Regular Decision Processes: Modelling Dynamic Systems without Using Hidden Variables[*]

Extended Abstract

Ronen I. Brafman
Ben-Gurion University
Beer-Sheva, Israel
brafman@cs.bgu.ac.il

Giuseppe De Giacomo
Sapienza Università di Roma
Rome, Italy
degiacomo@dis.uniroma1.it

ABSTRACT

We describe *Regular Decision Processes* (RDPs) a model in between MDPs and POMDPs. Like in POMDPs, the effect of an action may depend on the entire history of actions and observations, but this dependence is restricted to regular functions only. This makes RDP a tractable, yet rich model, that does not hypothesize hidden state, and could possibly be useful for learning dynamic systems.

ACM Reference Format:
Ronen I. Brafman and Giuseppe De Giacomo. 2019. Regular Decision Processes: Modelling Dynamic Systems without Using Hidden Variables. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 3 pages.

1 INTRODUCTION

MDPs are a well known model of decision making in stochastic environments. POMDPs extend them to partially observable environments. An optimal policy for an MDP can be computed in polynomial time, and there are good algorithms for PAC learning a near-optimal policy in an MDP [3, 7]. Solving a POMDP is much harder. Depending on the precise assumptions, its complexity ranges from PSPACE-hard to undecidable [8–10]. Learning them is also much harder: few algorithms exist that provide some gurantees (e.g., [14]) but they are not too useful w.r.t. sample complexity.

This paper introduces *Regular Decision Processes*, or RDPs, an intermediate model between MDPs and POMDPs. Various ideas behind RDPs, such as MDPs with temporally extended rewards [1, 2, 13], and k-order MDPs have been discussed in the literature, as well as models such as Predictive State Representations (PSRs), that also attempt to do away with latent variables [11]. However, we believe this paper is the first to formally define a model that embodies these intuitions, grounding it in advances in the area of temporal logic [5].

RDPs generalize k-order MDPs, and can capture some POMDPs models, but rely on observable variables only. In k-order MDPs, the future depends on the last k states. In POMDPs, the future depends on the entire history of observations. In RDPs, the future depends, also on the entire history of observations, but this dependence is restricted – the past determines the future only through regular functions of the sequence of past observations. This means that that this dependence can be described declaratively and succinctly using regular expressions, or more precisely, through formulas in dynamic logic over finite traces – LDL_f [5] that have the same expressive power, but nicer properties. Thus, like PSRs [11], RDPs make no reference to hidden state. This makes them a promising target model for studying learning in non-Markovian environments and in partially observable environments.

In this extended abstract, we provide a few central results on RDPs, their complexity, and how they can be solved, and briefly discuss the potential for learning them,

2 RDPS

A (factored) RDP $M_L = \langle \mathcal{P}, A, S, tr_L, r_L, s_0 \rangle$ consists of a set of propositions $\mathcal{P}$, defining a set of states S, which are all possible truth assignments over $\mathcal{P}$, a set of actions A, and an initial state s_0. Its 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 $\exists\, (\varphi, a, P', \pi(P')) \in T$ such that $s_0, ..., s_k \models \varphi$, and s_k and s' agree on all variables in $\mathcal{P} \setminus P'$.
(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: $r_L(s_0, \ldots, s_k) = \sum_{(\varphi, r) \in R \wedge s_0, \ldots, s_k \models \varphi} r$. As before, by definition r_L is bounded above and below.

[*]This work was supported by Israel Ministry of Science and Technology Grant 54178, the Helmsley Charitable Trust through the Agricultural, Biological and Cognitive Robotics Center of Ben-Gurion University, the Lynn and William Frankel Center, and Sapienza project "Immersive Cognitive Environments"

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada. © 2019 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

Consider the following LDL_f formulas and their use to model tr_L and r_L $\varphi_1 = \langle true^*; Rain; true^* \rangle end$ – it rained in the past. $\varphi_2 = \langle true^*; Rain; Rain; Rain \rangle end$ – it rained in the last 3 time steps (e.g., days).[1] $\varphi_3 = \langle true^*; Rain; (\neg(Tmp{>}5))^* \rangle end$ – 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. Now, if when driving from A to B after it has rained, followed by sub-zero temperature 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). Let A, B, d denote *at_A*, *at_B*, *damaged*. We can write: $(\varphi_4 \wedge \langle true^*; (A \wedge \neg d) \rangle end, drive, \{A, B, d\}), \pi)$, where $\pi(\neg A \wedge B \wedge d) = 0.1, \pi(\neg A \wedge \neg B \wedge d) = 0.1, \pi(\neg A \wedge B \wedge \neg d) = 0.8$. Now, suppose that if it has only rained and the temperature was not high since, then these probabilities drop to 0.01. We can write: $(\varphi_3 \wedge \neg\varphi_4 \wedge \langle true^*; (A \wedge \neg d) \rangle end, drive, \{A, B, d\}), \pi)$, 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 use LDL_f to capture the regular expression: $(\langle true^*; RqstAnn; (\neg DlvdAnn)^*; DlvdAnn \rangle end, 10)$.

3 SOLVING RDPS

RDPs 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. They are solved by transforming them into MDPs, but unlike the belief-space transformation of POMDPs, this transformation yields a finite MDP. The transformation process can be aided by automated tools for constructing a DFA from an LTL_f /LDL_f formula such as https://flloat.herokuapp.com [6], saving the modeller the effort and potential errors associated with attempting to transform them manually into MDPs.

Given RDP $M_L = \langle \mathcal{P}, A, S, tr_L, r_L, s_0 \rangle$, 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 quadruples in T: $(\varphi_1, a_1, P'_1, \pi(P'_1)), \ldots, (\varphi_m, a_m, P'_m, \pi(P'_m))$, 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. $2^{\mathcal{P}}$, 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 [2, 5]. The complexity of generating a deterministic automaton for φ_i is 2EXPTIME, and its size is doubly exponential in the worst case. It has been observed that, in practice, this transformation often does not involve exponential blow up and yields compact automata [12]. 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.)
- $tr((s, q_1, \ldots, q_m), a, (s', q'_1, \ldots, q'_m)) = tr_L(\bar{s}, a, s')$ if (1) there exists a (unique by assumption) $1 \le i \le m$ such that $q_i \in F_i$, (2) $\bar{s}$ is some trace that satisfies φ_i, and (3) for every $1 \le j \le m$ 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. 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.

We can show that the original RDP and the MDP generated using the method above have isomorphic computation trees (and hence sets of traces). That is, a given sequence of actions will yield identical distributions over sequences of states, and each such sequence of states will result in the same reward.

Complexity-wise, an optimal RDP policy can be computed in 2EXPTIME, while finding a policy with value $\ge c$ is 2EXPTIME-hard. 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 [4] which is a special case of stochastic planning, and a goal can be captured by a reward + transition to a sink state.

Furthermore, it is not difficult to show that every RDP has a *regular* policy that is optimal. 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 $s_0, \ldots, s_k \models \varphi_i$.

4 RDPS AND POMDPS

We conclude by discussing the relationship between RDPs and POMDPs. 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 is related to the transformation above, and in it, each hidden variable is essentially capturing the state of one of the automata. (This of course, can be encoded using boolean variables). By definition of the DFA, its state is a regular function of the history of observations.

The converse requires a little more effort to prove, but is also true. That is, if we have a factored POMDP with propositions defining its state, such that the value of each proposition is a regular function of the current history of observations, then there exists an equivalent RDP. Here equivalence implies again, that the computation trees are isomorphic, where each branch refers only to the *observable* aspects of the trace.

And what about general POMDPS? We can show that every infinite horizon POMDP can be approximated by an RDP, such that the optimal policy for the RDP, when applied to the POMDP, will yield an approximately optimal behavior for the latter.

[1]Note that, $\langle true^*; \varrho \rangle end$ means that sequence ϱ holds at the end.

REFERENCES

[1] Fahiem Bacchus, Craig Boutilier, and Adam J. Grove. 1996. Rewarding Behaviors. In *AAAI*.

[2] Ronen Brafman, Giuseppe De Giacomo, and Fabio Patrizi. 2018. LTLf/LDLf Non-Markovian Rewards. In *AAAI*.

[3] R. Brafman and M. Tennenholtz. 2002. R-max – A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research* 3 (2002), 213–231.

[4] Giuseppe De Giacomo and Sasha Rubin. 2018. Automata-Theoretic Foundations of FOND Planning for LTLf and LDL_f Goals. In *IJCAI*.

[5] Giuseppe De Giacomo and Moshe Y. Vardi. 2013. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *IJCAI*.

[6] Marco Favorito. 2018. *Reinforcement Learning for LTLf/LDLf Goals: Theory and Implementation*. Master's thesis. DIAG, Sapienza Univ. Rome.

[7] Michael J. Kearns and Satinder P. Singh. 2002. Near-Optimal Reinforcement Learning in Polynomial Time. *Machine Learning* 49, 2-3 (2002), 209–232.

[8] M. L. Littman, J. Goldsmith, and M. Mundhenk. 1998. The computational complexity of probabilistic planning. *Journal of AI Research* 9 (1998), 1–36.

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

[10] C. Papadimitriou and J. N. Tsitsiklis. 1987. The complexity of Markov decision processes. *Math. Oper. Res.* 12, 3 (1987), 441–450.

[11] S. Singh, M. L. Littman, and R. S. Sutton. 2001. Predictive Representations of State. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*. MIT Press, 1555–1561.

[12] Deian Tabakov and Moshe Y. Vardi. 2005. Experimental Evaluation of Classical Automata Constructions. In *LPAR*.

[13] Sylvie Thiébaux, Charles Gretton, John K. Slaney, David Price, and Froduald Kabanza. 2006. Decision-Theoretic Planning with non-Markovian Rewards. *J. Artif. Intell. Res. (JAIR)* 25 (2006), 17–74.

[14] Zongzhang Zhang, Michael L. Littman, and Xiaoping Chen. 2012. Covering Number as a Complexity Measure for POMDP Planning and Learning. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.* http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/4906