Expediting Reinforcement Learning by Incorporating Temporal Causal Information

Jan Corazza
TU Dortmund University
jan.corazza@tu-dortmund.de

Hadi Partovi Aria
Arizona State University
hpartovi@asu.edu

Daniel Neider
TU Dortmund University
daniel.neider@tu-dortmund.de

Zhe Xu
Arizona State University
xzhe1@asu.edu

Abstract

Reinforcement learning (RL) algorithms struggle with learning optimal policies for tasks where reward feedback is sparse and depends on a complex sequence of events in the environment. Probabilistic reward machines (PRMs) are finite-state formalisms that can capture temporal dependencies in the reward signal, along with nondeterministic task outcomes. While special RL algorithms can exploit this finite-state structure to expedite learning, PRMs remain difficult to modify and design by hand. This hinders the already difficult tasks of utilizing high-level causal knowledge about the environment, and transferring the reward formalism into a new domain with a different causal structure. This paper proposes a novel method to incorporate causal information in the form of Temporal Logic-based Causal Diagrams into the reward formalism, thereby expediting policy learning and aiding the transfer of task specifications to new environments.

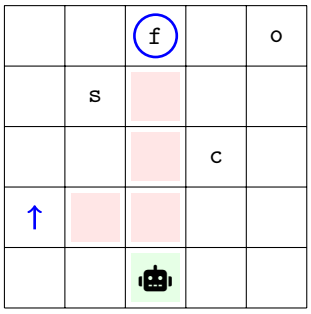
1 Introduction

In RL the interaction between the agent and the environment happens step by step. Starting in state s the agent chooses an action a with probability $\pi(a \mid s)$ (the policy), and the environment transitions into a new state s' and gives a reward r. This interaction is formalized in the concept of an MDP, a tuple $M = (S, A, R, p, \gamma)$ where S is the set of states, A the set of actions available to the agent, $R : (S \times A)^\star \times S \to \mathbb{R}$ the reward function mapping trajectories in the MDP to rewards, $p(s' \mid s, a)$ a probabilistic transition function, and $\gamma \in (0, 1)$ the discount factor. The agent's goal is to maximize the expected discounted return, $\max_\pi \mathbb{E}_\pi[\sum_{i=0}^{\infty} \gamma^i r_i]$. A labeling function $L : S \times A \times S \to 2^{\mathrm{AP}}$ can be provided to attach descriptive propositional variables to transitions in the MDP. An MDP together with a labeling function is called a labeled MDP.

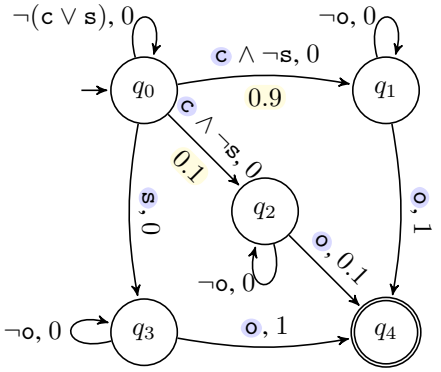
Although MDPs can have a large number of states and a complex transition function, one often has access to high-level causal knowledge of the environment. Figure 1a illustrates this point on a small example MDP. To complete the task, the agent must choose to bring either coffee or a soda to the office. The high-level knowledge one may supply is that any path from the soda to the office is later blocked by a flower pot, which the agent must avoid. This is due to walls and a one-way door, which constrain the agent's movement. Although special RL algorithms can find the optimal policy for this task, they will not take these temporal-causal constraints into account, and will explore the environment in an inefficient manner. Unfortunately, employing high-level knowledge about causality has shown to be a difficult task, as the current causal RL approaches (e.g., [1, 2, 3, 4, 5, 6, 7, 8]) mostly do not take into account the *temporal* aspect of the causal knowledge. This paper aims to

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

address this issue by proposing a novel method that incorporates knowledge about causality directly into the reward function.



(a) A labeled 5x5 Gridworld with coffee (c), soda (s), an office (o), and a flower pot (f). The agent can move in the four cardinal directions, and starts in the cell labeled. Other shaded cells are impassable walls. One-way doors are represented by the upwards arrow. The flower pot acts as a sink state.



(b) A PRM for the task in Figure 1a (left). Transitions are labeled with propositional formulas and reward outputs. Only transition probabilities different from 1 are shown. State q_4 is a terminal state which ends the task. Formulas on transitions from q_0 to q_1 and q_2 ($\texttt{c} \wedge \neg\texttt{s}$) contain $\neg\texttt{s}$ in order to disambiguate the transition function in state q_0 on the input $\{\texttt{c}, \texttt{s}\}$. The same could be achieved by using the formula $\neg\texttt{c} \wedge \texttt{s}$ for the transition from q_0 to q_3, and just c for transitions into q_1, q_2.

Figure 1: An MDP (left) and a PRM (right) that captures the task of bringing either coffee or soda to the office. The coffee machine has a probability of 10% to malfunction and produce bad coffee, leading to a reduced reward of 0.1 instead of 1. Bringing soda to the office results in a reward of 1 deterministically. An example input for the PRM is $\{\texttt{c, s}\}, \emptyset, \{\texttt{o, c}\}$ (a sequence of three labels), which will induce the run $q_0 \mapsto q_3 \mapsto q_3 \mapsto q_4$ with a reward of 1. It is important to note that inputs for PRMs are sets of descriptive propositional variables that are true in a given step, hence why a single label such as $\{\texttt{c, s}\}$ can include multiple (or 0) variables.

1.1 Probabilistic Reward Machines

Common RL algorithms such as Q-learning struggle with tasks where rewards are sparse and depend on a complex sequence of actions that the agent must perform in a specific order. Reward machines [9] are a finite-state formalism that can capture the reward function in such cases. Q-learning for Reward Machines (QRM) [9] can exploit this reward structure to expedite learning the optimal policy. A more general variant of reward machines, called *probabilistic reward machines* [10], use a nondeterministic transition function that can capture uncertainty in task outcomes. In the example from Figure 1, uncertainty comes from the fact that the coffee machine may malfunction. Definition 1.1 formalizes this notion of a finite-state representation of a temporally extended task with probabilistic outcomes.

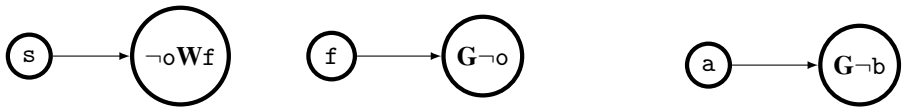
Definition 1.1 (Probabilistic Reward Machine (PRM))**.** A PRM $\mathsf{A} = (U, u_I, 2^{\text{AP}}, \Gamma, \tau, \sigma, F)$ is a tuple where U is a finite set of states with a distinguished initial state $u_I \in U$, AP is a set of atomic propositions and 2^{AP} is the set of labels, $\Gamma \subset \mathbb{R}$ is a finite set of rewards, $\tau : (U \times 2^{\text{AP}} \times U) \to [0, 1]$ is a probabilistic transition function, $\sigma : (U \times 2^{\text{AP}} \times U) \to \Gamma$ is a function mapping each transition to a reward in Γ, and $F \subseteq U$ is a finite set of terminal states that signal the end of the interaction.

The agent-environment interaction generates a trajectory $s_0, a_0, s_1, \ldots, a_{n-1}, s_n$ and the corresponding label sequence $\ell_0\ell_1 \cdots \ell_{n-1}$, where $L(s_i, a_i, s_{i+1}) = \ell_i$ for all $i = 0, \ldots, n-1$. The state s_0 may be a unique initial state, or drawn from an initial distribution. After reading a label ℓ in state u, the PRM executes a nondeterministic transition into a new state u' with probability $\tau(u, \ell, u')$, and the agent receives a reward $r = \sigma(u, \ell, u')$. A run of a PRM A on a label sequence $\ell_0\ell_1 \cdots \ell_{n-1}$ is a sequence $u_0, r_0, u_1, \ldots, r_{n-1}, u_n$ where $u_0 = u_I$, and for all $i = 0, \ldots, n-1$, $\tau(u_i, \ell_i, u_{i+1}) > 0$ and $\sigma(u_i, \ell_i, u_{i+1}) = r_i$.

1.2 Temporal Logic-based Causal Diagrams

Linear temporal logic over finite sequences (LTL_f) is a formal reasoning system that can capture causal and temporal properties of label sequences and labeled MDPs. Aside from Boolean operators like $\neg$ and $\vee$, LTL_f introduces temporal operators such as $\mathbf{G}\psi$ (true if and only if ψ holds for every element in the sequence), $\mathbf{X}\psi$ (true iff. ψ holds for the next element of the sequence), and $\psi\mathbf{U}\varphi$ (true iff. ψ holds until φ becomes true, and φ is true in some element of the sequence). We also rely on the weak until operator $\psi\mathbf{W}\varphi$ (true iff. ψ holds until φ becomes true, but φ is not required to become true).

In order to encode knowledge about causality in the underlying MDP, we rely on Temporal Logic-based Causal Diagrams (TL-CDs) [11]. TL-CDs are a special notation that expresses the causal relationship between formulas in LTL_f. The first conjunct induced by the TL-CD in Figure 2a, $\mathbf{G}(\texttt{s} \rightarrow \neg\texttt{o}\mathbf{W}\texttt{f})$, means that if the agent observes `s` (soda) in any step, then it will not observe `o` (the office) before it observes `f` (the flower pot). This part of the TL-CD encodes knowledge that soda may only be reached via a one-way door, and the only other exit towards the office will be blocked by the flower pot.



(a) $\mathbf{G}(\texttt{s} \rightarrow \neg\texttt{o}\mathbf{W}\texttt{f}) \wedge \mathbf{G}(\texttt{f} \rightarrow \mathbf{G}\neg\texttt{o})$ (b) $\mathbf{G}(\texttt{a} \rightarrow \mathbf{G}\neg\texttt{b})$

Figure 2: Figure 2a (left) is the TL-CD which captures relevant causal information in the environment from Figure 1a. Figure 2b (right) is a TL-CD that holds for the case study in Figure 6.

Formally, a TL-CD is a directed graph whose nodes are labeled with LTL_f formulas. For a TL-CD $\mathcal{C}$ one may construct an equivalent LTL_f formula $\varphi^{\mathcal{C}} = \bigwedge_{\varphi \blacktriangleright \psi} \mathbf{G}(\varphi \rightarrow \psi)$, where $\varphi \blacktriangleright \psi$ iterates over edges that connect formulas φ and ψ in the TL-CD. If $\varphi^{\mathcal{C}}$ is true for a label sequence ℓ, we will write $\ell \models \varphi^{\mathcal{C}}$. A label sequence $\ell = \ell_0\ell_1 \cdots \ell_{n-1}$ is attainable in an MDP $M = (S, A, R, p, \gamma)$ if there exists a trajectory $s_0, a_0, s_1, \ldots, a_{n-1}, s_n$ in M such that $L(s_i, a_i, s_{i+1}) = \ell_i$ and $p(s_i, a_i, s_{i+1}) > 0$ for all $i = 0, 1, \ldots, n-1$. We will say that a TL-CD $\mathcal{C}$ holds for an MDP M if for every label sequence ℓ attainable in M, we have $\ell \models \varphi^{\mathcal{C}}$. In order to simplify working with TL-CDs, we leverage the notion of deterministic finite automata (DFAs). We formalize this notion in Definition 1.2.

Definition 1.2 (Deterministic Finite Automaton (DFA))**.** A DFA is a tuple $\mathsf{C} = (Q, q_I, \Sigma, \delta, F)$ consisting of a finite set of states Q with an initial state q_I, input alphabet Σ, deterministic transition function $\delta : Q \times \Sigma \rightarrow Q$, and a finite set of accepting states $F \subseteq Q$.

If the run of the DFA C on an input string ℓ ends in an accepting state $q \in F$, we will write $\ell \in L(\mathsf{C})$. Every TL-CD $\mathcal{C}$ can be converted into an equivalent DFA C, in the sense that for every ℓ, we have $\ell \in L(\mathsf{C}) \iff \ell \models \varphi^{\mathcal{C}}$. We will refer to C as the *causal DFA*.

2 Problem statement

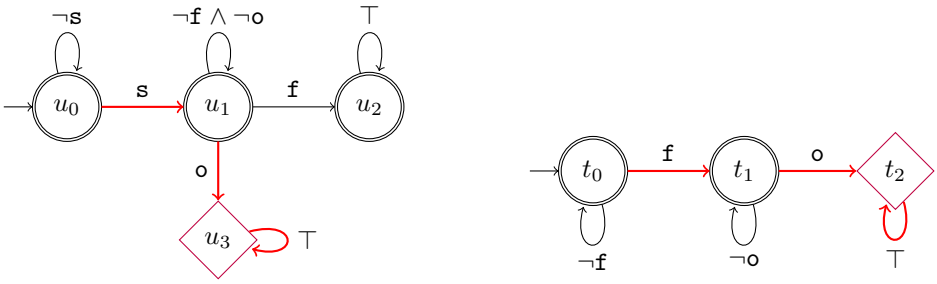
One may use QRM to find the optimal policy for the task in Figure 1. However, the PRM in Figure 1b does not take flower pots and one-way doors into account. Because the agent does not know that knocking over flower pots is forbidden or that choosing soda causes him to enter a room blocked by a flower pot, it will waste time exploring those fruitless trajectories. As PRMs are in essence task specifications, and one may also wish to transfer them into a new environment while preserving the overall goal. In both cases, high-level insights about causality, especially its temporal aspects, could prove helpful by reflecting the dynamics of the MDP in condensed form.

Unfortunately, incorporating knowledge about causality into the reward function remains a difficult and error-prone manual task. In PRMs, this would necessitate adding new states and reasoning about a different, more complicated transition function. Some methods such as JIRP [12] and SRMI [13] assume that a suitable but unknown representation of the reward function exists, and attempt to recover it from interaction traces. This work proposes an alternative method that leverages TL-CDs in order to automate the process of incorporating knowledge about causality into PRMs. More formally, the problem can be stated as follows. Given a TL-CD $\mathcal{C}$ which holds for an MDP M and a PRM A,

produce a PRM B that induces the same optimal policy as A, but utilizes causal information in $\mathcal{C}$ to expedite learning.

3 Method

We first consider the equivalent causal DFA for a given TL-CD. As explained in Section 1.2, the equivalent causal DFA captures the same semantics as the given TL-CD. While TL-CDs are an intuitive notational tool, DFAs are easier to work with computationally. The causal DFA for the TL-CD in Figure 2a is shown in Figure 3 (in two parts for convenience). State u_3 is a *sink* state, meaning that any run of the DFA which enters u_3 will never leave it. It is also a *rejecting* state. Taken together, this means that any label sequence for which the causal DFA enters u_3 is not the prefix of an attainable sequence in an MDP M if we assume that the TL-CD holds for M.



(a) $\mathbf{G}(\mathtt{s} \rightarrow \neg\mathtt{o}\mathbf{W}\mathtt{f})$ (b) $\mathbf{G}(\mathtt{f} \rightarrow \mathbf{G}\neg\mathtt{o})$

Figure 3: Two factors of the causal DFA for the TL-CD in Figure 2a. Rejecting sink states are rectangular. Their parallel composition is the true causal DFA, and its states come from the Cartesian product of states in this Figure. For example, the initial state is (u_0, t_0).

In other words, when a causal DFA run reaches a rejecting sink state on some prefix of input labels, then the entire input label sequence is unattainable. The reason is that there is no suffix of labels which can cause the run to transition into an accepting state. From now on we will implicitly consider causal DFAs to have at most one rejecting sink state, and that an accepting state is reachable from all other states. This can be achieved by minimization [14].

We propose to incorporate causal information from a TL-CD $\mathcal{C}$ into a PRM A by computing state values in a new PRM $\mathsf{B}_1 = \mathcal{C} \times \mathsf{A}$, which is a product of the TL-CD (represented by the causal DFA C) and A. The product PRM B_1 synchronizes the runs of the original PRM A and the causal DFA C. B_1 mirrors the output of A, except when C transitions into a rejecting sink state. Then the output of B_1 is set to a minimal value m that is lesser than any possible immediate reward and resulting future gain, and will remain there for the rest of the run as C will not leave the sink state. We also compute state values in a "pessimistic" PRM $\mathsf{B}_2 = \mathcal{C} \times (-\mathsf{A})$ in order to uncover temporal-causal information about worst-case reward outcomes. While B_1 outputs the same rewards as A, B_2 negates outputs of A (but also gives minimal outputs m for transitions into rejecting sink states). Because of the minimal reward output m, value iteration in either B_1 or B_2 will disregard transitions that lead C into a rejecting sink state, as explained under Figure 4. Due to negating reward outputs, label sequences that maximize return in B_2, minimize the return in B_1. We combine state value information from B_1 and B_2 into a final PRM B. To obtain B, we start from B_1, and add all states $u \in U^{\mathsf{B}_1}$ that have 0 value in both the machine B_1 and B_2 ($v^\star_{\mathsf{B}_1}(u) = v^\star_{\mathsf{B}_2}(u) = 0$) into the set of terminal states F^{B_1}. Such states have the property that no matter the policy, the future return is constrained with 0 from above and below (and thus, the choice of actions is of no consequence). The product $\mathcal{C} \times \mathsf{A}$ is formalized in Definition 3.1. We define the value of a PRM state u via the Bellman optimality equation 1, where γ matches the discount factor in the MDP.

$$v^\star(u) = \max_{\ell \in 2^{\mathsf{AP}}} \sum_{u' \in U} \tau\left(u, \ell, u'\right) \cdot \left(\sigma\left(u, \ell, u'\right) + \gamma v^\star\left(u'\right)\right) \tag{1}$$

As Equation 1 is an optimality equation, $v^\star(u)$ is the expected return of a PRM run starting in u and following the most optimistic label sequence (which may or may not be attainable in the MDP). We define the minimal reward output m as $m = -1 - \max_{r \in \Gamma^{\mathsf{A}}} |r| - \max_{u \in U^{\mathsf{A}}} v^\star(u)$.

Definition 3.1 (PRM & TL-CD product)**.** Let $M = (S, A, R, p, \gamma)$ be an MDP where the reward function $R : (2^{\text{AP}})^\star \to \Gamma$ is given by the PRM $\mathsf{A} = (U^{\mathsf{A}}, u_I^{\mathsf{A}}, 2^{\text{AP}}, \Gamma^{\mathsf{A}}, \tau^{\mathsf{A}}, \sigma^{\mathsf{A}}, F^{\mathsf{A}})$, $\mathcal{C}$ a TL-CD that holds for M, and $\mathsf{C} = (Q, q_I, 2^{\text{AP}}, \delta, F_{\mathsf{C}})$ its equivalent minimal causal DFA with states Q, initial state q_I, a set of accepting states $F_{\mathsf{C}} \subseteq Q$, and transition function δ. Let $Q_{\text{r.s.}} \subseteq Q \setminus F_{\mathsf{C}}$ be the set of rejecting sink states of C.

We define the product $\mathcal{C} \times \mathsf{A}$ as a new PRM $(U, u_I, 2^{\text{AP}}, \Gamma, \tau, \sigma, F)$, where

1. $U = U^{\mathsf{A}} \times Q$, a state of $\mathcal{C} \times \mathsf{A}$ is a pair of states (u, q) with $u \in U^{\mathsf{A}}$ and $q \in Q$;
2. $u_I = (u_I^{\mathsf{A}}, q_I)$, the initial state in $\mathcal{C} \times \mathsf{A}$ is the pair of initial states of A and C;
3. $\Gamma = \Gamma^{\mathsf{A}} \cup \{m\}$, the output alphabet of $\mathcal{C} \times \mathsf{A}$ is expanded with a possible reward output that is lesser than any future gain in A;
4. $\tau((u, q), \ell, (u', q')) = \tau^{\mathsf{A}}(u, \ell, u') \cdot \mathbb{1}_{\{\delta(q,\ell)=q'\}}$, the probability of $\mathcal{C} \times \mathsf{A}$ transitioning from (u, q) to (u', q') upon reading ℓ is the same as the probability of A transitioning from u to u', given that C transitions from q to q' (otherwise, the probability is 0);
5. $\sigma((u, q), \ell, (u', q')) = \begin{cases} \sigma^{\mathsf{A}}(u, \ell, u') & q' \notin Q_{\text{r.s.}} \\ m = -1 - \max_{r \in \Gamma^{\mathsf{A}}} |r| - \max_{u \in U^{\mathsf{A}}} v^\star(u) & \text{otherwise} \end{cases}$, the output of the product PRM agrees with A except when C transitions into a rejecting sink state; and
6. $F = \{(u, q) : u \in F^{\mathsf{A}}\}$, terminal states in $\mathcal{C} \times \mathsf{A}$ correspond to terminal states in A.

Performing value iteration acts as a form of look-ahead in the product $\mathcal{C} \times \mathsf{A}$, whose output function is defined so that transitions which lead the causal DFA into a rejecting sink state do not contribute to overall state value. The same is true for $B_2 = \mathcal{C} \times (-\mathsf{A})$, which is defined in the same way, except the output function $-\sigma^{\mathsf{A}}(u, \ell, u')$ provides look-ahead information about the worst-case future outcome. Our method, given in Algorithm 1, improves the convergence speed of QRM by utilizing information about expected rewards that better reflects the temporal causal structure of the environment. See the Appendix for further details about the function that computes the PRM and causal DFA intermediate product.

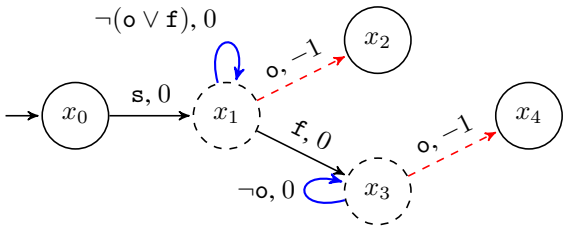


Figure 4: A fragment of the product of the PRM from Figure 1b and the TL-CD from Figure 2a. Inheriting the q, u, and t names of PRM and causal DFA states from previous figures, $x_0 = (q_0, u_0, t_0)$, $x_1 = (q_3, u_1, t_0)$, $x_2 = (q_4, u_3, t_0)$, $x_3 = (q_3, u_2, t_1)$, and $x_4 = (q_4, u_2, t_2)$. Due to the maximum in Equation 1, dashed transitions do not contribute to state value. Dashed states x_1 and x_3 have 0 value in both B_1 (depicted) and B_2, and will be added to the set of terminal states.

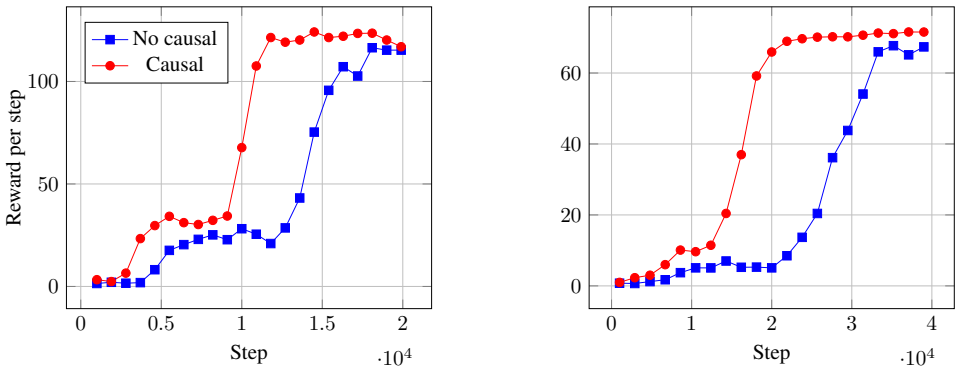
4 Case Studies

Our method shows promising results across two case studies. The first case study (results in Figure 5a) is based on the *coffee vs. soda* example from Figure 1. The second case study (results in Figure 5b) is described in Figure 6. We compared our method against QRM without access to knowledge about causality. In both case studies, our method takes significantly fewer steps to converge to the optimal policy.

Algorithm 1 Reinforcement Learning With Temporal-Causal Information

Require: MDP M, PRM A, minimal causal DFA C with rejecting sink states $Q_{\text{r.s.}}$

$\mathsf{B}_1, \mathsf{B}_2 = \text{computeProduct}(\mathsf{A}, \mathsf{C}), \text{computeProduct}(-\mathsf{A}, \mathsf{C})$
$v^\star_{\mathsf{B}_1}, v^\star_{\mathsf{B}_2} = \text{valueIteration}(\mathsf{B}_1, \gamma), \text{valueIteration}(\mathsf{B}_2, \gamma)$
$\mathsf{B} = \mathsf{B}_1$
for every state $u \in U^{\mathsf{B}}$ **do**
    if $v^\star_{\mathsf{B}_1}(u) = v^\star_{\mathsf{B}_2}(u) = 0$ **then**
        Add u to the set of terminal states of B
    end if
end for
$Q = \text{initializeQFunction}()$
while termination criteria not met **do**
    $Q = \text{RunQRMEpisode}(Q, \mathsf{B})$
end while
return Q



(a) Coffee vs. soda task

(b) Two-doors task

Figure 5: Reward per step averaged over 20 runs. "No causal" refers to using QRM with the original PRM that does not account for additional causal information in the environment. "Causal" are the results for our method. Both graphs showcase QRM convergence to the optimal policy.

5 Conclusion and Further Work

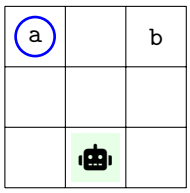
The method proposed in this paper addresses the difficult problem of accounting for knowledge about temporal causality in the RL environment. We have shown that an expressive and concise description of temporal and causal relations in the form of a Temporal Logic-based Causal Diagram can be integrated into the reward function formalism. Furthermore, we have shown how the added information about temporal and causal relations can be leveraged to expedite learning without changing the optimal policy.

While our method performs well in case studies, we are convinced that this work can be continued to integrate knowledge about causality even more tightly into the reward function. In particular, look-ahead information contained in state-values of the product PRM may be further utilized by methods like reward shaping. We are also interested in further exploring the interplay between probabilistic outcomes and causal information.

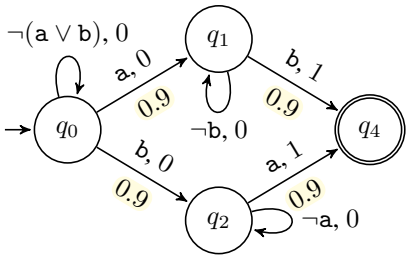
References

[1] Junzhe Zhang. Designing optimal dynamic treatment regimes: A causal reinforcement learning approach. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11012–11022. PMLR, 13–18 Jul 2020.

[2] Yangyi Lu, Amirhossein Meisami, and Ambuj Tewari. Causal Markov decision processes: Learning good interventions efficiently. *CoRR*, abs/2102.07663, 2021.



(a) 3x3 Gridworld environment where the agent must open both door A (a) and door B (b) in any order. However, the cell with door A traps the agent. The agent can fail at opening the doors with probability 0.1.



(b) The PRM without causal info about the two-door task. Missing transitions are all self-loops with probability 0.1.

Figure 6: The MDP and PRM for the second case study. The TL-CD that adds causal information regarding the sink door A can be found on Figure 2b. It states that after seeing door A, the agent can not later see door B.

[3] Lingxiao Wang, Zhuoran Yang, and Zhaoran Wang. Provably efficient causal reinforcement learning with confounded observational data. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 21164–21175, 2021.

[4] Elias Bareinboim, Andrew Forney, and Judea Pearl. Bandits with unobserved confounders: A causal approach. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[5] Sanghack Lee and Elias Bareinboim. Structural causal bandits: Where to intervene? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[6] Thomas Mesnard, Theophane Weber, Fabio Viola, Shantanu Thakoor, Alaa Saade, Anna Harutyunyan, Will Dabney, Thomas S. Stepleton, Nicolas Heess, Arthur Guez, Eric Moulines, Marcus Hutter, Lars Buesing, and Rémi Munos. Counterfactual credit assignment in model-free reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 7654–7664. PMLR, 2021.

[7] Andrew Forney, Judea Pearl, and Elias Bareinboim. Counterfactual data-fusion for online reinforcement learners. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1156–1164. PMLR, 06–11 Aug 2017.

[8] Jin Li, Ye Luo, and Xiaowei Zhang. Causal reinforcement learning: An instrumental variable approach. *ERN: Computational Techniques (Topic)*, 2021.

[9] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Anthony Valenzano, and Sheila A. McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *CoRR*, abs/2010.03950, 2020.

[10] Alvaro Velasquez, Andre Beckus, Taylor Dohmen, Ashutosh Trivedi, Noah Topper, and George K. Atia. Learning probabilistic reward machines from non-markovian stochastic reward processes. *CoRR*, abs/2107.04633, 2021.

[11] Yash Paliwal, Rajarshi Roy, Jean-Raphaël Gaglione, Nasim Baharisangari, Daniel Neider, Xiaoming Duan, Ufuk Topcu, and Zhe Xu. Reinforcement learning with temporal-logic-based causal diagrams. In Andreas Holzinger, Peter Kieseberg, Federico Cabitza, Andrea Campagner, A. Min Tjoa, and Edgar Weippl, editors, *Machine Learning and Knowledge Extraction*, pages 123–140, Cham, 2023. Springer Nature Switzerland.

[12] Zhe Xu, Ivan Gavran, Yousef Ahmad, Rupak Majumdar, Daniel Neider, Ufuk Topcu, and Bo Wu. Joint inference of reward machines and policies for reinforcement learning. *CoRR*, abs/1909.05912, 2019.

[13] Jan Corazza, Ivan Gavran, and Daniel Neider. Reinforcement learning with stochastic reward machines. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6):6429–6436, Jun. 2022.

[14] Michael Sipser. *Introduction to the Theory of Computation*. Course Technology, Boston, MA, third edition, 2013.

6 Appendix

Section 6.1 contains the algorithm for the `computeProduct` function used in Algorithm 1. In Section 6.2 we present two additional case studies.

6.1 Computing the PRM and causal DFA product

Algorithm 2 computes the (intermediate) product B_1 (B_2) of a PRM A and causal DFA C. This function is called in Line 1 of Algorithm 1 in Section 3. Note that this function does not perform value iteration, this is done in Line 2 of Algorithm 1. The set of terminal states in the intermediate product B_1 is a subset of the set of terminal states in the final product B.

The `transitions` dictionary used in Algorithm 2 represents the transition and output functions of B_1. It maps triplets $((u, q), \ell, (u', q'))$ to to pairs (p, r), where $p = \tau^{\mathsf{B}_1}((u, q), \ell, (u', q'))$, and $r = \sigma^{\mathsf{B}_1}((u, q), \ell, (u', q'))$.

Algorithm 2 computeProduct(A, C)

Require: PRM A, minimal causal DFA C with rejecting sink states $Q_{\text{r.s.}}$

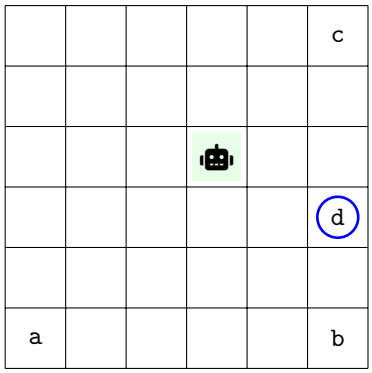
appears $\leftarrow$ A.appears $\cup$ C.appears ▷ M.appears is the set of relevant atomic propositions in M.
pairToSelfStateMap $\leftarrow \{\}$ ▷ Dictionary mapping pairs of states in A and C to a single state in B_1
selfToPairStateMap $\leftarrow \{\}$
nonTerminalStates $\leftarrow \emptyset$ ▷ Contains non-terminal states of the intermediate product PRM.
terminalStates $\leftarrow \emptyset$ ▷ Contains terminal states of the intermediate product PRM.
transitions $\leftarrow \{\}$ ▷ Dictionary representation of τ and σ functions of B_1.
stateCounter $\leftarrow 0$
for u in A.states **do**
    for q in C.states **do**
        pairToSelfStateMap$[(u, q)] \leftarrow$ stateCounter
        selfToPairStateMap[stateCounter] $\leftarrow (u, q)$
        if u in A.terminalStates **then**
            terminalStates $\leftarrow \{$stateCounter$\} \cup$ terminalStates
        else
            nonTerminalStates $\leftarrow \{$stateCounter$\} \cup$ nonTerminalStates
        end if
        stateCounter $\leftarrow$ stateCounter $+ 1$
    end for
end for
for u in A.nonTerminalStates **do**
    for q in C.states **do**
        state $\leftarrow$ pairToSelfStateMap$[(u, q)]$
        transitions[state] $\leftarrow \{\}$
        for inputSymbol in GENERATEINPUTS(appears) **do**
            inputSymbolPrm $\leftarrow$ A.appears $\cap$ inputSymbol
            inputSymbolDfa $\leftarrow$ C.appears $\cap$ inputSymbol
            transitions[state][inputSymbol] $\leftarrow \{\}$
            nextDfaState $\leftarrow$ C.transitions$[q]$[inputSymbolDfa]
            for nextPrmState in A.transitions$[u]$[inputSymbolPrm] **do**
                nextState $\leftarrow$ pairToSelfStateMap$[$(nextPrmState, nextDfaState)$]$
                probability, reward $\leftarrow$ A.transitions$[u]$[inputSymbolPrm][nextPrmState]
                if nextDfaState in $Q_{\text{r.s.}}$ **then**
                    reward $\leftarrow m$
                end if
                transitions[state][inputSymbol][nextState] $\leftarrow$ (probability, reward)
            end for
        end for
    end for
end for
return PRM(transitions, appears, terminalStates)

6.2 Additional Case Studies

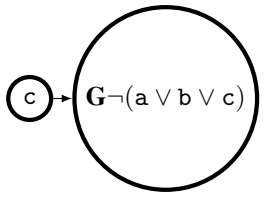
For a more thorough comparison and analysis of the method's efficiency, we implemented it across two distinct case studies: a four-door task and a small office world domain.

6.2.1 Four Door Case Study

The four door case study entails an agent navigating through a scenario where it must open four doors in any arbitrary order, as illustrated in Figure 7a. This task involves a significantly more complex PRM owing to the number of possible orders. To evaluate the method's performance and its efficacy in this case study, we use a grid world configuration of 6×6.



(a) 6x6 Gridworld environment where the agent must open door A (a), door B (b), door C (c), and door D (d) in any order. However, the cell with door D traps the agent. The agent may fail to open door B with a probability of 0.1, and it will go to door D instead.



$\mathbf{G}(\mathtt{d} \rightarrow \mathbf{G}\neg(\mathtt{a} \vee \mathtt{b} \vee \mathtt{c}))$

(b) TL-CD that holds for the case study of four-doors task.

Figure 7: The MDP and TL-CD for the third case study.

The agent here must open door A, door B, door C, and door D in any order. However, door D is a trap, and the agent cannot see doors A, B, or even C after seeing door D. This knowledge, in fact, is encoded in Figure 7b. As this task requires a complex Probabilistic Reward Machine (PRM), we deemed it prudent to relegate its detailed explanation to the Appendix.

Furthermore, Figure 8 compares our method on the four-doors task to QRM without additional causal information. It can be seen that QRM with causal information results in much higher rewards with faster convergence. We exclude the PRM from the Four Door Case Study because it is substantial in size.

6.2.2 Small Office World Case Study

Another case study in which we implemented this method is the small office world domain. For this specific exploration, we took into consideration a small office world with a spatial layout of 17×9, similar to the setup in [11]. Within the scope of this case study, the procedure to exit the grid entails a two-step process for the agent: first, it must obtain one of the two available keys, denoted as k_1 or k_2, and then navigate to exit e_1 or e_2, correspondingly aligned with the key acquired. Through one-way doors (indicated by blue arrows), keys, and walls, the agent interacts with the environment. A graphical illustration of this environment, capturing the elements and challenges the agent faces, is provided in Figure 9, providing a better understanding of the structural and operational complexities of the small office world being explored.

As a result of c being a one-way door, the agent will not be able to pick up key k_2 and exit at e_2, due to the information encoded in figure 10b. In addition, if the agent passes through the door b, it will not be able to exit through the door e_1. Furthermore, Figure 10a displays the PRM, omitting the causal information regarding the small office world. In order to succeed in exiting the maze and receiving reward 1, the agent must complete both sequences a$-k_1 - e_1$ (open door a, pick up key k_1,

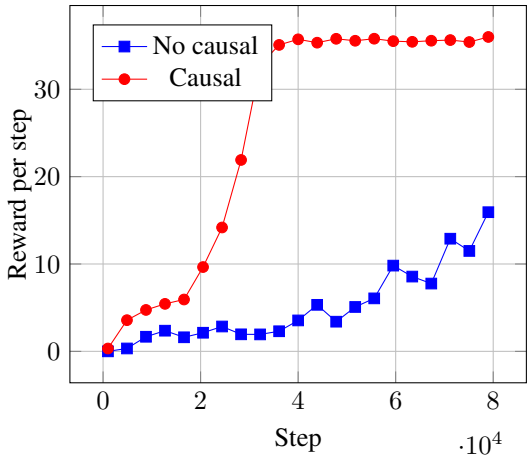


Figure 8: Four-doors task results, using the same reward per step metric averaged over 20 runs as in previous case studies.

and leave at e_1) or b$-k_2 - e_2$ (open door b, pick up key k_2, and exit at e_2). However, a probability of 0.9 suggests a likelihood of the agent exiting through e_1, while a probability of 0.1 indicates a risk of the agent getting stuck.

Figure 11 depicts the performance comparison of our method on the small office world scenario to QRM without additional causal information. In the figure, it can be seen that if the RL agent knows the causal DFA and learns never to open door b, the agent can obtain their optimal reward faster with higher accumulated rewards.

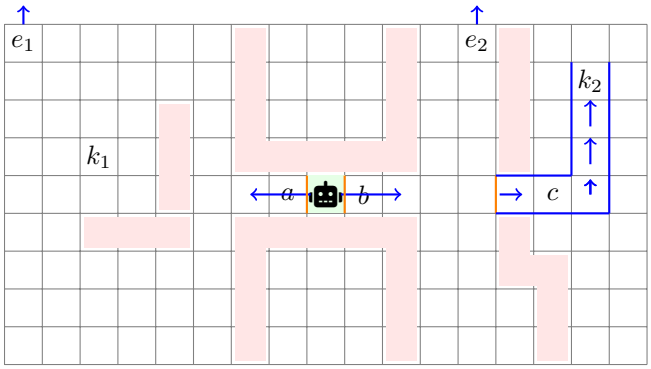
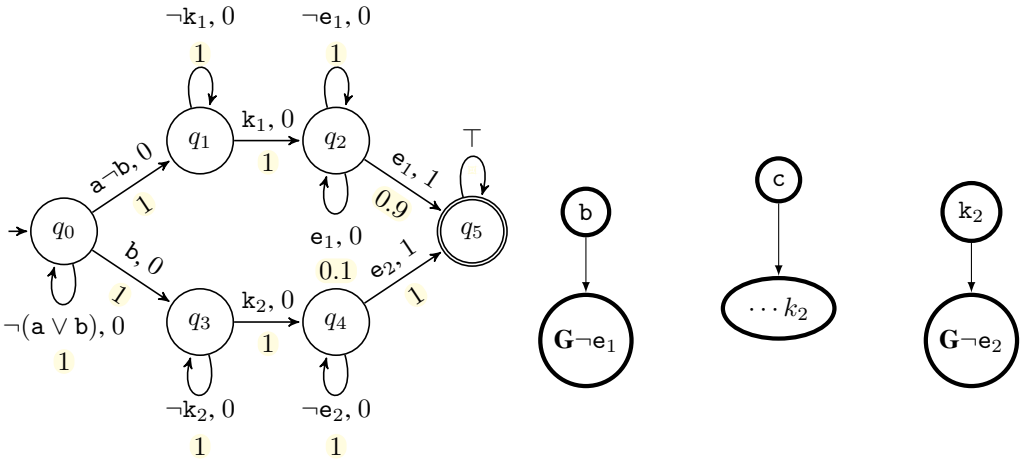


Figure 9: Map of the small office world. Shaded cells are impassable walls



(a) The PRM without causal info about the small office world

(b) TL-CD that holds for the case study of small office.

Figure 10: The PRM and TL-CD for the fourth case study.

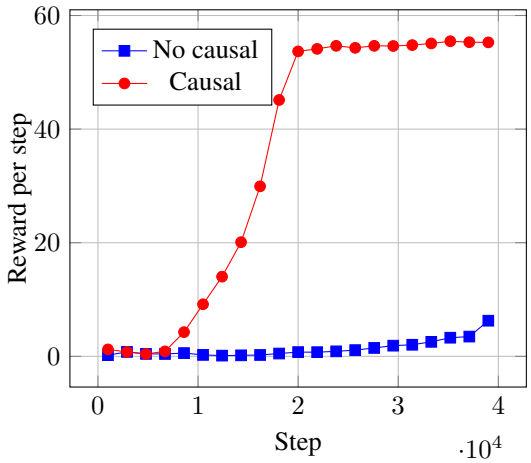


Figure 11: Small office world results, using the same reward per step metric averaged over 20 runs as in previous case studies.