t, a_2^t, r_2^t)\}_{t=1}^T$
$\{(\mathbf{x}_3^t, a_3^t, r_3^t)\}_{t=1}^T$
Object-centric model
$\{\mathbf{o}_1^t, a_1^t, r_1^t\}_{t=1}^T$
$\{\mathbf{o}_2^t, a_2^t, r_2^t\}_{t=1}^T$
$\{\mathbf{o}_3^t, a_3^t, r_3^t\}_{t=1}^T$
Object class inference
C_1, C_2, C_3
Learning class template graph
$\mathcal{G}_{C_1}, \mathcal{G}_{C_2}, \mathcal{G}_{C_3}$
Stage 2: Interaction Learning and Latents Inference
$\{(\mathbf{x}^t, a^t, r^t)\}_{t=1}^T$
$\{(\mathbf{x}^t, a^t, r^t)\}_{t=1}^T$
$\{(\mathbf{x}^t, a^t, r^t)\}_{t=1}^T$
Object-centric model
$\{\mathbf{o}_1^t, \mathbf{o}_2^t, \mathbf{o}_3^t, a^t, r^t)\}_{t=1}^T$
Learn the object selector
α^t
Learn the object interaction graph
$\mathcal{G}_{\text{inter}}$
Learn the interaction pattern graphs
$\mathcal{G}_{C_i \to C_j}, i \in \{1,2,3\}, j \in \{1,2,3\}$
Learn the latent constant parameters
$o_1 . \theta, o_2 . \theta, o_3 . \theta$
Stage 3: Policy Learning
$\{(\mathbf{x}^t, a^t, r^t,)\}_{t=1}^T$
Object-centric model
$\{\mathbf{o}_1^t, \mathbf{o}_2^t, \mathbf{o}_3^t, \mathbf{o}_4^t, a^t, r^t)\}_{t=1}^T$
Generate imagined trajectories
$\{(\hat{\mathbf{o}}_1^t, \hat{\mathbf{o}}_2^t, \hat{\mathbf{o}}_3^t, \hat{\mathbf{o}}_4, \hat{a}^t, \hat{r}^t)\}_{t=T_1}^{T_2}$
Learned policy
$\pi^*(\mathbf{a}^t \;\; \mathbf{s}^t, \boldsymbol{\theta}, \mathcal{G}_{\text{inter}})$
Stage 4: Adaptation
$\{(\mathbf{x}^t, a^t, r^t)\}_{t=1}^T$
Object-centric model
$\{\mathbf{o}_1^t, \mathbf{o}_2^t, \mathbf{o}_3^t, \mathbf{o}_4^t, \mathbf{o}_5^t, a^t, r^t)\}_{t=1}^T$
Inferred constant latent parameters
$o_i . \theta$
Inferred object inertaction graph
$\mathcal{G}_{\text{inter}}$
Apply policy
$\pi^*(\mathbf{a}^t \;\; \mathbf{s}^t, \boldsymbol{\theta}, \mathcal{G}_{\text{inter}})$

Figure A2: The learning pipelines of the DAFT-RL framework.

For $l = \{1, \ldots, n\}$ we define:

- $C_j.G_{s \to s_l} \in \{0,1\}^n$ as the subset of the adjacency matrix of $\mathcal{G}_{C_j}$ that describes the edges incoming into $C_j.s_l^{t+1}$ from each $C_j.s_i^t$ for $i \in \{1, \ldots, n\}$. In other words this binary vector is 1 only for the components $C_j.s_i^t$ for $i \in \{1, \ldots, n\}$ that have an edge to $C_j.s_l^{t+1}$ in $\mathcal{G}_{C_j}$.
- $C_j.G_{a \to s_l} \in \{0,1\}$ as a binary value that represents the existence of an edge between $\mathbf{a}^t$ and $C_j.s_l^{t+1}$, denoting how the action will affect the dynamics at the attribute s_l at next time step.
- $C_j.G_{\theta \to s_l} \in \{0,1\}$ as a binary value that represents the existence of an edge between $C_j.\theta^t$ and $C_j.s_l^{t+1}$

We additionally define:

- $C_j.G_{s \to r} \in \{0,1\}^n$ is the subset of the adjacency matrix of $\mathcal{G}_{C_j}$ that describes the edges incoming into r^{t+1} from each $C_j.s_i^t$ for $i \in \{1, \ldots, n\}$
- $C_j.G_{a \to r} \in \{0,1\}$ is a binary value that represents the existence of an edge between $\mathbf{a}^t$.

For each object class C_j all of these five groups of binary matrices together represent the adjacency matrix of graph $\mathcal{G}_{C_j}$.

Binary matrices in interaction pattern graphs. Interaction pattern graphs G_{C_i,C_j} represent the interaction structures between attributes of the object with class C_i and the object with class C_j, which are assumed to be constant in time. For each $l = \{1, \ldots, n\}$ we define $G_{C_i \to C_j.s^l} \in \{0,1\}^n$ to represent the binary vector that describes how the attributes of an object with class C_i influence the attribute $C_j.s_l^{t+1}$ of the object with class C_j. All of these vectors together represent the adjacency matrix of graph G_{C_i,C_j}.

Binary values in Dynamic interaction graphs The dynamic interaction graphs G_{inter}^t represent the interactions between the objects $\mathcal{O}$ at timestep t, which include the action selector $\{\boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2, \ldots, \boldsymbol{\alpha}^T\}$, where $\boldsymbol{\alpha}^i \in \{0,1\}^m$ represents which object is affected by $\mathbf{a}^t$ at all time steps $t = 1, \ldots, T$. We use the action selection $\boldsymbol{\alpha}^i$ in our losses. For each $o_i, o_j \in \mathcal{O}$ we define a binary value $G_{\text{inter}(i,j)}^t \in \{0,1\}$

to represent whether object o_i and o_j will have the interaction at time step t. All of these values together represent the adjacency matrix of graph G^t_{inter}.

C.1.1 Detailed Loss Functions

We provide the detailed loss functions for learning the DAFT-MDP below. In each step, we mark in blue which of the structures defined in the previous subsections are learn at that step.

Step 1: Class learning in single-object environments As described in Sec 3.1, we learn the class template graphs $\{\mathcal{G}^{\text{C}_1}, \mathcal{G}^{\text{C}_2}, \dots, \mathcal{G}^{\text{C}_\text{k}}\}$ using the collected $\mathcal{D}^{\text{single}}$. We give the detailed loss functions, including the prediction loss $\mathcal{L}^1_{\text{pred}}$ and the sparsity regularization $\mathcal{L}^1_{sparse}$ for dynamics and reward models, for this step below:

$$\begin{aligned}\mathcal{L}^1_{\text{pred}} = \sum_{t=1}^{T}\sum_{i=1}^{m}\sum_{l=1}^{n}\Big(&\log p_{\lambda_s}(o_i.s_l^{t+1} \mid C_j.G_{s\to s_l} \odot \mathbf{o_i}^t, C_j.G_{a\to s_l} \odot \mathbf{a}_i^t) \\ &+ \log p_{\lambda_r}(r_i^t \mid C_j.G_{s\to r} \odot \mathbf{o_i}^t, C_j.G_{a\to r} \odot \mathbf{a}_i^t)\Big)\end{aligned} \tag{A1}$$

$$\mathcal{L}^1_{\text{sparse}} = \sum_{j=1}^{k}\left(\sum_{l=1}^{n}\|C_j.G_{s\to s_l}\|_1 + \|C_j.G_{s\to r}\|_1 + \sum_{l=1}^{n}\|C_j.G_{a\to s_l}\|_1 + \|C_j.G_{a\to r}\|_1\right) \tag{A2}$$

where we update the prediction models including λ_s, λ_r, as well as the binary vectors defined previously $\{\{C_j.G_{s\to s_l}\}_{l=1}^n, \{C_j.G_{a\to s_l}\}_{l=1}^n, C_j.G_{s\to r}, C_j.G_{a\to r}\}$ (marked in blue). The complete loss $\mathcal{L}^1$ for Step 1 is:

$$\mathcal{L}^1 = \mathcal{L}^1_{\text{pred}} + \gamma^1_{\text{sparse}}\mathcal{L}^1_{\text{sparse}} \tag{A3}$$

where γ^1_{sparse} is a hyper-parameter.

Step 2.1: Learning the action binding In this step, we learn the dynamic action selector $\boldsymbol{\alpha}^t$ by using the soft attention networks. We have collected $\mathcal{D}^{\text{multi}}$ which have multiple objects in the scene. We consider that the class template graphs learned in the previous phase are fixed and update the prediction loss with the action selector. The detailed loss function for this stage is:

$$\mathcal{L}^2_{\text{pred}} = \sum_{t=1}^{T}\sum_{i=1}^{m}\sum_{l=1}^{n}\log p_{\lambda_s}\left(o_i.s_l^{t+1} \mid C_i.G_{s\to s_l} \odot \mathbf{o_i}^t, C_i.G_{a\to s_l} \odot \alpha_i^t \cdot f_v(\mathbf{a}^t)\right) \tag{A4}$$

where f_v is the value network to be updated. We also need to learn the key and query networks f_k and f_q to update α_i^t for each object o_i (marked in blue). Additionally, we use the same dynamic network λ_s and also update it in this stage.

Step 2.2: learning dynamic interaction graph As described in Sec 3.2.2, we use dNRI [37] to learn the dynamic interaction graph. Additionally, we also learn all the interaction pattern graphs, as well as learn how to infer latent parameters, and how they influence each attribute, at this step. We consider the class template graphs and action binding are fixed from the previous step. Detailed loss functions are:

$$\begin{aligned}\mathcal{L}^3_{\text{pred}} = \sum_{t=1}^{T}\sum_{i=1}^{m}\sum_{l=1}^{n}\log p_{\lambda_s}(o_i.s_l^{t+1} \mid\ & C_i.G_{s\to s_l} \odot \mathbf{o_i}^t, C_i.G_{a\to s_l} \odot \alpha_i^t \cdot f_v(\mathbf{a}^t), \\ & C_i.G_{\theta\to s_l} \odot o_i.\theta, \{G^t_{\text{inter}(i,j)} \cdot G_{C_j\to C_i.s_l} \odot \mathbf{o}_j^t\}_{j\in\{1,\dots,m\}\setminus i})\end{aligned} \tag{A5}$$

$$\mathcal{L}^3_{\text{KL}} = \sum_{t=2}^{T}\text{KL}\left(q_\phi\left(\mathbf{z}^t \mid \mathbf{s}^{1:T}\right)\|p_\phi\left(\boldsymbol{z}^t \mid \boldsymbol{s}^{1:t}, \boldsymbol{z}^{1:t-1}\right)\right) \tag{A6}$$

$$\mathcal{L}^3_{\text{sparse}} = \sum_{i=1}^{k}\sum_{j=1}^{k}\sum_{l=1}^{n}\|G_{C_i\to C_j.s_l}\|_1 + \sum_{i=1}^{k}\sum_{l=1}^{n}\|C_i.G_{\theta\to s_l}\|_1 \tag{A7}$$

where λ_s is the dynamics model we have employed in the previous steps. We learn the encoder/prior networks ϕ to generate the latent vectors $\mathbf{z}$, where we sample the graph G_{inter} from. Through the binary vectors we also learn the interaction pattern graphs $\{G_{C_i,C_j}\}_{i,j\in\{1,\dots,k\}}$, latent parameters $\{o_i.\theta\}_{i=1}^{m}$ and the binary vectors representing the edges from latent parameters to each attribute $\{C_i.G_{\theta\to s_l}\}_{i=1}^{k}$. We have completed the loss for step 2.2:

$$\mathcal{L}^3 = \mathcal{L}^3_{\text{pred}} + \gamma^3_{\text{KL}}\mathcal{L}^3_{\text{KL}} + \gamma^3_{\text{sparse}}\mathcal{L}^3_{\text{sparse}} \tag{A8}$$

where γ^3_{KL} and γ^3_{sparse} are the hyper-parameters.

C.1.2 Details for the Other Algorithms

Object class learning For the case with pixel observation as input, we choose to learn the object classes in a supervised manner, where the input includes the extracted feature vectors $\mathbf{o}_i$ for each object and we have the labels y_i for each single objects. We apply a cross-entropy loss to learn the mapping.

Interaction graph sampling During the learning of dynamic interaction graphs, we generate the edge distribution $\mathbf{z}^t$ at each time step t. We sample the edge matrix $\mathbf{M}^t$ of the graph $\mathcal{G}^t_{\text{inter}}$. Specifically, $\mathbf{M}^t \sim \text{Bern}(\mathbf{z}^t)$, where Bern is the multivariate Bernoulli distribution and all elements are mutually independent. We also employ the Gumbel-Softmax trick [40] to make the sampling process differentiable.

D Experimental Details

In this section, we summarize all the experimental details. We first discuss our baselines and how we modified them to evaluate them in a fair comparison with DAFT-RL, focusing in particular on adapting symbolic input approaches to pixel inputs and image-based approaches to symbolic inputs. We then describe the experimental setups, including each task description and the modifications we made to the benchmarks.

D.1 Baseline Modifications

Symbolic input approaches with pixel inputs. We adapt various approaches that rely on symbolic inputs, such as self-attention [15], deep sets [15], GNN [4], SRICS [6] and LRN [7], to handle scenarios where the inputs are raw pixels. To accomplish this, we leverage pre-trained object-centric models, specifically AIR [31], to obtain the state representation of each object within the scene. The pre-trained object-centric models, specifically AIR, are employed to extract object-specific information from the raw pixel inputs. These object-factored states, which represent the individual objects, are then used as inputs for the approaches mentioned above. By incorporating the object-centric models and leveraging their extracted object representations, we enable the symbolic approaches originally designed for other input formats to be applicable in scenarios where raw pixels serve as the input representation.

Image-based approaches with symbolic inputs. To modify the image-based approaches to fit with the benchmarks with symbolic inputs, we adopt the following changes:

- SMORL [5]: we remove the SCALOR encoder and directly obtain $\mathbf{z}$ from the simulator.
- STOVE [8]: similarly, we remove the SuPAIR model and directly observe object-factored states for dynamics and policy modeling.
- NCS [12]: we directly observe the type and state variables from the simulator without the slot attention models.

Notably, we do not and cannot modify COBRA [9] for the symbolic case, because COBRA is mostly built upon the scene encoder and decoder.

D.2 Experimental Setups

In this section, we give the task description for each of the three benchmarks and describe the detailed experimental settings for both model estimation and policy learning.

D.2.1 Task Description

OpenAI Fetch - Push and Switch. In this benchmarks the tasks are to push N boxes or flip M switches. The agent can obtain both the object factored states and the global (agent) states. Specifically, the object states are the object's pose and velocity. For switches, the states are also the switch angle and the position. The action is the 3D continuous control of the end effector displacement and 1D control of the distance between gripper fingers. The goals are the target position for cubes (pushing task) and the target angle of the switches (switching task). In the experiment, we consider a dense reward, computed by the distance between the object's states and the desired states in the goals.

Spriteworld. There are four tasks in the Spriteworld benchmark, as listed below. We follow the task sets as well as the modifications in [13]. The agent directly observes the 2D images as the observation states.

- **Object goal task**: the goal is to move the object to the target position without touching other distractor objects. The action includes four cardinal directions. A positive reward will be given if the goal is achieved.
- **Object interaction task**: the goal is to push the target to a given location in the scene. The reward will be obtained if the agent pushes the target to the goal position and the action also includes four cardinal directions.
- **Object comparison task**: the agent needs to figure out which object is different from other objects and move this object to the given position. The reward will be given if the goal is achieved and the action also includes four cardinal directions.
- **Property comparison task**: similarly, the agent needs to find out the object with a different color or shape from other objects. This task is generally more challenging as the agent needs to reason the property-level difference while the object comparison task only requires object-level reasoning. The reward and action are the same as the object comparison task.

Stacking. We follow the block-stacking experimental setups in [17]. In the block-stacking task, the agent can observe the images from the current scene and the goal. The action includes picking and placing the objects and coordinates [61]. The goal is to stack m objects with a given position.

D.2.2 Benchmark Modifications

Color, shape, friction and mass in Push & Switch To make the benchmark more realistic, we add two variables into the original states in the benchmark, the object color and shape, both are represented as one-hot vector. Additionally, we also use a different friction coefficient and mass for each object. We implement this by conducting a post-processing filter for each object in the simulator. Specifically, different friction coefficients will result in slower velocity and action effects. Different masses will result in different action effects of the agent. During the training of the estimation model, we use the objects with masses and friction coefficients uniformly sampled from $\{4, 6, 8, 10\}$ and $\{0.4, 0.6, 0.8, 1.0\}$ respectively. During testing, the object masses and friction coefficients are sampled from $\{1, 2, 3, 11, 13\} \cup \{3, 5, 7, 9\}$ and $\{0.1, 0.2, 1.1, 1.3\} \cup \{0.5, 0.7, 0.9\}$ respectively.

Unseen colors, shapes, and numbers of objects in Spriteworld To evaluate the generalization capabilities of our model, we take into account the number of unseen objects, shapes, and colors. We achieve this without directly modifying the benchmark, as it offers the flexibility to alter these properties. During the training phase for model estimation, we use the colors green, red, and yellow, along with a varied number of objects ranging from $\{3, 5, 7\}$. We consider as shapes circles and boxes. During the testing phase, we introduce new elements that were not seen during training. Specifically, we incorporate the color blue and triangle shapes. Additionally, the number of objects is extended to include $\{2, 4, 6, 8\}$. By considering these changes in colors, shapes, and object numbers, we aim to assess the model's ability to generalize to novel configurations and demonstrate its robustness.

Mass in Stacking We adjust the mass of each block by modifying the action effects on the object, treating them as post-processing filters applied to the image frames. In particular, picking up a heavier object will require more time compared to selecting a lighter one due to gravity. Placing a heavier one will make it faster than the lighter one. During training, we consider the masses ranging from $\{1, 1.5, 2, 2.5\}$. During testing, we make the masses of the objects ranging from $\{0.5, 1.25, 2.25, 3\}$.

D.2.3 Offline Model Estimation

In Step 1, we use the GRU model [39] as both the dynamics and reward prediction model to learn the class template graph. The hidden size of the GRU is set to 128. Additionally, we incorporate two MLP layers before and after the GRU, each with a hidden size of 64. During training, we continue optimizing the model until the dynamics and reward models converge, employing the mean squared error (MSE) loss with the Adam optimizer. In Step 2, we use MLP layers to parameterize the soft attention layers [35]. Specifically, for the functions f_k, f_q, and f_v, the hidden sizes are all set to 128, with only one hidden layer for each of them. To learn the dynamic interaction graph, we follow the hyperparameters and model architectures outlined in [37]. However, as opposed to them, we use the previously learned GRU in Step 1 as the decoder. For various experiments, we adopt different hyperparameters for the GRU decoder. The training details for each experiment are provided below.

Push & Switch In Step 1, we gather a total of 400 trajectories for both boxes and switches (200 for each). Each trajectory consists of 100 time steps, and the actions are generated using a random policy. The batch size for this stage is set to 80, and the weighting parameter before the regularization term is 0.015. In this step, in each trajectory, there is only one single object and different trajectories have different objects with different types (e.g., box or cubes), different colors (e.g., blue, red, etc), and different latent parameters (e.g., friction coefficients). In Step 2, we collect 500 trajectories, where each trajectory consists of 50 time steps, for learning the action binding selector. The batch size for training the soft attention network is set to 100. Regarding the learning of dynamic interaction graphs, we use the same parameters and model architectures as described in [37]. For the decoder, we reuse the decoder used in Step 1, which includes the interaction pattern graphs and the class template graphs. The balancing weights before the KL divergence and the sparsity regularization are set to 0.9 and 0.2, respectively. During training, we set the learning rate to 2.5×10^{-5}, and we apply a decaying factor of 0.5 every 100 training epochs.

Spriteworld We adopt the same pre-trained procedures outlined in [13] for the object-centric models. Specifically, for both SA (Slot Attention) [30] and AIR (Attend, Infer, Repeat) [31], we generate a dataset containing various objects with distinct colors and shapes. The dataset comprises 200 trajectories, each consisting of 50 time steps. Within each frame, there are typically 2-5 objects exhibiting a diverse range of colors and shapes. For Step 1, we use the collected dataset of 300 trajectories (with 50 time steps each) to learn the class template graphs with different objects. Each trajectory has the transitions of one single object. The weighting parameter before the sparsity regularizer is set to 0.3. The batch size is 64 and the learning rate is 0.002. In our Spriteworld scenario, dense reward signals are absent. Therefore, we solely focus on learning the graphs between states and actions for most steps, excluding those that achieve the goal. Additionally, there is no direct interaction among objects, which means we do not train Step 2 for inferring the interactions. This setup aligns with [13] and is consistent across all the baseline methods we employ.

Stacking We also pre-train the SA and AIR using the randomly collected dataset containing 200 trajectories, each with 30 time steps. The blocks are initialized with different colors in the dataset. To train Step 1, we collect the single-block dataset with 200 trajectories, each with 30 time steps. The balancing weight for sparsity regularization is 0.05. For Step 2, we have the dataset with 300 trajectories, each with 30 time steps and multiple objects. We use the same set of hyper-parameters for learning the soft attention networks as Push & Switch. For learning the dynamic interaction graph, we balanced weights before the KL divergence and the sparsity regularization is 0.2 and 0.1, respectively. We use the same learning rate to 2.5×10^{-5}, and we apply a decaying factor of 0.5 every 100 training epochs.

D.2.4 Policy Learning and Planning

For the Push & Switch and Spriteworld experiments, we use PPO [42] for all baselines with the trajectories generated by the learned dynamics and reward functions. The learning rate for PPO is 0.005 and 0.003 in these two benchmarks, respectively. The coefficient of the entropy term in PPO is 0.05. The policy is parameterized with 3 MLP layers of size 256 and 128 in both experiments. For Stacking, we follow all the MPC hyper-parameter settings in [17] for the planning.

D.3 Full Results

We provide the full results of the experiments in this section, including the quantitative results for all experiments, ablation studies, and the visualized graphs.

D.3.1 Quantitative Results

Table A3 and A4 give the results of the single training task in Push & Switch and Spriteworld benchmarks. Table A5 and A6 provide the results of Spriteworld with 1) changing object numbers and 2) changing object colors and shapes simultaneously during the testing phase, respectively.

Experiment Settings	**Method**								
	DAFT (Symbolic)	Deep Set	Self-attention	SRICS	GNN	STOVE	Self-attention	NCS	Relational RL
2-Push	0.959 ± 0.031	0.985 ± 0.025	0.943 ± 0.024	0.981 ± 0.015	0.923 ± 0.047	0.973 ± 0.028	0.968 ± 0.036	**0.993** ± **0.013**	0.916 ± 0.054
2-Switch	**0.982** ± **0.013**	0.869 ± 0.032	0.954 ± 0.015	0.978 ± 0.029	0.931 ± 0.028	0.916 ± 0.045	0.943 ± 0.016	0.977 ± 0.038	0.945 ± 0.042
3-Push	0.961 ± 0.035	0.753 ± 0.031	0.670 ± 0.023	0.931 ± 0.027	0.784 ± 0.041	0.954 ± 0.033	**0.972** ± **0.019**	0.893 ± 0.024	0.929 ± 0.022
3-Switch	**0.907** ± **0.066**	0.879 ± 0.077	0.805 ± 0.089	0.732 ± 0.095	0.295 ± 0.169	0.640 ± 0.105	0.660 ± 0.083	0.831 ± 0.112	0.748 ± 0.089

Table A3: Average success rate over 3 random seeds for Push & Switch environments testing (single task training mode). The numbers in bold highlight the top-performing method.

Experiment Settings	**Method**								
	DAFT (SA)	**DAFT** (AIR)	SMORL	SRICS	GNN	STOVE	COBRA	NCS	LRN
Object Goal	0.916 ± 0.032	0.920 ± 0.037	0.745 ± 0.066	0.784 ± 0.058	0.464 ± 0.091	0.643 ± 0.049	0.715 ± 0.049	**0.925** ± **0.021**	0.846 ± 0.067
Object Interaction	0.909 ± 0.068	0.896 ± 0.053	0.758 ± 0.063	0.824 ± 0.096	0.396 ± 0.146	0.683 ± 0.094	0.746 ± 0.073	**0.931** ± **0.061**	0.812 ± 0.115
Object Comparison	0.917 ± 0.065	0.902 ± 0.060	**0.923** ± **0.049**	0.812 ± 0.071	0.476 ± 0.128	0.625 ± 0.091	0.738 ± 0.057	0.901 ± 0.051	0.693 ± 0.097
Property Comparison	**0.930** ± **0.034**	0.905 ± 0.075	0.918 ± 0.088	0.810 ± 0.095	0.369 ± 0.174	0.602 ± 0.108	0.732 ± 0.083	0.897 ± 0.112	0.644 ± 0.091

Table A4: Average success rate over 3 random seeds for Spriteworld environments training (single task training mode). The numbers in bold highlight the top-performing method.

Experiment Settings	**Method**								
	DAFT (SA)	**DAFT** (AIR)	SMORL	SRICS	GNN	STOVE	COBRA	NCS	LRN
Object Goal	**0.928** ± **0.038**	0.899 ± 0.043	0.720 ± 0.069	0.766 ± 0.064	0.450 ± 0.095	0.672 ± 0.052	0.699 ± 0.055	0.883 ± 0.028	0.791 ± 0.071
Object Interaction	**0.900** ± **0.074**	0.886 ± 0.059	0.735 ± 0.067	0.802 ± 0.100	0.380 ± 0.142	0.665 ± 0.100	0.728 ± 0.077	0.851 ± 0.066	0.789 ± 0.119
Object Comparison	**0.904** ± **0.071**	0.890 ± 0.065	0.876 ± 0.054	0.795 ± 0.076	0.459 ± 0.129	0.674 ± 0.096	0.724 ± 0.061	0.864 ± 0.058	0.800 ± 0.102
Property Comparison	**0.911** ± **0.070**	0.875 ± 0.081	0.865 ± 0.093	0.782 ± 0.099	0.355 ± 0.173	0.680 ± 0.113	0.711 ± 0.087	0.875 ± 0.116	0.772 ± 0.097

Table A5: Average success rate over 3 random seeds for Spriteworld environments testing (with unseen object numbers). The numbers in bold highlight the top-performing method.

Experiment Settings	**Method**								
	`DAFT` (SA)	`DAFT` (AIR)	SMORL	SRICS	GNN	STOVE	COBRA	NCS	LRN
Object Goal	**0.902** ± **0.036**	0.897 ± 0.041	0.694 ± 0.071	0.738 ± 0.062	0.420 ± 0.099	0.654 ± 0.050	0.681 ± 0.053	0.835 ± 0.024	0.775 ± 0.069
Object Interaction	**0.895** ± **0.072**	0.880 ± 0.057	0.710 ± 0.065	0.772 ± 0.102	0.350 ± 0.140	0.635 ± 0.098	0.698 ± 0.075	0.859 ± 0.064	0.769 ± 0.117
Object Comparison	0.878 ± 0.069	**0.895** ± **0.063**	0.854 ± 0.052	0.765 ± 0.074	0.429 ± 0.125	0.631 ± 0.092	0.704 ± 0.059	0.824 ± 0.056	0.724 ± 0.100
Property Comparison	**0.912** ± **0.068**	0.867 ± 0.079	0.885 ± 0.091	0.742 ± 0.097	0.325 ± 0.171	0.660 ± 0.109	0.690 ± 0.085	0.809 ± 0.114	0.725 ± 0.093

Table A6: Average success rate over 3 random seeds for Spriteworld environments testing (with unseen object colors and shapes). The numbers in bold highlight the top-performing method.

D.3.2 Full Ablation Studies

Fig. A3 gives the full ablation studies on Push & Switch and Stacking benchmarks. We consider the following cases:

- DAFT-RL w/o latent parameters;
- DAFT-RL w/o factored class template graph;
- DAFT-RL w/o dynamic interaction graph;
- DAFT-RL w/o factored interaction pattern;
- Using single class learning instead of multiple object classes;
- DAFT-RL w/o action binding graph;
- Using hard attention networks for action binding;

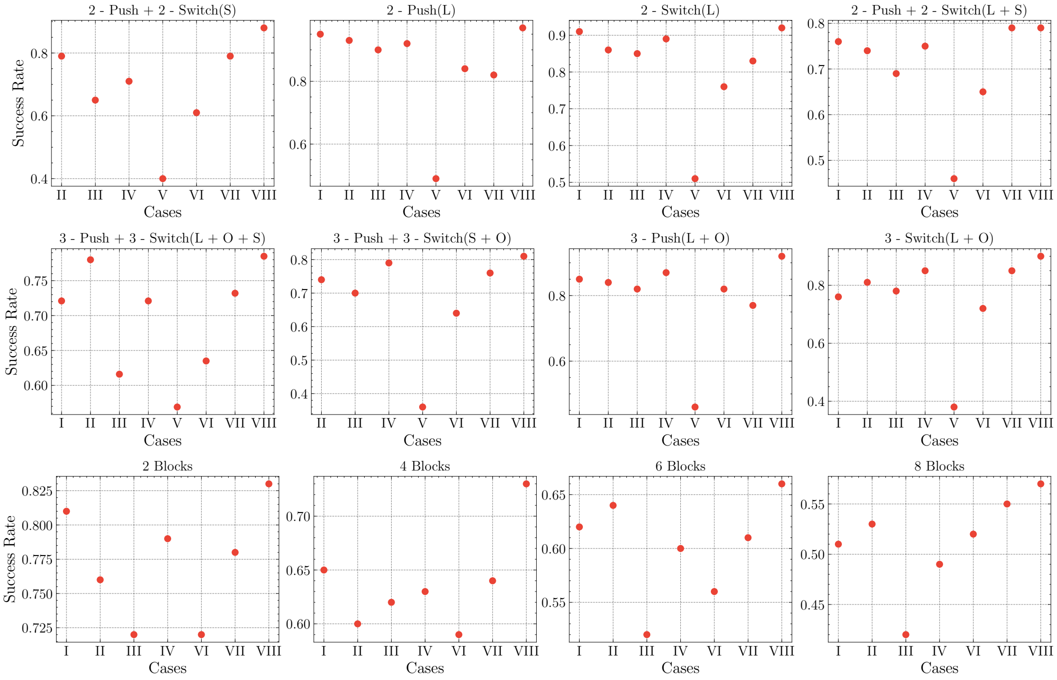

Figure A3: Ablation studies on Push & Switch and stacking tasks. I. without latent parameters; II. without factored class template graph; III. without dynamic interaction graph; IV. without factored interaction pattern; V. using single object class; VI. without action binding graph; VII. using hard attention networks; VIII. Original DAFT-RL. For those tasks where the latent parameters do not exist, we did not conduct ablations on case I. Similarly, for those tasks with only one object, we did not include the ablations on case V.

Based on the obtained results, we observe that each learned component plays a crucial role in optimizing the policy. Notably, the dynamic interaction graph consistently demonstrates a significant contribution across the majority of experiments in Push & Switch and stacking benchmarks.

Task	DAFT (SA)	DAFT (SA) w/o class template graphs
Object goal	0.897	0.767
Object interaction	0.890	0.739
Object comparison	0.893	0.685
Property comparison	0.907	0.832

Table A7: Ablation studies on the Spriteworld benchmark.

For Spriteworld, since there is no object interaction and latent parameters in this task, we only conduct the ablation studies on the class template graph. The results (Table A7) suggest that the class template graph can benefit policy optimization as well.

These findings highlight the importance of factored and dynamic graphs in capturing and modeling the attribute-level interactions between objects. They suggest that understanding and incorporating the dynamic relationships and dependencies among objects have a substantial impact on policy optimization.

D.4 Results on End-to-end Learning

Table A8 presents the results of end-to-end training. For a fair comparison, we use the same number of data samples as in the original multi-step training. In this end-to-end approach, only $\mathcal{D}^{\text{multi}}$ is employed as both the interaction model and object template graphs are trained simultaneously. The results indicate that while the multi-step training consistently outperforms, the agent can still achieve respectable compositional generalization performance with the end-to-end training method.

Experiment Settings	**Method**							
	2-P+2-S (S)	3-P+3-S (S+O)	2-P (L)	2-S (L)	3-P (L+O)	3-S (L+O)	2-P+2-S (L+S)	3-P+3-S (L+O+S)
Original (main paper)	0.881 ± 0.038	0.805 ± 0.024	0.968 ± 0.036	0.923 ± 0.065	0.921 ± 0.037	0.903 ± 0.023	0.793 ± 0.026	0.783 ± 0.025
End-to-end	0.821 ± 0.076	0.765 ± 0.023	0.806 ± 0.048	0.852 ± 0.038	0.875 ± 0.029	0.796 ± 0.047	0.635 ± 0.024	0.707 ± 0.062

Table A8: Results on end-to-end training in Push & Switch compositional task. m-P and n-S denote the m-Push and n-Switch task. The numbers in bold highlight the top-performing method.

D.5 Effect of Imagination

Table A9 gives the results where we disable the imagination for all methods. The results indicate that imagination helps improve the performances of all methods.

Experiment Settings	**Method**							
	DAFT-RL (SA)	DAFT-RL (AIR)	SMORL	SRICS	GNN	STOVE	NCS	LRN
Original	0.506 ± 0.083	0.571 ± 0.039	0.386 ± 0.062	0.420 ± 0.061	0.334 ± 0.047	0.278 ± 0.086	0.397 ± 0.052	0.463 ± 0.077
No imagination	0.469 ± 0.063	0.508 ± 0.064	0.401 ± 0.055	0.434 ± 0.078	0.319 ± 0.040	0.224 ± 0.071	0.264 ± 0.059	0.398 ± 0.061

Table A9: Results on the cases where imagination is disabled. Average success rate over 3 random seeds for Block-stacking with 8 boxes and different masses.

D.6 Effect of the Choice of Object-centric Models for Symbolic Methods

Table A10 shows the results of adding AIR and SA to the symbolic methods on the 8-blocking with variants in terms of masses. Results indicate that both choices are comparable in terms of the final success rate.

D.7 Effects of Learned Latent Parameters and Graphs

To assess the efficacy of the learned latent parameters, we carried out two sets of experiments: (i) incorporating the actual latent parameters (in this case, the mass value) into the state vector for policy learning across all methods; and (ii) integrating the latent parameters learned by DAFT-RL into the state vectors for all baseline methods. The results are presented in Table A11. Notably, while models

Experiment Settings	Method			
	DAFT-RL	SRICS	GNN	LRN
AIR	0.506 ± 0.083	**0.420** ± 0.061	**0.334** ± 0.047	0.463 ± 0.077
SA	**0.571** ± 0.039	0.395 ± 0.046	0.293 ± 0.036	**0.482** ± 0.052

Table A10: Comparisons on the using AIR or SA as the object-centric model for symbolic methods.

using the true latent parameters did outperform those utilizing DAFT-RL's learned latents, the latents derived from DAFT-RL notably enhanced the performance of all baseline methods. These findings suggest the significance of accurate latent parameter estimation and suggest that DAFT-RL can learn a set of valuable latents to improve generalization.

Additionally, we investigated the quality of learned graphs and the latent parameters, and their effect on the RL performance w.r.t. the number of samples for the 3-Push + 3-Switch (L+O+S) task. In particular, we varied the amount of training data, ranging from $10\%, 20\%, 40\%, 60\%, 80\%$ of the original sample size (900 trajectories). We measured the R^2 coefficient of our learned parameters with the true latent parameters, both for a random policy (Fig. A4a) and for a pre-trained policy (Fig. A4b). These results show that the performance degrades with smaller sample sizes, but it is still acceptable with 60% data, and that the difference between data collected with a random or pre-trained policy is negligible at higher sample sizes; the normalized Structural Hamming Distance between the reconstructed graph and the ground truth graph, as well as the success rate of the learned policy. As expected, the more data, the more accurate the graph, and the better the performance of the policy trained with the more accurate world model. For the number of samples in the main paper, the graph is perfectly reconstructed, which also means a good RL performance. Additionally, for limited data (e.g. 0.2 or 0.4 of the original dataset) leveraging a pre-trained policy enhances both graph and policy learning. However, as the amount of data increases, the benefits of using a pre-trained policy diminish.

Experiment Settings	Method							
	DAFT-RL (SA)	DAFT-RL (AIR)	SMORL	SRICS	GNN	STOVE	NCS	LRN
Original	0.506 ±0.083	**0.571 ±0.039**	0.386 ±0.062	0.420 ±0.061	0.334 ±0.047	0.278 ±0.086	0.397 ±0.052	0.463 ±0.077
True latent parameter	0.525 ±0.062	**0.598 ±0.023**	0.475 ±0.081	0.501 ±0.072	0.386 ±0.035	0.315 ±0.049	0.410 ±0.075	0.492 ±0.068
Parameter learned by DAFT-RL	0.506 ±0.083	**0.571 ±0.039**	0.415 ±0.059	0.493 ±0.073	0.359 ±0.041	0.263 ±0.085	0.399 ±0.049	0.484 ±0.061

Table A11: Effects of the latent parameter modeling. Original: Results in the original experiments, where for DAFT-RL, we use the learned latent and the others are without latent parameter estimations; True latent parameter: All methods directly use the true latent as part of the states. Parameters learned by DAFT-RL: all baselines are using the latent learned by DAFT-RL as part of the state vectors. Average success rate over 3 random seeds for Block-stacking with 8 boxes and different masses. The numbers in bold highlight the top-performing method.

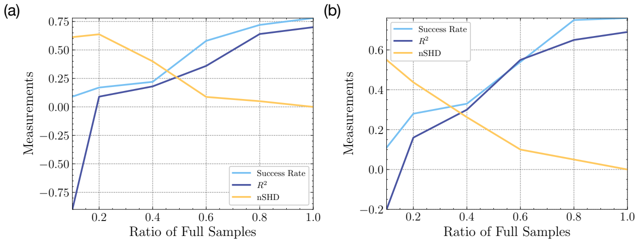

Figure A4: Quality of learned graphs w.r.t. the number of samples for 3-Push + 3-Switch (L+O+S). We plot the success rate of the RL task, the R^2 coefficient for learned representation vs the true latent parameters, and normalized Structured Hamming Distance (nSHD) between the learned graph and true graph under different number of training samples (measured as a ratio with the ones in the main paper) for (a) data collected by random policy, and (b) data collected by pre-trained policy.

D.7.1 Visualization of the Learned Graphs

In Figure A5, which illustrates the learned graphs for boxes and switches in the Push & Switch benchmarks, we denote the class template graphs for boxes and switches as A1 and A2. The variables

$C_1.s_1$, $C_1.s_2$, $C_1.s_3$, and $C_1.s_4$ represent the color, shape, velocity, and position of the boxes, respectively. Similarly, $C_2.s_1$, $C_2.s_2$, $C_2.s_3$, and $C_2.s_4$ correspond to the color, shape, position, and angle of the switches. In both cases, $C_i.\theta$ and $C_j.\theta$ indicate the friction coefficients.

As expected, in the learned class template graph, we observe that shape and color do not have a direct effect on the dynamics, and they remain unchanged throughout the dynamics. On the other hand, velocity directly influences the position of the boxes, while the position itself does not affect the velocity. Regarding the switches, their positions do not directly impact the angles since they are fixed at specific positions on the panel. Also, as expected, actions applied to the switches have an effect solely on their angles. The reward is based on the actions, positions, and angles for both boxes and switches. Additionally, the latent parameters influence the velocity and position of the boxes, while they specifically affect the angle of the switches.

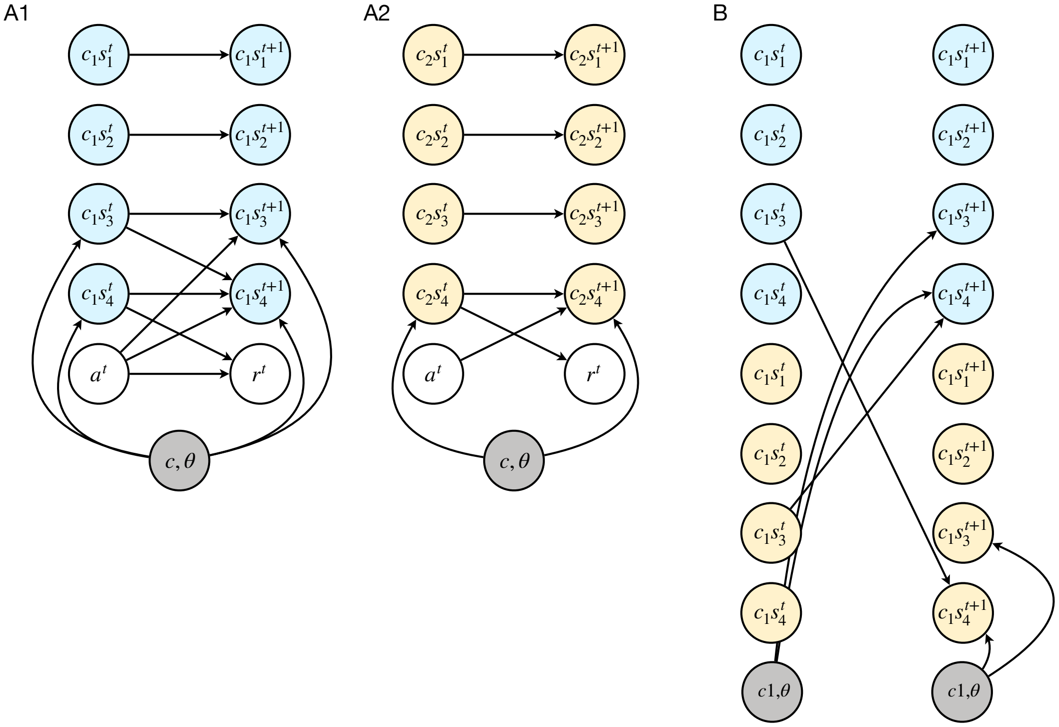

Figure A5: Learned class template graph and interaction pattern graph in push & switch task. A1-A2: Class template graphs for box and switch. B: interaction pattern graph among two boxes.

As for the interaction graph between two boxes, shown in Figure A5B, the velocity of one box can impact the position of the other box, while the other variables do not have a direct influence on each other. This learned interaction graph successfully captures the underlying physical processes of the system, enabling an attribute-level understanding of the object interactions. Overall, these learned graphs effectively recover the underlying physical processes of the system.

References

[1] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning (ICML)*, pages 2555–2565. PMLR, 2019.

[2] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

[3] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vhFu1Acb0xb.

[4] Richard Li, Allan Jabri, Trevor Darrell, and Pulkit Agrawal. Towards practical multi-object manipulation using relational reinforcement learning. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 4051–4058. IEEE, 2020.

[5] Andrii Zadaianchuk, Maximilian Seitzer, and Georg Martius. Self-supervised visual reinforcement learning with object-centric representations. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=xppLmXCbOw1.

[6] Andrii Zadaianchuk, Georg Martius, and Fanny Yang. Self-supervised reinforcement learning with independently controllable subgoals. In *Conference on Robot Learning (CoRL)*, pages 384–394. PMLR, 2022. URL https://openreview.net/forum?id=TEQWRlncJVm.

[7] Davide Mambelli, Frederik Träuble, Stefan Bauer, Bernhard Schölkopf, and Francesco Locatello. Compositional multi-object reinforcement learning with linear relation networks. In *ICLR2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality*, 2022. URL https://openreview.net/forum?id=HFUxPr_I5ec.

[8] Jannik Kossen, Karl Stelzner, Marcel Hussing, Claas Voelcker, and Kristian Kersting. Structured object-aware physics prediction for video modeling and planning. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=B1e-kxSKDH.

[9] Nicholas Watters, Loic Matthey, Matko Bosnjak, Christopher P Burgess, and Alexander Lerchner. Cobra: Data-efficient model-based rl through unsupervised object discovery and curiosity-driven exploration. *arXiv preprint arXiv:1905.09275*, 2019.

[10] Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=mLcmdlEUxy-.

[11] Jongwook Choi, Sungtae Lee, Xinyu Wang, Sungryull Sohn, and Honglak Lee. Unsupervised object interaction learning with counterfactual dynamics models. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*, 2023. URL https://openreview.net/forum?id=dYjH8Nv81K.

[12] Michael Chang, Alyssa Li Dayan, Franziska Meier, Thomas L. Griffiths, Sergey Levine, and Amy Zhang. Hierarchical abstraction for combinatorial generalization in object rearrangement. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=fGG6vHp3W9W.

[13] Jaesik Yoon, Yi-Fu Wu, Heechul Bae, and Sungjin Ahn. An investigation into pre-training object-centric representations for reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 40147–40174. PMLR, 2023.

[14] Linfeng Zhao, Lingzhi Kong, Robin Walters, and Lawson LS Wong. Toward compositional generalization in object-oriented world modeling. In *International Conference on Machine Learning (ICML)*, pages 26841–26864. PMLR, 2022.

[15] Allan Zhou, Vikash Kumar, Chelsea Finn, and Aravind Rajeswaran. Policy architectures for compositional generalization in control. In *ICML Workshop on Spurious Correlations, Invariance and Stability*, 2022.

[16] Ondrej Biza, Thomas Kipf, David Klee, Robert Platt, Jan-Willem van de Meent, and Lawson LS Wong. Factored world models for zero-shot generalization in robotic manipulation. *arXiv preprint arXiv:2202.05333*, 2022.

[17] Akihiro Nakano, Masahiro Suzuki, and Yutaka Matsuo. Interaction-based disentanglement of entities for object-centric world models. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=JQc2VowqCzz`.

[18] Brian Sallans and Geoffrey E Hinton. Reinforcement learning with factored states and actions. *The Journal of Machine Learning Research*, 5:1063–1088, 2004.

[19] Michael Kearns and Daphne Koller. Efficient reinforcement learning in factored mdps. In *Proceedings of the 16th international joint conference on Artificial Intelligence (IJCAI)*, pages 740–747, 1999.

[20] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial intelligence*, 121(1-2):49–107, 2000.

[21] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

[22] Assaf Hallak, François Schnitzler, Timothy Mann, and Shie Mannor. Off-policy model-based learning under unknown factored dynamics. In *International Conference on Machine Learning (ICML)*, pages 711–719. PMLR, 2015.

[23] Sammie Katt, Frans A. Oliehoek, and Christopher Amato. Bayesian reinforcement learning in factored pomdps. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, page 7–15, 2019.

[24] Natalia Gardiol and Leslie Kaelbling. Envelope-based planning in relational mdps. *Advances in Neural Information Processing Systems (NIPS)*, 16, 2003.

[25] Carlos Guestrin, Daphne Koller, Chris Gearhart, and Neal Kanodia. Generalizing plans to new environments in relational mdps. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1003–1010, 2003.

[26] Martijn Van Otterlo. A survey of reinforcement learning in relational domains. *Centre for Telematics and Information Technology (CTIT) University of Twente, Tech. Rep*, 2005.

[27] Carlos Diuk, Andre Cohen, and Michael L Littman. An object-oriented representation for efficient reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 240–247, 2008.

[28] Jonathan Scholz, Martin Levihn, Charles Isbell, and David Wingate. A physics-based model prior for object-oriented mdps. In *International Conference on Machine Learning (ICML)*, pages 1089–1097. PMLR, 2014.

[29] Arthur Wandzel, Yoonseon Oh, Michael Fishman, Nishanth Kumar, Lawson L.S. Wong, and Stefanie Tellex. Multi-object search using object-oriented pomdps. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7194–7200, 2019. doi: 10.1109/ICRA.2019.8793888.

[30] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:11525–11538, 2020.

[31] SM Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Geoffrey E Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. *Advances in neural information processing systems*, 29, 2016.

[32] Biwei Huang, Fan Feng, Chaochao Lu, Sara Magliacane, and Kun Zhang. AdaRL: What, where, and how to adapt in transfer reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=8H5bpVwvt5.

[33] Fan Feng, Biwei Huang, Kun Zhang, and Sara Magliacane. Factored adaptation for non-stationary reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems (NIPS)*, 30, 2017.

[35] Ondrej Biza, Robert Platt, Jan-Willem van de Meent, Lawson L.S. Wong, and Thomas Kipf. Binding actions to objects in world models. In *ICLR2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality*, 2022. URL https://openreview.net/forum?id=HImz8BuUclc.

[36] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning (ICML)*, pages 2688–2697. PMLR, 2018.

[37] Colin Graber and Alexander G Schwing. Dynamic neural relational inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8513–8522, 2020.

[38] Kevin Murphy. Dynamic bayesian networks: Representation, inference and learning. *UC Berkeley, Computer Science Division*, 2002.

[39] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS 2014 Deep Learning and Representation Learning Workshop*, 2014.

[40] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

[41] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

[42] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[43] Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate DALL-e learns to compose. In *International Conference on Learning Representations (ICML)*, 2022. URL https://openreview.net/forum?id=h0OYV0We3oh.

[44] Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=H1gax6VtDB.

[45] Caleb Chuck, Kevin Black, Aditya Arjun, Yuke Zhu, and Scott Niekum. Granger-causal hierarchical skill discovery. *arXiv preprint arXiv:2306.09509*, 2023.

[46] Jiaheng Hu, Zizhao Wang, Peter Stone, and Roberto Martin-Martin. Elden: Exploration via local dependencies. *arXiv preprint arXiv:2310.08702*, 2023.

[47] Rim Assouel, Pau Rodriguez, Perouz Taslakian, David Vazquez, and Yoshua Bengio. Object-centric compositional imagination for visual abstract reasoning. In *ICLR2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality*, 2022.

[48] Jason Hartford, Devon Graham, Kevin Leyton-Brown, and Siamak Ravanbakhsh. Deep models of interactions across sets. In *International Conference on Machine Learning*, pages 1909–1918. PMLR, 2018.

[49] Somjit Nath, Gopeshh Subbaraj, Khimya Khetarpal, and Samira Ebrahimi Kahou. Discovering object-centric generalized value functions from pixels. In *International Conference on Machine Learning (ICML)*, pages 25755–25768. PMLR, 2023.

[50] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[51] John Banister Lanier. *Curiosity-driven multi-criteria hindsight experience replay*. University of California, Irvine, 2019.

[52] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[53] Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. Recent advances in open set recognition: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(10): 3614–3631, 2020.

[54] Phillip Lippe, Sara Magliacane, Sindy Löwe, Yuki M Asano, Taco Cohen, and Efstratios Gavves. BISCUIT: Causal representation learning from binary interactions. In *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 216 of *Proceedings of Machine Learning Research*, pages 1263–1273. PMLR, 2023.

[55] Anurag Ajay, Seungwook Han, Yilun Du, Shaung Li, Abhi Gupta, Tommi Jaakkola, Josh Tenenbaum, Leslie Kaelbling, Akash Srivastava, and Pulkit Agrawal. Compositional foundation models for hierarchical planning. *arXiv preprint arXiv:2309.08587*, 2023.

[56] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 2023.

[57] Yifeng Zhu, Zhenyu Jiang, Peter Stone, and Yuke Zhu. Learning generalizable manipulation policies with object-centric 3d representations. In *7th Annual Conference on Robot Learning (CoRL)*, 2023.

[58] Jiayuan Mao, Tomás Lozano-Pérez, Joshua B Tenenbaum, and Leslie Pack Kaelbling. Learning reusable manipulation strategies. In *7th Annual Conference on Robot Learning (CoRL)*, 2023.

[59] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[60] Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning (CoRL)*, pages 80–93. PMLR, 2023.

[61] Rishi Veerapaneni, John D Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning. In *Conference on Robot Learning (CoRL)*, pages 1439–1456. PMLR, 2020.