2024 American Control Conference (ACC)
July 8-12, 2024. Toronto, Canada

Counterfactually-Guided Causal Reinforcement Learning with Reward Machines

Nasim Baharisangari, Yash Paliwal, and Zhe Xu

Abstract— **In causal reinforcement learning (RL), counterfactual reasoning deals with "what if" situations and allows for investigating the potential consequences of actions or events that did not actually happen. In this paper, we combine counterfactual reasoning and reinforcement learning (RL) and propose Counterfactually-Guided Causal Reinforcement Learning with Reward Machines (CGC-RL). In CGC-RL, using observational data, we first compute the optimal *counterfactual sequence* with the highest probability of completing a given task. Then, we construct an RM compatible with the *counterfactual sequence*. We use the constructed RM to apply dynamic potential-based reward shaping to encourage the agent to follow the *counterfactual sequence*. We prove the policy-invariance under dynamic reward shaping with RMs. Finally, we implement CGC-RL in one case study and compare the results with three baselines. Our results show that CGC-RL outperforms the baselines.**

I. Introduction

In causal learning/inference, counterfactual reasoning is a cognitive process that involves mentally exploring and evaluating alternative scenarios or outcomes that differ from reality. It allows individuals to imagine "what if" situations, enabling them to understand the potential consequences of different actions or events that did not actually happen. In reinforcement learning (RL), we utilize Partially Observable Markov Decision Processes (POMDPs) to model scenarios with partially observable environments. In such cases, states must be inferred from limited observations. Counterfactual reasoning, when incorporated, can help us learn the optimal policy from historical observational data. This is especially useful for completing complex tasks.

In the realm of RL, the integration of causal inference/learning involves the utilization of three distinct categories of data: observational data, experimental data, and counterfactual data, coupled with the causal diagram delineating the RL framework, if accessible. The agent's access to observational data can stem from diverse sources such as observing fellow agents, scrutinizing the environment, offline learning, and acquiring a foundational understanding of the underlying structure. Counterfactual data can be generated through a specified model, or estimated via active empirical learning methods [1], [2].

In the intersection of CI and RL, these aforementioned data types have been employed both individually and in various combinations. As an illustration, in [3], an agent enhances policy optimization through the acquisition of observational data within novel environments, enabling the agent to enact minimal yet pivotal adjustments based on the structural relationships among the RL framework's variables, as depicted in associated diagrams. In [4], a synthesis of observational and experimental data is harnessed to apprehend "causal states," which represent the coarsest partition of concatenated historical actions and observations, offering maximal predictiveness for future outcomes within partially observable Markov decision processes (POMDPs). In [5], a counterfactually-guided policy improvement search is proposed that uses observational data to find the best policy in POMDPs.

Moreover incorporating reward machines (RM) in RL has shown noticeable advantages [6]. For example, in [7], a framework for learning RMs for POMDPs is proposed which outperforms some of the common RL algorithms used in POMDPs such as DDQN, DQN and AC3.

In this paper, we propose a counterfactually-guided causal RL with RMs (CGC-RL) in POMDPs. In CGC-RL, we exploit observational data to infer abstract counterfactual guidance for the agent in RL such that the agent can learn a suboptimal policy that is closest to the optimal policy. In CGC-RL, we use counterfactual guidance to build a reward machine that encourages the agent to follow the counterfactual guidance to complete a given task. Our results show that CGC-RL outperformed the baselines including LRM+DQRM which is proposed in [8] as an RL method with reward machines in POMDPs.

II. Preliminaries

Partially Observable Markov Decision Process (POMDP): In partially observable problems, the underlying environment model is usually considered to be POMDP. A POMDP is a tuple $\mathcal{P}_{\mathcal{O}} = \langle S, O, A, r, p, w, \gamma \rangle$ where S is a finite set of *states*, A is a finite set of *actions*, $r : S \times A \to \mathbb{R}$ is the *reward function*, $p(s, a, s')$ is the *transition probability distribution*, γ is the *discount factor*, O is a finite set of *observations*, and $\omega(s, o)$ is the *observation probability distribution*. At each time step t, the agent is at state $s_t \in S$, executes an action $a_t \in A$, receives reward $r_t = r(s_t, a_t)$, and moves to state s_{t+1} according to $p(s_t, a_t, s_{t+1})$. In POMDP, the agent does not observe s_{t+1} but only receives an observation $o_{t+1} \in O$. This observation provides the agent with a clue about s_{t+1} through ω. RL methods can not be immediately applied to POMDPs due to the fact that the transition probabilities and reward function are generally non-Markovian w.r.t observations. Hence, for

The authors are with the School for Engineering of Matter, Transport and Energy, Arizona State University, Tempe, AZ 85287. {nbaharis, ypaliwal, xzhe1}@asu.edu

This work is partially supported by NSF CNS 2304863 and CNS 2339774 and ONR N00014-23-1-2505.

converging to optimal policies, one may need to consider the complete history $o_0, a_0, ..., a_{t-1}, o_t$ of observations and actions. Many RL methods for POMDPs, incorporate the history of observations and actions through recurrent neural network (RNN), and then use a policy gradient method to train the RNN [8].

Definition 1. *For an environment Env and a set of labels $\mathcal{P}$ that correspond to a set of features of Env, we define a labeling function $L : O_\emptyset \times A_\emptyset \times O \rightarrow 2^{\mathcal{P}}$ that assigns truth values to the symbols in $\mathcal{P}$ given an environment experiment $ex = (o, a, o')$. For an arbitrary set X, $X_\emptyset \triangleq X \cup \emptyset$ [8].*

Definition 2. *A deterministic finite automaton (DFA) is a finite state machine described using a tuple $\mathcal{W} = (V, 2^{\mathcal{P}}, \delta_w, v_I, F)$ where V is a finite set of states, $2^{\mathcal{P}}$ is the alphabet, v_I is the initial state, $F \subseteq V$ is the set of final states, and $\delta_W : V \times 2^{\mathcal{P}} \rightarrow V$ is the deterministic transition function. A run of DFA $\mathcal{W}$ on an arbitrary label sequence $z_0 z_1, ...z_k \in (2^{\mathcal{P}})^*$ is a sequence of states and labels $\tau := v_0 z_0 v_1 z_1, ...z_k v_{k+1}$, such that $v_0 = v_I$ and for each $0 \leq i \leq k$, $v_{i+1} = \delta_W(v_i, z_i)$. An accepted run is a run that ends in a final state $v_{k+1} \in F$. Finally, we define the language of $\mathcal{W}$ as $La(\mathcal{W}) = \{\tau \in (2^{\mathcal{P}})^* | \tau$ is accepted by $\mathcal{W}\}$.*

Definition 3. *Given a set of labels $\mathcal{P}$, a Reward Machine (RM) is a tuple $\mathcal{R}_{\mathcal{P}} = \langle U, u_0, \delta_u, \delta_r \rangle$ where U is a finite set of states, $u_0 \in U$ is an initial state, δ_u is the state-transition function, $\delta_u : U \times 2^{\mathcal{P}} \rightarrow U$, and δ_r is the reward-transition function, $\delta_r : U \times 2^{\mathcal{P}} \rightarrow \mathbb{R}$.*

III. Policy-Invariance Under Dynamic Reward Shaping with Reward Machines

The idea of potential-based reward shaping has been used in RL algorithms to take into consideration of prior knowledge/understanding of the environment that can improve the performance of RL algorithm. Potential-based reward shaping refers to shaping the reward function r as $r + F(s, s')$ where $F(s, s') = \gamma\Phi(s') - \Phi(s)$ with $\Phi(s)$ being some potential function. In [9], it is proven that the optimal policy remains unchanged under potential-based reward shaping. Furthermore, dynamic potential-based reward shaping using the reward shaping function $F(s, s', t) = \gamma\Phi(s', t) - \Phi(s, t)$ is proven to preserve the optimal policy [10] where $\Phi(s, t)$ is some potential function which is a function of the state of the environment and time.

In this paper, we write the augmented potential-based reward shaping function as $F(s, u, t, s', u', t') = \gamma\Phi(s', u', t') - \Phi(s, u, t)$ where t', s', and u' are respectively the current time-step, the current state in which the agent is, and the current state of $\mathcal{R}_{\mathcal{P}}$, and t, s, and u are respectively the previous time-step, the previous state in which the agent was, and the previous state of $\mathcal{R}_{\mathcal{P}}$. We can prove that the optimal policy π^* does not change under this reward shaping. In other words, the difference between the shaped Q-values and the true Q-values does not depend on the action a in an arbitrary augmented state (s, u). We formalize this using the following theorem.

Theorem 1. *Given a reward machine $\mathcal{R}_{\mathcal{P}}$, discount factor γ, a state $s \in S$, and an action $a \in A$, for the optimal Q-value $Q^*(s, a)$, using reward shaping function $F(s, u, t, s', u', t') = \gamma\Phi(s', u', t') - \Phi(s, u, t)$, the shaped optimal Q-value for (s, a) is $Q^*_\Phi = Q^*(s, a) - \Phi(s, u, t)$ where $\Phi(s, u, t)$ is some potential function.*

This shows that the difference between the true Q-values and the shaped Q-values does not depend on the action taken in an arbitrary augmented state (s, u).

IV. Counterfactually-Guided Causal RL with Reward Machines

In this section, we introduce counterfactually-guided causal RL with RMs (CGC-RL). Before explaining the framework in detail, we define the necessary concepts and notations. To clarify the concepts and definitions, we use Ex. 1 as a running example. Here, we represent a given task Υ with a DFA $\mathcal{W}$.

Example 1. *In a grid world with 9 cells, there are three airports Airport 1, Airport 2, and Airport 3. The task Υ of a taxi agent is to first pick up traveler number 1 from an airport and then pick up traveler number 2 from an airport. Fig. 1 illustrates the task DFA of this example. Due to a lack of communication, the agent does not know from which airport it should pick up each traveler. In this environment, $\mathcal{P} = \{$🚶,🧍,$✈^1$,$✈^2$,$✈^3\}$ and $A = \{$go right, go left, go up, go down$\}$. 🚶 is $True$ if the agent picked up Traveler 1 with its last action (by going to the airport where Traveler 1 was) and 🧍 is $True$ if the agent picked up Traveler 2 with its last action (by going to the airport where Traveler 2 was). $✈^1$,$✈^2$, or $✈^3$ is $True$ if the agent is at the airport with the corresponding number.*

Definition 4. *Given the set of states S, for an observation o, we define $\mathcal{F}^o$ as the set of all the states $s \in S$ such that $\omega(o|s) > 0$.*

In CGC-RL, we divide the observations $o \in O$ into two types of observations: *random observations* and *fixed observations*. We formally define these two types as follows.

Definition 5. *We define a fixed observation o^{F} as an observation such that $\omega(o^{\mathrm{F}}|s) = 1, \forall s \in \mathcal{F}^{\mathrm{F}}$. We denote the set of fixed observations by $O^{o^{\mathrm{F}}}$.*

Definition 6. *We define a random observation o^{R} as an observation that $\omega(o|s) \in (0, 1), \forall s \in \mathcal{F}^{\mathrm{R}}$. We denote the set of random observations by $O^{o^{\mathrm{R}}}$.*

For brevity, we write $O^{o\mathrm{F}}$ as O^{F} and $O^{o\mathrm{R}}$ as O^{R}. Based on this categorization, we can divide the labels in $\mathcal{P}$ into fixed and random labels. A fixed label denoted by e^{F} is a label that becomes $True$ when the agent observes a fixed observation.

$$\begin{tikzcd}
\text{start} \arrow[r] & v_0 \arrow[loop above, "\neg 🚶"] \arrow[r, "🚶"] & v_1 \arrow[loop above, "\neg 🧍"] \arrow[r, "🧍"] & \boxed{v_2} \arrow[loop above, "True"]
\end{tikzcd}$$

Fig. 1: Task DFA $\mathcal{W}$ in Example 1.

A random label denoted by e^{R} is a label that becomes $True$ when the agent observes a random observation. We denote the set of fixed labels and random labels by $\mathcal{P}^{\mathrm{F}}$ and $\mathcal{P}^{\mathrm{R}}$, respectively. In Ex. 1, the set of fixed labels is $\mathcal{P}^{\mathrm{F}} = \{✈^1, ✈^2, ✈^3\}$ and the set of random labels is $\mathcal{P}^{\mathrm{R}} = \{🚺, 🚹\}$.

For an arbitrary label $e \in \mathcal{P}$, we can define a set of states $\mathcal{F}^e$ similar to the sets of states $\mathcal{F}^o$ as in Def. 4. For an observation $o \in O$ that causes a label $e \in \mathcal{P}$ to become $True$, we have $\mathcal{F}^o = \mathcal{F}^e$. Hereafter, we use the notation of $\mathcal{F}$ for the labels.

Assumption 1. *In this paper, we assume that only random labels appear on task DFA $\mathcal{W}$ related to a given task Υ because the environment is partially observable.*

In developing CGC-RL, we further prune the set of labels and use one subset of fixed labels and one subset of random labels to increase the efficiency of the proposed algorithm.

Remark 1. *The subset of random labels $\mathcal{P}^{\mathrm{R}_{\mathrm{sub}}} \subseteq \mathcal{P}^{\mathrm{R}}$ that we use in CGC-RL include the random labels that appear in the task DFA $\mathcal{W}$ related to a given task Υ. We use $N_{|\mathcal{P}^{\mathrm{R}_{\mathrm{sub}}}|}$ to denote the number of random labels in $\mathcal{P}^{\mathrm{R}_{\mathrm{sub}}}$.*

Remark 2. *The subset of fixed labels $\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}} \subseteq \mathcal{P}^{\mathrm{F}}$ that we use in CGC-RL are those fixed labels that meet the following conditions. We use $N_{|\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}|}$ to denote the number of fixed labels in $\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}$.*

1) $\exists i \in \{1, ..., N_{|\mathcal{P}^{\mathrm{R}_{\mathrm{sub}}}|}\}$, *such that* $Pr\big(\mathcal{F}^{e^{\mathrm{F}}_q} \cap \mathcal{F}^{e^{\mathrm{R}}_i} \neq \emptyset\big) > 0, \forall q \in \{1, ..., N_{|\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}|}\}$, *and*
2) $\bigcap_{q=1}^{N_{|\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}|}} \{\mathcal{F}^{e^{\mathrm{F}}_q}\} = \emptyset$.

Re. 2 states that we are interested in properties of the environment that give us deterministic clues about the states s in which the subtasks in a given task DFA $\mathcal{W}$ have a nonzero probability of being satisfied (Item 1). The states associated with each fixed label are different from each other so each clue guides the agent to a distinct part of the environment (Item 2). In Ex. 1, the set of fixed labels e^{F} is $\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}} = \{✈^1, ✈^2, ✈^3\}$. Here, there is a nonzero probability that the e^{R} labels in $\mathcal{P}^{\mathrm{R}_{\mathrm{sub}}} = \{🚺, 🚹\}$ get $True$ at the states related to $Airport$ 1 (denoted by $\mathcal{F}^{✈^1}$), i.e., we have $Pr(\mathcal{F}^{✈^1} \cap \mathcal{F}^{🚺} \neq \emptyset) > 0$ and $Pr(\mathcal{F}^{✈^1} \cap \mathcal{F}^{🚹} \neq \emptyset) > 0$, and we have the same pair of probabilities for the other airports (Item 1 in Def. 2). Also, we have $\mathcal{F}^{✈^1} \cap \mathcal{F}^{✈^2} \cap \mathcal{F}^{✈^3} = \emptyset$ which means there is no state s such that any two of the labels $✈^1, ✈^2, ✈^3$ are $True$ at s simultaneously (Item 2).

We use fixed labels to construct *counterfactual guidance* for the agent such that a given task is satisfied with the highest probability following the guidance. In doing so, we incorporate *counterfactual composite actions*. Hereafter, we refer to *counterfactual guidance* as *counterfactual sequence*.

Definition 7. *In an environment Env, for each fixed label $e^{\mathrm{F}} \in \mathcal{P}^{\mathrm{F}}$, we define a counterfactual composite action which is a function of e^{F} denoted by $\chi(e^{\mathrm{F}}) \in \mathcal{X}$ as a sequence of actions $a \in A$ that may take some time steps to be completed such that executing $\chi(e^{\mathrm{F}})$ ends up in e^{F} to be $True$.*

Counterfactual composite actions provide a compact representation of the actions needed to be taken according to the *counterfactual sequence*. In Ex. 1, the set of counterfactual composite actions $\chi(e^{\mathrm{F}})$ is $\mathcal{X} = \{\mathbf{Navigate}(✈^1), \mathbf{Navigate}(✈^2), \mathbf{Navigate}(✈^3)\}$. If at time-step t, the agent is at an arbitrary state s and the counterfactual composite action $\mathcal{X}(✈^1) = \mathbf{Navigate}(✈^1)$, then the agent needs to navigate to a state s' where $✈^1$ is $True$ and completing this may take some time steps.

In the next step, we construct the ordered sequence of random labels that appear in a given accepting run of a task DFA $\tau := v_0 e^{\mathrm{R}}_0 v_1 e^{\mathrm{R}}_1 ... e^{\mathrm{R}}_{N^\tau - 1}$ as $\theta^\tau := e^{\mathrm{R}}_0, ..., e^{\mathrm{R}}_{N^\tau - 1}$ where N^τ is the number of random labels in τ. In Ex. 1, for the accepting run $\tau := v_0 🚺 v_1 🚹 v_2$, we have $\theta^\tau := 🚺, 🚹$. In what follows, we define the *counterfactual sequence* that *satisfies* the sequence of random labels e^{R} corresponding to an accepting run τ.

Definition 8. *For a given $\theta^\tau := e^{\mathrm{R}}_0, ..., e^{\mathrm{R}}_{N^\tau - 1}$ where $\tau \in La(\mathcal{W})$, we define an ordered sequence of counterfactual composite actions $\chi(e^{\mathrm{F}}) \in \mathcal{X}$ as $\xi := \chi_0(e^{\mathrm{F}}_0), ..., \chi_{N^\tau - 1}(e^{\mathrm{F}}_{N^\tau - 1})$ such that $\xi \models \theta^\tau$, i.e., executing $\chi_i(e^{\mathrm{F}}_i)$ leads the agent to the states where e^{F}_i is $True$ which then leads to e^{R}_i to be $True$, $\forall i \in \{0, ..., N_\tau - 1\}$.*

In Ex. 1, if the ground truth reality in an episode is that Traveler 1 is at $Airport$ 1 and Traveler 2 is at $Airport$ 2, then $\theta^\tau := 🚺, 🚹$ can be satisfied by $\xi := \mathbf{Navigate}(✈^1), \mathbf{Navigate}(✈^2)$. This means that if the agent first go to $Airport$ 1 where $✈^1$ is $True$, then it observes 🚺 which causes 🚺 to be $True$. The same description holds for 🚹.

We use counterfactual sequences to provide the best guidance through RMs that lead the agent to complete a given task. Doing so, we counterfactually reason how the agent achieves the best suboptimal policy. We expand on this matter later in this section.

Definition 9. *In the environment Env, given a task DFA $\mathcal{W}$, the set of all the counterfactual composite actions $\mathcal{X}$, the set E containing all the possible sequences of the composite actions $\chi(e^{\mathrm{F}}) \in \mathcal{X}$, and the set of fixed labels $\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}$, we define the optimal counterfactual sequence ξ^* as $\xi^* := \chi_0(e^{\mathrm{F}}_0), ..., \chi_{N^\tau - 1}(e^{\mathrm{F}}_{N^\tau - 1})$ such that $\xi^*, \tau^* = \arg\max_{\xi \in E, \tau \in La(\mathcal{W})} Pr(\xi \models \theta^\tau)$. We denote the run corresponding to ξ^* by τ^* where $Pr(\xi \models \theta^\tau)$ denotes the probability of ξ satisfying θ^τ.*

In this paper, the optimal counterfactual sequence ξ^* is the intervention that we make to figure out what would happen if the agent follows this sequence hypothetically. We expect that this intervention causes the agent to learn the best suboptimal policy. To learn this sequence, we need to know the *prior probability distribution of the occurrence of the labels in $\mathcal{P}^{\mathrm{R}_{\mathrm{sub}}}$* which is unknown. We define this prior distribution in Def. 10. Later in this section, we explain that we calculate the posterior probability distribution of occurrence of the labels in $\mathcal{P}^{\mathrm{R}_{\mathrm{sub}}}$from observational data. We use this probability distribution to find the optimal counterfactual sequence ξ^*, i.e., the sequence of composite actions that aid the agent in completing a given task with the highest probability of success.

Definition 10. *Given the sets* $\{\mathcal{F}^{e^{\mathrm{F}}_q}\}_{q=1}^{N_{|\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}|}}$*, for each radnom label* $e^{\mathrm{R}} \in \mathcal{P}^{\mathrm{R}_{\mathrm{sub}}}$*, we define the categorical probability distribution* $\{Pr(e^{\mathrm{R}} = True|\mathcal{F}^{e^{\mathrm{F}}_q})\}_{q=1}^{N_{|\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}|}}$ *with* $N_{|\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}|}$ *categories.*

In Ex. 1, for $\text{🚺} \in \{\text{🚺},\text{🚹}\}$, we can have the following categorical distribution $Pr(\text{🚺} = True|\mathcal{F}^{\text{✈}^1}) = 0.5$, $Pr(\text{🚺} = True|\mathcal{F}^{\text{✈}^2}) = 0.35$, and $Pr(\text{🚺} = True|\mathcal{F}^{\text{✈}^3}) = 0.15$.

Now, we explain the algorithms we use in CGC-RL in detail. In CGC-RL, we first compute a counterfactual sequence ξ^* from the observational data $\mathcal{D}$ and then compute its corresponding θ^{τ^*} using the subroutine `FindSE()` (Alg. 1). In the next step, we construct $\mathcal{R}_{\mathcal{P}}$ compatible with ξ^* and θ^{τ^*} as illusatred in Fig. 2. Then, we use the subroutine `FindPolicy()` (Alg. 2) to compute the policy π using $\mathcal{R}_{\mathcal{P}}$. Below, we explain Alg. 1 and Alg. 2 in detail, respectively.

In Alg. 1, for a task DFA $\mathcal{W}$, we first compute the posterior probability distribution $\{Pr(e^{\mathrm{R}}_i = True|\mathcal{F}^{e^{\mathrm{F}}_q}, \mathcal{D})\}_{q=1}^{N_{|\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}|}}$ for each $e^{\mathrm{R}}_i \in \mathcal{P}^{\mathrm{R}_{\mathrm{sub}}}$ (Line 3 in Alg. 1). Then, we compute $La(\mathcal{W})$ where $|La(\mathcal{W})|$ denotes the number of runs in $La(\mathcal{W})$ and initialize $|La(\mathcal{W})|$ empty sets $\{E_{n^\tau}\}_{n^\tau=1}^{|La(\mathcal{W})|}$ each corresponding to each $\tau \in La(\mathcal{W})$ (Lines 5-6 in Alg. 1). Then, for each $\tau \in La(\mathcal{W})$, we first compute the corresponding θ^τ (Line 10 in Alg. 1). After that, Alg. 1 generates N_g generations of random sequences of $\chi(e^{\mathrm{F}}) \in \mathcal{X}$ (i.e., ξ) with the population size of N_p stored in the set $\mathcal{S}$ (Lines 11-12). In the next step, Alg. 1 randomly generates a truth assignment for each $e^{\mathrm{R}}_i \in \mathcal{P}^{\mathrm{R}}$ according to $\{Pr(e^{\mathrm{R}}_i = True|\mathcal{F}^{e^{\mathrm{F}}_q}, \mathcal{D})\}_{q=1}^{N_{|\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}|}}$ (Line 13 in Alg. 1). In the next step, for each $\xi \in \mathcal{S}$, we check whether $\xi \models \theta^\tau$. If yes, we assign a point of 1 to ξ and store it in the set E_{n^τ} corresponding to the run used in iteration n^τ. In all of the generations, a sequence ξ that satisfies θ^τ might appear more than one time. Hence, for each ξ that satisfies θ^τ, `FindSE()` keeps track of the cumulative points val_{ξ,n^τ} (Lines 14-20 in Alg. 1). Then, `FindSE()` returns ξ^* with the highest val_{ξ,n^τ} and its corresponding θ^{τ^*}.

Theorem 2. *If for all* $i \in \{1, ..., N_{|\mathcal{P}^{\mathrm{R}_{\mathrm{sub}}}|}\}$*, in* $\{Pr(e^{\mathrm{R}}_i = True|\mathcal{F}^{e^{\mathrm{F}}_q}, \mathcal{D})\}_{q=1}^{N_{|\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}|}}$*, there exists a category that has a higher probability than other categories then, Alg. 1 converges to the sequence* ξ^* *when* $N_g \to \infty$*.*

In the next step, we construct the RM $\mathcal{R}_{\mathcal{P}}$, where $\mathcal{P} = \mathcal{P}^{\mathrm{R}_{\mathrm{sub}}} \cup \mathcal{P}^{\mathrm{F}_{\mathrm{sub}}} \cup \mathcal{Y}$ and $\mathcal{Y}$ is the set of other labels in $\mathcal{P}$ and we have $\mathcal{P}^{\mathrm{R}_{\mathrm{sub}}} \cap \mathcal{P}^{\mathrm{F}_{\mathrm{sub}}} \cap \mathcal{Y} = \emptyset$. $\mathcal{R}_{\mathcal{P}}$ is deterministic and its main job is to guide the agent to follow ξ^* and θ^{τ^*}. The structure of $\mathcal{R}_{\mathcal{P}}$ is a linear chain (Fig. 2), of all of the states of RM ending in the final state u_F and encourages the agent to follow ξ^* to complete the task corresponding to θ^{τ^*}. We denote the number of the states of RM except for u_0 and u_F by $N_u \in \{1, 2, ...\}$. $\mathcal{R}_{\mathcal{P}}$ gives potential-based rewards to the agent during the *navigation* transitions where *navigation* transitions are those transitions in $\mathcal{R}_{\mathcal{P}}$ that encourage the agent to go to states s where labels e^{F} in ξ^* become $True$. $\mathcal{R}_{\mathcal{P}}$ starts from the initial state u_0 and transitions to u_1 once $y_0 \in \mathcal{Y}$ is set to $True$ and this is not a navigation transition and no potential-based reward will be given. y_0 is a label related to Env that triggers starting θ^{τ^*}. $\mathcal{R}_{\mathcal{P}}$ transitions to u_2 from u_1 once $e^{\mathrm{F}}_0 = True$ and this is a navigation transition. The agent receives the potential-based reward of $r^{\xi}_0 = \gamma\Phi(s', u_2, t) - \Phi(s, u_1, t)$ upon transitioning to u_2. Finally, $\mathcal{R}_{\mathcal{P}}$ transitions from u_{N_u} to u_F upon completing the task in θ^{τ^*}, and the agent receives a reward of R_f.

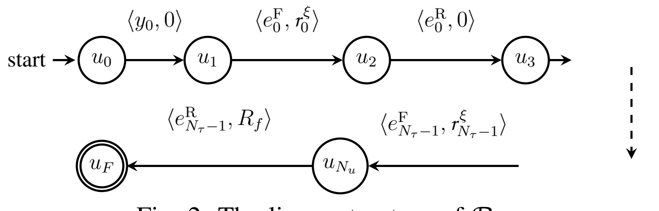
Finally, we use `FindPolicy()` (Alg. 2), to compute the policy π with $\mathcal{R}_{\mathcal{P}}$. Alg. 2 first initializes the environment state, RM, and the policy π which is a random policy (Lines 3-6). In Line 3, σ denotes the truth value of any possible label in $\mathcal{P}$ upon observing o. Then Alg. 2 starts the training of the agent in $eplength$ episodes. Then, in Alg. 2, the agent selects an action based on the current $\pi, o,$ and u, and receives the reward r and observation o from the environment, and then, $\mathcal{R}_{\mathcal{P}}$ transition from u to u' (Lines 8-10). In line 9, the Boolean variable $done$ sets to true if a terminal condition is met upon executing a. At Line 11, the agent gets the predefined potential values $\Phi(s, u, t)$ and $\Phi(s', u', t)$ based on the type of the transition between u and u'. If the transition between u and u' is a navigation transition, then $\Phi(s, u, t) = 0$ and $\Phi(s', u', t) = m$ where $m \in \mathbb{R}^+$ is an arbitrary positive value. Else, $\Phi(s, u, t) = \Phi(s', u', t) = 0$. At Line 12, the agent receives the potential-based reward r' from $\mathcal{R}_{\mathcal{P}}$ on top of the reward r. Finally, if at any point during an episode, a terminal condition is met, then, the current episode terminates and the next episode starts (Lines 14-15).

Remark 3. *Fig. 2 illustrates the core linear structure of the reward machine we use in CGC-RL. Other labels in* $\mathcal{Y}$ *can appear in non-navigation transitions in* $\mathcal{R}_{\mathcal{P}}$ *as needed.*

Remark 4. *The proposed algorithm can be applied to the task DFAs that have fixed labels that are not used in the set* $\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}$*. In this case, we modify the counterfactual sequence* ξ *according to the location of those fixed labels deterministically.*

V. Simulations

In this section, we implement CGC-RL in the Locked-Doors Domain. Locked-Doors domain and the task defined in this domain are the more complicated versions of the 2-Keys domain and its corresponding task in [8], [11]. Here we use CGC-RL with DQRM (CGC-DQRM). DQRM is a deep Q-learning method for reward machines proposed in



$\langle y_0, 0\rangle$
$\langle e^{\mathrm{F}}_0, r^{\xi}_0\rangle$
$\langle e^{\mathrm{R}}_0, 0\rangle$
start
u_0
u_1
u_2
u_3
$\langle e^{\mathrm{R}}_{N_\tau-1}, R_f\rangle$
$\langle e^{\mathrm{F}}_{N_\tau-1}, r^{\xi}_{N_\tau-1}\rangle$
u_F
u_{N_u}

Fig. 2: The linear structure of $\mathcal{R}_{\mathcal{P}}$.

Algorithm 1: Computation of the Optimal Counterfactual Sequence from Observational data.

```
Input: Observational Data $\mathcal{D}$, the set of counterfactual composite actions $\mathcal{X}$, the set of all the random labels $\mathcal{P}^{\mathrm{R}_{\mathrm{sub}}}$, the set of fixed labels $\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}$ and task DFA $\mathcal{W}$
Parameter: Number of generations $N_g$ and the size of the population $N_p$
function FindSE($\mathcal{D}, \mathcal{X}, \mathcal{P}^{\mathrm{R}_{\mathrm{sub}}}, \mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}, \mathcal{W}$)
    for $i = 1, ..., |\mathcal{P}^{\mathrm{R}_{\mathrm{sub}}}|$ do
        Compute $\{Pr(e_i^{\mathrm{R}} = True|\mathcal{F}^{e_q^{\mathrm{F}}}, \mathcal{D})\}_{q=1}^{N_{|\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}|}}$ using $\mathcal{D}$
    end for
    $La(\mathcal{W}) \leftarrow$ compute-language($\mathcal{W}$)
    $\emptyset \leftarrow E_1, ..., E_{|La(\mathcal{W})|}$
    $0 \leftarrow n^\tau$
    for $\tau \in La(\mathcal{W})$ do
        $n^\tau \leftarrow n^\tau + 1$
        $\theta^{n^\tau} \leftarrow$ compute-run-seq($\tau$)
        for $i' = 1, ..., N_g$ do
            Generate the set $\mathcal{S}$ consists of $N_p$ random sequences of counterfactual actions in $\mathcal{X}$
            Randomly generate the truth assignment of each $e_i^{\mathrm{R}} \in \mathcal{P}^{\mathrm{R}_{\mathrm{sub}}}$ according to $\{Pr(e_i^{\mathrm{R}} = True|\mathcal{F}^{e_q^{\mathrm{F}}}, \mathcal{D})\}_{q=1}^{N_{|\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}|}}$
            for $k = 1, ..., N_p$ do
                if $\xi_k \models \theta^{n^\tau}$ then
                    if $\xi_k \in E_{n^\tau}$ then
                        $val_{\xi_k, n^\tau} \leftarrow val_{\xi_k, n^\tau} + 1$
                    else
                        $val_{\xi_k, n^\tau} \leftarrow 1$
                        Add $\xi_k$ to $E_{n^\tau}$
            end for
            $\emptyset \leftarrow \mathcal{S}$
        end for
    end for
    $\xi^* \leftarrow \arg\max_{\xi \in \bigcup_{n^\tau=1}^{|La(\mathcal{W})|} E_{n^\tau}} val_{\xi, n^\tau}$
    return $\xi^*, \theta^{\tau^*}$
```

[7]. DQRM uses DDQN to learn a subpolicy for each state of a given RM. A DQN or Deep Q-Network, approximates a state-value function in a Q-learning framework with a neural network [12]. DDQN refers to Double Deep Q-Network which is a variant of the deep Q-network (DQN) [13].

The baselines we use are as follows.

1) **DDQN-RM:** This method uses DDQN for learning the optimal policy where we incorporate the extra information about the state of the reward machine with an extra binary vector. We concatenate this binary vector to the state of the agent.
2) **LRM-DQRM:** This method which is proposed in [8], [11] simultaneously learns a reward machine and exploits that reward machine to learn a policy. In this approach, if the learned reward machine is not the best one then it attempts to find a new reward machine. If the reward machine is updated, a new policy will be learned from scratch. LRM-DQRM outperforms several baselines that were used in [8], [11].

Algorithm 2: RL with Dynamic Potential-Based Reward Shaping using RMs

```
Input: Reward Machine $\mathcal{R}_{\mathcal{P}}$
Parameter: Episode length $eplength$, Discount factor $\gamma$, Number of episodes $N_s$
function FindPolicy($\mathcal{R}_{\mathcal{P}}$)
    for $i = 1, ..., N_s$ do
        $o \leftarrow$ env-get-initial-state(),
        $\sigma \leftarrow L(\emptyset, \emptyset, o)$
        $u \leftarrow \delta_u(u_0, \sigma)$
        $\pi \leftarrow$ initialize-policy()
        for $0 \leq t < eplength$ do
            $a \leftarrow$ select-action($\pi, o, u$)
            $o', r,$ done $\leftarrow$ env-execute-action($a$)
            $u' \leftarrow \delta_u(u, L(o, a, o')), \sigma' \leftarrow L(o, a, o')$
            $\Phi(s, u, t), \Phi(s', u', t) \leftarrow$ get-potential-value($u, u'$)
            $r' \leftarrow r + \gamma\Phi(s', u', t) - \Phi(s, u, t)$
            $\pi \leftarrow$ improve($\pi, o, u, \sigma, a, r', o', u', \sigma',$ done)
            if done then
                start-next-episode()
            $o \leftarrow o', u \leftarrow u', \sigma \leftarrow \sigma'$
        end for
    end for
    return $\pi$
```

3) **CGC-DQRM-No:** This method is CGC-DQRM without potential-based reward shaping.

We use DDQN to synthesize observational data $\mathcal{D}$ to calculate the categorical distribution $\{Pr(e_i^{\mathrm{R}} = True|\mathcal{F}^{e_q^{\mathrm{F}}}, \mathcal{D})\}_{q=1}^{N_{|\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}|}}$. Also, we use 1500 episodes with a maximum length of 1000 time steps.

A. *Locked-Doors Domain*

The map of the Locked-Doors Domain is shown in Fig. 5. In this domain $\mathcal{P}^{\mathrm{R}_{\mathrm{sub}}} = \{$🍏,✊^{1},✊^{2},🔓^{1},🔓$^2\}$, and $\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}} = \{$🟨,🟩,🟦,🟥$\}$, $\mathcal{Y} = \{$🍎,🔑^{1},🔑^{2},🔒^{1},🔒^{2},,●,🟪,🚪^{1},🚪$^2\}$, and $A = \{$go up, go down, go right, go left$\}$. 🔑^{1} is $True$ if the agent is currently in the same room as the first key, 🍎 is $True$ if the agent is currently in the same room as apple, 🔒^{1} and 🔒^{2} are $True$ if the agent is in the same room where the doors are locked. 🍏 is $True$ if the agent ate the apple with its last action, 🔓^{1} is $True$ if the agent opened the first door with its last action, ✊^{1} is $True$ if the agent grabbed the first key with its last action. In the Locked-Doors Domain, there is a button ● yellow room. Once the agent presses the button, an apple randomly appears in either of the green, blue, or red rooms and two distinct keys that respectively unlock the first door and the second door appear in either of the two other rooms in which the apple has not appeared. Both the first and the second doors get locked in the room where the apple appeared.

Using observational data, we compute $\{Pr($🍏$ = True|\mathcal{F}^{e_q^{\mathrm{F}}}, \mathcal{D})\}_{q=1}^{|\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}|}$ as $Pr($🍏$ = True|\mathcal{F}^{\text{🟦}}, \mathcal{D}) = 0.053$, $Pr($🍏$ = True|\mathcal{F}^{\text{🟥}}, \mathcal{D}) = 0.674$, and $Pr($🍏$ = True|\mathcal{F}^{\text{🟩}}, \mathcal{D}) = 0.272$. We compute $\{Pr($✊$^1 = True|\mathcal{F}^{e_q^{\mathrm{F}}}, \mathcal{D})\}_{q=1}^{|\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}|}$ as $Pr($✊$^1 = True|\mathcal{F}^{\text{🟦}}, \mathcal{D}) =$

0.177, $Pr(\text{✊}^1 = True|\mathcal{F}^{🟥}, \mathcal{D}) = 0.312$, and $Pr(\text{✊}^1 = True|\mathcal{F}^{🟩}, \mathcal{D}) = 0.511$. We compute $\{Pr(\text{✊}^2 = True|\mathcal{F}^{e^{\mathrm{F}}_q}, \mathcal{D})\}_{q=1}^{|\mathcal{P}^{\mathrm{F}_{\mathrm{sub}}}|}$ as $Pr(\text{✊}^2 = True|\mathcal{F}^{🟦}, \mathcal{D}) = 0.793$, $Pr(\text{✊}^2 = True|\mathcal{F}^{🟥}, \mathcal{D}) = 0.118$, and $Pr(\text{✊}^2 = True|\mathcal{F}^{🟩}, \mathcal{D}) = 0.089$. The posterior probability distributions of 🔓^1 and 🔓^2 are the same as 🍎.

The given task DFA $\mathcal{W}$ for this case study is shown in Fig. 3. Using Alg. 1, we compute $\xi^* :=$ **Navigate**(🟩), **Navigate**(🟥), **Navigate**(🟦), **Navigate**(🟥) and $\theta^{\tau^*} := \text{✊}^1, \text{🔓}^1, \text{✊}^2, \text{🔓}^2$, 🍎. In this case study, we build the reward machine RM as in Fig. 4. Fig. 6 illustrates the results for the Locked-Doors Domain. As can be seen, CGC-DQRM is the only algorithm that converges to 0.6 of the cumulative reward. For LRM-DQRM, we set $u_{\max} = 15$ in this case study. As can be seen, LRM-DQRM did not successfully learn a reward machine for the Locked-Doors Domain; hence, LRM-DQRM could not learn a policy for this case study.

VI. Conclusion

We proposed counterfactually-guided causal RL with RMs (CGC-RL) in partially observable environments. In CGC-RL, we use observational data to infer the best counterfactual sequence for an agent to complete a given task. We then, build a reward machine that incorporates the counterfactual sequence to guide the agent to complete a given task. As a future direction, we plan to apply CGC-RL to multi-agent RL.

References

[1] A. Forney, J. Pearl, and E. Bareinboim, "Counterfactual data-fusion for online reinforcement learners," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 1156–1164. [Online]. Available: https://proceedings.mlr.press/v70/forney17a.html

[2] S. Pitis, E. Creager, and A. Garg, "Counterfactual data augmentation using locally factored dynamics," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS'20, Red Hook, NY, USA, 2020.

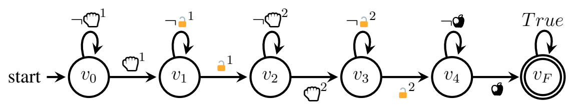
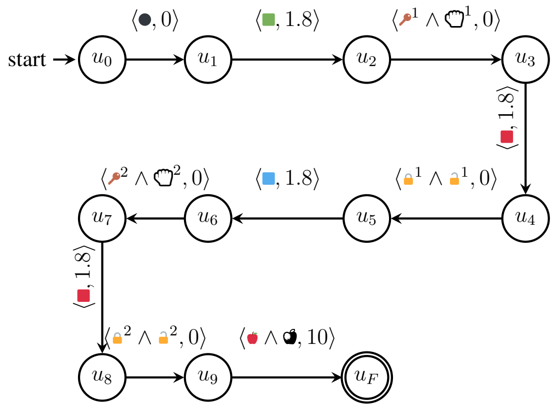


Fig. 3: Task DFA $\mathcal{W}$ in the Locked-Doors Domain.



start
u_0
u_1
u_2
u_3
u_4
u_5
u_6
u_7
u_8
u_9
u_F
$\langle ●, 0\rangle$
$\langle 🟩, 1.8\rangle$
$\langle \text{🔑}^1 \wedge \text{✊}^1, 0\rangle$
$\langle 🟥, 1.8\rangle$
$\langle \text{🔓}^1 \wedge \text{🔓}^1, 0\rangle$
$\langle 🟦, 1.8\rangle$
$\langle \text{🔑}^2 \wedge \text{✊}^2, 0\rangle$
$\langle 🟥, 1.8\rangle$
$\langle \text{🔓}^2 \wedge \text{🔓}^2, 0\rangle$
$\langle 🍎 \wedge 🍎, 10\rangle$

Fig. 4: The linear structure of $\mathcal{R}_{\mathcal{P}}$ in Locked-Doors Domain.

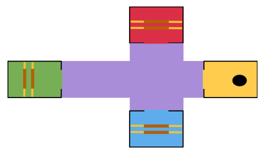


Fig. 5: The map of the rooms in Locked-Doors Domain.

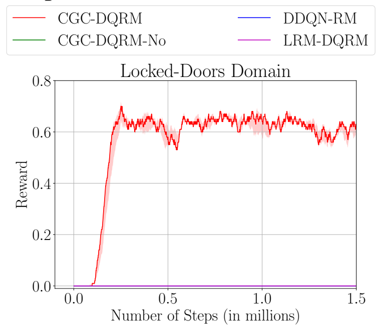


Fig. 6: The obtained results for the Locked-Doors Domain averaged over three independent runs.

[3] B. Huang, F. Feng, C. Lu, S. Magliacane, and K. Zhang, "AdaRL: What, where, and how to adapt in transfer reinforcement learning," *ArXiv*, vol. abs/2107.02729, 2021.

[4] A. Zhang, Z. C. Lipton, L. Pineda, K. Azizzadenesheli, A. Anandkumar, L. Itti, J. Pineau, and T. Furlanello, "Learning causal state representations of partially observable environments," *ArXiv*, vol. abs/1906.10437, 2019.

[5] L. Buesing, T. Weber, Y. Zwols, S. Racaniere, A. Guez, J.-B. Lespiau, and N. Heess, "Woulda, coulda, shoulda: Counterfactually-guided policy search," 2018.

[6] Z. Xu, I. Gavran, Y. Ahmad, R. Majumdar, D. Neider, U. Topcu, and B. Wu, "Joint inference of reward machines and policies for reinforcement learning," in *Proceedings of the 30th International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI Press, 2020, pp. 590–598.

[7] R. T. Icarte, T. Q. Klassen, R. A. Valenzano, and S. A. McIlraith, "Using reward machines for high-level task specification and decomposition in reinforcement learning," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 2112–2121.

[8] R. Toro Icarte, T. Q. Klassen, R. Valenzano, M. P. Castro, E. Waldie, and S. A. McIlraith, "Learning reward machines: A study in partially observable reinforcement learning," *Artificial Intelligence*, vol. 323, p. 103989, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370223001352

[9] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the Sixteenth International Conference on Machine Learning*, ser. ICML '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, p. 278–287.

[10] S. Devlin and D. Kudenko, "Dynamic potential-based reward shaping," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, ser. AAMAS '12. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2012, p. 433–440.

[11] R. Toro Icarte, E. Waldie, T. Klassen, R. Valenzano, M. Castro, and S. McIlraith, "Learning reward machines for partially observable reinforcement learning," *Advances in neural information processing systems*, vol. 32, 2019.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[13] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," 2015.