Regular Decision Processes: A Model for Non-Markovian Domains

Ronen I. Brafman[1] and **Giuseppe De Giacomo**[2]
[1]Ben-Gurion University, Israel
[2]Sapienza Università di Roma, Italy
brafman@cs.bgu.ac.il, degiacomo@dis.uniroma1.it

Abstract

We introduce and study Regular Decision Processes (RDPs), a new, compact, factored model for domains with non-Markovian dynamics and rewards. In RDPs, transition and reward functions are specified using formulas in linear dynamic logic over finite traces, a language with the expressive power of regular expressions. This allows specifying complex dependence on the past using intuitive and compact formulas, and provides a model that generalizes MDPs and k-order MDPs. RDPs can also approximate POMDPs without having to postulate the existence of hidden variables, and, in principle, can be learned from observations only.

1 Introduction

The Markov assumption (MA) plays a key-role in the definition of MDPs. It states that the next-state distribution following an action is independent of the history, given the current state. Yet, MA does not hold in many domains: one is likelier to be able to enter a restricted area if permission was obtained in the past; a car is likelier to skid if it rained in the past with no sunshine since; and even more so, if the rain was followed by below zero temperature; a person is likelier to contract mumps flying, if she has not contracted it in the past; or, a robot should be rewarded for delivering coffee only if its owner previously requested it. The first examples describe non-Markovian dynamics, and the last one, a non-Markovian reward, first studied in [Bacchus *et al.*, 1996], and recently advocated as useful for reinforcement learning in robotics [Littman *et al.*, 2017; Camacho *et al.*, 2017].

Often, MA is treated as a simple technical restriction because one can always modify the state description to make a non-Markovian model Markovian. For example, we can add indicator variables that record whether access was granted, coffee was requested, etc. More generally, the belief-space MDP can be viewed as maintaining information about the entire history of a POMDP using an infinitely larger state space.

This approach suffers from two problems: First, the agent's state space may be hard-coded. For example, a robot learning its environment may have a built-in state representation, related to its sensing abilities, set by a designer who was possibly unfamiliar with this environment. Second, even when the designer has the freedom to modify the state space, it may be much simpler to specify non-Markovian dynamics or reward explicitly than to design the correct equivalent Markovian model which (as we show) can be exponentially larger.

One well-known non-Markovian model is k-order MDPs, in which the next state distribution induced by an action depends on the last k states. However, k-order MDPs are impractical for large k, as their equivalent MDP is exponential in k. Moreover, they cannot express simple properties that depend on the arbitrary past, such as the examples above.

To address these issues, we introduce and study *Regular Decision Process* (RDP), a non-Markovian decision model and language that can succinctly represent dependence on the arbitrary past. In RDPs, the next state distribution and the reward function are conditioned on logical formulas, like in factored MDPs [Boutilier *et al.*, 1999]. But while formulas in factored MDPs are propositional (or first-order) and refer to the current state only, formulas in RDPs are in LDL_f– linear dynamic logic over finite traces, and refer to the entire history.

The main contributions of this paper are to introduce RDPs, study their properties and complexity, describe how to optimize them, and postulate a potential method for learning them from observations. We also briefly explore the potential role of RDPs in approximating POMDPs. It is well known that POMDPs are difficult to solve [Papadimitriou and Tsitsiklis, 1987; Littman *et al.*, 1998; Madani *et al.*, 2003] – and difficult to learn: only a few algorithms exist that provide some guarantees (e.g., [Zhang *et al.*, 2012]) but they are not too useful w.r.t. sample complexity. RDPs offer three potential benefits: First, they are easier to solve than POMDPs. Second, they are more expressive than MDPs, and hence may be able to provide a better approximation. Indeed, we show that for every POMDP, there exists a RDP that can ϵ-approximate it. Third, they do not require hypothesizing the existence of hidden variables – they are based on observable variables only. This latter property is promising from the learning perspective.

2 Background

We assume familiarity with MDPs and POMDPs, and only recall basic notation and ideas.

A Markov Decision Process (MDP) $\mathcal{M} = \langle S, A, Tr, R, s_0 \rangle$ contains a set S of states, a set A of actions, a transition function $Tr : S \times A \rightarrow \Pi(S)$ that returns for every state s and action a a distribution over the next state; a reward

function $R : S \times A \to \mathbb{R}$ that specifies the real-valued reward received by the agent when applying action a in state s; and an initial state $s_0 \in S$. In this paper, states in S are truth assignments to a set $\mathcal{P}$ of primitive propositions. This allows for using a more compact factored description of the reward and transition function. To emphasize this, we write $\mathcal{M} = \langle \mathcal{P}, S, A, Tr, R, s_0 \rangle$. The horizon of the MDP is the number of actions/steps that the agent executes.

A solution to an MDP is a *policy* that maps histories of states and actions to actions. The *value* of policy ρ at s, $v^\rho(s)$, is the expected discounted sum of rewards when starting at s and selecting actions based on ρ. Every MDP has an *optimal* policy, ρ^*, that maximizes the expected discounted sum of rewards for every starting state $s \in S$. When the horizon is infinite, a deterministic stationary policy $\rho^* : S \to A$ that is optimal, exists [Puterman, 2005]. Many methods for solving MDPs rely on the Markov assumption, and their theoretical and practical complexity is strongly impacted by $|S||A|$.

A partially observable MDP (POMDP) is a tuple $M = \langle S, A, Tr, R, O, \Omega, b_0 \rangle$. S, A, Tr, R are as in MDPs, but instead of observing the entire state, the agent observes an element of Ω. $O : A \times S \to \Pi(\Omega)$ specifies the probability of observing $o \in \Omega$ if $a \in A$ was executed and led to state $s \in S$, denoted $o(a, s, o)$. Observations can be noisy, and the same observation may be possible in different state-action pairs. b_0 is the initial distribution over states. In a factored POMDP [Boutilier *et al.*, 1999], $M = \langle \mathcal{P}, S, A, Tr, R, O, \mathcal{P}_O, b_0 \rangle$, $\mathcal{P}$ is a set of propositions, S contains assignments to $\mathcal{P}$, and $\mathcal{P}_O \subset \mathcal{P}$ are the observable propositions. Transition, reward, and observation functions are specified more succinctly by alluding to changes in the values of propositions as a function of the current state's properties.

POMDPs can be solved by compiling them to belief-space MDPs – MDPs in which the states (called *belief states*) are distributions over S. The initial belief state is b_0, and given the current belief state, action and observation, the next belief state can be computed using standard probabilistic inference. While this state space is uncountable, it enjoys special properties that are exploited by various solution algorithms.

LDL_f (*linear dynamic logic on finite traces*)[De Giacomo and Vardi, 2013] combines linear-time temporal logic LTL [Pnueli, 1977] with the syntax of PDL, *propositional dynamic logic* [Fischer and Ladner, 1979; Harel *et al.*, 2000; Vardi, 2011], but interpreted over finite traces. LDL_f is as expressive as regular expressions (RE) and both have the expressive power of monadic second order logic over finite traces. Thus, much of this paper can be understood without familiarity with LDL_f, by substituting "regular expressions" wherever "LDL_f formula" appears. However, RE themselves *are not convenient* for expressing temporal specifications: To negate a RE, one must generate the corresponding deterministic finite automaton (DFA), which can be exponential in the size of the RE, negate it, and generate its corresponding RE. One can conjoin two RE in polynomial time by constructing their NFAs, generating their product NFA, and then moving to RE. Polynomial, but quite inconvenient. LDL_f was explicitly introduced in [De Giacomo and Vardi, 2013] to provide a more convenient specification language.

Here, we consider a variant of LDL_f that works also on empty traces [Brafman *et al.*, 2018]. Formally, LDL_f formulas φ are built as follows:

$$\begin{array}{rcl} \varphi & ::= & tt \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \langle \varrho \rangle \varphi \\ \varrho & ::= & \phi \mid \varphi? \mid \varrho_1 + \varrho_2 \mid \varrho_1 ; \varrho_2 \mid \varrho^* \end{array}$$

tt stands for logical true; ϕ is a propositional formula over $\mathcal{P}$ (including *true*, not to be confused with tt); ϱ denotes path expressions, which are RE over propositional formulas ϕ with the addition of the test construct $\varphi?$ typical of PDL. We use abbreviations $[\varrho]\varphi \doteq \neg\langle\varrho\rangle\neg\varphi$, $ff \doteq \neg tt$ for false, and $\phi \doteq \langle\phi\rangle tt$ to denote occurrence of propositional formula ϕ.

Intuitively, $\langle\varrho\rangle\varphi$ states that from the current step in the trace, there exists an execution satisfying the RE ϱ, such that its last step satisfies φ, while $[\varrho]\varphi$ states that from the current step, all executions satisfying the RE ϱ are such that their last step satisfies φ. Tests are used to insert into the execution path checks for satisfaction of additional LDL_f formulas.

The semantics of LDL_f is given in terms of *finite traces*, i.e., finite sequences $\tau = \tau_0, \ldots, \tau_n$ of elements from the alphabet $2^{\mathcal{P}}$. In decision processes, traces are sequences of states *and* actions: $\langle s_0, a_1, s_1, \ldots, a_n, s_n \rangle$. We represent them as a trace by extending the set $\mathcal{P}$ to include one proposition p_a per action a, assigned true if a was the last action, and setting $\tau_i \doteq s_i \cup \{p_a \mid a = a_i\}$. In this way, τ_i denotes the pair (a_i, s_i) (with a dummy action a_0 for the initial state s_0). Henceforth, we assume this form, even if verbally referring to state-action sequences and sometimes representing the actions explicitly. We define $\tau(i) \doteq \tau_i$, $length(\tau) \doteq n + 1$, and $\tau(i, j) \doteq \tau_i, \tau_{i+1}, \ldots, \tau_{j-1}$. When $j > length(\tau)$, $\tau(i, j) \doteq \tau(i, length(\tau))$. Given a finite trace τ, an LDL_f formula φ, and a position i, we define when φ *is true* at step i, written $\tau, i \models \varphi$, by (mutual) induction, as follows:

- $\tau, i \models tt$;
- $\tau, i \models \neg\varphi$ iff $\tau, i \not\models \varphi$;
- $\tau, i \models \varphi_1 \wedge \varphi_2$ iff $\tau, i \models \varphi_1$ and $\tau, i \models \varphi_2$;
- $\tau, i \models \langle\varrho\rangle\varphi$ iff there exists $i \leq j$ such that $\tau(i, j) \in \mathcal{L}(\varrho)$ and $\tau, j \models \varphi$, where the relation $\tau(i, j) \in \mathcal{L}(\varrho)$ is as follows:
  - $\tau(i, j) \in \mathcal{L}(\phi)$ if $j{=}i{+}1$, $i < length(\tau)$, and $\tau(i) \models \phi$ (ϕ propositional);
  - $\tau(i, j) \in \mathcal{L}(\varphi?)$ if $j = i$ and $\tau, i \models \varphi$;
  - $\tau(i, j) \in \mathcal{L}(\varrho_1{+}\varrho_2)$ if $\tau(i, j) \in \mathcal{L}(\varrho_1)$ or $\tau(i, j) \in \mathcal{L}(\varrho_2)$;
  - $\tau(i, j) \in \mathcal{L}(\varrho_1; \varrho_2)$ if there exists $k \in [i, j]$ such that $\tau(i, k) \in \mathcal{L}(\varrho_1)$ and $\tau(k, j) \in \mathcal{L}(\varrho_2)$;
  - $\tau(i, j) \in \mathcal{L}(\varrho^*)$ if $j = i$ or there exists k such that $\tau(i, k) \in \mathcal{L}(\varrho)$ and $\tau(k, j) \in \mathcal{L}(\varrho^*)$.

Note that if $i \geq length(\tau)$, the above definitions still apply; though, $\langle\phi\rangle\varphi$ (ϕ prop.) and $\langle\varrho\rangle\varphi$ become trivially false.

We say that a trace τ *satisfies* an LDL_f formula φ, written $\tau \models \varphi$, if $\tau, 0 \models \varphi$.

In LDL_f we can easily express temporal operators such as *next* ($\bigcirc$) and *until* ($\mathcal{U}$) using abbreviations $\bigcirc\varphi \doteq \langle true\rangle(\varphi \wedge \neg end)$ and $\varphi_1 \mathcal{U} \varphi_2 \doteq \langle(\varphi_1?; true)^*\rangle(\varphi_2 \wedge \neg end)$, and any RE r, with the formula $\langle r\rangle end$, where $end \doteq [true]ff$ expresses that the trace has ended. We also use the abbreviation $atlast(\varrho) \doteq \langle true^*; \varrho\rangle end$ to say that the sequence ϱ happened just before the end. For example, $atlast(Rain)$ means that at the very end of the trace it rained.

Example 1. *Here are some* LDL$_f$ *formulas and their intuitive semantics:* $\varphi_1 = \langle true^*; Rain; true^* \rangle end$ *– it rained in the past.* $\varphi_2 = atlast(Rain; Rain; Rain)end$ *– it rained in the last 3 time steps (e.g., days).* $\varphi_2' = atlast(Rain) \vee atlast(Rain; true) \vee atlast(Rain; true; true)$ *– it rained in one of the last 3 time steps.* $\varphi_3 = \langle true^*; Rain; (\neg(Tmp{>}5))^* \rangle end$ *– it rained in the past, and the temperature was not above* 5^oC *since.* $\varphi_4 = \langle true^*; Rain; Tmp{<}0; (\neg(Tmp{>}2))^* \rangle end$ *– it rained in the past, then the temperature was below 0, and since, it was not above* 2^oC.

3 Non-Markovian Decision Processes

We define the basic semantic model of a Non-Markovian Decision Process (NMDP) and the more restricted case of a RDP.

3.1 NMDPs

A NMDP is identical to an MDP, except that the transition and reward functions depend on the *entire* history (trace). Formally, a Non-Markovian Decision Process (NMDP) is a tuple $M = \langle \mathcal{P}, A, S, tr, r, s_0 \rangle$, where $\mathcal{P}, A, S, s_0$ are as in an MDP, and we assume there are propositions $A \subseteq \mathcal{P}$ that denote the action that just occurred; $tr : S^+ \times A \times S \to \Pi(S)$ is the transition function, i.e., $tr((s_0, \ldots, s_k), a, s')$ is the probability of reaching state s' when executing action a given history $s_0, ..., s_k$; and $r : S^+ \to [r_{min}, r_{max}]$. Here S^+ is the set of all finite, non-empty, state sequences.

A *policy* for a NMDP is a partial function $\rho : S^+ \to \mathcal{A}$ such that ρ is defined on every sequence $w \in S^+$ reachable from s_0 under ρ, where reachability under ρ is defined inductively as follows: s_0 is reachable; if $w \in S^+$ is reachable and $tr(w, a, s') > 0$ then $w \cdot s'$ is reachable.

The *value* of a trace $s_0, s_1, \ldots, s_n$ is its discounted sum of rewards: $v(s_0, \ldots, s_n) = \sum_{i=0}^{n} \gamma^n r(s_0, \ldots s_i)$, where $0 < \gamma < 1$ is the discount factor. Because we assume the reward value is lower and upper bounded, this discounted sum is always finite and bounded from above and below. For every finite horizon n, ρ and M define a distribution over possible traces, and $v_n(\rho)$ denotes the expected value of length n traces with respect to this distribution. We denote the *value of a policy* ρ by $v(\rho) = \liminf_{n \to \infty} v_n(\rho)$. Finally, ρ is an *optimal policy* for M if for every policy ρ', we have that $v(\rho) \geq v(\rho')$.

For MDPs with infinite horizon, we know that there exists an optimal stationary policy, i.e., a policy $\rho : S \to \mathcal{A}$. Thus, in MDPs we can restrict attention to this finite set of policies. For NMDPs, it is clear that we cannot restrict our attention to stationary policies. In fact, it is not a-priori clear that there exists an optimal policy. However, since the value of a policy is bounded, there exists some value $v^* = \sup_{\text{policy } \rho} v(\rho)$.

3.2 RDPs

NMDPs can refer to infinite objects and may not be finitely representable, so one cannot specify or learn them, in general. Hence, we restrict attention to a class of NMDPs in which a finite, logic-based description is used to capture properties of traces. Specifically, we use dynamic logic over finite trace, LDL$_f$, whose formulas are interpreted as sets of traces. We note that there is much past work in the area of probabilistic model-checking and control on computing the probability that a trace will satisfy a given temporal formula [Kwiatkowska *et al.*, 2011], and on generating policies that ensure (or maximizing the probability) that it will be satisfied [Ding *et al.*, 2014]. Unlike them, we use LDL$_f$ formula to describe the inherent dynamics of the system.

We define a *Regular Decision Processes* (RDP) to be a NMDP $M_L = \langle \mathcal{P}, A, S, tr_L, r_L, s_0 \rangle$ whose transition and reward functions are specified as follows: tr_L is represented by a *finite* set T of quadruples of the form: $(\varphi, a, P', \pi(P'))$, where φ is an LDL$_f$ formula over $\mathcal{P}$, $a \in A$, $P' \subseteq \mathcal{P}$ is the set of propositions affected by a when φ holds, and $\pi(P')$ is a joint-distribution over P' describing its post-action distribution. The basic assumption is that the value of variables not in P' is not impacted by a.

If $\{(\varphi_i, a, P_i', \pi_i(P') | i \in I_a\}$ are all quadruples for a, then the φ_i's must be *mutually exclusive*, i.e., $\varphi_i \wedge \varphi_j$ is inconsistent, for $i \neq j$. But they need not be exhaustive, so that no a transition may be possible given some traces.

$tr_L((s_0, ..., s_k), a, s')$ is now defined as follows:

1. $tr_L((s_0, ..., s_k), a, s') = \pi(s'|_{P'})$ if exists a quadruple $(\varphi, a, P', \pi(P'))$ such that $s_0, ..., s_k \models \varphi$, and s_k and s' agree on all variables in $\mathcal{P} \setminus P'$. Here, $s'|_{P'}$ denotes the restriction of s' to the propositions in P'.
2. $tr_L((s_0, ..., s_k), a, s') = 0$ otherwise.

That is, given current trace $s_0, ..., s_k$ and action a, if no quadruple has a condition φ satisfied by $s_0, ..., s_k$, then no transition is possible on a. Otherwise, let $(\varphi, a, P', \pi(P'))$ be such a quadruple. By our assumptions, it is the only one. The next state s' must assign exactly the same value to propositions not in P' as in s_k – i.e., they are *not* impacted by the action. Its probability is equal to the probability that π assigns to the value of the P' propositions in s'.

The reward function r_L is specified using a *finite* set R of pairs of the form (φ, r), where φ is an LDL$_f$ formula over $\mathcal{P}$, and $r \in \mathbb{R}$ is a real-valued reward. Given a trace $s_0, \ldots, s_k$, the agent receives the reward: $r_L(s_0, \ldots, s_k) = \sum_{(\varphi, r) \in R \wedge s_0, ..., s_k \models \varphi} r$. As before, by definition r_L is bounded above and below.

Example 2. *When driving from* A *to* B *after it has rained, followed by temperature below zero and very low temperature since, there is a 0.1 probability of reaching* B *with some damage, and 0.1 probability of not reaching* B *at all (with some damage). Using* φ_4 *defined earlier, and* A, B, d *for* $at\_A, at\_B,$ damaged*, respectively, we can write:* $(\varphi_4 \wedge atlast(A \wedge \neg d), drive, \{A, B, d\}), \pi)$*, where* $\pi(\neg A \wedge B \wedge d) = 0.1, \pi(\neg A \wedge \neg B \wedge d) = 0.1, \pi(\neg A \wedge B \wedge \neg d) = 0.8$*. If it has only rained and the temperature was not high since, then these probabilities drop to 0.01. We can write:* $(\varphi_3 \wedge \neg\varphi_4 \wedge atlast(A \wedge \neg d), drive, \{A, B, d\}), \pi)$*, where* $\pi(\neg A \wedge B \wedge d) = 0.01, \pi(\neg A \wedge \neg B \wedge d) = 0.01, \pi(\neg A \wedge B \wedge \neg d) = 0.98$*. To reward the robot for delivering coffee to Ann only if she requested it earlier, we can write* $(\langle true^*; RqstAnn, (\neg DlvdAnn)^*; DlvdAnn \rangle end, 10)$.

LDL$_f$ has the same expressive power as RE. Hence, dependence on non-regular constructs cannot be specified by RDPs. For example, it is not possible to express something that says that the robot will be rewarded if the number of coffee cups

served is equal precisely to the number of requests made – this requires the strength of a context-free language.

4 Solving RDPs

RDPs are attractive because they provide a natural way of using the rich, yet intuitive, language of regular expressions to specify a decision process in which transitions and rewards can depend on an unbounded history, unlike, for example, k-order MDPs. Of course, the ability to specify them is of little use if we cannot solve them. Using the well-known relationship between LDL_f formulas and automata, we can exploit a construction used in past work on decision processes with non-Markovian reward, to automatically transform RDPs into MDPs, on which we can apply known methods. This makes RDPs a useful tool for specifying more complex decision processes, that combined with existing automated tools for constructing a DFA from an LTL_f/LDL_f formula (e.g., https://flloat.herokuapp.com [Favorito, 2018] http://ltlf2dfa.diag.uniroma1.it [Fuggitti, 2018]), can save the modeler the effort and potential errors associated with attempting to transform them manually into MDPs.

In principle, one could directly apply Monte-Carlo methods to solve RDPs. Yet, one must still be able to recognize which formulas are satisfied by the current trace, and this is most efficiently and conveniently done by tracking the state of the formula using the construction described above.

Given a RDP $M_L = \langle \mathcal{P}, A, S, tr_L, r_L, s_0 \rangle$ as above, we can construct an equivalent MDP M as follows: Let T be the set of quadruples $(\varphi, a, P', \pi(P'))$ defining tr_L. Let R be the set of pairs (φ, r) defining r_L. Enumerate the quadruples in T as $(\varphi_1, a_1, P'_1, \pi(P'_1)), \ldots, (\varphi_m, a_m, P'_m, \pi(P'_m))$, and the pairs in R as $(\varphi_{m+1}, r_{m+1}), \ldots, (\varphi_n, r_n)$. For each formula φ_i, build the corresponding DFA $A_i = \langle 2^{\mathcal{P}}, Q_i, \delta_i, F_i, , q_{i,0} \rangle$ that accepts exactly those traces that satisfy φ_i. Here, $2^{\mathcal{P}}$, the set of all truth assignments to the propositions in $\mathcal{P}$, is A_i's alphabet, Q_i is its state space, $q_{i,0}$ is the initial state, δ_i is its transition function, and F_i is the set of accepting/goal states. For details of this well known construction see [De Giacomo and Vardi, 2013; Brafman *et al.*, 2018]. The complexity of generating a deterministic automaton for φ_i is 2EXPTIME, and its size is doubly exponential in the worst case. In practice, this transformation often does not involve exponential blow up and yields compact automata [Tabakov and Vardi, 2005]. We define the MDP $M = \langle \mathcal{P}', Q, tr, r, q_0 \rangle$ where

- $Q = S \times Q_1 \times \cdots \times Q_n$
- $\mathcal{P}'$ extends $\mathcal{P}$ with propositions that capture the states of $A_1, \ldots, A_n$. (The finite state of A_i can be encoded using $log(|Q_i|)$ propositions.)
- $tr((s, q_1, \ldots, q_n), a, (s', q'_1, \ldots, q'_n)) = tr_L(\bar{s}, a, s')$ if (1) there exists a (unique by assumption) $1 \leq i \leq m$ such that $q_i \in F_i$, (2) $\bar{s}$ is some trace that satisfies φ_i, and (3) for every $1 \leq j \leq n$ we have $q'_i = \delta_i(q_i, s')$. Otherwise, $tr((s, q_1, \ldots, q_n), a, (s', q'_1, \ldots, q'_n)) = 0$.
- $r((s, q_1, \ldots, q_n), a) = \sum_{\{i \in \{m+1 \ldots, n\} | q_i \in F_i\}} r_i$
- $q_0 = (s_0, q_{1,0}, \ldots, q_{n,0})$

In words: the MDP state reflects the states of the RDP and all automata tracking the satisfaction of $\varphi_1, \ldots, \varphi_n$. The initial state is the RDP's initial state combined with the initial states of all automata. The transition function updates the RDP state component identically to tr_L, and deterministically updates the state of each automaton using its transition function. Note that the automata transitions depend on the new RDP state s', and recall s' reflects the last action, too. Our requirement on the quadruples ensures that for every action, there is exactly one formula that is satisfied (or, for inapplicable actions, no formula). Finally, the reward is the sum of the rewards associated with formulas satisfied by the current trace, which correspond exactly to all automata entering an accepting state.

The computation tree Υ_M of NMDP $M = \langle \mathcal{P}, A, S, tr, r, s_0 \rangle$ is defined by induction as follows.

- The root of Υ_M is s_0.
- If $s_0, \ldots, s_k$ is a node of Υ_M, then for each action a and state s' such that $tr((s_0, \ldots, s_k), a, s') > 0$, the node $s_0, \ldots, s_k$ in Υ_M has a successor $s_0, \ldots, s_k, s'$, and the edge between them is labeled $a, (tr((s_0, \ldots, s_k), a, s'), r(s_0, \ldots, s_k, s')$.

The computation tree of an MDP is similarly defined.

Lemma 1. *Given RDP $M_L = \langle \mathcal{P}, A, S, tr_L, r_L, s_0 \rangle$ let $M = \langle \mathcal{P}, A, S, tr, r, s_0 \rangle$ be the MDP constructed as above. M_L and M generate computation trees (and hence sets of traces) that are isomorphic.*

Proof (sketch). We map nodes in M_L's computation tree to nodes in M's computation tree as follows: The RDP state component of the MDP state is identical to the RDP state in M_L. The value of $Q_1 \times \cdots \times Q_n$ is a deterministic function of the sequence of RDP states. Since the Q_i states encode the satisfiability of the formulas $\varphi_1, \ldots, \varphi_n$, there is a one-to-one correspondence between traces in which Q_i reaches a final state and ones in which φ_i is satisfied. This implies that the same transitions are possible in both cases, and the same rewards are obtained. The formal proof proceeds by induction on the length of the trace/branch. □

Theorem 1. *An optimal RDP policy can be computed in 2EXPTIME; finding a policy with value $\geq c$ is 2EXPTIME-hard.*

Proof (sketch). Membership comes from the construction above: the generation of the automata from the formulas is in 2EXPTIME (but their factored encoding is only exponential) while computing a policy for an MDP is polynomial in $|S||A|$. Hardness comes from Planning for LDL_f (actually its fragment LTL_f) goals in fair FOND, see [De Giacomo and Rubin, 2018] which is a special case of stochastic planning, and a goal can be captured by a reward + transition to a sink state. □

A policy ρ is *regular* if it has the form $\{(\varphi_i, a_i)\}$ where φ_i is an LDL_f formula and a_i an action, such that for every trace $s_0, \ldots, s_k$ reachable given ρ, either no transition is possible after $s_0, \ldots, s_k$, or there exists exactly one $(\varphi_i, a_i) \in \rho$ such that $s_0, \ldots, s_k \models \varphi_i$. A by-product of the results above is:

Theorem 2. *Every RDP has a regular policy which is optimal.*

Due to space restrictions, proofs of most results are omitted.

5 Partially Observable Stochastic Domains

MDPs assume the agent is always aware of its current state. This is unrealistic for many, if not most, real-world domains in which the state is only partially observable. Consequently, much effort has been made to solve partially observable Markov decision processes (POMDPs) [Cassandra *et al.*, 1994] and partially observable contingent planning problems [Albore *et al.*, 2009].

The standard approach for solving partially observable problems is to transform them into fully observable problems over a different state space. POMDPs can be transformed into an (fully observable) MDP over (probabilistic) belief states, i.e., distributions over world states, and contingent planning problems can be transformed into classical planning problems over classical belief states, i.e., sets of world states.

However, just the formulation of a partially observable model presupposes knowledge of the nature of the unobservable part of the world. This is a serious drawback when we consider the problem of learning such domains from observable data when no additional domain knowledge exists. This problem is well-known, and has given rise to models such as predictive-state representations (PSR) [Singh *et al.*, 2001] whose core idea is to work with the observations only. NMDPs and RDPs have this feature, too.

Lemma 2. *For every POMDP $M = \langle \mathcal{P}, S, A, Tr, R, O, \mathcal{P}_O, b_0 \rangle$, there exists an NMDP $M' = \langle \mathcal{P}_O, A, 2^{\mathcal{P}_O}, tr, r, \epsilon \rangle$, where ϵ, the empty string, denotes the initial, empty observation, and tr and r are defined based on Tr, R, and O, such that the computation trees for M and M' are isomorphic.*

This essentially follows from the history-based MDP formulation of a POMDP.

The key difference between the NMDP and POMDP formulation is that the former makes no reference to hidden variables/state. Thus, its policy is also based on observables only. But as noted, general NMDPs are not practically useful. Therefore, the history-based view is used only to represent finite-horizon policies. We suggest investigating RDPs as a tractable approximation of POMDPs. To understand this option, first we characterise the type of POMDPs that RDPs can capture – this can be understood from the construction of the equivalent MDP in Section 4.

Lemma 3. *For every POMDP $M = \langle \mathcal{P}, S, A, Tr, R, O, \mathcal{P}_O, b_0 \rangle$ if in all reachable states the value of every $p \in \mathcal{P}$ is a regular function of the past and current values (trace) of $\mathcal{P}_O$ (i.e., p is true* iff *some* RE *over $\mathcal{P}_O$ is satisfied), there exists an RDP with an equivalent computation tree.*

Proof (sketch). If the POMDP's hidden variables are a regular function of the observable variables, then we know that they can be represented using an automaton whose alphabet is $2^{\mathcal{P}_O}$, and so its language can be represented by an LDL_f formula. □

The converse is also easy to show:

Lemma 4. *Every RDP can be transformed into a POMDP with the same set of observable variables in which the hidden variables are a regular function of the observable variables.*

This establishes the equivalence between RDPs and the restricted class of POMDPs whose hidden variables are a regular function of their observable variables.

However, we can show the following:

Theorem 3. *For every infinite horizon discounted reward POMDP M, and for every $\epsilon > 0$, there exists a RDP M_L over $\mathcal{P}_O$, such that the optimal policy for M_L is ϵ-optimal for M.*

Proof (sketch). First, we prove the following claim: *every finite horizon POMDP M is equivalent to a RDP with variables $\mathcal{P} = \mathcal{P}_O$.* Starting from the POMDP $M = \langle \mathcal{P}, S, A, Tr, R, O, \mathcal{P}_O, b_0 \rangle$, we construct the history-based MDP $M_h = \langle Q, A, tr, r, \epsilon \rangle$. Q contains all histories bounded by the horizon, and is thus finite. We define the corresponding RDP $M_L = \langle \mathcal{P}, A, S, tr_L, r_L, s_0 \rangle$. tr_L is defined as follows: first, we partition, for each action, the states in Q into subsets $\{Q_i | i \in I_a\}$ such that a changes the values of exactly the same set of propositions when applied in each $q \in Q_i$, and such that the marginal distribution over P' values in $a(q)$ is identical for all $q \in Q_i$. Note that some Q_i can be singletons. Let $\{(Q_i, a_j) | i \in I_{a_j}, a_j \in A\}$ denote all pairs of such state set and relevant action.

For every (Q_i, a_j) pair we consider the DFA $A^{Q_i}_{Tr}$ defined as $A^{Q_i}_{Tr} = \langle 2^{\mathcal{P}}, Q, \delta, Q_i, q_0 \rangle$, where $\delta(q, s) = q \cdot s$. That is the new state is the history obtained by concatenating the new state with the current history. (Recall the last action is represented in the new state). The final states are Q_i. Notice that all the DFAs $A^{Q_i}_{Tr}$ for all Q_i are identical except for their accepting state. From basic automata theory, we know that there is a regular expression r_{Q_i} that accepts the paths, i.e., the traces in S^+ leading from the initial state q_0 to Q_i. Hence we can express reaching Q_i by the LDL_f formula $\varphi_i = \langle r_{Q_i} \rangle end$.

As noted above, the next state distribution for states in Q_i and action a is characterised by some common $(P', \pi(P'))$. Hence, we can represent this transition by the quadruple: $(\varphi_i, a, P', \pi(P'))$, and we add it to the set of quadruples T. A similar method will let us generate the elements of R. Thus, this concluded the proof of the claim.

Because of discounting, we can compute an ϵ-optimal policy to an infinite horizon POMDP by solving a finite-horizon POMDP with a sufficiently long horizon. □

An important challenge for future work is finding more effective strategies for generating approximate RDP, and bounding their size. Such strategies would open the possibility of solving POMDPs by first approximating them using RDPs, and then solving the resulting RDP.

6 RDP Learning

We now outline a potential approach for learning RDPs with no background information by combining ideas from automata learning and RL. This is a purely theoretic proposal attempting to lay foundations for future practical methods. Learning algorithms might also prove useful for constructing RDPs that approximate a known POMDP.

Algorithm 1 outlines an RDP learning method. The algorithm requires access to an automata learning algorithm, such as [Lang *et al.*, 1998; Lambeau *et al.*, 2008], and assumes

restarts. Recall that we assume there are observable variables denoting the action performed. Hence, we do not mention actions explicitly in the algorithm. Also, note that the automata for transitions are constructed so that only traces from one set H_i^{tr} are accepted. This ensures that we are never in an accepting state in more than one component M_i^{tr}, and so only one quadruple formula is satisfied.

Algorithm 1 Pseudocode for Learning Finite NMDPs

1: Generate a set of traces with their associated rewards
2: The set of propositions of the RDP corresponds to the set of observable propositions
3: **for all** traces τ **do**
4: Compute $(a, P', \pi(P'))_\tau$ the empirical distribution over traces extending τ with action a and one observation.
5: Group traces with identical $(a, P'_i, \pi(P'_i))$ into sets $\{H_i^{tr} | i \in \mathcal{T}\}$.
6: Group traces with identical reward into sets $\{H_j^r | j \in \mathcal{R}\}$
7: **for all** $i \in \mathcal{T}$ **do**
8: Learn an automaton M_i^{tr} that accepts H_i^{tr} and rejects all H_j^{tr} for $j \neq i$
9: Generate formula φ_i^{tr} that captures language of M_i^{tr}
10: Insert the quadruple $(\varphi_i^{tr}, a, P'_i, \pi(P'_i))$ to tr_L
11: **for all** $i \in \mathcal{R}$ **do**
12: Learn an automaton M_i^r that accepts H_i^r and rejects all H_j^r for $j \neq i$
13: Generate formula φ_i^r that captures language of M_i^r
14: Insert the pair (φ_i^r, r_i) into r_L

We can control the quality of the approximation by increasing the number of samples in each set of traces to obtain more accurate estimates of $(P'_i, \pi(P'_i))$. We note that, unfortunately, automata learning, used in Lines 8 and 12, is a difficult problem, with many hardness results as far as sample complexity, unless input beyond positive and negative instances is available [Angluin *et al.*, 2013]. However, practical algorithms developed in the area of grammatical inferences exist [Lang *et al.*, 1998; Lambeau *et al.*, 2008].

7 Discussion and Summary

It is worthwhile to compare RDPs with the Predictive State Representation model (PSR) [Singh *et al.*, 2001]. PSRs were introduced explicitly to model dynamical systems without referring to a hidden state, and RDPs are also motivated by the desire to model rich dynamical systems without postulating some hidden state structure. A basic element of a PSR is its set of *core tests*, where a test is simply a sequence of actions and observations $q = a^1 o^1 \cdots a^n o^n$. A key computation for a test is its probability given a history $h = a_1 o_1 \cdots a_m o_m$. That is, $p(q|h)$ is the probability that following history h, one will observe $o^1 \cdots o^n$ if one performs $a^1 \cdots a^n$. In a linear PSR, one can compute $p(q|h)$ for every q and every finite h as a linear combination of $p(q_1|h), \cdots, p(q_n|h)$, where $q_1, \ldots q_n$ are the core tests. n is called the dimensionality of the system. [Singh *et al.*, 2001] proved that a linear PSR can represent any POMDP using no more core tests than the number of underlying states of the POMDP. Later work showed examples of non-linear PSRs that require exponentially fewer tests than an equivalent POMDP model [Rudary and Singh, 2003] and algorithms for learning PSRs [Singh *et al.*, 2003].

Given this representational power of PSRs, we have the following progression in terms of expressivity:

$$MDP < k{-}orderMDP < RDP \leq POMDP < PSR$$

Earlier, we saw that RDPs are equivalent to a (most likely strict) sub-class of POMDPs. So RDPs are most likely less expressive than POMDPs. Support for this conjecture comes from the fact that, unlike POMDPs, RDPs can be translated into a, possibly much larger, but still finite MDP. Interestingly, if we replace the stochastic transition function in RDPs and POMDPs with a non-deterministic one (obtaining the contingent planning model in the latter), the two models have equivalent expressive power [Brafman and De Giacomo, 2019].

Complexity-wise, a similar relation exists. There is a polynomial reduction from POMDPs to linear PSRs, so PSRs are at least as hard to solve as POMDPs. POMDP optimization is *undecidable* for the infinite horizon case, while for (factored) RDPs, it is 2EXPTIME, thus much easier. For finite horizons, solving flat POMDPs is in PSPACE. For a factored model, one likely incurs another exponential blowup (thus EXPSPACE) so it is likely no easier then RDP optimization. However, the 2EXPTIME complexity of RDPs is for unbounded plan sizes, and with bounded horizon, the problem should be easier.

Moreover, one exponential in the complexity of RDP optimization comes from the need to generate a factored representation of the automaton for each formula, which is worst-case exponential in the size of the formulas used (and thus, the reduction from RDPs to POMDPs is not polynomial). There is evidence that this transformation is fast in practice, and yields compact automata [Tabakov and Vardi, 2005], and if one were to restrict attention to RDPs with bounded size formulas, one could still refer to arbitrary events in the past, yet get rid of this exponential factor. Hence, RDPs with bounded formulas should be much easier to solve than POMDPs.

So while precise characterization of the finite-horizon case for RDPs is still missing, RDPs seem to offer a middle ground between MDPs and POMDPs: they are more expressive and potentially more succinct than k-order MDPs, are easier to solve than POMDPS, yet refer only to *observable* variables.

Summary. We studied a new model for controlled dynamical systems. This model allows for compact and convenient specification of non-Markovian transitions and rewards, generalizes k-order MDPs, can ϵ-approximate POMDPs, yet makes no reference to hidden variables. This last property makes it a potential target model for learning dynamical system without background knowledge. In future work we hope to farther explore methods for approximating POMDPs using RDPs, and to develop practical RDP learning algorithms. Finally, RDPs suggest that it may be worth studying additional models that restrict the nature of the hidden state of a POMDP for which better inference and learning algorithms may exist.

Acknowledgements

This work is supported in part by the Israel Ministry of Science and Technology grant 877017, the Lynn and William Frankel Center for Computer Science, and Sapienza project "Immersive Cognitive Environments".

References

[Albore *et al.*, 2009] Alexandre Albore, Héctor Palacios, and Hector Geffner. A translation-based approach to contingent planning. In *IJCAI'09*, pages 1623–1628, 2009.

[Angluin *et al.*, 2013] Dana Angluin, James Aspnes, Sarah Eisenstat, and Aryeh Kontorovich. On the learnability of shuffle ideals. *JMLR*, 14:1513–1531, 2013.

[Bacchus *et al.*, 1996] Fahiem Bacchus, Craig Boutilier, and Adam J. Grove. Rewarding behaviors. In *AAAI*, 1996.

[Boutilier *et al.*, 1999] Craig Boutilier, Tom Dean, and Steve Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *JAIR*, 11:1–94, 1999.

[Brafman and De Giacomo, 2019] Ronen Brafman and Giuseppe De Giacomo. Planning for LTLf/LDLf goals in non-markovian fully observable nondeterministic domains. In *IJCAI'19*, 2019.

[Brafman *et al.*, 2018] Ronen Brafman, Giuseppe De Giacomo, and Fabio Patrizi. LTLf/LDLf non-Markovian rewards. In *AAAI*, 2018.

[Camacho *et al.*, 2017] Alberto Camacho, Oscar Chen, Scott Sanner, and Sheila A. McIlraith. Non-markovian rewards expressed in LTL: guiding search via reward shaping. In *SOC*, pages 159–160, 2017.

[Cassandra *et al.*, 1994] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *AAAI'94*, volume 2, pages 1023–1028, 1994.

[De Giacomo and Rubin, 2018] Giuseppe De Giacomo and Sasha Rubin. Automata-theoretic foundations of FOND planning for LTLf and LDL_f goals. In *IJCAI'18*, 2018.

[De Giacomo and Vardi, 2013] Giuseppe De Giacomo and Moshe Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI'13*, 2013.

[Ding *et al.*, 2014] Xu Chu Ding, Stephen L. Smith, Calin Belta, and Daniela Rus. Optimal control of markov decision processes with linear temporal logic constraints. *IEEE Trans. Automat. Contr.*, 59(5):1244–1257, 2014.

[Favorito, 2018] Marco Favorito. Reinforcement learning for LTLf/LDLf goals: Theory and implementation. Master's thesis, DIAG, Sapienza Univ. Rome, 2018.

[Fischer and Ladner, 1979] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Com. Systems and Science*, 18, 1979.

[Fuggitti, 2018] Francesco Fuggitti. LTL and past LTL on finite traces for planning and declarative process mining. Master's thesis, DIAG, Sapienza Univ. Rome, 2018.

[Harel *et al.*, 2000] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[Kwiatkowska *et al.*, 2011] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *ICGI'11*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[Lambeau *et al.*, 2008] Bernard Lambeau, Christophe Damas, and Pierre Dupont. State-merging DFA induction algorithms with mandatory merge constraints. In *Grammatical Inference: Algorithms and Applications, 9th International Colloquium, ICGI 2008, Saint-Malo, France, September 22-24, 2008, Proceedings*, pages 139–153, 2008.

[Lang *et al.*, 1998] Kevin J. Lang, Barak A. Pearlmutter, and Rodney A. Price. Results of the abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In *ICGI'98*, pages 1–12, 1998.

[Littman *et al.*, 1998] Michael L. Littman, Judy Goldsmith, and Martin Mundhenk. The computational complexity of probabilistic planning. *Journal of AI Research*, 9:1–36, 1998.

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

[Madani *et al.*, 2003] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artif. Intell.*, 147(1-2):5–34, 2003.

[Papadimitriou and Tsitsiklis, 1987] Christos Papadimitriou and John N. Tsitsiklis. The complexity of markov decision processes. *Math. Oper. Res.*, 12(3):441–450, 1987.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. In *FOCS*, 1977.

[Puterman, 2005] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 2005.

[Rudary and Singh, 2003] Matthew R. Rudary and Satinder P. Singh. A nonlinear predictive state representation. In *NIPS'03*, pages 855–862, 2003.

[Singh *et al.*, 2001] Satinder P. Singh, Michael L. Littman, and Richard S. Sutton. Predictive representations of state. In *NIPS'01*, pages 1555–1561. MIT Press, December 2001.

[Singh *et al.*, 2003] Satinder P. Singh, Michael L. Littman, Nicholas K. Jong, David Pardoe, and Peter Stone. Learning predictive state representations. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 712–719, August 2003.

[Tabakov and Vardi, 2005] Deian Tabakov and Moshe Y. Vardi. Experimental evaluation of classical automata constructions. In *LPAR*, 2005.

[Vardi, 2011] Moshe Y. Vardi. The rise and fall of linear time logic. In *GandALF*, 2011.

[Zhang *et al.*, 2012] Zongzhang Zhang, Michael L. Littman, and Xiaoping Chen. Covering number as a complexity measure for POMDP planning and learning. In *AAAI'12*, 2012.