Task-Independent Causal State Abstraction

Zizhao Wang[1], **Xuesu Xiao**[1], **Yuke Zhu**[1], **Peter Stone**[1,2]
[1]The University of Texas at Austin, [2]Sony AI
`zizhao.wang@utexas.edu, {xiao, yukez, pstone}@cs.utexas.edu`

Abstract

Learning dynamics models accurately and learning policies sample-efficiently are two important challenges for Model-Based Reinforcement Learning (MBRL). Regarding dynamics accuracy, in contrast to the sparse dynamics exhibited in many real world environments, most MBRL methods learn a dense dynamics model which is vulnerable to spurious correlations and therefore generalizes poorly to unseen states. Meanwhile, existing state abstractions can improve sample efficiency, but their dependence on specific reward functions constrains their applications to limited tasks. In this paper, we introduce a novel state abstraction called *Task-Independent Causal State Abstraction* (TICSA). Exploiting sparsity exhibited in the real world, the proposed method first learns a causal dynamics model that generalizes to unexplored states. A state abstraction can then be derived from the learned dynamics, which not only improves sample efficiency but also applies to many tasks. Using a simulated manipulation environment and two different tasks, we observe that both the dynamics model and policies learned by the proposed method generalize well to unseen states and that TICSA also improves sample efficiency compared to learning without state abstraction.

1 Introduction

The core idea of MBRL is to learn a dynamics model of the environment, which is later used to solve the task via planning [1, 2] or other methods [3–5]. The **accuracy** of the learned dynamics models is the key to the success of MBRL: small inaccuracies get compounded during lookahead search and can therefore lead the policy to completely untrained states during deployment. For higher-dimensional state spaces, even if it is possible to learn an accurate dynamics model, simply increasing model capacity can lead to poor **sample efficiency** and may need impractical amounts of data for training.

For model **accuracy**, most current methods learn *dense* dynamics models which do not generalize well to unexplored parts of the state space [6, 7]. To be specific, when predicting the next step value of each state variable, dense dynamics models take the current values of *all* state variables and action dimensions as input. Learning with this unnecessary large input space makes the dynamics model vulnerable to spurious correlations between unrelated features. In contrast, in many real world problems, the dynamics is *sparse*, meaning that the next step value of each state variable only depends on a small subset of state variables. Therefore, we propose to incorporate such sparsity into the dynamics model where only the relevant state variables are used to model the transition function. With sparsity as the inductive bias, the model can generalize well when seeing out-of(-training)-distribution samples for irrelevant features. In this paper, we loosely call a model with such sparse dependence on necessary features as a **causal** model (see a more formal definition in [8]).

One way to address **sample efficiency** is to learn with state abstraction. State abstraction maps multiple states into a single abstract state, and thus simplifies the policy learning to a smaller state space [9]. However, existing state abstraction approaches, like bisimulation [10–12], depend on the reward function, and thus the application of the learned abstraction is constrained to a limited range of tasks (see details in Sec. A.4). In this paper, we aim at finding a state abstraction that is applicable to many tasks. Our inspiration is that, based on the agent's ability to influence the environment via its actions, we can split state variables into three types: (1) *controllable* state variables that can be affected by some action directly or indirectly; (2) *action-relevant* state variables that cannot be affected by any action (even though they might change on their own) but can influence the action's effects on controllable state variables; and (3) *action-irrelevant* state variables that are neither affected by any action nor will affect any action's consequences. Then for the robot to finish most tasks, it just needs to consider: (1) what are the effects of its action on controllable state variables; and (2) how action-relevant state variables will influence its action results. Therefore, the robot can solve the task with a state abstraction that omits action-irrelevant state variables. Compared to most existing state abstraction approaches which only apply to one single task (where we think of a task as being defined by its reward function), the proposed state abstraction can support learning of *any* task with a reward dependent only on controllable and action-relevant state variables.

Moreover, causal (sparse) dynamics models provide a way to derive the proposed state abstraction, as they directly represent and learn whether each state variable depends on other state variables and the action. To combine the advantages of both methods (better model accuracy and sample-efficiency), we propose a new state abstraction framework called Task-Independent Causal State Abstraction (TICSA) that (1) learns a causal dynamics model among all state variables, (2) divides state variables into three types with the learned causal model, and (3) derives a state abstraction and dynamics model in the abstract space by ignoring action-irrelevant state variables.

We evaluate TICSA in a robot manipulation environment. TICSA learns an accurate dynamics model and generalizes better to out-of(-training)-distribution states than a non-causal model. When applied to downstream tasks, policies with the derived state abstraction learn with higher sample efficiency and again generalize better than those that use a non-abstracted model.

2 Approach

2.1 Problem Definition

We consider a Markov Process with a $d_{\mathcal{S}}$-dimensional $\mathcal{S}$ state space, a $d_{\mathcal{A}}$-dimensional $\mathcal{A}$ action space, and $\mathcal{P}$ as the transition probability (we will slightly abuse $\mathcal{P}$ for other transitions in the rest of the paper). There are $d_{\mathcal{S}}$ state variables, one for each dimension of the state, denoted as $s^{1:d_{\mathcal{S}}}$.

We begin with a formal definition of controllable, action-relevant, and action-irrelevant state variables. Given the causal graphical model ([13], see appendix Sec. A.2 for details) of a Markov Process with $V = \{s_t^{1:d_{\mathcal{S}}}, a_t, s_{t+1}^{1:d_{\mathcal{S}}}\}$ as nodes and E as edges describing causal relationships from $s_t^{1:d_{\mathcal{S}}}/a_t$ to $s_{t+1}^{1:d_{\mathcal{S}}}$, ancestors of a state variable s^i are defined as all nodes that have a directed path leading to node s^i (not necessarily from the immediate previous time step but can be from any previous step). For example, for the causal dynamics model show in Fig. 1 (a), s^4 is an ancestor of s^2 as there is a path of $s_t^4 \to s_{t+1}^3 \to s_{t+2}^2$. Decedents of nodes are defined in the same way but in the opposite direction. Then we have:

Definition (Controllable State Variables $s^{\mathcal{C}}$, Action-Relevant State Variables $s^{\mathcal{R}}$, and Action-Irrelevant State Variables $S^{\mathcal{I}}$) $s^{\mathcal{C}}$ are state variables that are the decedents of the action a_t; $s^{\mathcal{R}}$ are those that are ancestors of controllable state variables, excluding those already belonging to $s^{\mathcal{C}}$; and $S^{\mathcal{I}}$ are those that belong to neither $s^{\mathcal{C}}$ nor $s^{\mathcal{R}}$, where $\mathcal{C}$, $\mathcal{R}$, and $\mathcal{I}$ are the set of state dimension indices for controllable, action-relevant, and action-irrelevant state variables respectively.

Given these definitions, the type of each state variable in the example causal dynamics model is shown in Fig. 1 (b). Further in (c), one may notice that the causal graph can be split into three parts, allowing us to rewrite the transition probabilities as $\mathcal{P}(s_{t+1}|s_t, a_t) = \mathcal{P}(s_{t+1}^{\mathcal{C}}|s_t^{\mathcal{C}}, s_t^{\mathcal{R}}, a_t) \cdot \mathcal{P}(s_{t+1}^{\mathcal{R}}|s_t^{\mathcal{R}}) \cdot \mathcal{P}(s_{t+1}^{\mathcal{I}}|s_t^{\mathcal{R}}, s_t^{\mathcal{I}})$. Then TICSA forms the state abstraction ϕ by omitting action-irrelevant state variables, i.e., $\phi(s_t) = (s_t^{\mathcal{C}}, s_t^{\mathcal{R}})$, and the dynamics in the abstract space can be expressed by removing the subgraph involving action-irrelevant state variables as follows, and it is still a causal dynamics model: $\mathcal{P}(\phi(s_{t+1})|\phi(s_t), a_t) = \mathcal{P}(s_{t+1}^{\mathcal{C}}|s_t^{\mathcal{C}}, s_t^{\mathcal{R}}, a_t) \cdot \mathcal{P}(s_{t+1}^{\mathcal{R}}|s_t^{\mathcal{R}})$. With this state abstraction ϕ, TICSA can be used to solve any actively-accomplishable downstream task (see detailed discussion for tasks that TICSA can solve in Appendix Sec. B.1).

So far, we have defined three types of state variables and TICSA for a known causal dynamics model. However, for real world problem, such a model is usually not accessible. Instead, agents are provided

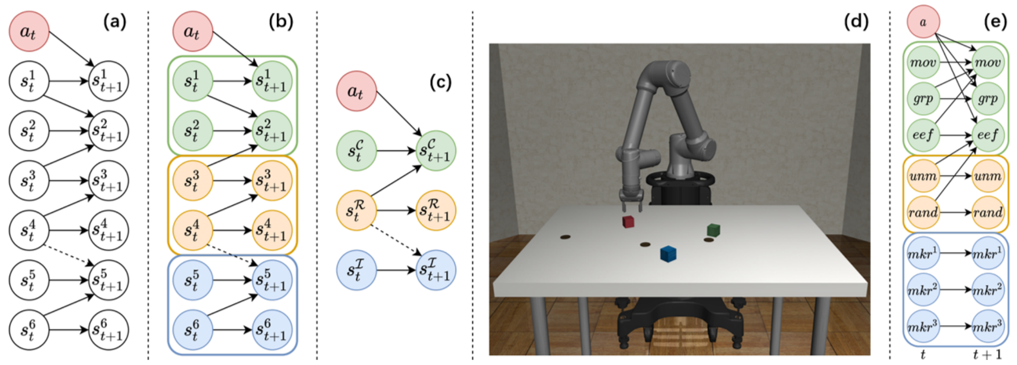
(a)
(b)
(c)
(d)
(e)
a_t
s_t^1 s_t^2 s_t^3 s_t^4 s_t^5 s_t^6
s_{t+1}^1 s_{t+1}^2 s_{t+1}^3 s_{t+1}^4 s_{t+1}^5 s_{t+1}^6
$s_t^{\mathcal{C}}$ $s_t^{\mathcal{R}}$ $s_t^{\mathcal{I}}$
$s_{t+1}^{\mathcal{C}}$ $s_{t+1}^{\mathcal{R}}$ $s_{t+1}^{\mathcal{I}}$
a
mov
grp
eef
unm
rand
mkr^1
mkr^2
mkr^3
t
t+1

Figure 1: **(a)** an example causal dynamics model. **(b)** state variables can be split into three types: controllable (green), action-relevant (orange), and action-irrelevant (blue). The dashed arrow of $s_t^4 \to s_{t+1}^5$ represents that whether this edge exists or not does not affect that s^5 is an action-irrelevant state variable. **(c)** the causal graph can be split into three subgraphs, one for each type of state variable. **(d)** simulated environments. **(e)** the learned causal graph in the object level accurately recovers the relationship between the state variables and the action.

with $\mathcal{S}, \mathcal{A}$, but can only collect transition data via its interactions with the environment. Hence, this paper introduces a novel method that: (1) learns a **causal** dynamics model $F_\theta : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, (2) derives the state abstraction $\phi : \mathcal{S} \to \mathcal{S}^{\mathcal{C}} \times \mathcal{S}^{\mathcal{R}}$ and dynamics F_θ^ϕ in the abstract space, (3) learns a reward predictor for any actively-accomplishable task in the abstract space $R_\varphi : \phi(\mathcal{S}) \times \mathcal{A} \to \mathbb{R}$, and (4) uses planning methods to solve the task with learned F_θ^ϕ and R_φ.

2.2 Causal Dynamics

To learn a causal dynamics model F_θ from the agent's experience, we adapt a differentiable causal inference method [14]. The parameter θ includes (1) $d_{\mathcal{S}}$ neural networks $f_\theta^{1:d_{\mathcal{S}}}$, one for each state variable, to model the conditional probability $\mathcal{P}(s_{t+1}^i | \mathbf{PA}_{s^i})$, where $\mathbf{PA}_{s^i}$ are parents of s^i in the learned causal graph, (2) a binary adjacency matrix $M \in \{0,1\}^{d_{\mathcal{S}} \times d_{\mathcal{S}}}$ and a binary intervention mask $I \in \{0,1\}^{d_{\mathcal{S}} \times d_{\mathcal{A}}}$ which select neural network inputs from all state variables and action dimensions. Here M and I model the causal graph, denoted as $\mathcal{G}_\theta$, of the causal dynamics model. Specifically, if matrix element M_{ij} (or I_{ij}) $= 1$, it means there is an edge from state variables s^j or action dimension a^j to state variable s^i, i.e., s^j (or a^j) is one for the parents for s^i. From the learned causal graph $\mathcal{G}_\theta$, one can derive which state variables are controllable, action-relevant, and action-irrelevant according to their definitions. Overall, the dynamics is modeled as:

$$F_\theta(s_{t+1}|s_t, a_t) = \prod_{i=1}^{d_{\mathcal{S}}} f_\theta^i(s_{t+1}^i | M_i \odot s_t, I_i \odot a_t), \tag{1}$$

where $\odot$ denotes element-wise multiplication. The neural network f_θ^i outputs the probabilistic distribution of s_{t+1}^i which in principle could be any distribution. We experiment with a Gaussian distribution whose mean and variance are computed by the network as two separate heads. Entries in the adjacency matrix M_{ij} and intervention mask I_{ij} are modeled as samples from independent Bernoulli distributions with success probability p_{ij}.

During the training of θ, to align with the sparsity in real world dynamics, our method encourages each state variable to depend on other nodes only if it is necessary (i.e., have as few parents as possible). To this end, TICSA trains θ by maximizing the following regularized H-step log-likelihood:

$$\theta^* = \arg\max_\theta \mathop{\mathbb{E}}_{\mathcal{D}} \log F_\theta(s_{t+1:t+H} | s_t, a_{t:t+H-1}) - \lambda_M \|M\|_1 - \lambda_I \|I\|_1, \tag{2}$$

where $\mathcal{D}$ is transition data collected so far (see Appendix Sec. B.2 for data collection details), λ_M and λ_I are regularization coefficients, and gradients through binary M and I are estimated using the Gumbel estimator [15].

After training, TICSA derives the learned state abstraction $\phi(s) = (\tilde{s}^{\mathcal{C}}, \tilde{s}^{\mathcal{R}})$ from the learned causal graph $\mathcal{G}_\theta = (M, I)$, according to **Definition** above (we denote $\tilde{x}$ as a learned prediction for the ground truth x). The dynamics in the abstract space F_θ^ϕ can also be derived by omitting the prediction networks for action-irrelevant state variables, i.e., $\{f_\theta^i\}^{i \in \mathcal{I}}$.

Table 1: **Left**: Prediction Accuracy for States (log-likelihood), **Right**: Episodic Cumulative Reward on Tasks.

Dynamics Learning			**Task Learning**			
# of OOD Markers	**Causal** (Ours)	**Non-Causal**		TICSA (Ours)	**No-Abstraction**	**Vanilla**
0	10.7 $\pm$ 1.2	**11.0** $\pm$ 0.9	**Reach**	**168.4** $\pm$ 60.7	2.1 $\pm$ 5.8	1.7 $\pm$ 4.7
1	**9.9** $\pm$ 1.3	8.2 $\pm$ 2.2	**Grasp**	47.9 $\pm$ 39.5	34.8 $\pm$ 30.4	**57.7** $\pm$ 69.1
2	**10.1** $\pm$ 1.6	5.1 $\pm$ 4.4	**Reach** (OOD)	**170.9** $\pm$ 55.9	0.9 $\pm$ 2.5	0.9 $\pm$ 2.0
3	**10.2** $\pm$ 1.1	1.3 $\pm$ 6.9	**Grasp** (OOD)	**41.3** $\pm$ 28.1	10.1 $\pm$ 12.1	0.17 $\pm$ 0.44

2.3 Policy Learning for Downstream Tasks

When solving for each actively-accomplishable downstream task, TICSA simultaneously (1) learns a reward predictor with the abstract state, action, and an *optional* goal as input, $R_\varphi : \phi(\mathcal{S}) \times \mathcal{A} \to \mathbb{R}$, and (2) uses a planning algorithm with the learned dynamics and reward predictor for action selection. As the reward predictor is learned in an abstract space rather than the full state space, it is more sample-efficient and less vulnerable to spurious correlations brought about by the excessive information (i.e., action-irrelevant state variables). Meanwhile, planning in the abstract space also reduces the computation cost by relieving the need to roll out action-irrelevant state variables.

The reward predictor is modeled as a neural network and trained by minimizing the prediction error, $\varphi^* = \arg\min_\varphi \mathbb{E}_{(s_t,a_t,r_t)\sim\mathcal{B}}\, \mathcal{L}(R_\varphi(\phi(s_t), a_t), r_t)$, where $\mathcal{B}$ is the task data collected so far, and $\mathcal{L}$ can take any loss function (we experiment with the absolute value of the prediction error). For planning, we use the cross entropy method (CEM [16]), a population-based optimization algorithm, to search for the best action with the learned dynamics and reward predictor, in the same way as [7] (see detailed setup is in Appendix Sec. B.3).

3 Experiments

We evaluate whether TICSA can learn an accurate causal dynamics model and state abstraction. Furthermore, compared to learning with the full state space, we hypothesize that learning with TICSA achieves better sample efficiency and generalization. (more details in Appendix Sec. C)

Setup We design a table-top manipulation environment shown in Fig. 1 (d), with three objects that the robot can interact with: a *movable* red cube that the robot can manipulate (denoted as mov), an *unmovable* green cube that is fixed on the table (unm), and a *randomly moving* blue cube that also cannot be manipulated ($rand$). There are also three non-interactable markers (mkr^1, mkr^2, and mkr^3) on the table, as distractors to introduce potential spurious correlations. The action controls robot's end-effector (*eef*) and gripper (*grp*). Two tasks are **reaching** whose goal is to move the robot's *eef* to a desired location and **grasping** task whose goal is to grasp the movable object.

Baselines For dynamics learning, we compare the generalizability of the learned causal dynamics model to a non-causal dynamics model. The non-causal model has the same architecture and training process as the causal model, except that its prediction for each state variable f_θ^i takes all state variables and action dimensions as input. To generate out-of-distribution (OOD) states, we change the initialization ranges of non-interactable markers from on the table to on the ground. We also vary the number of OOD markers.

For downstream task learning, we compare against: **No-Abstraction** which learns the task with the learned causal dynamics model, but does not use the derived state abstraction for reward prediction (uses all state variables instead). **Vanilla**: like standard MBRL, learning the task with non-causal dynamics model and a reward predictor depending on all state variables.

Results For dynamics learning, the learned causal graph is shown in Fig. 1 (e). On the object level, TICSA learns an accurate causal relationship between the state variables and the action. For dynamics OOD generalizability (Table. 1 Left), compared to the non-causal dynamics model whose prediction performance degrades severely as more markers are at unseen positions, the causal model maintains the same prediction accuracies.

For task learning (Table. 1 Right), in the reaching task with relatively sparse rewards, learning with TICSA achieves the best performance, as training the reward predictor in a smaller state space improves sample efficiency. For policy generalizability, learning with TICSA maintains similar performances, while performances of No-Abstraction and Vanilla drop significantly.

References

[1] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*. Citeseer, 2011, pp. 465–472.

[2] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *arXiv preprint arXiv:1805.12114*, 2018.

[3] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *ACM Sigart Bulletin*, vol. 2, no. 4, pp. 160–163, 1991.

[4] A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine, "Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model," *arXiv preprint arXiv:1907.00953*, 2019.

[5] A. Zhang, R. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," *arXiv preprint arXiv:2006.10742*, 2020.

[6] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *arXiv preprint arXiv:1912.01603*, 2019.

[7] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2555–2565.

[8] J. Peters, P. Bühlmann, and N. Meinshausen, "Causal inference by using invariant prediction: identification and confidence intervals," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, pp. 947–1012, 2016.

[9] D. P. Bertsekas, D. A. Castanon *et al.*, "Adaptive aggregation methods for infinite horizon dynamic programming," 1988.

[10] E. Even-Dar and Y. Mansour, "Approximate equivalence of markov decision processes," in *Learning Theory and Kernel Machines*. Springer, 2003, pp. 581–594.

[11] B. Ravindran, *An algebraic approach to abstraction in reinforcement learning*. University of Massachusetts Amherst, 2004.

[12] L. Li, *A unifying framework for computational reinforcement learning theory*. Rutgers The State University of New Jersey-New Brunswick, 2009.

[13] J. Peters, D. Janzing, and B. Schölkopf, *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[14] P. Brouillard, S. Lachapelle, A. Lacoste, S. Lacoste-Julien, and A. Drouin, "Differentiable causal discovery from interventional data," *arXiv preprint arXiv:2007.01754*, 2020.

[15] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[16] R. Y. Rubinstein, "Optimization of computer simulation models with rare events," *European Journal of Operational Research*, vol. 99, no. 1, pp. 89–112, 1997.

[17] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1714–1721.

[18] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7559–7566.

[19] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," *arXiv preprint arXiv:1802.10592*, 2018.

[20] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," *arXiv preprint arXiv:1906.08253*, 2019.

[21] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine, "Model-based value estimation for efficient model-free reinforcement learning," *arXiv preprint arXiv:1803.00101*, 2018.

[22] B. Amos, S. Stanton, D. Yarats, and A. G. Wilson, "On the model-based stochastic value gradient for continuous reinforcement learning," in *Learning for Dynamics and Control*. PMLR, 2021, pp. 6–20.

[23] X. Fu, G. Yang, P. Agrawal, and T. Jaakkola, "Learning task informed abstractions," in *International Conference on Machine Learning*. PMLR, 2021, pp. 3480–3491.

[24] N. Ferns, P. Panangaden, and D. Precup, "Bisimulation metrics for continuous markov decision processes," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1662–1714, 2011.

[25] N. Ferns and D. Precup, "Bisimulation metrics are optimal value functions." in *UAI*. Citeseer, 2014, pp. 210–219.

[26] A. Zhang, C. Lyle, S. Sodhani, A. Filos, M. Kwiatkowska, J. Pineau, Y. Gal, and D. Precup, "Invariant causal prediction for block mdps," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 214–11 224.

[27] C. Boutilier, T. Dean, and S. Hanks, "Decision-theoretic planning: Structural assumptions and computational leverage," *Journal of Artificial Intelligence Research*, vol. 11, pp. 1–94, 1999.

[28] T. Dietterich, G. Trimponias, and Z. Chen, "Discovering and removing exogenous state variables and rewards for reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1262–1270.

[29] R. Chitnis and T. Lozano-Pérez, "Learning compact models for planning with exogenous processes," in *Conference on Robot Learning*. PMLR, 2020, pp. 813–822.

[30] S. Volodin, N. Wichers, and J. Nixon, "Resolving spurious correlations in causal models of environments via interventions," *arXiv preprint arXiv:2002.05217*, 2020.

[31] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, "robosuite: A modular simulation framework and benchmark for robot learning," in *arXiv preprint arXiv:2009.12293*, 2020.

Appendix A Backgrounds and Related Work

A.1 Markov Process

We consider the agent's interaction with the environment as a Markov Process defined as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P})$, where $\mathcal{S} \subseteq \mathbb{R}^{d_\mathcal{S}}$ is the $d_\mathcal{S}$-dimensional state space, $\mathcal{A} \subseteq \mathbb{R}^{d_\mathcal{A}}$ is the $d_\mathcal{A}$-dimensional action space, $\mathcal{P}$ is the transition probability (we will slightly abuse $\mathcal{P}$ for other transitions in the rest of the paper). The state space consists of $d_\mathcal{S}$ state variables, one for each dimension of the state space, denoted as $\{\mathcal{S}^i\}_{i=1}^{d_\mathcal{S}}$. In this paper, we assume that the transition over each state variable $\mathcal{P}(s_{t+1}^i | s_t, a_t)$ is independent. Formally, for states $s_t, s_{t+1} \in \mathcal{S}$, and action $a_t \in \mathcal{A}$, we can express the state transition probability as

$$\mathcal{P}(s_{t+1} | s_t, a_t) = \prod_i \mathcal{P}(s_{t+1}^i | s_t, a_t). \tag{3}$$

A.2 Causal Graphical Models (CGM)

In the most general form, a causal graphical model [13] is defined by a distribution P_X over d random variables $X^1, \ldots, X^d$ and a Directed Acyclic Graph $\mathcal{G} = (V, E)$. Each node $i \in V = \{1, \ldots, d\}$ is associated with a random variable X^i and each edge $(i \to j) \in E$ represents that X^i is a **direct cause** of X^j, i.e., X^j depends on X^i during the data generation process (note it is different from conditional dependence calculated from observed data). Then the distribution can be specified as

$$P_X(x^1, \ldots, x^d) = \prod_{i=1}^{d} P_{X^i}(x^i | \mathbf{PA}_i), \tag{4}$$

where $\mathbf{PA}_i$ is the set of parents of the node i in the graph $\mathcal{G}$.

For simplicity, in the remainder of the paper, we denote $v^{1:n}$ (or $v_{1:n}$) for an abbreviation of a set of variables $\{v^i\}_{i=1}^n$ (or $\{v_i\}_{i=1}^n$).

When formalizing a Markov process as CGM, the random variables are state variables at the current time step $s_t^{1:d_\mathcal{S}}$, current action a_t, and state variables at the next time step $s_{t+1}^{1:d_\mathcal{S}}$. We further assume all edges in the graph $\mathcal{G}$ are directed from s_t^i / a_t to s_{t+1}^j, i.e., there are no simultaneous or backward causal relationships. Then Eq. (3) can be rewriten as

$$\mathcal{P}(s_{t+1} | s_t, a_t) = \prod_{i=1}^{d_\mathcal{S}} \mathcal{P}(s_{t+1}^i | \mathbf{PA}_{s^i}). \tag{5}$$

Compared to Eq. (3) where s_{t+1}^i depends on all current state variables ($s_t = s_t^{1:d_\mathcal{S}}$) and action, in this re-formalization, s_{t+1}^i only depends on a subset of those variables. This sparse dependence matches the sparsity exhibited in the dynamics of many real world problems and provides an inductive bias for our dynamics model learning.

A.3 Model-based Reinforcement Learning

Model-based RL typically involves learning a dynamics model of the environment (including a reward predictor) by maximizing the likelihood of collected trajectories, then the dynamics model and reward predictor are used for planning [17, 2, 18], providing synthetic data [19, 20], or improving Q-value estimates [21, 22]. However, most existing approaches model dynamics in a dense form of Eq. (3), where each next state variable depends on all current state variables and action. This non-causal formalization suffers from spurious correlations and generalizes poorly for out-of-distribution states. Moreover, the reward predictor typically also has the same dense architecture, thus exacerbating the generalization issue.

To mitigate this issue, some existing approaches learn task-relevant representations via bisimulation metrics and show certain causal property [5, 23]. However, those methods restrict to a causal reward predictor, while the dynamics model is still non-causal and subject to the same generalization issue. Furthermore, their learned state representations and corresponding dynamics models can barely be applied to other tasks (detailed in the next subsection).

A.4 State Abstraction

Aiming at improving sample efficiency, state abstractions map equivalent states into a single abstract state (usually by just keeping state variables relevant to the task) while preserving certain property (e.g., the agent still can learn the optimal policy in the abstract space), thus enabling the agent to learn on a potentially much smaller space. Bisimulation [10–12], as a popular state abstraction, aggregates states that generate equivalent reward sequences given any action sequence. As exact equivalence is not practical, prior work has used approximate variants like bisimulation metrics [24, 25]. Although it may seem to be an orthogonal research direction, bisimulation is closely tied to causality, as the causal graphical model directly models whether certain state variable is relevant to the reward. Previous work [26] shows that state variables kept by the bisimulation abstraction consist of and only consist of the causal ancestors of the reward variable.

Though bisimulation successfully reduces the learning space, as the abstraction is derived from the reward function, it is often limited to a small range of tasks. Specifically, the abstraction learned from bisimulation (metrics) can only generalize to tasks where the new reward function has a subset of the same causal ancestors. For example, in a kitchen work space consisting of a robot, a bowl, and a cup, the learned abstraction from robot picking up the bowl will likely generalize to learning to place the bowl. However, as the cup is irrelevant with the bowl pick-up reward, the learned abstraction excludes cup information and thus cannot learn to pick up the cup. To learn a state abstraction for cup related tasks, the method needs to be trained from scratch with a task related to the cup. For real world problems, we usually want the robot learn to finish a wide range of tasks (e.g., manipulating different objects), and training this bisimulation abstraction for each task (object) is obviously intractable.

In the same spirit with state abstraction, works on *endogenous* state variables (those can be affected by actions [27]) and *exogenous* ones (those cannot be affected) try to reduce state space by removing irrelevant exogenous state variables [28, 29]. However, their methods still depend on the reward and thus have the same issues mentioned earlier.

Instead, the proposed TICSA derives the state abstraction from the learned dynamics model only and does not require a reward function. As a result, its abstraction can be applied to a wide range of tasks.

Appendix B Approach Details

B.1 Solvable Tasks for TICSA

TICSA can be used to solve any actively-accomplishable downstream task. Here, downstream means the task is defined in the same environment (Markov Process) as the dynamics model is learned so that the agent can use the state abstraction ϕ to solve the task via learning from the provided rewards. Meanwhile, actively-accomplishable means that, for the reward function of the task, its inputs only consist of controllable and action-related state variables $(s^{\mathcal{C}}, s^{\mathcal{R}})$. Additionally, if the task is to let certain controllable state variables reach a *varying* goal $g_t \in \underline{\mathcal{S}}^{\mathcal{C}}$, the goal must be provided. Otherwise, it is impossible for the agent to predict the reward accurately without knowing where the goal is. Notice again that actively-accomplishable tasks do not include tasks whose reward functions also take action-irrelevant state variables as input, for example, giving the robot a $+1$ reward for picking up the cup only when it rains outsides and 0 otherwise. However, as the reward function of a task can be arbitrarily designed using any state variable, being able to solve all tasks means that no state variables can be omitted (i.e., no state abstraction). Meanwhile, we maintain that in practice, it is relatively uncommon for a task's reward function to involve action-irrelevant state variables, making TICSA fairly generally applicable in practice.

B.2 Data Collection for Dynamics Learning

The collected transitions $\mathcal{D}$ are important to the quality of the learned causal dynamics [30]. Hence, a desired data collection policy should explore the state-action space as much as possibles to expose and examine all potential dependence (between state variables and the action, and between state variables themselves). In this work, we use a scripted policy to collect data and leave learning a policy for future work.

B.3 Planning Algorithm

For planning, we use the cross entropy method (CEM [16]), a population-based optimization algorithm, to search for the best action with the learned dynamics and reward predictor, in the same way as [7]. For each time step t, CEM initializes a time-dependent diagonal Gaussian belief over optimal action sequence $a_{t:t+L} \sim \mathcal{N}(\mu_{t:t+L}, \sigma^2_{t:t+L})$ where L is the planning length. Starting from zero mean and unit variance, it repeatedly samples J candidate action sequences, evaluates them based on cumulative rewards, and updates $\mu_{t:t+L}, \sigma^2_{t:t+L})$ to the mean and variance of the top K candidates. After N iterations, the planner returns μ_t as the current optimal action.

Appendix C Experiments Details

C.1 Environment

We design a table-top manipulation environment shown in Fig. 1 (d) using the `robosuite` simulation framework [31]. At each episode, the objects and markers are reset to randomly sampled poses on the table. The experiment is conducted on a UR5e robot arm with a Rethink gripper. The state space is 32-dimensional, consisting of robot end-effector (EEF) location ($\mathbb{R}^3$), gripper (grp) joint angles ($\mathbb{R}^2$), poses of the three objects ($3 \times SE(3)$), and locations of the three markers ($3 \times \mathbb{R}^3$). The action space is 4-dimensional, consisting of EEF location displacement ($\mathbb{R}^3$) and the degree to which the gripper is opened ($[0, 1]$). Given this setup, as ground truth, the controllable state variables are the robot EEF location, the gripper joint angles, and the movable object pose; the action-relevant state variables are the poses of the unmovable and the randomly moving objects as they may block arm motion; the action-irrelevant state variables are locations of the markers as they are non-interactable.

C.2 Implementation

Causal Dynamics Model For the prediction network of each state variable $f^i_\theta(s^i_{t+1})$, we use a 3-layer (64, 32, and 32 units) fully connected network with Leaky ReLU activations. For transition data collection, a scripted policy with mixed behaviors (10% random motion, 50% pushing, and 40% picking) is used. During training, a prediction horizon of $H = 5$ is used, and λ_M is gradually increased from 0.01 to 0.15 and λ_I from 0.01 to 0.2 in the first 500K steps. The model is trained for 900K steps, where, at each step, one transition is collected and the model has one gradient update using the data collected so far.

Causal State Abstraction After the training of the causal dynamics model, the learned adjacency matrix M and intervention mask I are used to construct the causal graph $\mathcal{G}_\theta$. For the learned Bernoulli success probability p_{ij} of each entry in M and I, we use a threshold of 0.5 to determine if an edge exists or not. Then the state abstraction and the dynamics in the abstract space are derived according to Definitions 1-3 given $\mathcal{G}_\theta$.

Reward Predictor and Planning The reward predictor R_φ is modeled as a fully connected network with ReLU activations. The number and sizes of layers depend on the task, so they will be introduced later. For CEM planning, for all tasks, we use planning length $L = 10$, $J = 128$ candidates, $K = 32$ top candidates, and $N = 10$ iterations.

C.3 Causal Dynamics

We begin by testing if TICSA can learn an accurate causal dynamics model qualitatively by examining the learned causal graph and quantitatively by evaluating its generalizability with respect to out-of-distribution states.

The learned causal graph is shown in Fig. **??** (Right), where we approximate it to the object level as it is hard to display the full M and I whose size is 32×36 in total[1]. At object level, TICSA learns an accurate causal relationship between the state variables and the action and correctly divides the state variables in the same way as described in Sec. C.1. Specifically, it learns that its actions can control EEF, the gripper, and the movable object. Meanwhile, unmovable and randomly moving objects

[1]The full M and I can be found in this video.

can influence the EEF by blocking its motions. Finally, markers are not interactable so each of them depends on nothing and influences nothing but itself. However, at the state variable level, the learned relationships are not perfectly accurate (see details in the associated video mentioned before).

To test the generalizability of the learned causal dynamics model, we compare it to a non-causal dynamics model. The non-causal model has the same architecture and training process as the causal model, except that its prediction for each state variable f_θ^i takes all state variables and action dimensions as input. To generate out-of-distribution data, we change the initialization ranges of non-interactable markers from on the table to on the ground. We also vary the number of markers. Then the same scripted policy introduced in Sec. C.2 is used to collect 10K transition data points to evaluate each model. Each model makes a 5-step prediction for all other state variables except for the out-of-distribution marker. The performances are measured by the mean and standard deviation of the log-likelihood and are listed in Table. 1 Left. Compared to the causal dynamics model, the non-causal model has more model capacity to overfit the training data, and thus it achieves slightly better prediction when no markers are changed. However, as more and more markers are in different positions than during training, its prediction performance degrades severely, whereas the causal model maintains the same prediction accuracies. This result validates that the causal dynamics model has better out-of-distribution generalization due to its inductive bias of sparsity.

C.4 Downstream Tasks

To validate that the state abstraction from TICSA can be applied to learning various downstream tasks in a sample-efficient and generalizable manner, we evaluate it on two tasks (reaching and grasping) and compare its performance against two baselines.

For the reaching task, the goal is to move the robot's EEF to a desired location g_t^{eef} that is randomly sampled for each episode, and its reward function is formulated as $r_t^{\text{reach}} = 1 - \tanh\left(10 \cdot \|x_t^{\text{eef}} - g_t^{\text{eef}}\|_1\right)$, where x_t^{eef} is the current EEF location. Notice that, though the reward is non-zero everywhere, it is actually pretty sparse (< 0.005 if the L^1 distance is greater than 0.3 m). For this task, the reward predictor uses a 2-layer fully connected network with 64 units each layer.

The goal of the grasping task is to grasp the movable object, and it is defined by the following 2-component reward $r_t^{\text{grasp}} = 0.5 \cdot \left(1 - \tanh\left(5 \cdot \|x_t^{\text{eef}} - x_t^{\text{mov}}\|_1\right)\right) + \mathbb{1}(\text{success})$, where x_t^{mov} is the current location of the movable object, and $\mathbb{1}(\text{success})$ is the indicator function judging whether the grasp is successful. The first component of the reward provides a denser reward (compared to the reaching task) to guide the EEF to explore the area. The reward predictor for this task uses a 3-layer fully connected network with 128 units each layer.

As random exploration is unlikely to finish the tasks and generate learning signals for the reward predictor, for each episode, there is 10% chance to use the same scripted policy as in Sec. C.2 to provide demonstrations for the agent.

For a fair comparison, we use the same architecture and training process for all methods. During training, the dynamics model is frozen and only the reward predictor is updated.

After training each method for 90K steps in the reaching task and 300K steps in the grasping task, we evaluate the trained policies using 50 runs for each task, and their performances (measured by the mean and standard deviation of the cumulative reward in the episode) are shown in Table. 1 Right top two rows. For the reaching task with relatively sparse rewards, learning with TICSA achieves the best performance, as training the reward predictor in a smaller state space reduces complexity and improves sample efficiency. However, for the grasping task, as it has a dense reward to guide exploration, all methods learn to finish it, and Vanilla has the highest cumulative rewards with its largest model capacity.

To test the generalizability of our method, we again change the initialization ranges of all 3 markers in the same way as in Sec. C.3 and test each method on 50 runs. The performances are shown in Table. 1 Right in the bottom two rows. For both tasks, learning with TICSA maintains similar performances as the state abstraction omits markers that are irrelevant to the manipulation dynamics and rewards. In contrast, performances of No-Abstraction and Vanilla drop significantly. Especially for the grasping task, the performance of No-Abstraction drops 71% even though it uses a causal dynamics model, and Vanilla degrades even more ($> 99\%$).