Non-Markovian Rewards Expressed in LTL: Guiding Search Via Reward Shaping (Extended Version)

Alberto Camacho [1] **Oscar Chen** [1] **Scott Sanner** [2] **Sheila A. McIlraith** [1]

Abstract

In many decision-making settings, reward is acquired in response to some complex behaviour that an agent realizes over time. An assistive robot receiving reward for ensuring that its patient takes their medication once daily soon after eating, is one such example. Reward of this sort is referred to as *non-Markovian* because it is predicated on state history rather than solely on the current state. Our concern in this paper is with Non-Markovian Decision Processes and in particular with the specification and effective exploitation of non-Markovian reward. To this end, we outline a means of specifying non-Markovian reward, expressed in Linear Temporal Logic (LTL) interpreted over finite traces. Central to our approach is the transformation of the reward specification language, here $\mathsf{LTL_f}$, into deterministic finite state automata. MDP planners based on heuristic-search and UCT struggle with non-Markovian rewards which provide little guidance to the relatively myopic lookahead of these solvers. Here we explore the use of *reward shaping* to automatically reshape the automata-based reward. These reshaped, automata-based rewards can be exploited by off-the-shelf MDP planners to guide search, while crucially preserving policy optimality guarantees. Experiments with augmented International Probabilistic Planning Competition domains demonstrate significantly improved performance via the exploitation of our techniques. The work presented here uses $\mathsf{LTL_f}$ to specify non-Markovian reward, but our approach will work for any formal language for which there is a corresponding automata representation.

[1]Department of Computer Science, University of Toronto, Toronto, Canada [2]Department of Mechanical Engineering, University of Toronto, Toronto, Canada. Correspondence to: Alberto Camacho <acamacho@cs.toronto.edu>.

Accepted at the 1st Workshop on Goal Specifications for Reinforcement Learning, FAIM 2018, Stockholm, Sweden, 2018.

Preamble

This work illustrates how non-Markovian reward functions can be translated to an automata representation and how reward shaping can be used to guide search in sequential decision-making when reward functions are expressed via automata. While this paper utilizes Linear Temporal Logic interpreted over finite traces ($\mathsf{LTL_f}$) to specify non-Markovian rewards, subsequently translating the LTL into automata, our approach is applicable to any formal language for which there is a corresponding automata representation. This includes a number of languages that have been used to express (temporally extended) goals and preferences for AI automated planning. Examples of such languages include Past LTL (PLTL) (Bacchus et al., 1996; Sohrabi et al., 2011); Golog, which provides procedural programming language constructs augmented with logical expressions (Baier et al., 2008; 2007; Fritz et al., 2008; Bienvenu et al., 2011); $\mathsf{LDL_f}$ (Brafman et al., 2018); and Hierarchical Task Networks (HTN) (Fritz et al., 2008). All of these languages can similarly be used to specify goals for reinforcement learning. There are merits and shortcomings to each of these approaches with respect to ease of specification of rewards. Toro Icarte et al. (2018) recently showed how to exploit exposed structure for so-called Reward Machines – automata-based reward functions – to improve the performance of tabular and deep reinforcement learning systems.

An extended abstract of this work appeared in the proceedings of the International Symposium on Optimization and Combinatorial Search (SoCS 2017) and was presented at RLDM-17 (Camacho et al., 2017b;a) accompanied by a technical report (Camacho et al., 2017c).

1 Introduction

In Markov Decision Processes agents typically receive positive or negative reward in response to their current state. Nevertheless, agents may also realize reward in response to more complex behaviour that is reflected over a sequence of states. For example, an autonomous electric vehicle may acquire reward for always recharging its battery after a trip. Similarly, a personal robot may acquire reward by opening the refrigerator, removing a prescribed item, and closing

the refrigerator immediately thereafter. Such reward is commonly referred to as non-Markovian reward because it is predicated on the state history rather than solely on the current state. Our concern in this paper is with both the specification and effective exploitation of non-Markovian reward in Markov Decision Processes (MDPs). Here we use Linear Temporal Logic intepreted over finite traces ($\mathsf{LTL_f}$) to specify non-Markovian rewards. Notwithstanding, our approach is applicable to other formal languages for which there exist corresponding automata representations.

Current state-of-the-art MDP planners are based on heuristic search and variants of UCT techniques (Kocsis & Szepesvári, 2006). UCT policies tend to make greedy and myopic decisions. As such, these planners struggle with non-Markovian rewards since there is little guidance for their relatively myopic lookahead. The impact of this myopic guidance can be seen in state-of-the-art MDP planner PROST (Keller & Eyerich, 2012), a UCT-based planner that generates high-quality solutions for moderately sized MDPs, but whose performance suffers in large problems that require significant lookahead.

In this paper we explore transformation of the reward function through reward shaping (Ng et al., 1999) as a means of mitigating for the myopic lookahead of UCT-based methods. To this end, we propose an approach to solving non-Markovian Reward Decision Problems (NMRDPs) by transforming our reward-worthy non-Markovian behaviour into corresponding deterministic finite state automata. The accepting conditions of these automata signify satisfaction of the reward-inducing behaviour in a manner that is solvable with off-the-shelf MDP planners, crucially preserving optimality guarantees. Moreover, we use reward shaping with these automata-based reward encodings in order to induce non-sparse, myopic-friendly rewards. This helps guide the accrual of non-Markovian reward. We evaluate our approach to solving NMRDPs via experimentation with off-the-shelf state-of-the-art heuristic and UCT-based MDP planners. Experiments with a set of International Probabilistic Planning Competition (IPPC) domains augmented with non-Markovian rewards show significantly improved performance using our automata representation together with reward shaping.

2 Background

2.1 Markov Decision Processes

Markov Decision Processes (MDPs) (Puterman, 1994) are popular models for decision-theoretic planning problems (Boutilier et al., 1999). An MDP is a tuple $M = \langle S, A, P, R, T, \gamma, s_0 \rangle$, where: S is a finite set of states; A is a finite set of actions; $P_a(s, s')$ is the probability of reaching the state $s' \in S$ after applying action a in state $s \in S$; $R : S \times A \times S \to \mathbb{R}$ is the reward function (sometimes $R : S \times A \to \mathbb{R}$); $T \in \mathbb{N}$ is the horizon; $\gamma \in (0, 1]$ is the discount factor; and $s_0 \in S$ is the *initial state* of the MDP.

Solutions to an MDP are a sequence of step-dependent *policies* $\Pi = (\pi_0, \ldots, \pi_{T-1})$ that map states $s \in S$ at step k ($0 \leq k < T$) to actions $\pi_k(s) \in A$. The *value* of a policy Π in state s at step k, $V_{\Pi,k}(s)$, is the expected discounted cumulative reward over the horizon $T - k$ following Π. Formally, $V_{\Pi,k}(s) = \mathbb{E}_\Pi\{\Sigma_{i=k}^{T-1} \gamma^i R_i\}$, where R_t denotes the immediate reward obtained at step i if the agent follows policy Π from s. An optimal policy sequence Π^* for an MDP over horizon T with initial state s_0 satisfies $\Pi^* = \text{argmax}_\Pi V_{\Pi,0}(s_0)$.

MDPs are commonly described using factored representations of the states and dynamics. In particular, RDDL (Sanner, 2010) is a modelling language that allows for a lifted, compact representation of factored MDPs. States are given by assignments to all ground state fluents of the RDDL specification, and transition dynamics are described by conditional probabilities that can be expressed as a stochastic form of *successor state axioms* (Reiter, 1991). Intuitively, the updates on the truth of each ground fluent are described with respect to what holds in the current state and the actions that are performed. Modern algorithms for MDPs are typically based on UCT-search techniques that sacrifice optimality in favor of scalability. The current state-of-the-art solution method for MDPs is PROST (Keller & Eyerich, 2012), a Monte-Carlo sampling algorithm based on UCT and heuristic search for finite-horizon MDPs. Whereas PROST generates good-quality solutions to moderate-sized MDPs, its performance suffers in large problems that require a significant look-ahead. In such cases, the Monte-Carlo roll-outs cannot sufficiently capture the structures inherent in the problem, leading to greedy/myopic search behavior.

2.2 Linear Temporal Logic over Finite Traces

Linear Temporal Logic (LTL) is a compelling language for expressing temporal properties over (infinite) sequences of states. It is a propositional modal logic with modalities referring to time (Pnueli, 1977). $\mathsf{LTL_f}$ has essentially the same syntax as LTL but is interpreted over finite traces. In particular, given a set of propositional symbols, P, $\mathsf{LTL_f}$ formula φ is defined as follows:

$$\varphi := \top \mid \bot \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2$$

where $p \in P$, and $\bigcirc$ (*next*) and $\mathcal{U}$ (*until*) are temporal operators. Intuitively, *next* specifies what needs to hold in the next time step, and *until* specifies what needs to hold at least until something else holds. Other temporal operators such as *eventually* ($\Diamond$), *always* ($\Box$), and *release* ($\mathcal{R}$) are defined by the standard equivalences: $\Diamond\varphi \equiv \top \,\mathcal{U}\, \varphi$, $\Box\varphi \equiv \neg\Diamond\neg\varphi$, and $\varphi_1 \,\mathcal{R}\, \varphi_2 \equiv \neg(\neg\varphi_1 \,\mathcal{U}\, \neg\varphi_2)$. $\mathsf{LTL_f}$ also has

a *weak-next* operator ($\bullet$), defined by $\bullet\varphi \equiv \neg\bigcirc\neg\varphi$, that tells that φ needs to hold in the next time step if such next time step exists. Note that $\neg\bigcirc\varphi \not\equiv \bigcirc\neg\varphi$ (De Giacomo & Vardi, 2013). We further extend the syntax of $\mathsf{LTL_f}$ with the modality **final** $\equiv \neg\bigcirc\top$ to specify properties that hold in the final state of the trace.

Within the automated planning literature, the study of LTL interpreted over *finite* traces dates back at least to the work on specifying temporally extended goals and preferences (e.g., (Bacchus & Kabanza, 2000; Baier & McIlraith, 2006; Baier et al., 2009)), and a fragment was incorporated into PDDL 3.0 in 2006 (Gerevini et al., 2009). Recent work uses $\mathsf{LTL_f}$ (De Giacomo & Vardi, 2013) as a specification language for synthesis (e.g. (De Giacomo & Vardi, 2015; Camacho et al., 2018)).

$\mathsf{LTL_f}$ formulae are interpreted over finite traces of propositional states, $\sigma = s_0 \cdots s_n$, where each s_i is a set of propositions from P that are true in s_i. We say that σ *satisfies* $\mathsf{LTL_f}$ formula φ, denoted $\sigma \models \varphi$, when $\sigma, 0 \models \varphi$, where:

- $\sigma, i \models p$, for each $p \in P \cup \{\top\}$ iff $s_i \models p$.
- $\sigma, i \models \neg\varphi$ iff $\sigma, i \models \varphi$ does not hold.
- $\sigma, i \models \varphi_1 \wedge \varphi_2$ iff $\sigma, i \models \varphi_1$ and $\sigma, i \models \varphi_2$.
- $\sigma, i \models \bigcirc\varphi$ iff $i < n$ and $\sigma, (i+1) \models \varphi$.
- $\sigma, i \models \varphi_1 \,\mathcal{U}\, \varphi_2$ iff there exists a $i \leq j \leq n$ such that $\sigma, j \models \varphi_2$, and $\sigma, k \models \varphi_1$, for each $i \leq k < j$.

The interpretation of other modal operators such as *eventually* and *always* follow from these definitions.

3 Non-Markovian Reward

Non-Markovian Reward Decision Processes (NMRDPs) (e.g., (Bacchus et al., 1996; 1997; Thiébaux et al., 2006)) generalize the MDP model by allowing reward functions to range over the history of visited states. In contrast to MDPs, the domain of the reward function R ranges over the set of finite state-sequences drawn from S, denoted $S^\star$. As in conventional MDPs, optimal solutions maximize the expected discounted cumulative reward. Adapted from (Bacchus et al., 1996):

Definition 1 (NMRDP)**.** *A Non-Markovian Reward Decision Process (NMRDP) is a tuple $M = \langle S, A, P, R, T, \gamma, s_0\rangle$, where S, A, P, T, γ, and s_0 are as defined in MDPs: S is a finite set of states; A is a finite set of actions; $P_a(s, s')$ is the probability of reaching the state $s' \in S$ after applying action a in state $s \in S$; $T \in \mathbb{N}$ is the horizon; $\gamma \in (0, 1]$ is the discount factor; and $s_0 \in S$ is the* initial state *of the NMRDP. In contrast to MDPs, the reward function is $R : S^* \to \mathbb{R}$ (sometimes $R : (S \times A)^* \to \mathbb{R}$).*

3.1 Temporally-Extended Reward Function

Following Bacchus et al. (1996), non-Markovian reward in an NMRDP M is described in terms of a finite set of temporally extended reward formulae, φ_i, each with an associated reward r_i, resulting in tuples $(\varphi_i : r_i)$. In this paper, we specify such reward formulae – the reward worthy temporally extended behaviour – using $\mathsf{LTL_f}$, following (Camacho et al., 2017b). Previous approaches to NMRDPs have used Past LTL (PLTL), a dialect of LTL whose temporal modalities refer to past events (Bacchus et al., 1996; 1997), and finitely interpreted future LTL, \$FLTL (Thiébaux et al., 2006) to describe temporally extended reward formulae. Recently Littman et al. (2017) devised a geometric variant of LTL, GLTL, that appeals to a geometric distribution on the temporal extent of formulae, mimicking the effect of discounting (2017). Brafman et al. (2018) proposed $\mathsf{LDL_f}$ to specify temporally extended reward formulae.

Example: By way of illustration, consider an assistive robot that accumulates reward by ensuring that its ward takes their medication daily and that they do so after eating lunch. Such behaviour might be expressed by the $\mathsf{LTL_f}$ formula:
$\varphi := (\Diamond ingested(medication))$
$\quad \wedge\ (\neg ingested(medication)\,\mathcal{U}\, ingested(lunch))$.
Associated with this reward-worthy behaviour is a reward of 100. A robot would have numerous $(\varphi_i : r_i)$ pairs.

Adapted from Bacchus et al. (1996), a *temporally-extended reward function* (TERF) is in turn defined in terms of a finite set of tuples $\{(\varphi_i : r_i)\}_{1 \leq i \leq m}$. Each φ_i is a formula that describes a behaviour, and $r_i \in \mathbb{R}$ is the reward given to the agent when its trajectory $\Gamma \in S^*$ satisfies φ_i (denoted $\Gamma \models \varphi_i$). Formally, a TERF R is defined by $R(\Gamma) = \Sigma_{i=1}^{m} R_i(\Gamma)$, where $R_i(\Gamma) = r_i$ if $\Gamma \models \varphi_i$, and $R_i(\Gamma) = 0$ otherwise.

TERFs can be specified in a variety of languages. Here we illustrate the specification of TERFS using $\mathsf{LTL_f}$ but the approach to solving NMRDPs proposed in this paper holds for any language that can be compiled into DFA. This includes PLTL, a subset of Golog, and $\mathsf{LDL_f}$ formulae. Temporally extended goals compliant with the PDDL3 standard (Gerevini et al., 2009) are also compilable into DFAs. We also support rewards specified directly in terms of NFAs and DFAs (c.f. (Toro Icarte et al., 2018)).

Example (cont.): Returning to our example, we can define a TERF with the tuple $(\varphi : 100)$, that gives a positive reward of 100 to temporally extended behaviour φ, as above. Note that $(\varphi : 100)$ rewards all sequences of states for which there is a prefix that satisfies φ. To only reward the first occurrence of the behaviour within a sequence of states, one could modify the above $\mathsf{LTL_f}$ formula as follows: $(\neg ingest(medication)\,\mathcal{U}\,(ingest(medication) \wedge \neg\bigcirc\top)) \wedge (\neg ingest(medication)\,\mathcal{U}\, ingest(lunch))$.

4 Solving NMRDPs via Automata

Bacchus et al. (1996)'s solved NMRDPs via a state-based construction method that kept track of the state history, but suffered from state space explosion. Bacchus et al. (1997) provided a method for translating NMRDPs to a structured-MDP representation that mitigated for this, but was only amenable to structural policy construction methods. Further work by Thiébaux et al. (2006), which is the current state-of-the-art method, introduced a framework for solving NMRDPs using anytime state-based heuristic search methods. These methods had the benefit of exploring a significantly reduced fraction of the state space at the cost of producing possibly sub-optimal policies.

4.1 Compiling NMRDP into Automata-Based MDPs

Our approach to solve an NMRDP, M, with TERF, R, transforms the problem into a standard MDP,M', (with Markovian reward) that can be solved with conventional MDP solvers. Key to our approach is the translation of temporally extended reward formulae into deterministic finite state automata (DFAs). The approach comprises four steps.

Step 1 Transform each φ_i in the TERF into a DFA

Step 2 Construct an MDP M' from M and the DFAs

Step 3 Solve M' – using any off-the-shelf MDP planner

Step 4 Extract solution to M from a solution to M'

In what follows, we elaborate on each step. For the purposes of this paper, we limit our explication to finite-horizon NMRDPs. Notwithstanding, our approach can be extended to infinite-horizon NMRDPs.

4.2 Transforming TERFs into Automata Functions

A *deterministic finite-state automaton* (DFA) is a tuple $\langle Q, \Sigma, \delta, q_0, Q_{Fin}\rangle$, where Q is a finite set of states, Σ is the *alphabet* of the automaton, $\delta : Q \times \Sigma \to Q$ is a transition function, $q_0 \in Q$ is the initial state, and $Q_{Fin} \subseteq Q$ is a set of accepting states. The transition dynamics of a DFA is defined over *finite words*, or sequences $w = s_0, s_1, \ldots, s_n$ of elements in Σ. Here, Σ are the states of an MDP. At every stage i, the automaton makes a deterministic transition from state q_i to state $q_{i+1} = \delta(q_i, s_i)$. The *guard* of a transition from q to q', denoted by $guard(q \to q')$, is a propositional formula so that $q' = \delta(q, s)$ iff $s \models guard(q \to q')$. We say that A accepts w if $q_n \in Q_{Fin}$. We say that M accepts w if $q_{n+1} \in Q_{Fin}$.

In Step 1 of our approach, we transform each tuple $(\varphi : r)$ in the TERF into a tuple $(A_\varphi : r)$, where A_φ is a DFA that accepts a word π iff it satisfies φ. The translation of $\mathsf{LTL_f}$ formula φ to a corresponding DFA typically involves conversion to a non-deterministic automaton followed by determinization (e.g. (Baier & McIlraith, 2006; De Giacomo & Vardi, 2015)). Other dialects of LTL can be transformed into DFA. In particular, Sohrabi et al. (2011) showed a method to transform PLTL formulae into an NBA (that can also be determinized to a DFA), and all basic temporal operators in PDDL3 have automaton representations (cf. (Gerevini et al., 2009)). Some fragments of LTL, such as safe and co-safe formulae, can be transformed into DFA (c.f. (Kupferman & Vardi, 2001)).

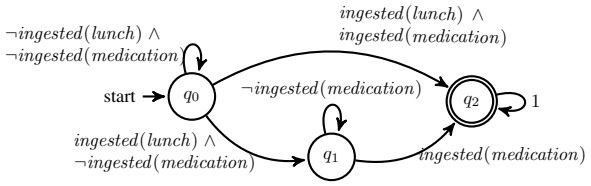

Figure 1. DFA for $\mathsf{LTL_f}$ formula $(\Diamond ingested(medication)) \wedge (\neg ingested(medication)\, \mathcal{U}\, ingested(lunch))$.

Example (cont.): Figure 1 depicts a DFA corresponding to $\mathsf{LTL_f}$ formula $\varphi := \Diamond ingested(medication) \wedge (\neg ingested(medication)\, \mathcal{U}\, ingested(lunch))$. Automaton states are represented by nodes, and transitions are represented by arcs. Transition labels describe the guards. Finally, accepting states are depicted by double-ringed nodes. The word $\pi = \{s_0, s_1, s_2\}$ with $s_1 \models \{\neg ingested(lunch) \wedge \neg ingested(medication)\}$, $s_2 \models \{ingested(lunch) \wedge \neg ingested(medication)\}$, and $s_3 \models \{ingested(lunch) \wedge ingested(medication)\}$ induces one and only one run in the automaton, $\{q_0, q_0, q_1, q_2\}$. As this run finishes in an accepting state, it follows that π satisfies the $\mathsf{LTL_f}$ formula.

4.3 Compiling NMRDPs to MDPs

In Step, we construct an MDP, M', by augmenting M with extra fluents and actions that integrate the dynamics of the DFAs within the MDP, making it possible for the reward function to be Markovian. The dynamics of M' expand *each* time step into three modes: **world**, **sync**, and **reward**. In **world** mode, an action from the NMRDP is applied. In **sync** mode, the automata states are synchronized according to the observed state. Intuitively, the automata states simulate the runs of the automata given the observed **world** state trajectories. The assignment of reward is delayed to **reward** mode and is performed upon satisfaction of each of the $\mathsf{LTL_f}$ reward formulae in the TERF. This is detected when an automaton reaches an accepting state. Table 1 contains technical details of the compilation. We provide a detailed account of the compilation below.

M' has the same fluents as M, plus the auxiliary fluents described below. The fluents $\mathbf{world}, \mathbf{sync}, \mathbf{reward}$ control the dynamics of the problem, forcing an alternation between three different modes: **world** mode, **sync** mode, and **reward** mode. For each automaton A_φ, and for each

	Original NMRDP	Compiled MDP
Initial state	s_0	$s'_0 := s_0 \cup \bigcup_{(\varphi:r)\in R}\{f_q \mid q \text{ initial state of } A_\varphi\} \cup \{\mathbf{sync}\}$
Successor state axioms	$p' \leftrightarrow (\phi_p^+) \vee (p \wedge \neg\phi_p^-)$	$p' \leftrightarrow (\phi_p^+ \wedge \mathbf{world}) \vee (p \wedge (\neg\mathbf{world} \vee (\neg\phi_p^- \wedge \mathbf{world}))$ $\mathbf{world}' \leftrightarrow \mathbf{reward}$ $\mathbf{sync}' \leftrightarrow \mathbf{world}$ $\mathbf{reward}' \leftrightarrow \mathbf{sync}$ $f_q \leftrightarrow (\mathbf{sync} \wedge \phi_q^+) \vee (f_q \wedge \neg\mathbf{sync})$
Reward function	$R = \{(\varphi_i : r_i)\}_{i=1..m}$	$\forall s$ such that $s \models \mathbf{reward}$, $R'(s) := \sum_{(\varphi:r)\in R} \sum_{q\in A_\varphi,\ q \text{ accepting}} r_i \mathbb{1}_{f_q}(s)$
Discound factor	γ	$\gamma' := \gamma^{1/3}$
Horizon	T	$T' := 3T$

Table 1. Dynamics of the compiled MDP in terms of the dynamics of the original NMRDP. Here, $p \in F$ are propositional variables and $q \in A_\varphi$ are automaton states for each pair $(\varphi : r)$ in R. The successor state axioms for a predicate fluent p are described in the form $p' \leftrightarrow \phi$, where ϕ are the conditions that make p true at time $t+1$ as a function of the values of fluents at time t. The function ϕ_p^+ (resp. ϕ_p^-) is a propositional formula that describes the conditions under which p is made true (resp. false). Likewise, $\phi_q^+ = \bigvee_{q'\in A_\varphi} guard(q' \to q) \wedge f_{q'}$ describes the conditions under which f_q is made true. In the definition of R', $\mathbb{1}_{f_q}(s)$ is the indicator function that evaluates to 1 when f_q holds in s, and 0 otherwise.

automaton state $q \in A_\varphi$, M' has fluents f_q. Intuitively, each fluent f_q simulates an automaton state q. The set of actions in M' contains the set of actions A in M. For simplicity, and without loss of generality, we assume A contains an action *no-op* – denoting that the agent performs no explicit action – and an action o_{end} that the agent is required to execute at the end of the search horizon.

In **world** mode, execution of actions from A emulates the stochastic transition model of the actions in the original NMRDP. After an action is applied in **world** mode, the dynamics of the MDP switches to **sync** mode. In **sync** mode, the truth of the automaton state fluents f_q is updated to simulate the state transition of each automaton A_ϕ with respect to the current state of the MDP. After an action is applied in **sync** mode, the dynamics of the MDP switches to **reward** mode. In **reward** mode, the agent collects reward upon satisfaction of each of the $\mathsf{LTL_f}$ reward formulae in the simulated NMRDP. More precisely, for each mapping $\varphi : r$ in the TERF, a reward r is given to the agent in state s when f_q holds in s for some accepting automaton state $q \in A_\varphi$. The resulting reward is Markovian, and is formalized by R' in Table 1. Without loss of generality, we assume that the agent is forced to perform o_{end} in the last **reward** mode (i.e., in the last turn) before reaching the search horizon, T'.

The horizon in M', T', is three times the horizon T in the original NMRDP M. This is because each step in M has three counterparts in M', corresponding to the **world**, **sync**, and **reward** modes. Likewise, the discount factor of M' is $\gamma' = \gamma^{1/3}$. Finally, the initial state s'_0 of M' has all the fluents in the initial state s_0 of M, plus fluents f_q for the initial automaton state q of each automaton A_φ, and fluent **sync** that forces the agent to start in **sync** mode.

The size of the compiled MDP M' is polynomial in the size of the DFAs from the TERF (Theorem 1). When the TERF is given in terms of $\mathsf{LTL_f}$ formulae, the size of M' is worst-case double-exponential in the size of the formulae.

Theorem 1. *The size of the compiled MDP is polynomial in the size of the DFAs.*

Proof sketch. The set of ground fluents in the compiled MDP is augmented with automaton state fluents f_q and f_q^c, one for each automaton state. Other fluents and parametrized actions in the compiled MDP are bounded in size. □

Corollary 1. *The size of the compiled MDP is worst-case double-exponential in the size of the* $\mathsf{LTL_f}$ *formulae.*

Proof sketch. It is well-known that $\mathsf{LTL_f}$ formulae can be transformed into DFAs that are worst-case double-exponential in the size of the formula. □

4.4 Solution and Optimality Preservation

The compilation described above transforms a finite-horizon NMRDP M into a finite-horizon MDP M' that preserves optimal, and near-optimal solutions. A key aspect to understand this property is to realize that the stochastic transition model in M' in **world** mode simulates the transitions in M, whereas automata transitions simulated in **sync** mode are deterministic. As such, there exists a correspondence between finite state-action trajectories in M and M'. For a state-action trajectory $\Gamma = s_0, a_1, s_1, a_2, \ldots, s_n$ in M one can construct a state-action trajectory in M' by interleaving synchronization and reward state-actions between each a_i. Conversely, for each state-action trajectory Γ' in M' one can construct Γ by removing synchronization and reward state-actions from Γ'. The association described above defines a correspondence between policies Π in M and policies Π' in M'. It is straightforward to see that $V_{\Pi,0}(s_0) = V_{\Pi',0}(s'_0)$.

Consequently, the compilation from NMRDP into MDPs preserves optimal, and near-optimal solutions.

Theorem 2. *The automata-based compilation from NMRDP to MDP preserves optimal and near-optimal solutions.*

Proof sketch. Follows from the observation above, establishing the correspondence between state-action sequences in M and M'. □

Example (cont.): Returning to our example with TERF, $\{(\varphi : 100)\}$, suppose the agent performs action $ingest(lunch)$ followed by $ingest(medication)$, inducing the state trajectory (only relevant subset of state shown):

$$\pi = \{\neg ingested(lunch), \neg ingested(medication);$$
$$ingested(lunch), \neg ingested(medication);$$
$$ingested(lunch), ingested(medication)\}$$

The dynamics in the compiled MDP start by processing the initial state, and self-transitioning from the automaton state q_0 to itself. In **reward** mode, no reward is given. Then, in **world** mode the action $ingest(lunch)$ is performed, leading to a state s_1 in which $\{\neg ingested(medication), ingested(lunch)\}$ holds. The following **sync** mode synchronizes the automaton state to q_1, and so on until reaching world state s_2, where $\{ingested(medication), ingested(lunch)\}$ holds. At this point, the automaton synchronizes to state q_3, that is accepting. In **reward** mode, a reward of 100 is given.

5 Reward Shaping to Improve Performance

The approach to solving NMRDPs presented in Section 4 preserves optimality (cf. Theorem 2). Here we augment our approach with reward shaping in an effort to mitigate for the sparse reward inherent in our non-Markovian rewards, which aggravates the weak guidance and lookahead of state-of-the-art UCT-based MDP planners.

In particular, state-of-the-art MDP planner PROST (Keller & Eyerich, 2012) suffers from such myopia. PROST expands a search tree by favoring the branches that achieve the most immediate reward. An estimation of the expected reward is computed in a leaf node by either (i) throwing a Monte-Carlo (MC) roll-out, (ii) running iterative deepening search (IDS) with a bounded horizon, or (iii) running depth-first search (DFS) with a bounded horizon. When the rewards are sparse, neither of the methods below provides good guidance. MC roll-outs make random moves that collect little non-Markovian reward. IDS does not collect reward most of the time, and it does not scale with long look-ahead horizon. Similarly, DFS is blind most of the time.

5.1 Reward Shaping

Reward shaping is a common technique in MDPs which aims to improve search by transforming the reward function. Such reward transformations have the form $R'(s, a, s') = R(s, a, s') + F(s, a, s')$, where R is the original reward function and F is a *shaping* reward function. The intuition behind reward shaping is that by increasing (resp. decreasing) the reward in states that lead to other high-value states or trajectories (resp. low-value states or trajectories), we can increase the effectiveness of search and the quality of solutions found, while reducing search memory and run times. Unfortunately, reward shaping with an arbitrary $F(s, a, s')$ may lead to an optimal policy that is suboptimal w.r.t. the original unshaped reward. However, as noted by (Ng et al., 1999), if $F(s, a, s')$ is chosen from a restricted class of potential-based reward shaping functions defined as $F(s, a, s') = \gamma\phi(s') - \phi(s)$ (for some real-valued function ϕ), then this guarantees preservation of optimal and near-optimal policies with respect to the original unshaped MDP. Preservation of near-optimality is desirable since it provides guarantees for suboptimal solutions obtained by state-of-the-art heuristic search approximate methods.

Theorem 3 ((Ng et al., 1999))**.** *Potential-based MDP reward shaping preserve optimal, and near-optimal solutions.*

5.2 Automata-based Potentials

Reward shaping can be applied to the compiled MDP, as with regular MDPs, with the aim of providing better guidance to MDP solvers. The particular structure of the compiled MDPs, where automata fluents capture relevant historical information, suggest that we can exploit automata to design a class of shaping rewards that provide effective guidance by exploiting automata. In this section, we introduce a class of automata-based shaping reward functions that is, by construction, potential based. In Section 6, we conduct a preliminary evaluation of this class of potentials.

We augment the MDP with additional fluents, f_q^c, one for each automaton fluent f_q. These fluents keep track of the previous configuration of the automata. Its value is updated in **sync** mode according to the successor state axioms:

$$f_q^{c\prime} \leftrightarrow (\mathbf{sync} \wedge f_q) \vee (f_q^c \wedge \neg \mathbf{sync})$$

Formally, our shaping reward function is defined over states s in **reward** mode (i.e. $s \models \mathbf{reward}$) as follows:

$$F(s, a) = \gamma \sum_{f_q} \phi(f_q) - \sum_{f_q^c} \phi(f_q^c)$$

for all actions $a \in A \setminus \{o_{end}\}$, and f_q and f_q^c that hold in s. Unlike Ng et al. (1999)'s method, our shaping function F does not depend on two consecutive states in the MDP, but on the two states visited *in the last two* **world** *modes*. Intuitively, F emulates a potential-based reward transformation of the form $F(s, a, s') = \gamma\phi(s') - \phi(s)$ (i.e., as defined by Ng et al.) in the original NMRDP. The important thing to notice here is that, in the compiled MDP, the application of

F is *delayed* to the **reward** mode. This is because many off-the-shelf MDP planners employ reward functions of the form $R(s, a)$, rather than the more general $R(s, a, s')$.

The potential function $\phi(s)$ decomposes into a sum of potential functions evaluated on the automaton states that hold true in state s. More precisely, the potential function has the form $\phi(s) = \sum_{f_q \in s} \phi(f_q)$. Automaton state copies f_q^c make it possible to evaluate the potential in the previous **world** mode state. In order to preserve optimality (and near-optimality) of solutions to finite-horizon NMRDPs, we need to substract the shaping rewards at the end of each finite execution trace. This is performed upon application of o_{end} action in **reward** mode, by extending the domain of F to states $s \models \mathbf{reward}$ as follows:

$$F(s, o_{end}) = -\sum_{f_q^c} \phi(f_q^c)$$

for all f_q^c that hold in s.

Theorem 4. *Automata-based reward shaping preserves optimal, and near-optimal solutions.*

Example (cont.): In our robot example, we may want to provide guidance by assigning potentials $\phi(q_0) = 0$, $\phi(q_1) = 50$, and $\phi(q_2) = 100$. Intuitively, these potentials assign positive reward for transitioning from q_0 to q_1, with the rationale that state trajectories that yield such transitions make progress towards achievement of an accepting state.

In what follows, we introduce different criteria to engineer potential functions. It is not our purpose to give an exhaustive list, nor to study theoretical guarantees. Rather, our purpose is to inspire the reader about the number of ways that reward shaping can be used to improve guidance in NMRDPs. In Section 6, we prove empirically that even naive potential functions can successfully guide search and result in significant improvements in terms of the quality of the solutions found by approximate methods.

Liveness Preservation: It is possible to incent exploration of those states for which satisfaction of the reward formula φ is, in principle, still reachable. This is the case for non-Markovian liveness behaviors where we can assign $\phi(f_q) = 0$ to non-accepting sink automaton states $q \in A_\varphi$, and $\phi(f_q) = c$ otherwise, with $c \in \mathbb{R}^+$.

Heuristic Guidance: In order to incentive the search process towards satisfaction of the TERFs – and, therefore, collect reward –, we can distribute the potentials $\phi(f_q)$ in a way that they increase monotonically w.r.t. the inverse of *a distance measure* between q and the set of accepting states in A_φ. For example, the potentials can be distributed proportionally according to the graph distance in the directed graph representation of A_φ.

Attenuation Factor: Similar to the discount factor in infinite-horizon MDPs, an attenuation factor can be applied to the potentials in order to incentive early achievement of rewards. Unlike the discount factor, the attenuation factor need not be uniform over all behaviours in the TERF. For example, we can attenuate potentials by a factor $T - k/T$, where k is the steps counter in the current state and T is the horizon limit. Whereas the potentials with this technique are not constant over time, it has been shown that optimality, and near-optimality guarantees are equally preserved in dynamic reward shaping (Devlin & Kudenko, 2012).

Given the linearity of the shaping function, linear combinations of the methods presented above preserve optimality guarantees. Similarly, the technique to attenuate the potentials is orthogonal to the liveness preservation and heuristic guidance techniques presented above.

6 Empirical Evaluation

We conducted experiments to evaluate the impact of different reward shaping techniques on the quality of the solutions obtained by MDPs compiled from NMRDPs. We conducted experiments over a range of RDDL-encoded MDP problems from International Probabilistic Planning Competitions (IPPCs) in which we replaced the Markovian rewards by TERFs. We used different configurations of PROST as the MDP planner, including, the current state-of-the-art UCT*, the configurations that won the IPPC 2011 and 2014, and a configuration of PROST that behaves like the basic UCT algorithm by Kocsis & Szepesvári (2006). For UCT*, we tested different heuristic evaluation functions based on iterative deepening search (IDS), and depth-first search (DFS).

6.1 Guiding Towards Long-term TERFs

In our first set of experiments, we evaluated the impact of reward shaping in guiding search for long-term non-Markovian rewards. We conducted our tests in a modification of the *academic-advising* domain. In these problems, the agent can take courses. Courses have prerequisites – e.g. second-year courses have as prerequisite all first-year courses, and so on –, that affect the probability to pass a course. The agent is given a (non-Markovian) reward upon completion of all courses, if these were taken in an order that satisfies all prerequisites. In $\mathsf{LTL_f}$, the reward formula is the conjunction of two families of subfomulae. The first family asks the agent to pass all courses, and is captured by formulae $\Diamond(passed(c))$, one for each course, c. The second family asks the agent to take courses in an order that satisfies the prerequisites, that is, if c' is a prerequisite of c, then c should not be taken until c' is passed. This is captured by formulae $\Box(\bigwedge_{c'}((taken(c) \wedge prereq(c', c)) \rightarrow passed(c'))$.

Table 2 summarizes the results of experiments with different *academic-advising* problems. Each problem `p_Y_C` has the TERF described above, where Y is the number of academic

MDP Planner	Compilation	p_3_3	p_4_2	p_4_3	p_4_4
PROST UCT*(IDS)	MDP	30	30	0	0
PROST UCT*(DFS)	MDP	30	30	0	0
PROST IPPC-2014	MDP	2	30	0	0
PROST IPPC-2011	MDP	27	30	2	0
UCT	MDP	0	0	0	0
PROST UCT*(IDS)	MDP + RS	30	30	30	30
PROST UCT*(DFS)	MDP + RS	30	30	30	30
PROST IPPC-2014	MDP + RS	30	30	30	30
PROST IPPC-2011	MDP + RS	30	30	30	30
UCT (3 steps look ahead)	MDP + RS	29	30	29	30

Table 2. Number of runs (over 30 trials) that achieved the non-Markovian reward in the *academic-advising* problems. Different MDP planners were used to solve the compiled MDP problem, with and without reward shaping (RS).

years, and C is the number of courses per academic year. We compiled each NMRDP problem into an MDP, with and without reward shaping. We evaluated the quality of the solutions using different configurations of PROST.

As predicted, state-of-the-art anytime planners are short-sighted and struggle to find good long-term strategies in MDP compilations of NMRDPs. To improve look ahead, we modified the compiled MDP to have only one mode, *world* mode, in which the automata is also progressed. The results of these experiments are reported in Table 2 as *MDP compilation*. We note an abrupt decrease in performance in larger problem instances. This is because the look ahead performed by PROST in the presence of sparse non-Markovian rewards is highly uninformed (i.e. close to blind search), and for sufficiently large instances the look ahead cannot provide any information. If this occurs, PROST makes early random moves that prevent the agent from collecting any long-term reward. When the compiled MDP is enhanced with reward shaping, the search performance improves significantly. In our tests with reward shaping, the potentials are distributed uniformly with the number of courses passed — i.e. inversely proportional to the distance to the accepting automaton states as described in Section 5.2 —, and go down to zero when the prerequisites are violated. State-of-the-art configurations of PROST achieved the TERF in all trials. Remarkably, reward shaping boosted the performance of the basic UCT algorithm from zero to achieving the non-Markovian reward in almost all trials.

6.2 Guiding Towards Liveness Tradeoffs

In our second set of experiments, we evaluated the practicality of our approach in problems where the stochasticity of the domain could make it infeasible to accomplish all reward-worthy behaviors. We used a modification of the *wildfire* domain. In the *wildfire* domain, some places in a grid field are originally burning. Fire can propagate to neighboring cells with certain probability. The agent can attempt to extinguish fire, one cell at a time. We consider a *wildfire* problem in a 3×3 grid with desired behaviours, one for each cell c: *never have fire in c for more than two turns in a row*. We assign each reward $r = 100$ if, at the end of an execution trace of length 10, the behaviour is satisfied. We experimented with different configurations of PROST, UCT, and a naive planner that takes no action. The results are summarized in Table 3. In these experiments, we noted that reward shaping is beneficial to the quality of the plans. Moreover, PROST easily runs out of memory (512MB) if no reward shaping is applied. This is because the sparsity of rewards forces it to expand a large search tree. Limiting the memory usage in PROST resulted in policies of lower quality than those obtained with reward shaping. On the other hand, naive reward shaping ($\phi(F_q) = r$ in accepting states, 0 otherwise) successfully guides search, with a significant reduction in memory, demonstrating improved scalability.

MDP Planner	No RS	RS
PROST UCT*(IDS)	mem	617
PROST UCT*(DFS)	mem	627
PROST IPPC-2014	mem	620
PROST IPPC-2011	mem	637
UCT (3 steps look ahead)	423	527
no actions taken	263	263

Table 3. Average reward achieved (over 30 trials) in a *wildfire* problem with desired behaviours, for each cell c: *never have fire in c for more than two turns in a row*. Different MDP planners were used to solve the compiled MDP problem, with and without reward shaping (RS).

7 Summary and Discussion

NMRDPs provide a powerful framework for modelling decision-making problems with behaviour-based rewards. In this paper we used LTL_f to specify rich non-Markovian rewards and presented a technique for solving NMRDPs via compilation to MDPs that can be solved with off-the-shelf MDP planners. Our approach employs automata representations of the LTL_f formulae into the compiled MDP. We leverage reward shaping to help guide search, mitigating for the sparseness of non-Markovian rewards and the poor lookahead of some state-of-the-art UCT-based methods. Our experiments demonstrate that automata-based reward shaping is an effective method to enhance search and obtain solutions of superior quality. While non-Markovian rewards were specified here in LTL_f, the proposed approach will work for rewards specified in any formal language for which there is a corresponding automata representation including PLTL and other dialects of LTL; Golog and other dialects of regular expressions including LDL_f; and compositions of these languages (c.f., (Baier et al., 2008)).

Acknowledgements: The authors gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada.

References

Bacchus, Fahiem and Kabanza, Froduald. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1-2):123–191, 2000.

Bacchus, Fahiem, Boutilier, Craig, and Grove, Adam J. Rewarding behaviors. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI)*, pp. 1160–1167, 1996.

Bacchus, Fahiem, Boutilier, Craig, and Grove, Adam J. Structured solution methods for non-markovian decision processes. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI)*, pp. 112–117, 1997.

Baier, Jorge A. and McIlraith, Sheila A. Planning with temporally extended goals using heuristic search. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 342–345, 2006.

Baier, Jorge A., Fritz, Christian, and McIlraith, Sheila A. Exploiting procedural domain control knowledge in state-of-the-art planners. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 26–33, Providence, Rhode Island, USA, September 22 - 26 2007.

Baier, Jorge A., Fritz, Christian, Bienvenu, Meghyn, and McIlraith, Sheila A. Beyond classical planning: Procedural control knowledge and preferences in state-of-the-art planners. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1509–1512, 2008.

Baier, Jorge A., Bacchus, Fahiem, and McIlraith, Sheila A. A heuristic search approach to planning with temporally extended preferences. *Artificial Intelligence*, 173(5-6): 593–618, 2009.

Bienvenu, Meghyn, Fritz, Christian, and McIlraith, Sheila A. Specifying and computing preferred plans. *Artificial Intelligence*, 175(7–8):1308–1345, 2011.

Boutilier, Craig, Dean, Thomas, and Hanks, Steve. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research (JAIR)*, 11:1–94, 1999.

Brafman, Ronen I., De Giacomo, Giuseppe, and Patrizi, Fabio. LTLf/LDLf non-Markovian rewards. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Camacho, Alberto, Chen, Oscar, Sanner, Scott, and McIlraith, Sheila A. Decision-making with non-markovian rewards: From LTL to automata-based reward shaping. In *3rd Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM)*, pp. 279–283, 2017a.

Camacho, Alberto, Chen, Oscar, Sanner, Scott, and McIlraith, Sheila A. Non-Markovian rewards expressed in LTL: Guiding search via reward shaping. In *Proceedings of the 10th Symposium on Combinatorial Search (SOCS)*, pp. 159 – 160, 2017b.

Camacho, Alberto, Chen, Oscar, Sanner, Scott, and McIlraith, Sheila A. Decision-making with non-markovian rewards: Guiding search via automata-based reward shaping. Technical Report CSRG-632, Department of Computer Science, University of Toronto, June 2017c.

Camacho, Alberto, Baier, Jorge A., Muise, Christian J., and McIlraith, Sheila A. Finite LTL synthesis as planning. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 29–38, 2018.

De Giacomo, Giuseppe and Vardi, Moshe Y. Linear temporal logic and linear dynamic logic on finite traces. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 854–860, 2013.

De Giacomo, Giuseppe and Vardi, Moshe Y. Synthesis for LTL and LDL on finite traces. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1558–1564, 2015.

Devlin, Sam and Kudenko, Daniel. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pp. 433–440, 2012.

Fritz, Christian, Baier, Jorge A., and McIlraith, Sheila A. ConGolog, sin Trans: Compiling ConGolog into basic action theories for planning and beyond. In *Proceedings of the 11th International Conference on Knowledge Representation and Reasoning (KR)*, pp. 600–610, Sydney, Australia, 2008.

Gerevini, Alfonso, Haslum, Patrik, Long, Derek, Saetti, Alessandro, and Dimopoulos, Yannis. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6):619–668, 2009.

Keller, Thomas and Eyerich, Patrick. PROST: probabilistic planning based on UCT. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.

Kocsis, Levente and Szepesvári, Csaba. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning (ECML)*, pp. 282–293, 2006.

Kupferman, Orna and Vardi, Moshe Y. Model checking of safety properties. *Formal Methods in System Design*, 19 (3):291–314, 2001.

Littman, Michael L., Topcu, Ufuk, Fu, Jie, Jr., Charles Lee Isbell, Wen, Min, and MacGlashan, James. Environment-independent task specifications via GLTL. *CoRR*, abs/1704.04341, 2017.

Ng, Andrew Y., Harada, Daishi, and Russell, Stuart. Policy invariance under reward transformations : Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, volume 3, pp. 278–287, 1999.

Pnueli, Amir. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 46–57, 1977.

Puterman, Martin L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1994.

Reiter, Raymond. *The Frame Problem in the Situation Calculus: A Simple Solution (sometimes) and a completeness result for goal regression*, pp. 359–380. Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy. Academic Press, San Diego, CA, 1991.

Sanner, Scott. Relational dynamic influence diagram language (RDDL): Language description. 2010. URL http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf.

Sohrabi, Shirin, Baier, Jorge A., and McIlraith, Sheila A. Preferred explanations: Theory and generation via planning. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 261–267, August 2011.

Thiébaux, Sylvie, Gretton, Charles, Slaney, John K, Price, David, Kabanza, Froduald, et al. Decision-theoretic planning with non-markovian rewards. *Journal of Artificial Intelligence Research (JAIR)*, 25:17–74, 2006.

Toro Icarte, Rodrigo, Klassen, Toryn, Valenzano, Richard, and McIlraith, Sheila A. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.