Sequential Causal Imitation Learning with Unobserved Confounders

Daniel Kumor
Purdue University
dkumor@purdue.edu

Junzhe Zhang
Columbia University
junzhez@cs.columbia.edu

Elias Bareinboim
Columbia University
eb@cs.columbia.edu

Abstract

“Monkey see monkey do” is an age-old adage, referring to naïve imitation without a deep understanding of a system’s underlying mechanics. Indeed, if a demonstrator has access to information unavailable to the imitator (monkey), such as a different set of sensors, then no matter how perfectly the imitator models its perceived environment (SEE), attempting to reproduce the demonstrator’s behavior (Do) can lead to poor outcomes. Imitation learning in the presence of a mismatch between demonstrator and imitator has been studied in the literature under the rubric of causal imitation learning (Zhang et al., 2020), but existing solutions are limited to single-stage decision-making. This paper investigates the problem of causal imitation learning in sequential settings, where the imitator must make multiple decisions per episode. We develop a graphical criterion that is necessary and sufficient for determining the feasibility of causal imitation, providing conditions when an imitator can match a demonstrator’s performance despite differing capabilities. Finally, we provide an efficient algorithm for determining imitability and corroborate our theory with simulations.

1 Introduction

Without access to observational data, an agent must learn how to operate at a suitable level of performance through trial and error (Sutton et al., 1998; Mnih et al., 2013). This from-scratch approach is often impractical in environments with the potential of extreme negative - and final - outcomes (driving off a cliff). While both Nature and machine learning researchers have approached the problem from a wide variety of perspectives, a particularly potent method which has been used with great success in many learning machines, including humans, is exploiting observations of other agents in the environment (Rizzolatti & Craighero, 2004; Hussein et al., 2017).

Learning to act by observing other agents offers a data multiplier, allowing agents to take into account others’ experiences. Even when the precise loss function is unknown (what exactly goes into being a good driver?), the agent can attempt to learn from “experts”, namely agents which are known to gain an acceptable reward at the target task. This approach has been studied under the umbrella of imitation learning (Argall et al., 2009; Billard et al., 2008; Hussein et al., 2017; Osa et al., 2018). Several methods have been proposed, including inverse reinforcement learning (Ng et al., 2000; Abbeel & Ng, 2004; Syed & Schapire, 2008; Ziebart et al., 2008) and behavior cloning (Widrow, 1964; Pomerleau, 1989; Muller et al., 2006; Mulling et al., 2013; Muñler & Goldberg, 2017). The former attempts to reconstruct the loss/reward function that the experts minimize and then use it for optimization; the latter directly copies the expert’s actions (behavior cloning).

Despite the power entailed by this approach, it relies on a somewhat stringent condition: the expert and imitator’s sensory capabilities need to be perfectly matched. As an example, self-driving cars rely solely on cameras or lidar, completely ignoring the auditory dimension - and yet most human demonstrators are able to exploit this data, especially in dangerous situations (car horns, screeching...
We start by introducing the notation and definitions used throughout the paper. In particular, we use capital letters for random variables (Z), and small letters for their values (z). Bolded letters represent sets of random variables and their samples (\(Z = \{Z_1, ..., Z_n\}\), \(z = \{z_1 \sim Z_1, ..., z_n \sim Z_n\}\)). \(|Z|\) represents a set’s cardinality. The joint distribution over variables \(Z\) is denoted by \(P(Z)\). To simplify notation, we consistently use the shorthand \(P(z_i)\) to represent probabilities \(P(Z_i = z_i)\).

1.1 Preliminaries

We start by introducing the notation and definitions used throughout the paper. In particular, we use capital letters for random variables (Z), and small letters for their values (z). Bolded letters represent sets of random variables and their samples (\(Z = \{Z_1, ..., Z_n\}\), \(z = \{z_1 \sim Z_1, ..., z_n \sim Z_n\}\)). \(|Z|\) represents a set’s cardinality. The joint distribution over variables \(Z\) is denoted by \(P(Z)\). To simplify notation, we consistently use the shorthand \(P(z_i)\) to represent probabilities \(P(Z_i = z_i)\).
We are interested in learning a policy over a series of actions before \( X \). The basic semantic framework of our analysis rests on structural causal models (SCMs) \cite{Pearl2000}. An SCM \( M \) is a tuple \((U, V, F, P(u))\) with \( U \) the set of endogenous, and \( V \) exogenous variables. \( F \) is a set of structural functions \( s.t. \ \text{for } f_V \in F, V \leftarrow f_V(\text{pa}_V, u_V), \) with \( \text{pa}_V \subseteq V, U \subseteq U \). Values of \( U \) are drawn from an exogenous distribution \( P(u) \), inducing distribution \( P(V) \) over the endogenous \( V \). Since the learner can observe only a subset of endogenous variables, we split \( V \) into \( O \subseteq V \) (observed) and \( L = V \setminus O \) (latent) sets of variables. The marginal \( P(O) \) is thus referred to as the observational distribution.

Each SCM \( M \) is associated with a causal diagram \( G \) where (e.g., see Fig. 2a) solid nodes represent observed variables \( O \), dashed nodes represent latent variables \( L \), and arrows represent the arguments \( \text{pa}(V) \) of each functional relationship \( f_V \). Exogenous variables \( U \) are not explicitly shown; a bi-directed arrow between nodes \( V_i \) and \( V_j \) indicates the presence of an unobserved confounder (UC) affecting both \( V_i \) and \( V_j \). We will use standard conventions to represent graphical relationships such as parents, children, descendants, and ancestors. For example, the set of parent nodes of \( V \) in \( G \) is denoted by \( \text{pa}(V)_G = \cup_{X \in X} \text{pa}(X)_G \). \( \text{ch}, \text{de} \) and \( an \) are similarly defined. Capitalized versions \( P_a, \text{Ch}, \text{De}, \text{An} \) include the argument as well, e.g. \( \text{De}(X)_G = \text{de}(X)_G \cup X \). An observed variable \( V_i \in O \) is an effective parent of \( V_j \in V \) if there is a directed path from \( V_i \) to \( V_j \) in \( G \) such that every internal node on the path is in \( L \). We define \( \text{pa}^+(S) \) as the set of effective parents of variables in \( S \), excluding \( S \) itself, and \( \text{pa}^+(S) \) as \( S \cup \text{pa}^+(S) \). Other relations, like \( \text{ch}^+(S) \) are defined similarly.

A path from a node \( X \) to a node \( Y \) in \( G \) is said to be “active” conditioned on a (possibly empty) set \( W \) if there is a collider at \( A \) along the path \( \rightarrow A \leftrightarrow \) only if \( A \in \text{An}(W) \), and the path does not otherwise contain vertices from \( W \) (d-separation, \cite{Koller2009}). \( X \) and \( Y \) are independent conditioned on \( W \) \((X \perp \perp Y | W)\) if there are no active paths between any \( X \in X \) and \( Y \in Y \). For a subset \( X \subseteq V \), the subgraph obtained from \( G \) with edges outgoing from \( X \) into \( X \) removed is written \( G_X/G_X \) respectively. Finally, we utilize a grouping of observed nodes, called confounded components (c-components).\cite{Tian2002}.

**Definition 1.1.** For a causal diagram \( G \), let \( N \) be a set of unobserved variables in \( L \cup U \). A set \( C \subseteq \text{Ch}(N) \cap O \) is a c-component if for any pair \( U_i, U_j \in N \), there exists a path between \( U_i \) and \( U_j \) in \( G \) such that every observed node \( V_k \in O \) on the path is a collider (i.e., \( V_k \leftrightarrow \)).

C-components correspond to observed variables whose values are affected by related sets of unobserved common causes, such that if \( A, B \in C, (A \perp B | O \setminus \{A, B\}) \). In particular, we focus on maximal c-components \( C \), where there doesn’t exist c-component \( C' \) s.t. \( C \subset C' \). The collection of maximal c-components forms a partition \( C_1, \ldots, C_m \) over observed variables \( O \). For any set \( S \subseteq O \), let \( C(S) \) be the union of c-components \( C_i \) that contain variables in \( S \). For instance, for variable \( Z \) in Fig. 1d, the c-component \( C(\{Z\}) = \{Z, X_1\} \).

### 2 Causal Sequential Imitation Learning

We are interested in learning a policy over a series of actions \( X \subseteq O \) so that an imitator gets average reward \( Y \in V \) identical to that of an expert demonstrator. More specifically, let variables in \( X \) be ordered by \( X_1, \ldots, X_n, n = |X| \). Actions are taken sequentially by the imitator, where only information available at the time of the action can be used to inform a policy for \( X_1 \in X \). To encode the ordering of observations and actions in time, we fix a topological ordering on the variables of \( G \), which we call the “temporal ordering”. We define functions \( \text{before}(X_i) \) and \( \text{after}(X_i) \) to represent nodes that come before/after an action \( X_i \in X \) following the ordering, excluding \( X_i \) itself. A policy \( \pi \) on actions \( X \) is a sequence of decision rules \( \{\pi_1, \ldots, \pi_n\} \) where each \( \pi_i(X_i|Z_i) \) is a function
mapping from domains of covariates $Z_i \subseteq \text{before}(X_i)$ to the domain of action $X_i$. The imitator
following a policy $\pi$ replacing the demonstrator in an environment is encoded by replacing the
expert’s original policy in the SCM $M$ with $\pi$, which gives the results of the imitator’s actions as
$P(V|\text{do}(\pi))$. Our goal is to learn an imitating policy $\pi$ such that the induced distribution $P(Y|\text{do}(\pi))$
perfectly matches the original expert’s performance $P(Y)$. Formally

**Definition 2.1.** (Zhang et al. [2020]) Given a causal diagram $G$, $Y \subseteq V$ is said to be imitable with
respect to actions $X \subseteq O$ in $G$ if there exists $\pi \in \Pi$ uniquely discernible from the observational
distribution $P(O)$ such that for all possible SCMs $M$ compatible with $G$, $P(\{Y|\text{do}(\pi)\})_M = P(\{Y\})_M$.

In other words, the expert’s performance on reward $Y$ is imitable if any set of SCMs must share the
same imitating policy $\pi \in \Pi$ whenever they generate the same causal diagram $G$ and the observational
distribution $P(O)$. Henceforth, we will consistently refer to Def. 2.1 as the fundamental problem of
causal imitation learning. For single stage decision-making problems ($X = \{X\}$), Zhang et al.
(2020) demonstrated imitability for reward $Y$ if and only if there exists a set $Z \subseteq \text{before}(X)$ such that
$(Y \perp X|Z)_{G_Z}$, called the backdoor admissible set, (Pearl 2000 Def. 3.3.1) $(Z = \{F, B, S\})$
in Fig. 1d. It is verifiable that an imitating policy is given by $\pi(X|F, B, S) = P(X|F, B, S)$.

Since the backdoor criterion is complete for the single-stage problem, one may be tempted to surmise
that a version of the criterion generalized to multiple interventions might likewise solve the imitatibility
problem in the general case ($|X| > 1$). Next we show that this is not the case. Let $X_{1:i}$ stand for
a sequence of variables $\{X_1, \ldots, X_i\}$; $X_{1:i} = \emptyset$ if $i < 1$. Pearl & Robins (1995) generalized the
backdoor criterion to the sequential decision-making setting as follows:

**Definition 2.2.** (Pearl & Robins [1995]) Given a causal diagram $G$, a set of action variables $X$, and
target node $Y$, sets $Z_{1} \subseteq \text{before}(X_{1}), \ldots, Z_n \subseteq \text{before}(X_n)$ satisfy the sequential backdoor for
$(G, X, Y)$ if for each $X_i \in X$ such that $(Y \perp X_i|X_{1:i-1}, Z_{1:i})_{G_{X_i}}$.

While the sequential backdoor is an extension of the backdoor to multi-stage decisions, its existence
does not always guarantee the imitatibility of latent reward $Y$. As an example, consider the causal
diagram $G$ described in Fig. 1d. In this case, $Z_1 = \{\}, Z_2 = \{Z\}$, $\{(X_1, Z_1), (X_2, Z_2)\}$ is a
sequential backdoor set for $(G, X, Y)$, but there are distributions for which no agent can
imitate the demonstrator’s performance ($Y$) without knowledge of either the latent $U_i$ or $U_2$. To
witness, suppose that the adversary sets up an SCM with binary variables as follows: $U_1, U_2 \sim Bern(0.5)$, with $X_1 := U_1$, $Z := U_1 \oplus U_2$, $X_2 := Z$ and $Y = \neg(X_1 \oplus X_2 \oplus U_2)$, with $\oplus$ as a
binary XOR. The fact that $U \oplus U = 0$ is exploited to generate a chain where each latent variable
appears exactly twice in $Y$, making $Y = \neg(U_1 \oplus (U_1 \oplus U_2) \oplus U_2)$ = 1. On the other hand, when
imitating, $X_1$ can no longer base its value on $U_1$, making the imitated $\hat{Y} = \neg(\hat{X}_1 \oplus \hat{X}_2 \oplus U_2)$. The
imitator can do no better than $E[\hat{Y}] = 0.5$! We refer readers to Kumor et al. [2021] Proposition C.1)
for a more detailed explanation.

### 2.1 Sequential Backdoor for Causal Imitation

We now introduce the main result of this paper: a generalized backdoor criterion that allows one
to learn imitating policies in the sequential setting. For a sequence of covariate sets $Z_i \subseteq \text{before}(X_i), \ldots, Z_n \subseteq \text{before}(X_n)$, let $G'_i$, $i = 1, \ldots, n$, be the manipulated graph obtained from
a causal diagram $G$ by first (1) removing all arrows coming into nodes in $X_{i+1:n}$; and (2) adding
arrows $Z_{i+1} \rightarrow X_{i+1}, \ldots, Z_n \rightarrow X_n$. We can then define a sequential backdoor criterion for causal
imitation as follows:

**Definition 2.3.** Given a causal diagram $G$, a set of action variables $X$, and target node $Y$, sets
$Z_1 \subseteq \text{before}(X_1), \ldots, Z_n \subseteq \text{before}(X_n)$ satisfy the “sequential $\pi$-backdoor” for $(G, X, Y)$ if at
each $X_i \in X$, either (1) $X_i \perp Y|Z_i$ in $(G'_i, X_i)$ or (2) $X_i \notin \text{An}(Y)$ in $G'_i$.

The first condition of Def. 2.3 is similar to the backdoor criterion where $Z_i$ is a set of variables that
effectively encodes all information relevant to imitating $X_i$ with respect to $Y$. In other words, if the
joint $P(Z_i \cup \{X_i\})$ matches when both expert and imitator are acting, then an adversarial $Y$ cannot
distinguish between the two. The critical modification of the original $\pi$-backdoor for the sequential
setting comes from the causal graph in which this check happens. $G'_i$ can be seen as $G$ with all future

---

1The $\pi$ in “$\pi$-backdoor” is part of the name, and does not refer to any specific policy.
actions of the imitator already encoded in the graph. That is, when performing a check for \( X_i \), it is done with all actions after \( i \) being performed by the imitator rather than expert, with the associated parents of each future \( X_{j>i} \) replaced with their corresponding imitator’s conditioning set. Several examples of \( \mathcal{G}'_i \) are shown in Fig. 3.

The second condition allows for the case where an action at \( X_i \) has no effect on the value of \( Y \) once future actions are taken. Since \( \mathcal{G}'_i \) has modified parents for future \( X_{j>i} \), the value of \( X_i \) might no longer be relevant at all to \( Y \), i.e. \( Y \) would get the same input distribution no matter what policy is chosen for \( X_i \). This allows \( X_i \) to fail condition (1), meaning that it is not imitable by itself, but still be part of an imitable set \( X \), because future actions can shield \( Y \) from errors made at \( X_i \).

Indeed, the sequential \( \pi \)-backdoor criterion can be seen as a recursively applying the single-action \( \pi \)-backdoor. Starting from the last action \( X_k \) in temporal order, one can directly show that \( Y \) is imitable using a backdoor admissible set \( Z_k \) (or \( X_k \) doesn’t affect \( Y \) by any causal path). Replacing \( X_k \) in the SCM with this new imitating policy, the resulting SCM with graph \( \mathcal{G}'_{k-1} \) has an identical distribution over \( Y \) as \( \mathcal{G} \). The procedure can then be repeated for \( X_{k-1} \) using \( \mathcal{G}'_{k-1} \) as the starting graph, and continued recursively, showing imitability for the full set.

**Theorem 2.1.** Given a causal diagram \( \mathcal{G} \), a set of action variables \( X \), and target node \( Y \), if there exist sets \( Z_1, Z_2, ..., Z_k \) that satisfy the sequential \( \pi \)-backdoor criterion with respect to \( (\mathcal{G}, X, Y) \), then \( Y \) is imitable with respect to \( X \) in \( \mathcal{G} \) with policy \( \pi(X_i|Z_i) = P(X_i|Z_i) \) for each \( X_i \in X \).

Thm. 2.1 establishes the sufficiency of the sequential \( \pi \)-backdoor for imitation learning. Consider again the diagram in Fig. 2c. It is verifiable that covariate sets \( Z_1 = \emptyset \), \( Z_2 = \{ Z \} \) are sequential \( \pi \)-backdoor admissible. Thm. 2.1 implies that the imitating policy is given by \( \pi_1(X_1) = P(X_1) \) and \( \pi_2(X_2|Z) = P(X_2|Z) \). Once \( \pi \)-backdoor admissible sets are obtained, the imitating policy can be learned from the observational data through standard density estimation methods for stochastic policies, and supervised learning methods for deterministic policies. This means that the sequential \( \pi \)-backdoor is a method for choosing a set of covariates to use when performing imitation learning, which can be used instead of \( P_{\theta}(X_i) \) for each \( X_i \in X \) in the case when the imitator does not observe certain elements of \( P_{\theta}(X) \). With the covariates chosen using the sequential \( \pi \)-backdoor, one can use domain-specific algorithms for computing an imitating policy based on the observational data.

### 3 Finding Sequential \( \pi \)-Backdoor Admissible Sets

At each \( X_i \), the sequential \( \pi \)-backdoor criterion requires that \( Z_i \) is a back-door adjustment set in the manipulated graph \( \mathcal{G}'_i \). There already exist efficient methods for finding adjustment sets in the
literature ([Tian & Paz, 1998], [van der Zander & Liśkiewicz, 2020]), so if the adjustment were with reference to $G$, one could run these algorithms on each $X_i$, independently to find each backdoor admissible set $Z_i$. However, each action $X_i$ has its $Z_i$ in the manipulated graph $G'_i$, which is dependent on the adjustment used for future actions $X_{i+1:n}$. This means that certain adjustment sets $Z_i$ for $X_{j>i}$ will make there not exist any $Z_i$ for $X_i$ in $G'_i$ that satisfies the criterion! As an example, in Fig. 2c, $X_2$ can use any combination of $Z, X_1, W$ as a valid adjustment set $Z_2$. However, if $Z$ comes after $X_1$ in temporal order, only $Z_2 = \{Z\}$ leads to valid imitation over $X = \{X_1, X_2\}$.

The direct approach towards solving this problem would involve enumerating all possible backdoor admissible sets $Z_i$ for each $X_i$, but there are both exponentially many backdoor admissible sets $Z_i$, and exponentially many combinations of sets over multiple $X_{i:n}$. Such a direct exponential enumeration is not feasible in practical settings. To address these issues, this section will see the development of Alg. [1], which finds a sequential $\pi$-backdoor admissible set $Z_{1:n}$ with regard to actions $X$ in a causal diagram $G$ in polynomial time, if such a set exists.

Before delving into the details of Alg. [1] we describe a method intuitively motivated by the “Nearest Separator” from [Van der Zander et al., 2015] that can generate a backdoor admissible set $Z_i$ for a single independent action $X_i$ in the presence of unobserved variables. While it does not solve the problem of multiple actions due to the issues listed above, it is a building block for Alg. [1].

Consider the Markov Boundary (minimal Markov Blanket, Pearl, 1988) for a set of nodes $O^X \subseteq O$, which is defined as the minimal set $Z \subset O \setminus O^X$ such that $(O^X \perp O \setminus O^X | Z)$. This definition can be applied to graphs with latent variables, where it can be constructed in terms of $c$-components:

**Lemma 3.1.** Given $O^X \subseteq O$, the Markov Boundary of $O^X$ in $G$ is $Pa^+(C(Ch^+(O^X))) \setminus O^X$.

If there is a set $Z \subseteq before(X_i)$ that satisfies the backdoor criterion for $X_i$ with respect to $Y$, then taking $G^Y$ as the ancestral graph of $Y$, the Markov Boundary $Z'$ of $X_i$ in $G^Y_{X_i}$ has $Z' \subseteq before(X_i)$, and also satisfies the backdoor criterion in $G$ (Lem. 3.1). The Markov Boundary can therefore be used to generate a backdoor adjustment set wherever one exists.

A naïve algorithm that uses the Markov Boundary of $X_i \in X$ in $(G'_i)_{X_i}$ as the corresponding $Z_i$, and returns a failure whenever $Z_i \notin before(X_i)$ for the sequential $\pi$-backdoor is equivalent to the existing literature on finding backdoor-admissible sets. It cannot create a valid sequential $\pi$-backdoor for Fig. 2c since $X_2$ would have $Z_2 = \{W\}$, but no adjustment set exists for $X_1$ that d-separates it from $Y$ in the resulting $G'_i$. We must take into account interactions between actions encoded in $G'_i$.

We notice that an $X_i$ does not require a valid adjustment set if it is not an ancestor of $Y$ in $G'_i$ (i.e. $X_i$ does not need to satisfy (1) of Def. 2.3 if it can satisfy (2)). Furthermore, even if $X_i$ is an ancestor of $Y$ in $G'_i$, and therefore must satisfy condition (1) of Def. 2.3, any elements of its $c$-component that are not ancestors of $Y$ in $G'_i$ won’t be part of $(G'_i)^Y$, and therefore don’t need to be conditioned.

It is therefore beneficial for an action $X_i$ to have a backdoor adjustment set that maximizes the number of nodes that are not ancestors of $Y$ in $G'_i$, so that actions $X_{i<j}$ can satisfy (2) of Def. 2.3 if possible, and have the smallest possible $c$-components in $(G'_i)^Y$ (increasing likelihood that backdoor set $Z_i \subseteq before(X_i)$ exists if $X_i$ must satisfy condition (1)).

To demonstrate this intuition, we once again look at Fig. 2c focusing only on action $X_2$. If we were to use $\{W\}$ as $Z_2$, we still have the same set of ancestors of $Y$ in $G'_i$. If we switch to $\{X_1\}$, then $W$ would no longer be an ancestor of $Y$ in $G'_1$ - meaning that $X_1$ is better as a backdoor adjustment set for $X_2$ than $\{W\}$ if we only know that $X_2$ is an action (i.e. $W$ would directly satisfy (2) of Def. 2.3 if it were the other action). Finally, using $\{Z\}$ as $Z_2$ makes both $X_1$ and $W$ no longer ancestors of $Y$ in $G'_i$, meaning that it is the best option for the adjustment set $Z_2$.

FINDOX in Alg. [1] employs the above ideas to iteratively grow a set $O^X \subseteq O$ of ancestors of $X$ (and including $X$) in $G^Y$ whose elements (possibly excluding $X$) will not be ancestors of $Y$ once the actions in their descendants are taken. That is, an element $O_i \in O^X$ where $ch^+(O_i) \subseteq O^X$ is not present in $(G'_i)^Y$ for all actions $X_i$ that come before it in temporal order. Combined with the Markov Boundary, FINDOX can be used to generate sequential $\pi$-backdoors.

We exemplify the use of Alg. [1] through Fig. 2c: $O^X$ represents a map of observed variables which are not ancestors of $Y$ in $G_{i<j}'$ to the earliest action $X_j$ in their descendants. The keys of $O^X$ will be the set $O^X$. Considering the temporal order $\{X_1, Z, W, X_2, Y\}$, the algorithm starts from the last...
Algorithm 1 Find largest valid $O^X$ in ancestral graph of $Y$ given $G$, $X$ and target $Y$

1: function HASVALIDADJUSTMENT($G, O^X, O_i, X_i$) 
2: $C \leftarrow$ the c-component of $O_i$ in $G^Y$
3: $G_C \leftarrow$ the subgraph of $G^Y$ containing only $Pa^+(C)$ and intermediate latent variables
4: $O_C \leftarrow C \setminus (O^X \cup \{O_i\})$ (elements of c-component that might be ancestors of $Y$ in $G'$)
5: return $(O_i \perp \!\!\!\!\perp O^C | (O^C \cap \text{before}(X_i)))$ in $G_C$

6: function FINDOX($G, X, Y$) 
7: $\theta^X \leftarrow$ empty map from elements of $O$ to elements of $X$
8: do
9: for $O_i \in O$ of $G^Y$ (ancestral graph of $Y$) in reverse temporal order do
10: if $|ch^+ (O_i)| > 0$ and $ch^+(O_i) \subseteq \text{keys($\theta^X$)}$ then
11: $X_i \leftarrow$ earliest element of $\theta^X[ch^+(O_i)]$ in temporal order
12: if HASVALIDADJUSTMENT($G, \text{keys($\theta^X$)}, O_i, X_i$) then
13: $\theta^X [O_i] \leftarrow X_i$
14: else if $O_i \in X$ and HASVALIDADJUSTMENT($G, \text{keys($\theta^X$)}, O_i, O_i$) then
15: $\theta^X [O_i] \leftarrow O_i$
16: while $|\theta^X|$ changed in most recent pass
17: return $\text{keys($\theta^X$)}$

Next, $W$ has $X_2$ as its only child, which itself maps to $X_2$ in $\theta^X$. The subgraph of $W$'s c-component and its parents is $\xymatrix{X \ar[r] & W}$, giving $(W \perp \!\!\!\!\perp O|X_1)_{G^X}$, and $\{X_1\} \subseteq \text{before}(X_2)$, allowing us to conclude that there is a backdoor admissible set for $X_2$ where $W$ is no longer an ancestor of $Y$. We set $\theta^X = \{X_2 : X_2, W : X_2\}$, and indeed with $O^X = \{X_2, W\}$, the Markov Boundary of $O^X$ in $G^X$ is $X_1$, and is once again a valid sequential $\pi$-backdoor for the single action $X_2$ (ignoring $X_1$), with policy $\pi(X_2|W) = P(X_2|W)$.

Since $Z$ doesn’t have its children in the keys of $\theta^X$, and is not an element of $X$, it is skipped, leaving only $X_1$. $X_1$’s children ($W$) are in $\theta^X$, we check conditioning using $X_2$ instead of $X_1$ (i.e. we check if $X_1$ can satisfy (2) of Def. 2.3 and not be an ancestor of $Y$ in $G'$). This time, we have $\xymatrix{X \ar[r] & Z}$ as the c-component subgraph, and $Z$ comes before $X_2$, satisfying the check $(X_1 \perp \!\!\!\!\perp Z|Z)$ in HASVALIDADJUSTMENT, resulting in $\theta^X = \{X_2 : X_2, W : X_2, X_1 : X_2\}$, and $O^X = \{X_2, W, X_1\}$. Indeed, the Markov Boundary of $O^X$ in $G^X$ is $\{Z\}$, and we can construct a valid sequential $\pi$-backdoor by using $Z_1 = \{\}$ and $Z_2 = \{Z\}$, where $X_1$ is no longer an ancestor of $Y$ in $G^X$! In this case, we call $X_2$ a “boundary action”, because it is an ancestor of $Y$ in $G'$. On the other hand, $X_1$ is not a boundary action, because it is not an ancestor of $Y$ in $G'$. 

Definition 3.1. The set $X^B \subseteq X$ called the “boundary actions” for $O^X := \text{FINDOX}(G, X, Y)$ are all elements $X_i \in X \cap O^X$ where $ch^+(X_i) \nsubseteq O^X$.

Alg. 1 is general; the set $O^X$ returned by FINDOX can always be used in conjunction with its Markov Boundary to construct a sequential $\pi$-backdoor if one exists: 

Lemma 3.2. Let $O^X := \text{FINDOX}(G, X, Y)$, and $X' := O^X \cap X$. Taking $Z$ as the Markov Boundary of $O^X$ in $G^X$, and $X^B$ as the boundary actions of $O^X$, the sets $Z_i = (Z \cup X^B) \cap \text{before}(X'_i)$ for each $X'_i \in X'$ are a valid sequential $\pi$-backdoor for $(G, X', Y)$. 

7
Lemma 3.3. Let $O^X := \text{FINDOX}(G, X, Y)$. Suppose that there exists a sequential $\pi$-backdoor for $X^* \subseteq X$. Then $X^* \subseteq O^X$.

Combined together, Lems. 3.2 and 3.3 show that FINDOX finds the maximal subset of $X$ where a sequential $\pi$-backdoor exists, and the adjustment sets $Z_{1:n}$ can be constructed using the subset of a Markov Boundary over $O^X$ that comes before each corresponding action $X_i$ (Lem. 3.2). FINDOX is therefore both necessary and sufficient for generating a sequential $\pi$-backdoor:

Theorem 3.1. Let $O^X$ be the output of FINDOX$(G, X, Y)$. A sequential $\pi$-backdoor exists for $(G, X, Y)$ if and only if $X \subseteq O^X$.

4. Necessity of Sequential $\pi$-Backdoor for Imitation

In this section, we show that the sequential $\pi$-backdoor is necessary for imitability, meaning that the sequential $\pi$-backdoor is complete.

A given imitation problem can have multiple possible conditioning sets satisfying the sequential $\pi$-backdoor, and a violation of the criterion for one set does not preclude the existence of another that satisfies the criterion. To avoid this issue, we will use the output of Algorithm FINDOX, which returns a unique set $O^X$ for each problem:

Lemma 4.1. Let $O^X := \text{FINDOX}(G, X, Y)$. Suppose $\exists X_i \in X$ s.t. $X_i \in X \setminus O^X$. Then $X$ is not imitable with respect to $Y$ in $G$.

Our next proposition establishes the necessity of the sequential $\pi$-backdoor criterion for the imitability of the expert’s performance (Def. 2.1), which follows immediately from Lem. 4.1 and Thm. 3.1.

Theorem 4.1. If there do not exist adjustment sets satisfying the sequential $\pi$-backdoor criterion for $(G, X, Y)$, then $X$ is not imitable with respect to $Y$ in $G$.

The proof of Lem. 4.1 relies on the construction of an adversarial SCM for which $Y$ can detect the imitator’s lack of access to the latent variables. For example, in Fig. 2a, $Z$ can carry information about the latent variable $U$ to $Y$, and is only determined after the decision for the value of $X$ is made. Setting $U \sim \text{Bern}(0.5), X := U, Z := U, Y := X \oplus Z$ leaves the imitator with a performance of $E[Y] = 0.5$, while the expert can get perfect performance ($E[Y] = 1$).

Another example with similar mechanics can be seen in Fig. 2c. If the variables are determined in the order $(X_1, W, X_2, Z, Y)$, then the sequence of actions is not imitable, since $Z$ can transfer information about the latent variable $U$ to $Y$, while $X_2$ has no way of gaining information about $U$, because the action at $X$ needed to be taken without context.

Finally, observe Fig. 2d. If $Z$ is determined after $X_1$, the imitator must guess a value for $X_1$ without this side information, which is then combined with $U_2$ at $W$. An adversary can exploit this trick to construct a distribution where guessing wrong can be detected at $Y$ as follows: $U_1 \sim \text{Bern}(0.5), Z, X := U_1, U_2 \sim \text{Bern}(0.5), \text{Bern}(0.5))$ (that is, $U_2$ is a tuple of two binary variables, or a single variable with a uniform domain of $0, 1, 2, 3$). Then setting $W = U_2[Z]$ ([]) represents array access, meaning first element of tuple if $Z = 0$ and second if $Z = 1$, and $X_2 := W, Y := (U_2[X_1] == X_2)$ gives $E[Y] = 1$ only if $\pi_1$ guesses the value of $U_1$, meaning that the imitator can never achieve the expert’s performance. This construction also demonstrates non-imitability when $X_1$ and $Z$ are switched (i.e., Fig. 2c, with $W \leftrightarrow Y$ added, and $X_1$ coming before $Z$ in temporal order).

Due to these results, after running Alg. 1 on the domain’s causal structure, the imitator gets two pieces of information:

1. Is the problem imitable? In other words, is it possible to use only observable context variables, and still get provably optimal imitation, despite the expert and imitator having different information?

2. If so, what context should be included in each action? Including/removing certain observed covariates in an estimation procedure can lead to different conclusions/actions, only one of which is correct (known as “Simpson’s Paradox” in the statistics literature (Pearl 2000)). Furthermore, as demonstrated in Fig. 2c, when performing actions sequentially, some actions might not be imitable themselves ($X_1$ if $Z$ after $X_1$), which leads to bias in observed
descendants (W) - the correct context takes this into account, using only covariates known not to be affected by incorrectly guessed actions.

Finally, the obtained context Zt for every action Xi could be be used as input to existing algorithms for behavioral cloning, giving an imitating policy with an unbiased result.

5 Simulations

We performed 2 experiments (for full details, refer to [Kumor et al., 2021 Appendix B]), comparing the performance of 4 separate approaches to determining which variables to include in an imitating policy:

1. **All Observed (AO)** - Take into account all variables available to the imitator at the time of each action. This is the approach most commonly used in the literature.

2. **Observed Parents (OP)** - The expert used a set of variables to take an action - use the subset of these that are available to the imitator.

3. **π-Backdoor** - In certain cases, each individual action can be imitated independently, so the individual single-action covariate sets are used.

4. **Sequential π-Backdoor (ours)** - The method developed in this paper, which takes into account multiple actions in sequence.

The first simulation consists of running behavioral cloning on randomly sampled distributions consistent with a series of causal graphs designed to showcase aspects of our method. For each causal graph, 10,000 random discrete causal models were sampled, representing the environment as well as expert performance, and then the expert’s policy X was replaced with imitating policies approximating π(Xi) = P(Xi|ctx(Xi)), with context ctx determined by each of the 4 tested methods in turn. Our results are shown in Table 1 with causal graphs shown in the first column, temporal ordering of variables in the second column, and absolute distance between expert and imitator for the 4 methods in the remaining columns.

In the first row, including Z when developing a policy for X leads to a biased answer, which makes the average error of using all observed covariates (red) larger than just the sampling fluctuations present in the other columns. Similarly, Z needs to be taken into account in row 2, but

<table>
<thead>
<tr>
<th>#</th>
<th>Structure</th>
<th>Order</th>
<th>Seq. π-Backdoor</th>
<th>π-Backdoor</th>
<th>Observed Parents</th>
<th>All Observed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>![Graph 1]</td>
<td>z, x1, x2, y</td>
<td>0.04 ± 0.04%</td>
<td>0.04 ± 0.03%</td>
<td>0.05 ± 0.04%</td>
<td>0.13 ± 0.18%</td>
</tr>
<tr>
<td>2</td>
<td>![Graph 2]</td>
<td>z, x1, x2, y</td>
<td>0.05 ± 0.03%</td>
<td>0.05 ± 0.03%</td>
<td><strong>0.20 ± 0.25%</strong></td>
<td>0.05 ± 0.03%</td>
</tr>
<tr>
<td>3</td>
<td>![Graph 3]</td>
<td>x3, z, x2, y</td>
<td>0.04 ± 0.03%</td>
<td><strong>Not Imitable</strong></td>
<td>0.27 ± 0.40%</td>
<td>0.26 ± 0.39%</td>
</tr>
<tr>
<td>4</td>
<td>![Graph 4]</td>
<td>x3, z, x2, y</td>
<td><strong>Not Imitable</strong></td>
<td><strong>Not Imitable</strong></td>
<td>0.19 ± 0.29%</td>
<td>0.19 ± 0.29%</td>
</tr>
</tbody>
</table>

Table 1: Values of |E[Y] − E[Ŷ]| from behavioral cloning using different contexts in randomly sampled models consistent with each causal graph. Cases with incorrect imitation are shown in red.

Figure 4: Results of applying supervised learning techniques to continuous data with different sets of variables as input at each action. OPT is the ground truth expert’s performance, SπBD represents our method, AO is all observed, and OP represents observed parents.
it is not explicitly used by $X$, so a method relying only on observed parents leads to bias here. In the next row, $Z$ is not observed at the time of action $X_1$, making the $\pi$-backdoor incorrectly claim non-imitability. Our method recognizes that $X_2$’s policy can fix the error made at $X_1$, and is the only method that leads to an unbiased result. Finally, in the 4th row, the non-causal approaches have no way to determine non-imitability, and return biased results in all such cases.

The second simulation used a synthetic, adversarial causal model, enriched with continuous data from the HighD dataset (Krajewski et al., 2018) altered to conform to the causal model, to demonstrate that different covariate sets can lead to significantly different imitation performance. A neural network was trained for each action-policy pair using standard supervised learning approaches, leading to the results shown in Fig. 4. The causal structure was not imitable from the single-action setting, so the remaining 3 methods were compared to the optimal reward, showing that our method approaches the performance of the expert, whereas non-causal methods lead to biased results. Full details of model construction, including the full causal graph are given in (Kumor et al., 2021 Appendix B)

6 Limitations & Societal Impact

There are two main limitations to our approach: (1) Our method focuses on the causal diagram, requiring the imitator to provide the causal structure of its environment. This is a fundamental requirement: any agent wishing to operate in environments with latent variables must somehow encode the additional knowledge required to make such inferences from observations. (2) Our criterion only takes into consideration the causal structure, and not the associated data $P(o)$. Data-dependent methods can be computationally intensive, often requiring density estimation. If our approach returns “imitable”, then the resulting policies are guaranteed to give perfect imitation, without needing to process large datasets to determine imitability.

Finally, advances in technology towards improving imitation can easily be transferred to methods used for impersonation - our method provides conditions under which an imposter (imitator) can fool a target ($Y$) into believing they are interacting with a known party (expert). Our method shows when it is provably impossible to detect an impersonation attack. On the other hand, our results can be used to ensure that the causal structure of a domain cannot be imitated, helping mitigate such issues.

7 Conclusion

Great care needs to be taken in choosing which covariates to include when determining a policy for imitating an expert demonstrator when expert and imitator have different views of the world. The wrong set of variables can lead to biased, or even outright incorrect predictions. Our work provides general and complete results for the graphical conditions under which behavioral cloning is possible, and provides an agent with the tools needed to determine the variables relevant to its policy.

Funding Transparency

The team is supported in part by funding from the NSF, Amazon, JP Morgan, and The Alfred P. Sloan Foundation, as well as grants from NSF IIS-1704352 and IIS-1750807 (CAREER).

References


A The Sequential $\pi$-Backdoor

This section contains proofs of the core theorems mentioned in the text. All results shown here are in reference to the sequential $\pi$-backdoor:

**Definition 2.3.** Given a causal diagram $G$, a set of action variables $X$, and target node $Y$, sets $Z_1 \subseteq \text{before}(X_1), \ldots, Z_n \subseteq \text{before}(X_n)$ satisfy the "sequential $\pi$-backdoor" for $(G, X, Y)$ if at each $X_i \in X$, either (1) $(X_i \perp \!\!\!\perp Y | Z_i)$ in $(G_i')_{X_i}$, or (2) $X_i \notin \text{An}(Y)$ in $G_i'$.

The first portion shows the proof of sufficiency of Def. 2.3 for imitation, the second proves that Alg. 1 finds valid sequential $\pi$-backdoors, and the third portion provides the construction of a counterexample for imitation whenever the algorithm fails to find a valid sequential $\pi$-backdoor, which then shows necessity.

### A.1 Proof of Sufficiency

The proof of sufficiency will make heavy use of sufficiency of Def. 2.3 for imitation, the second proves that Alg. 1 finds valid sequential $\pi$-backdoors, and the third portion provides the construction of a counterexample for imitation whenever the algorithm fails to find a valid sequential $\pi$-backdoor, which then shows necessity.

**Theorem A.1.** (Zhang et al., 2020) $Y$ is imitable w.r.t. $X$ in $G$ if there is $Z \subseteq \text{before}(X)$ such that $(X \perp \!\!\!\perp Y | Z)$ in $G_{X, Y}$. Moreover, the imitating policy is given by $\pi(Z) = P(X | Z)$.

We will show that the sequential version can be proved with a recursive application of Thm. A.1.

**Theorem 2.1.** Given a causal diagram $G$, a set of action variables $X$, and target node $Y$, if there exist sets $Z_1, Z_2, \ldots, Z_k$ that satisfy the sequential $\pi$-backdoor criterion with respect to $(G, X, Y)$, then $Y$ is imitable with respect to $X$ in $G$ with policy $\pi(X_i | Z_i) = P(X_i | Z_i)$ for each $X_i \in X$.

**Proof.** We will proceed by induction on the $X$ in reverse temporal order. In the base case we have $G_k'$ with distribution over $Y$ identical to the distribution over $Y$ in $G$ ($G_k' = G$ so the distribution of $Y$ when no action is taken by the imitator is identical to the original distribution).

In the inductive step, we have graph $G_i'$, which explicitly encodes actions of $X_{i+1}, \ldots, X_k$. We know that the distribution over $Y$ is identical in both $G$ and $G_i$.

If condition (2) is satisfied for $X_i$ in $G_i$, then $X_i$ is not an ancestor of $Y$, and therefore $P(Y | \text{do}(X_i)) = P(Y)$ for any action $X_i$. We can therefore take any action from $X_i$ without affecting the distribution of $Y$, and therefore the distribution is identical for $Y$ in $G$ and $G_{i-1}$ (which is the graph explicitly encoding whichever policy was chosen for $X_i$).

If condition (2) is not satisfied, condition (1) must be satisfied in $G_i$, and by Thm. A.1 the policy $\pi_{X_i}(Z_i) = P(X_i | Z_i)$ perfectly imitates $Y$ in $G_i$, meaning that once the policy is explicitly encoded to construct $G_{i-1}$ we have $P(Y|G_{i-1}) = P(Y|G_i) = P(Y|G)$ where $P(Y|G)$ is the probability of $Y$ assuming the given policy is followed for $X_{i+1}, \ldots, X_k$. We therefore have $G_{i-1}$ which encodes the causal graph of the imitator acting on $X_i, \ldots, X_k$ has an identical distribution over $Y$ if the given policies are taken, proving the inductive hypothesis.

Finally, once all actions are taken according to the described policies, the distribution over $Y$ remains identical to the distribution over $Y$ in the original mode with all actions taken by expert, meaning that $Y$ is imitable with respect to $X$ in $G$. $\square$

### A.2 Algorithm Proofs

**Lemma A.1.** Suppose that there exists a sequential $\pi$-backdoor for $X$. Then $X \subseteq \text{FINDOX}(G, X, Y)$.

**Proof.** We first observe that keys can only be added to $\Theta^X$ in Alg. 1. Furthermore, while the algorithm runs a loop in reverse temporal order, the outer "do" runs until there is no change in $\Theta^X$ for a pass through all of the variables.

We can therefore do a proof by induction over $X$ in reverse temporal order. In particular, on each successive pass of the outer loop, we focus on the element before it, $X_i \in X$ and its ancestors - since

\footnote{The $\pi$ in “$\pi$-backdoor” is part of the name, and does not refer to any specific policy.}
adding elements to $\mathcal{O}^X$ can only help in the conditioning checks, the focus on reverse temporal order is, in a sense, a worst-case scenario of the algorithm being artificially limited.

We will show that after the $i$th successive outer-loop, all elements of $X_{k-i+1}, \ldots, X_k$ will be in the set $\text{keys}(\mathcal{O}^X)$, as well as all elements that are not ancestors of $Y$ in $G'_{k-1}$ of the ancestral graph of $Y$.

In the first outer loop, we have $X_k \in X$ that is last in temporal order. When performing the check for $X_k$, a conditioning check is performed ensuring that all values of the c-component of $X$.

Suppose for contradiction that $V_j$ is not an ancestor of $Y$ in $G'_{k-1}$ or is $X_k$, and yet $V_i \notin \text{keys}(\mathcal{O}^X)$. Since the check is performed in the ancestral graph of $Y$, the only way for this to be true is if all of its directed paths to $Y$ pass through $X_k$, and as such are cut when the parents of $X_k$ are replaced with the conditioning set $Z_k$. We therefore conclude that $V_i$ and has at least one directed path (possibly of size 0 if $V_i = X_k$) to $X_k$ in $G'_k$, and no element along any path from $V_i$ to $X_k$ is in $Z_k$ (otherwise it would be an ancestor of $Y$ in $G'_{k-1}$). For $V_i$ to not be in $\text{keys}(\mathcal{O}^X)$, the algorithm’s conditioning check must have failed somewhere along one of the directed paths to $X_k$. Call the node of failure $V_j$.

The conditioning check ensures that all elements in the c-component of $V_j$ reachable without using colliders at elements in $\mathcal{O}^X$ are in before$(X_i)$. Since the keys of $\mathcal{O}^X$ represent non-ancestors of $Y$, the check failing means that there is a path to an element of the c-component that is in before$(X_k)$. Each collider that the path passes is either an ancestor of $Y$, and therefore either it, or one of its descendants must be in $Z_k$, or was not yet added to $\text{keys}(\mathcal{O}^X)$, but it itself is not an ancestor of $Y$, in which case the same proof can be repeated using that element as $V_j$, since it would also fail the conditioning check. Since any conditioning set will have a path to an element before $X_k$, which itself is an ancestor of $Y$ in $G'_{k-1}$, it and its descendants can’t be part of a conditioning set $Z_k$, meaning that there was a contradiction ($Z_k$ was a d-separating set - since $X_k$ was the last element in temporal order, and we are in the ancestral graph of $Y$, it must satisfy condition (1).

Now suppose we are on the $i$th loop. After the first $i-1$ loops, all elements $X_{k-i+1}, \ldots, X_k$ will be in the set $\text{keys}(\mathcal{O}^X)$, and all elements that are not ancestors of $Y$ in $G'_{k-i}$ are also in $\text{keys}(\mathcal{O}^X)$. If $X_{k-i}$ satisfies condition (2) of the sequential $\pi$-backdoor, then it is already in $\text{keys}(\mathcal{O}^X)$, and any policy chosen will not have any effect on the elements that are ancestors of $Y$.

However, if it satisfies condition (1), we proceed in the same way as before (we show a shortened repeat here). Suppose for contradiction that $V_i$ is not an ancestor of $Y$ in $G'_{k-i-1}$, but is not in $\text{keys}(\mathcal{O}^X)$ after the $i$th loop. Since all non-ancestors of $Y$ that don’t have paths through $X_i$ were already in $\text{keys}(\mathcal{O}^X)$ after the $i-1$st loop, $V_i$ must have a directed path to $X_i$. For the algorithm to not include the element, the conditioning check must have failed at some node on one of the paths from $V_i$ to $X_i$. Call this element $V_j$. Once again, the conditioning $Z_k$ that satisfies the sequential $\pi$-backdoor must block all elements of the c-component of $V_j$ that have directed paths to $Y$ which are reachable without conditioning on non-ancestors, which is violated if any such element comes before $X_i$. This is a contradiction, since the conditioning set satisfies the sequential $\pi$-backdoor.

We have therefore shown that after repeating the proof inductively for all variables, using $k$ passes of the algorithm through the nodes, we have $X \subseteq \text{FINDOX}(\mathcal{G}, X, Y)$.}

The next theorem and proof will use the following definition of boundary nodes:

**Definition 3.1.** The set $X' \subseteq X$ called the “boundary actions” for $O^X := \text{FINDOX}(\mathcal{G}, X, Y)$ are all elements $X_i \in X \cap O^X$ where $ch^+(X_i) \not\subseteq O^X$.

**Lemma 3.2.** Let $O^X := \text{FINDOX}(\mathcal{G}, X, Y)$, and $X' := O^X \cap X$. Taking $Z$ as the Markov Boundary of $O^X$ in $G^Y_X$, and $X$ as the boundary actions of $O^X$, the sets $Z_i := (Z \cup X') \cap \text{before}(X'_i)$ for each $X'_i \in X'$ are a valid sequential $\pi$-backdoor for $(\mathcal{G}, X', Y)$.

**Proof.** We will perform a proof by induction over the algorithm’s $\mathcal{O}^X$ map. Suppose that the algorithm is given $\mathcal{G}, X, Y$ - we show that at each step of the algorithm, returning $\mathcal{O}^X$ would satisfy the theorem.

In the base case $\mathcal{O}^X$ is empty, making $O^X' = \emptyset$, so $X' = X \cap O^X' = \emptyset$. There is therefore a valid sequential $\pi$-backdoor of size 0.

Next, suppose that $\mathcal{O}^X$ can be used to construct a valid sequential $\pi$-backdoor. Suppose we are now checking a node $V_i \in V$ for inclusion in $\mathcal{O}^X$, following Alg. [1].

14
If the check fails, no node is added, and the theorem remains true. We therefore focus on the situations where an element is added to \( O^X \).

Suppose \( ch(V_i) \subseteq \text{keys}(O^X) \), and the algorithm chooses the earliest element of \( O^X[ch(V_i)] \) in temporal order, or the element is a boundary action of the current \( V^X \). For \( V_i \) to be added to the keys of \( O^X \), we will show that the resulting conditioning set corresponds to a valid sequential \( \pi \)-backdoor. If the added element was in \( X \), but is not a boundary action, then we know it satisfies condition (2), since by the given construction, all directed paths from \( V_i \) to \( Y \) pass through elements of \( X \), which in turn don’t have any descendants of \( V_i \) in their conditioning sets. Furthermore, all previously added elements of \( \text{keys}(O^X) \cap X \) that have all children in \( \text{keys}(O^X) \) satisfy condition (2), since no elements were removed from \( \text{keys}(O^X) \). We therefore only need to prove that the boundary actions satisfy (1).

We will prove this by contradiction. Suppose that \( V_i \) is added to \( O^X \), but after adding \( V_i \), there is a boundary action \( X_j \) which has \( (X_j \perp \! \! \! \perp Y|Z_j) \) before \( V_i \) was added, but has \( (X_j \not\perp \! \! \! \perp Y|Z_j) \) after it was added. Using Lem. [A.1] we know that the only difference in \( Z_j \) before and after \( V_i \) is added to \( \text{keys}(O^X) \) is that \( V_i \) is removed from the conditioning set, and elements of its c-component and its parents that are not in \( \text{keys}(O^X) \) added. Because \( X_j \) and \( Y \) are no longer independent, there exists a path from \( X_j \) to \( Y \) in \( G'_i \) with colliders at elements of \( Z_j \). Since the entire conditioning set remains identical as before, and was a valid markov boundary, adding the elements of \( V_i \)'s c-component cannot open any paths to \( Y \). However, the removal of \( V_i \) from the conditioning set can open paths. We therefore know that the path from \( X_j \) to \( Y \) must pass through \( V_i \). Suppose \( P \) is this path. There are two possibilities for this path:

1. The path comes to \( V_i \), and continues into the descendants to \( V_i \) (\( V_i \to \ldots \)). All paths into descendants pass to either elements of \( X \) that come after \( X_j \), in which case they are not ancestors of \( Y \) in \( G'_i \), and \( Z_j \) does not condition on them, meaning that the path can’t get to \( Y \). On the other hand, if the path passes to elements of \( X \cap \text{before}(X_j) \), it has a collider at that element (call it \( X_o \)), and then continues back up. However, if the path continues back passing \( V_i \) again, we know that it must be blocked, since in the original conditioning set, we can modify the path to simply use \( V_i \) as a collider, showing a contradiction. On the other hand, if the path does not pass back up to \( V_i \)'s ancestors, either \( (X_j \not\perp \! \! \! \perp Y|Z_j) \), in which case we can repeat this proof using \( X_j \) instead of \( X_j \) as the starting node (since \( Z_l \subseteq Z_j \)), or the path uses an element \( X_o \in \text{after}(X_i) \) but \( X_o \in \text{before}(X_j) \) as a collider. In that case, the same argument can be repeated: either \( (X_o \not\perp \! \! \! \perp Y|Z_o) \), in which case the graph can be restarted with \( X_o \) or the path continues to another such element. Since there are finite elements of \( X \), at some point we have \( (X_o \not\perp \! \! \! \perp Y|Z_o) \), and can restart the proof from there.

2. The path comes to \( V_i \) from the descendants (\( \leftarrow V_i \leftarrow \)) or \( V_i \) is the starting node. The path must therefore either start at a descendant to \( V_i \) (\( X_i \)) or have a collider at \( X_i \). In both cases, since \( V_i \) passed the conditioning check that includes all such possible descendants, the set \( Z_j \) contains all the elements of the c-component and their parents that are not in \( \text{keys}(O^X) \).

We can therefore repeat the argument made in possibility 1 for the continuation of the path, showing that there must exist another element \( X_o \) where \( (X_o \not\perp \! \! \! \perp Y|Z_o) \) along this path, from which the proof can be restarted.

Since both cases recursively shorten \( P \) on each restart of the proof, there will be an \( X_o \) at which one of the requirements will be violated, showing a contradiction.

**Theorem 3.1.** Let \( O^X \) be the output of \( \text{FINDOX}(G, X, Y) \). A sequential \( \pi \)-backdoor exists for \( (G, X, Y) \) if and only if \( X \subseteq O^X \).

**Proof.** Lem. [A.1] shows that \( X \subseteq O^X \), if a \( \pi \)-backdoor exists, and ?? shows that we can construct a valid \( \pi \)-backdoor whenever \( X \subseteq O^X \), which shows that the algorithm returns an answer if and only if a \( \pi \)-backdoor exists.

**Lemma A.2.** \( O^X := \text{FINDOX}(G, X, Y) \), and let \( O^X' := \text{FINDOX}(G, X', Y) \), with \( X' \subseteq X \). Then \( O^X' \subseteq O^X \).
Proof. We first observe that keys can only be added to $O^X$ in Alg. 1. Furthermore, we observe that the conditioning checks can only be helped by adding values of $X$ to consider in the algorithm (i.e. adding an element to $O^X$ cannot cause a conditioning check to fail that would otherwise succeed, but it can cause a conditioning check to succeed where it would have failed). Therefore, all of the conditioning checks that succeeded when using $X$ will also succeed when using $X'$, and so $\text{keys}(O^X) \subseteq \text{keys}(O^X)$.

Lemma 3.3. Let $O^X := \text{findOX}(G, X, Y)$. Suppose that there exists a sequential $\pi$-backdoor for $X^n \subseteq X$. Then $X^n \subseteq O^X$.

Proof. Suppose not, that is, suppose that a sequential $\pi$-backdoor exists for $X^n$, but $X^n \not\subseteq V^X$. However, using Lem. A.2 it also means that $V^{X^*} := \text{findOX}(G, X^n, Y)$ does not contain $X^n$, and so using Thm. 3.1. no sequential $\pi$-backdoor exists for $X^n$, a contradiction.

A.3 Proof of Necessity

Definition A.1. A $O^X$-adversarial directed path is a directed path from a node $V_i$ to $Y$ in the ancestral graph of $Y$, such that given $O^X$ from $\text{findOX}(G, X, Y)$, is constructed iteratively starting from $V_i$ as follows:

- If $\text{ch}(V_i) \subseteq \text{keys}(O^X)$, choose the one that maps to the earliest element of $X$ in $O^X$ ($V_{i+1} \in \{V_j | V_j \in \text{ch}(V_i), O^X[\text{ch}(V_i)] \subseteq \text{after}(O^X[V_j])\}$)
- If $V_c := \text{ch}(V_i) \setminus \text{keys}(O^X)$ is non-empty, choose any $V_{i+1} \in V_c$ as next element in path.
- If the resulting path intersects with another, the path continues with the other to $Y$.

Definition A.2. A $O^X$-adversarial directed path set is a set of paths starting at a set of nodes $V^P$ to $Y$, where each path carries a tuple of values, and two paths merging concatenates the tuples of both paths, such that $Y$ obtains a tuple containing all of the values of nodes at $V^P$.

Definition A.3. A $O^X$-adversarial latent path is a path starting at $V_i \in V^Y$ and ending at $V_j \in V^Y$, of the form $V_i \leftarrow \ldots \rightarrow C_1 \leftarrow \ldots \rightarrow V_j$, with colliders $C_1, C_2, \ldots, C_k \in C \subseteq V^Y$ and:

1. $\forall C_i \in C \cup \{V_i, V_j\}, C_i \notin \text{keys}(O^X)$
2. All elements along the path except $C, V_i, V_j$ are latent
3. $V_j$ comes after $V_i$ and all elements of $X$ where $\{X_i \in X | \exists C_i \in C \cup \{V_i\}, \text{ch}(C_i) \subseteq \text{keys}(O^X), X_i \in O^X[C_i], X_i \in \text{before}(X_j) \forall X_j \in O^X[C_i]\}$
4. $|\text{ch}(V_j) \setminus \text{keys}(O^X)| > 0$

Lemma A.3. Given $O^X$, and $V_i \in X$ or $\text{ch}(V_i) \subseteq \text{keys}(O^X)$, but $V_i \notin \text{keys}(O^X)$, then there exists a $O^X$-adversarial latent path from $V_i$ to a node $V_j$, crossing colliders $A_1, \ldots, A_k$.

Proof. Given the c-component $C$ of $V_i$ (including $V_i$), every $C_i \in C$ which is not in $\text{keys}(O^X)$, but is either in $X$ or has all its children in $\text{keys}(O^X)$ had $(V_i \perp \perp C \setminus (O^X \cup \{V_i\})) \cap (O^X \cup \{V_i\}) \subseteq \text{after}(C_i)$ in $G_C$ evaluated in $\text{findOX}(G, X, Y)$ on the last iteration, and failed the check, without any changes to $O^X$ in the iteration. This means that each $C_i$ has an element of the c-component, $C_j \in \text{after}(C_i)$ (otherwise it would be part of the conditioning in the check) such that there is a path in the c-component’s subgraph to it crossing only colliders $A_1, \ldots, A_k \in C$ not in $O^X$.

Let $P$ be the path corresponding to $V_i$. Conditions 1,2 hold by definition. However, this path does not necessarily satisfy conditions 3,4. Let $A_1$ be the first element along the path from $V_i$ which violates condition 3, meaning that $V_i$ comes before $A_i$. Since $A_i$ must be either in $X$ or has all its children in $\text{keys}(O^X)$ by the requirements of condition 3’s constraint, it has a path to a different end element, $V_{a_1}$ which comes after $A_i$. We can therefore concatenate $P$ with the new path, removing all loops and intersections, such that we are left with a new path $P'$ ending at $V_{a_1}$. The elements of the path before $A_1$ satisfied condition 3, so they must satisfy the condition with an end element coming even later in temporal order. We can repeat this procedure until we have a path from $V_i$ where none of the colliders come after the last element $V_{a_1}$, meaning that condition 3 is also satisfied.
Finally, suppose \( V_j \) has all of its children in \( \text{keys}(\mathcal{O}^X) \), violating condition 4. This is once again sufficient for a path to exist from \( V_j \) to an element before \( V_j \), which is concatenated in the same manner to \( P \). We repeat the procedure for violations of 3 and 4 until all conditions are satisfied (which can be repeated up to \(| C | \) times, since each iteration uses a later end node in the c-component).

The above procedure is sufficient for the resulting path to be a \( \mathcal{O}^X \)-adversarial latent path starting from \( V_i \), which completes the proof.

\[ \square \]

**Lemma 4.1.** Let \( \mathcal{O}^X := \text{FindOX}(\mathcal{G}, X, Y) \). Suppose \( \exists X_i \in X \) s.t. \( X_i \in X \setminus \mathcal{O}^X \). Then \( X \) is not imitable with respect to \( Y \) in \( \mathcal{G} \).

**Proof.** We prove this by induction, starting with a proof that \( X \cap \text{Before}(X_i) \) is not imitable.

**Base Case:** Define \( X_1 \) to be the earliest element of \( \mathcal{O}^X \setminus \text{ch}(X_i) \) if all children of \( X_i \) are in \( \mathcal{O}^X \), otherwise let it be \( X_i \).

Since \( X_1 \) was not in \( \mathcal{O}^X \), it means that there is a \( \mathcal{O}^X \)-adversarial latent path from \( X_1 \) to element \( V_j \in \text{after}(X_j) \), crossing colliders \( A_1, \ldots, A_k \) (Lem. A.3). Between each two successive observed nodes of the latent path, there is a latent variable that is a common ancestor. That is, the path is of the form \( X_1 \leftarrow \ldots \leftarrow U_1 \rightarrow \ldots \rightarrow A_1 \leftarrow \ldots \rightarrow V_j \). The set of these latent variables can be \( \{ U_1, U_2, \ldots, U_{k+1} \} \). Set these variables to each be \( U_i \sim \text{Bern}(0.5) \) (i.e. random coin flips), and have the directed paths from \( U^{X_1} \) to the observed nodes of the path simply pass through the values. Then, let each \( A_i = U_i \oplus U_{i+1} \), \( X_i = U_1 \), and \( V_i = U_{k+1} \). Finally, construct a \( \mathcal{O}^X \)-adversarial directed path set (Def. A.2) from \( (A_1, \ldots, V_i, X_i) \) to \( Y \) that passes these values to \( Y \). \( Y \) then accepts the set of values \( (A_1, \ldots, V_i, X_i) \) which conform to the condition \( 0 = X_1 \oplus V_j \oplus \bigoplus_{A_i \in A} A_i \).

Note that we do not have \( Y \) as a mathematical function of its arguments, but rather as the set of tuples that are considered "correct", with \( Y = 1 \) if the inputs to \( Y \) are in the “accepted” set, and \( Y = 0 \) otherwise.

When the expert is acting, the result is \( 0 \), since \( M \oplus M = 0 \) for any variable \( M \), making \( Y = 1 \):

\[
X_1 \oplus \left( \bigoplus_{A_i \in A} A_i \right) \oplus V_j = U_1 \oplus (U_1 \oplus U_2) \oplus (U_2 \oplus U_3) \oplus \ldots \oplus (U_k \oplus U_{k+1}) \oplus U_{k+1} = 0
\]

We observe that when imitating \( X_i \), the imitator does not have access to \( U_1 \), and all imitated elements can only use \( A \) as context (since \( X_i \) is the last element of \( X \cap \text{Before}(X_i) \) in temporal order, and \( V_i \in \text{after}(X_j) \)). Since many paths for the path set can have crossed \( X_i \), we can create a generalized imitator which has control over all of the inputs to \( Y \) except for \( V_j \). This means that the requirement is now:

\[ 0 = V_j \oplus f(A_1, \ldots, A_k) \oplus U_{k+1} \oplus f(U_1 \oplus U_2, \ldots, U_k \oplus U_{k+1}) \]

\( U_{k+1} \) is only present in \( A_k \), but then \( U_1 \) must be isolated to extract the value of \( U_{k+1} \). This proceeds recursively along the chain until \( U_1 \) is reached, which is not present in any other observed variable, allowing us to conclude that it is impossible to isolate \( U_{k+1} \), and therefore impossible to guess correctly all the time, and so it is impossible to match the expert’s performance (which is 100%). We have therefore shown that \( X \cap \text{Before}(X_i) \) is not imitable.

**Inductive Step:** Define \( X_j \in X \). Suppose that \( X \cap \text{before}(X_j) \) is not imitable, with a given adversarial circuit for \( X \cap \text{before}(X_j) \) as constructed in this proof bearing witness to non-imitability. We will prove that \( X \cap \text{Before}(X_j) \) (i.e. including \( X_j \)) is not imitable, and construct a corresponding circuit.

We are adding another imitator \( X_j \) to the previous path sets, which can possibly come after the previous circuit’s adversarial path \( V_j \) (which came after \( X_{j-1} \)). This means that we must modify the circuit to make sure that \( X_j \) cannot use its ability to observe previous values to “fix” any mistakes made by \( X_{j-1} \).

If none of the paths from the previous circuit pass through \( X_j \), then it cannot affect the value of \( Y \), and thus \( X \cap \text{Before}(X_j) \) is not imitable with an identical circuit as the previous inductive step. Similarly, any adversarial directed paths that enter \( \mathcal{O}^X \) in the second node of the path and move across \( X_j \) before exiting to \( V^{X_j} \) (first node of any adversarial directed path in the circuit is never in \( \mathcal{O}^X \)) means that there is an element in \( \text{after}(X_j) \) in the \( \mathcal{O}^X \)-adversarial latent path that created the node starting this path set, and therefore \( X_j \) cannot know at least one of the values required to

17
“correct” for the value passed down this path (each adversarial latent path corresponds to a sequence of xors of bernoulli random variables - to set these values to be consistent with the full xor chain, $X_j$ would need to have access to the entire chain).

Finally, suppose there is a set $P$ of $\mathcal{O}^X$-adversarial directed paths which enters $O^X$ and passes through $X_j$ before exiting to $V^Y$, and the last node $V_b$ of each such path before entering $O^X$ to $X_j$ is not the first node of the path. Note that the value of $\mathcal{O}^X$ stays the same in each fragment of $O^X$ before exiting to $Y$, since otherwise it would mean that either the path moved across a boundary node directly back into $O^X$ (disallowed by definition of adversarial path), or that the conditioning check succeeded using $V_i, V_i$, but failed when using $X_i, V_i$, where $X_i$ comes after $V_i$ - where there are less elements not conditioned. For each path $p_i \in P$ we have two cases. In the first case, the path did not cross any other $X_i$ after entering the $O^X$ of $X_j$, and in the second case it does.

We start by tackling the first case. We know that $X \cap \text{before}(X_j)$ is not imitable, so with a certain probability, the full set of paths gives a set of values that are not in the acceptable set for $Y$. In particular, it is now possible that by modifying the values that pass through the paths across $X_j$, the imitator might set the values to their correct settings given the context of all previous values. The adversary can prevent this by recognizing that $V_b$ (the last node of $p_i$ before entering $O^X$ to $X_j$) must have an associated $\mathcal{O}^X$-adversarial latent path from $V_b$ to element $V_j \in \text{after}(X_j)$.

Suppose that the path $p_i$ passes $n$ binary values to $Y$ across $V_b$. We combine this with an adversarial latent path, which is of the form $V_b \leftarrow \ldots \leftarrow U_1 \rightarrow \ldots \rightarrow A_1 \leftarrow \ldots \rightarrow V_j$. The set of these latent variables can be $\{U_1, U_2, \ldots, U_{k+1}\} = U^X_j$. Set these variables to each be $U_i \sim \text{Bern}(0.5)^{2n}$ (i.e. $2n$ random coin flips), and have the directed paths from $U^X_j$ to the observed nodes of the path to simply pass through the values. Once again, each $A_i = U_i \oplus U_{i+1}$ where the xor is performed elementwise, and $V_i = U_{k+1}$. At $V_b$, however, we now perform a different operation. We take the $2n$-dimensional vector coming from $U_1$, reshape it to $(n, 2)$, and have as output be $U_1[p_i]$, in other words, the binary values passing through $p_i$ are now indices that choose the binary values from $U_1$ for each element. In other words, if $p_i$ carries the tuple $(0, 1)$, and $U_1 = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, then $U_1[p_i] = (a, d)$. Then, the path $p_i$ replaces its value at $V_b$ with this output, carrying it to $Y$. At $Y$, each element of the set of valid values has its corresponding $p_i$ element replaced with the set of values compatible with:

$$0 = V_b[p_i] \oplus \left( \bigoplus_{A_i \in A} A_i[p_i] \right) \oplus V_j[p_i]$$

This is because once again:

$$U_1[p_i] \oplus (U_1 \oplus U_2)[p_i] \oplus (U_2 \oplus U_3)[p_i] \oplus \ldots \oplus (U_k \oplus U_{k+1})[p_i] \oplus U_{k+1}[p_i] = 0$$

Once again, without being able to change the value of $p_i$ to the correct one before it gets to $V_b$, $X_j$ only has access to the imitated value of $p_i$, which only matches the correctly imitated elements. Without knowledge of $V_j$, $X_j$ cannot correctly account for the elements of $p_j$ which were not imitated/guessed correctly (inductive hypothesis).

Finally, in the second case, the path crossed through another element $X_i$ after entering the $O^X$ of $X_j$. This means that $\mathcal{O}^X$ at $V_b$ for $X_i$ was either $X_i$ or an element after it, meaning that the circuit constructed for $X_i$ still has its $V_i$ value unobserved by $X_j$, and once again cannot be imitated.

By repeating this step for each path crossing $X_j$, and performing the procedure for each $X_j \in X$, we can construct a full circuit for the entire graph, which is not imitable at each step in temporal order after the element $X_i$, completing the proof.

**Theorem 4.1.** If there do not exist adjustment sets satisfying the sequential $\pi$-backdoor criterion for $(\mathcal{G}, X, Y)$, then $X$ is not imitable with respect to $Y$ in $\mathcal{G}$.

**Proof.** By Thm. 3.1 and Lem. 4.1. 

---

18
B Simulations

This section describes implementation details of each experiment.

B.1 Synthetic Binary Simulations

This experiment first generates a randomly sampled model consistent with the causal graphs in Table 1. All variables are binary. This means that each variable $X$ contains 2 possible actions. $Y$ is also binary, such that $E[Y]$ is sufficient for characterizing the imitation results.

For each variable in the causal graph, the model was generated by sampling each element of the conditional probability table (CPT) uniformly at random. For example, $X_2$ in Table 1#1 would have its CPT generated as follows for every combination of values $x_1$, $u_2$ of its parents:

$$P(X_2 = 1 | X_1 = x_1, U_2 = u_2) = \text{Uniform}(0, 1)$$

$Y$ was also generated in the same way:

$$P(Y = 1 | Pa(Y)) = \text{Uniform}(0, 1)$$

$$P(Y = 0 | Pa(Y)) = 1 - P(Y = 1 | Pa(Y))$$

This type of modeling ensures that the resulting bias is an average of randomly chosen models, proving that distributions where the bias is non-negligible consistent with the given graphs are common.

With the models produced as above, for each action, we sampled 10,000 runs, creating a dataset over the observed variables (including expert actions). We subsequently found the empirical CPT for each action $X_i$ to reproduce $P(X_i | Context)$ from the observational data. Context is chosen according to the given policy (Seq $\pi$-Backdoor, $\pi$-Backdoor, Observed Parents, All Observed).

Finally, the trained policies were plugged back into the model, giving their performance in the actual system for a total of 10000 runs. The absolute value of the difference between the expectation of $Y$ from expert and the imitator is recorded.

This procedure, including sampling of new models consistent with the graph, is performed 1000 times for each graph, and the resulting average performance is recorded in Table 1.

B.2 Adversarial Graph with Continuous Data

The causal graph in Fig. 5 was generated to demonstrate a larger problem that uses continuous observed variables rather than the small binary graphs demonstrated in Appendix B.1.

The implemented structure intuitively represents a simplified radar-based cruise control system’s observations of a driver’s behavior. At the time of each action $X_i$, the radar ($R_i$) detects the distance to the car in front, as well as the change in distance over time. The driver is increasing/decreasing ($X_i$) the speed based on both the distance to the car in front, and their own mental state ($H$, for “in a hurry”), which is unobserved by the driving system. The driver’s action results in a new distance/relative velocity between cars ($R_{i+1}$).
The driving system also has access to the full vehicle state, which includes whether the air conditioning (A) has been turned on. The AC state is determined by the driver’s internal thoughts (H), as well as the driving conditions (C), including outside temperature, which are not directly sensed by the system.

Finally, whether the driver is considered a “safe driver” according to highway patrol is dependent on the overall conditions (C), and the distance between cars throughout the measuring period (R_i).

The goal of the system is to drive in such a way that the anti-robot highway patrol will not notice any erratic behavior.

In this model, we once again compare the 4 approaches.

1. **All Observed**: We assume that A is observed before all actions. This gives us the following policy:
   \[
   \begin{align*}
   \pi(X_1|R_1, A) &= P(X_1|R_1, A) \\
   \pi(X_2|R_1, X_1, R_2, A) &= P(X_2|R_1, X_1, R_2, A) \\
   \pi(X_3|R_1, X_1, R_2, X_2, R_3, A) &= P(X_3|R_1, X_1, R_2, X_2, R_3, A)
   \end{align*}
   \]

2. **Observed Parents**: In this approach, we only condition each policy on the subset of the action’s parents that are observed. Namely:
   \[
   \begin{align*}
   \pi(X_1|R_1, A) &= P(X_1|R_1) \\
   \pi(X_2|R_1, X_1, R_2, A) &= P(X_2|R_2) \\
   \pi(X_3|R_1, X_1, R_2, X_2, R_3, A) &= P(X_3|R_3)
   \end{align*}
   \]

3. **π-Backdoor**: The π-backdoor cannot be applied in this situation, since X1 would need to be imitable by itself, whereas there is a path to Y through H that cannot be taken into account at X1.

4. **Sequential π-Backdoor**: The sequential π-backdoor returns the following policy:
   \[
   \begin{align*}
   \pi(X_1|R_1, A) &= P(X_1|R_1) \\
   \pi(X_2|R_1, X_1, R_2, A) &= P(X_2|R_1, X_1, R_2) \\
   \pi(X_3|R_1, X_1, R_2, X_2, R_3, A) &= P(X_3|R_1, X_1, R_2, X_2, R_3)
   \end{align*}
   \]

The only difference between policies here is the data that is taken into account for imitation purposes. In particular, the sequential π-backdoor explicitly ignores the state of the air conditioning, which is available to it, and would alter the outcome (see the “All Observed” policy).

### B.2.1 Generating an Adversarial Environment

The cruise control system has no knowledge of “correct” behavior, and does not understand humans, which includes both the driver and highway patrol. Whatever policy is generated by the algorithm must therefore take into account any possible behaviors that are consistent with the causal graph.

To demonstrate the issues stemming from an incorrectly chosen covariate set, we are free to construct an adversarial model. To do this, we followed a procedure similar to Zhang et al. (2020), maximizing error from naïve behavioral cloning. We first construct a binary model consistent with the above causal graph, then overlay continuous data over the R_i. The imitator will only have access to the continuous data, and will remain ignorant to the underlying data-generating mechanics, except for the causal graph. In particular, we will focus on two sub-structures in the model.

1. The first sub-structure, shown in Fig. [ba] can have the following binary representation:
   \[
   \begin{align*}
   R_1 &\sim \text{Bern}(0.5) \\
   H &\sim \text{Bern}(0.5) \\
   X_1 &= R_1 \oplus H \\
   R_2 &= X_1 \\
   X_2 &= R_2 \oplus H \\
   R_3 &= X_2 \\
   Y &= R_1 \equiv R_3
   \end{align*}
   \]
Given this structure, using only $R_2$ as $X_2$’s parent when imitating will give $P(X_2|R_2) = P(X_2)$, but the correct policy would be $X_2 = R_1$ - meaning that the imitator can only guess correctly half the time.

2. The second sub-structure, from Fig. 6b, has the following binary representation:

- $H \sim \text{Bern}(0.62)$
- $C \sim \text{Bern}(0.62)$
- $A = C \land H$
- $X_3 = H$
- $R_4 = X_3$
- $Y = R_4 \oplus C$

With this structure, not including $A$ in $X_3$’s policy gives an average performance $E[Y] = 0.47$, but including it gives a maximum of $E[Y|do(\pi)] = 0.18$.

Combining these two substructures gives us the following binary version of the full graph:

- $C \sim (\text{Bern}(0.5), \text{Bern}(0.62))$
- $H \sim (\text{Bern}(0.5), \text{Bern}(0.62))$
- $A = C[1] \land H[1]$
- $R_1 = C[0]$
- $X_1 = R_3 \oplus H[0]$
- $R_2 = X_1$
- $X_2 = R_2 \oplus H[0]$
- $R_3 = X_2$
- $X_3 = H[1]$
- $R_4 = X_3$
- $Y = (R_1 == R_3) \land (R_4 \oplus C[1])$

Since the resulting samples of the above model would be binary, we generate an overlay of continuous data which has identical underlying mechanics.

### B.2.2 Applying Continuous Data to a Binary Structure

While we could sample from arbitrary continuous distributions, we choose to use data from the HighD dataset (Krajewski et al., 2018), which includes vehicle trajectories gathered from drones flying over a section of highway. This dataset does not include a causal graph, and hypothesizing a graph from the data is beyond the scope of our contribution. Instead, since this data does not conform to the distribution implied by the adversarial binary structure generated in Appendix B.2.1, we alter the data to match the adversarial structure before performing imitation. This gives us a non-trivial continuous distribution conforming to the causal graph. We are effectively creating a new dataset...
using the randomness/distribution present in the HighD dataset’s trajectories, for which we know the
ground-truth causal diagram and model.

The procedure used to generate a trajectory conforming to a sequence of actions determined by the
causal model from a trajectory in the HighD dataset is:

1. Sample each real trajectory at 4 points (one for each \( R_i \)), giving a tuple for the dis-
tance/change of distance between cars ((\( D_1, \Delta D_1 \), \( D_2, \Delta D_2 \), \( D_3, \Delta D_3 \), \( D_4, \Delta D_4 \))

2. Set each new trajectory to be 

\[
D_i' = D_1 + \sum_{j=2}^{4} |\Delta D_j| \times (-1)^{1-X_j} \text{ based on the actions,}
\]

with \( \Delta D_i = |\Delta D_i| \times (-1)^{1-X_i-1} \).

This gives continuous trajectories conforming to the adversarial causal model.

By using \( Y = ((\Delta D_1 > 0) == (\Delta D_3 > 0))(\delta D_4 > 0)^C[1] \), we get a simulator for trajectories,
and can use the above conversion to evaluate imitator decisions.

B.2.3 Imitators

A 2 hidden layer neural network with (50,20) neurons with ReLU activation, and Adam optimizer
(lr=5e-5) was trained for each \( X_i \) and context pair. At each \( X_i \), the network inputs were the context
variables specific to the method being tested, and output being a prediction of probability of \( X_i \)
(sigmoid activation with BCE loss).

The outputs \( (X_1, X_2, X_3) \) were binarized by sampling from a bernoulli distribution using the outputs
of the network as probabilities.

B.2.4 Results

The testing dataset was converted using the same method as described above, using the outputs
\( X_1, X_2, X_3 \) from the learned policies instead of the ground-truth causal graph as inputs to the
synthetic trajectory generation. This gave an expected performance average for each policy type,
shown in Fig. 4.
C Examples and Simplified Proofs

Lemma C.1. If there is a set $Z \subseteq \text{before}(X_i)$ that satisfies the backdoor criterion for $X_i$, then taking $G^Y$ as the ancestral graph of $Y$, the Markov Boundary $Z'$ of $X_i$ in $G^Y_{X_i}$ also satisfies the backdoor criterion in $G$.

Proof. We know that if $Z'$ exists, $(Y \perp\!\!\!\perp X_i|Z')$ in $G^Y_{X_i}$ by definition of Markov Boundary.

All we need to show is that if $Z \subseteq \text{before}(X_i)$ exists, then $Z' \subseteq \text{before}(X_i)$. With outgoing edges from $X_i$ removed in $G^Y_{X_i}$, the boundary simplifies to $Pa^+(C(X_i)) \setminus \{X_i\}$, and in the ancestral graph of $Y$, each element of $C(X_i)$ is an ancestor of $Y$, and so has an element of $Z \subseteq \text{before}(X_i)$ blocking each such path - and therefore $Pa^+(C(X_i)) \subseteq \text{before}(X_i)$ too. □

Proposition C.1. The distribution of $X_1, X_2$ is not imitable with respect to $Y$ in Fig. 1d.

Proof. Define the following functional dependence between the nodes in the causal diagram Fig. 1d where $\oplus$ represents XOR:

\[
U_1 := \text{Bernoulli}(0.5) \quad U_2 := \text{Bernoulli}(0.5) \\
Z := U_1 \oplus U_2 \quad X_1 := Z \\
X_2 := U_2 \quad Y := ((X_1 \oplus X_2) == U_1)
\]

The idea above is to encode information about $U_1$ in $X_1, X_2$, which can then be verified by $Y$. In particular,

\[
Y = ((X_1 \oplus X_2) == U_1) = (((U_1 \oplus U_2) \oplus U_2) == U_1) = (U_1 == U_1) = 1
\]

Notice that the value of $Y$ compares the imitated values $X_1, X_2$ to $U_1$. Without any way to observe $U_1$, the imitator has no way of guessing correctly more than 50% of the time.

More rigorously, the imitator has a distribution determined by a function $f(Z) \rightarrow (X_1, X_2)$, since only $Z$ is observed. The problem of imitation here can therefore be reduced to finding values for an output distribution for $f$ that result in a distribution over $Y$ identical to the demonstrator’s observational distribution:

The demonstrator’s (i.e. natural) distribution is as follows, which has $P(Y = 1) = 1$

<table>
<thead>
<tr>
<th>$U_1$</th>
<th>$U_2$</th>
<th>$Z$</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$Y$</th>
<th>$P(___)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.25</td>
</tr>
</tbody>
</table>

The target distribution over the imitating function can be written as a set of variables:

| $Z$ | $X_1$ | $X_2$ | $P(f(Z) \rightarrow (X_1, X_2)|Z)$ |
|-----|-------|-------|----------------------------------|
| 0   | 0     | 0     | $a_0$                            |
| 0   | 0     | 1     | $a_1$                            |
| 0   | 1     | 0     | $a_2$                            |
| 0   | 1     | 1     | $1 - a_0 - a_1 - a_2$             |
| 1   | 0     | 0     | $b_0$                            |
| 1   | 0     | 1     | $b_1$                            |
| 1   | 1     | 0     | $b_2$                            |
| 1   | 1     | 1     | $1 - b_0 - b_1 - b_2$             |

Using this, we can now compute $P(Y = 1|U_1 = 0, U_2 = 0)$, where $Z = U_1 \oplus U_2 = 0$, giving $P(X_1 \oplus X_2 = 1) = a_1 + a_2$, this means that $P(Y = 1|U_1 = 0, U_2 = 0) = 1 - a_1 - a_2$, meaning that $a_1 = 0$ and $a_2 = 0$ to match the demonstrator’s distribution (demonstrator never has $Y = 0$).
Next, we compute \( P(Y = 1|U_1 = 1, U_2 = 1) \), with \( Z = U_1 \oplus U_2 = 0 \) once again. \( P(X_1 \oplus X_2 = 1) = a_1 + a_2 \) once again, giving \( P(Y = 1|U_1 = 1, U_2 = 1) = a_1 + a_2 \), which must add up to 1 - but we already required that both variables were 0 to satisfy the previous requirement.

This means that there exists no assignment to the imitator’s probabilities that has \( P(Y = 1) \).

**Proposition C.2.** The distribution of \( X_1, X_2, X_3 \) is not imitable with respect to \( Y \) in Fig. 7c.

**Proof.** This example is used as a demonstration of the ideas behind the proof of Thm. C.1. We notice that there is a latent chain \( Y \leftarrow U_3 \rightarrow Z_3 \leftarrow U_2 \rightarrow Z_2 \leftarrow U_1 \rightarrow X_1 \) (Fig. 7b), which shows that \( Y \) and \( X_1 \) are in the same ancestral c-component.

We will construct an equation of XORs made up of the values of the latent variables in the above chain, such that each latent variable except \( U_3 \) is used twice, canceling itself, so that \( Y \) can check if the chain correctly cancelled with a comparison to \( U_3 \).

To witness, we have:

\[
\begin{align*}
U_1, U_2, U_3 &\sim \text{Bernoulli}(0.5) \\
Z_1 &= 1 \\
X_1 &= U_1 \\
X_2 &= X_1 \\
Z_2 &= U_1 \oplus U_2 \\
X_3 &= X_2 \oplus Z_2 \quad \Rightarrow U_1 \oplus U_1 \oplus U_2 = U_2 \\
Z_3 &= X_3 \oplus U_2 \oplus U_3 \quad \Rightarrow U_2 \oplus U_2 \oplus U_3 = U_3 \\
Y &= (Z_3 == U_3) \quad \Rightarrow (U_3 == U_3) = 1
\end{align*}
\]

Critically, we can now show that without a way to observe \( X_1 \) or \( U_1 \), it is now impossible to correctly set the values incoming to \( Y \):

\[
Y = (Z_3 == U_3) = ((X_3 \oplus U_2 \oplus U_3) == U_3) = ((f(Z_1, Z_2) \oplus U_2 \oplus U_3) == U_3)
\]
Since $X_3$ is imitated, it must be a function of the observed variables ($X_1, X_2$ are generated, the imitator does not observe what would have been had the imitator not performed any action).

The only way to satisfy the above equation is $f(Z_1, Z_2) = U_2$. However, we can substitute: $f(Z_1, Z_2) = f(1, U_1 \oplus U_2)$. With no additional knowledge, there is no way to disentangle $U_1 \oplus U_2$, meaning that there does not exist a function $f$ that outputs $U_2$ from the given inputs. \qed

**Theorem C.1.** Let $G^Y$ be the ancestral graph of $Y$ in $G$. If there is $X_i \in X$ such that $X_i \in C(Y)$, then $X$ is not imitable with respect to $Y$ in $G$.

**Proof.** An example of the steps taken in this proof is given in Prop. \cite{prop-c2}. Let the corresponding chain of latent variables be $Y \leftarrow U_1 \rightarrow V_1 \leftarrow U_2 \rightarrow \ldots \leftarrow U_n \rightarrow X_i$. WLOG, we can assume that the chain does not repeat nodes (if it did, then it has a cycle, and we can create another chain with the cycle removed), and that none of the $V_j$ are imitated (we can shorten the chain to the first element of $X$ along it).

We construct a model for the given graph as follows:

- Create a tree $T$ rooted at $Y$ which will hold the scaffolding for our constructed distribution. Since this is an ancestral graph, each node has a directed path to $Y$. For each $V_1, \ldots, V_{n-1}, X_i$ in reverse topological order, find a single directed path from the node to either $Y$ or a node along a previously found path, whichever intersects first. Define the set of nodes along these paths (including $V$ and $X_i$) as $P$.
- Each of $V_1, \ldots, V_{n-1}$ has 2 inputs from the chain (as well as possibly other inputs). Let the value $V_i = U_i \oplus U_{i+1} \bigoplus_{P_j \in (Pa(V_i) \cap P)} P_j$.
- Each element $P \setminus V$ is defined as $P_i = \bigoplus_{P_j \in (Pa(P_i) \cap P)} P_j$.
- $X_1 = U_n \bigoplus_{P_j \in (Pa(X_i) \cap P)} P_j$.
- $Y = (U_1 \oplus \bigoplus_{P_j \in (Pa(Y) \cap P)} P_j)$.
- Let all $U_i \sim Bernoulli(0.5)$.
- All other variables are set to $1$.

The model’s construction is consistent with the causal graph, and results in $Y = 1$ with probability $1$.

We now show that the imitator has no way of reconstructing $Y = 1$. Define $T'$ as the subtree of $T$ which, starting at $Y$, stops at the first element of $X$, or at the end of the path in $P$. Let these sub-paths be $P'$. The equation resulting for the imitated $Y$ is therefore:

$$Y = \left( U_1 \oplus \bigoplus_{V_i \in V \cap T'} (U_i \oplus U_{i+1}) \right) \left( \bigoplus_{X_i \in X \cap T'} X_i \right)$$

We now imagine a stronger version of the imitator, which replaces the possibly multiple $\bigoplus_{X_i \in X \cap T'} X_i$ with a single function $f$, which has as inputs all possible observed information, including information computed after the action $X$ is taken.

To achieve this, we observe that only the $V_i$ have values not completely determined by their observed parents. We define $V'_i = U_i \oplus U_{i+1}$, which can be computed from the observed values by xoring with its parents. This means the $V'_i$ contain all of the information from the observed values - and are well-defined even for future nodes (nodes that depend on an imitation decision).

This means that the function $f$ has more information than the actual imitator. We will show that even this weaker version of the problem is not imitable:

$$Y = \left( U_1 \oplus \bigoplus_{V_i \in V \cap T'} (U_i \oplus U_{i+1}) \right) \oplus f(V'_1, \ldots, V'_{n-1})$$

25
Since \( V'_i = U_i \oplus U_{i+1} \), we can redefine \( f' = \bigoplus_{V'_i \in V \cap \mathcal{P}_i} (V'_i) \oplus f \) to reduce the above equation to:

\[
Y = (U_1 \equiv f'(U_1 \oplus U_2, ..., U_{n-1} \oplus U_n))
\]

\( U_1 \) is only present in \( V'_1 \), so it must be used in \( f \), but then \( U_2 \) must be isolated to extract the value of \( U_1 \). This proceeds recursively along the chain until \( U_n \) is reached, which is not present in any other observed variable, allowing us to conclude that it is not possible to isolate \( U_1 \) in \( f' \).

This shows that even a generalized imitator that has access to future information cannot perform imitation here, and so the above distribution is not imitable.

**Theorem C.2.** Suppose that there exists an m-factor \((V^X, X, Z, Y) \in \mathcal{G}\), and let \( X' = \{X_2, ..., X_n\}\) (X with the first element in temporal order removed). Then there exists sets \( V^{X'}\), \( Z' \) such that \((V^{X'}, X', Z', Y) \) is an m-factor for \( \mathcal{G} \).

**Proof.** Let \( X_1 \) be the removed variable. Note that since \( X_1 \) comes first in temporal order, none of its ancestors are in \( X \). There are two cases of interest.

The first is when \( X_1 \notin X^B \). In this case, the new m-factor is simply \((V^X, X', Z, Y) \). Since the conditioning sets and \( V^Y \) remain identical, the m-factor conditions hold directly for the new set.

In the case where \( X_1 \in X^B \), we cannot directly remove \( X_1 \), since there can be elements of \( V^X \) that are ancestors of \( X_1 \). We instead show that we can create a new set \( V^{X'} \) and \( Z' \) which removes ancestors of \( X_1 \) from \( V^X \). In particular, we can instead set \( Z' = Z \cup (V^X \cap An(X_1)_{\mathcal{G}^{X^B}}) \), and \( V^{X'} = V^X \setminus An(X_1)_{\mathcal{G}^{X^B}} \). We show that the resulting set satisfies the requirements of an m-factor.

1. Since \( An(X_1)_{\mathcal{G}^{X^B}} \) were the only elements removed from \( V^X \) and added to \( V^Y \), we know that \( X' \in V^{X'} \), since \( X_1 \) was the first element of \( X \) in temporal order (so none of its ancestors are in \( X' \)). Likewise, \( Z' \subseteq V^Y \), since \( V^{X'} = V^Y \cup An(X_1)_{\mathcal{G}^{X^B}} \), which includes \( Z \subseteq V^Y \) and \((V^X \cap An(X_1)_{\mathcal{G}^{X^B}}) \subseteq An(X_1)_{\mathcal{G}^{X^B}}\).

2. Since all ancestors of \( X_1 \) in \( \mathcal{G}^{X^B} \) are removed from \( V^{X'} \), any element in \( X^{X'} \) has the same descendants in \( \mathcal{G}^{X^B} \) as the original m-factor.

3. Suppose not. That is, \( \exists V_x \in V^{X'} \) and \( V_y \in V^{Y'} \) such that \((V_x \not\perp V_y | Z') \) in \( \mathcal{G}^{X^B} \). We know that \( V_y \notin V^X \cap An(X_1)_{\mathcal{G}^{X^B}} \), because those elements are part of \( Z' \), and all elements are independent of \( V_y \) conditioned on \( V_y \). We therefore know that \( V_y \in V^Y \). Likewise, since \( V^{X'} \subseteq V^X, V_x \in V^{X'} \). Suppose that the path crosses colliders \( Z_1, Z_2, ..., Z_k \in Z' \). Suppose that all colliders are from the set \( Z \), and there are none from \( Z' \setminus Z \). This path also exists in the original m-factor, so we have created an unblocked path which violates condition 2 of the original m-factor, \((V^X \not\perp V^Y | Z)\) - a contradiction. Next, suppose that the colliders can be elements of \( Z' \setminus Z \) (i.e. elements added for the new m-factor). Let \( Z_j \) be the last such element along the path. This means that \( Z_j \in V^X \cap An(X_1)_{\mathcal{G}^{X^B}} \). Taking only the portion of the path from \( Z_j \) to \( V_y \), the remaining colliders are from \( Z \), we have created an unblocked path from \( Z_j \in V^X \) to \( V_y \in V^Y \). However, this path cannot exist, since \((V^X \not\perp V^Y | Z)\) by condition 2 of the original m-factor.

4. Suppose not. That is, suppose that there is an unblocked path in \( \mathcal{G}^{X^B} \) from \( X_j \in X' \) to \( V^{Y'} \) conditioned on \((X' \cup Z') \cap \text{before}(X_j)\). Let \( Z_1, ..., Z_k \) be the colliders along this path. For each such collider \( Z_i \) in \( Z' \setminus Z \), we know that it is an ancestor of \( X_1 \), and that it has a directed path to it in \( V^X \). Since \( X_1 \in X \cap \text{before}(X_j) \) (\( X_1 \) is first element of \( X \) in temporal order), we can replace each \( Z_i \) collider in the original path with the directed path from \( Z_i \) to \( X_1 \), and the path repeated back to \( Z_i \), effectively replacing the collider at \( Z_i \) with a collider at \( X_1 \), creating a path from \( X_j \) to \( V_y \) conditioned on \((X \cup Z) \cap \text{before}(X_j)\), violating condition 4 of the original m-factor.

This completes the conditions, showing that the smaller m-factor always exists when removing the first element of \( X \) in temporal order from the set of actions. \( \square \)
When replacing the mechanisms of the imitator, we can now decompose the probability of \( Y \) using the independence relations from the given graph:

\[
P(Y) = \sum_{A, X_2, X_3, C} P(Y, A, X_2, X_3, C) = \sum_{A, X_2, X_3, C} P(Y|A, X_2, X_3, C)P(A, X_2, X_3, C)
\]

\[
= \sum_{A, X_2, X_3, C} P(Y|A, X_2, X_3, C)P(A)P(C)P(X_2|A)P(X_3|AX_2C)
\]

When replacing the mechanisms of \( X \) with their imitated counterparts, we get the following probability for \( Y \):

\[
\hat{P}(Y) = \sum_{V \setminus \{Y\}} P(V, ..., Y) = \sum_{V \setminus \{Y\}} \sum_{U} P(V, ..., Y, U, \ldots)
\]

\[
= \sum_{V \setminus \{Y\}} P(Y|AX_2X_3C)P(A|U_1)P(U_1)P(X_1|B|X_1)P(X_2|A)P(U_2)P(X_3|CX_2A)P(U_3)P(C|U_3)P(U_3)
\]

\[
= \sum_{V \setminus \{Y\}} P(Y|AX_2X_3C)P(A)P(B, X_1)P(X_2|A)P(X_3|CX_2A)P(C)
\]

\[
= \sum_{A, X_2, X_3, C} P(Y|AX_2X_3C)P(A)P(X_2|A)P(X_3|CX_2A)P(C)
\]

These equations match, showing that the given mechanisms are sufficient for imitation. \qed