**Abstract.** Reward Machines (RMs) provide a structured, automata-based representation of a reward function that enables a Reinforcement Learning (RL) agent to decompose an RL problem into structured subproblems that can be efficiently learned via off-policy learning. Here we show that RMs can be learned from experience, instead of being specified by the user, and that the resulting problem decomposition can be used to effectively solve partially observable RL problems. We pose the task of learning RMs as a discrete optimization problem whose objective is to find an RM that decomposes the problem into a set of subproblems such that the combination of their optimal memoryless policies is an optimal policy for the original problem. We show the effectiveness of this approach on three partially observable domains, where it significantly outperforms A3C, PPO, and ACER, and discuss its advantages, limitations, and broader potential.

**Learning Reward Machines for Partially Observable Reinforcement Learning**

**Rodrigo Toro Icarte**† Ethan Waldie‡ Toryn Q. Klassen†† Richard Valenzano§ Margarita P. Castro‡‡ Sheila A. McIlraith††

†Dept of Computer Science, University of Toronto ‡Dept of Mechanical and Industrial Engineering, University of Toronto §Vector Institute ‡‡Element AI

†(toro, ewaldie@toronto), toshi, shula@cs.toronto.edu, ‡mpcastro@ai.toronto.ca, ‡‡rick.valenzano@elementai.com

---

**Q-Learning Reward Machines (QRM)**

- **Q-Learning** policy (q-function) per state in the RM.
- Select actions using the policy of the current RM state.
- Reuse experience to update all the q-values at the same time.

**Q-Learning**

1. First approach for learning RMs from experience.
2. Extended RMs and QRM to work under partial observability.
3. Developed a theory for 1 and 2.

**Main Contributions**

**Our 2018 RM Results**

- **Learning Reward Machines (RM)**
- **Q-Learning**
- **QRM**

**RMs under Partial Observability**

RMs are defined over a set of propositional symbols \( \mathcal{P} \) that correspond to high-level events (e.g., \( \mathcal{P} = \{ \text{good}, \text{bad}, \text{repeat} \} \)) that the agent can detect using a labelling function \( L : O \times A \rightarrow 2^\mathcal{P} \).

**Def.** A Reward Machine is a tuple \( R = (U, \delta, \rho, \theta) \) where \( U \) is a finite set of states, \( \delta \in U \times A \times U \) is an initial state, \( \theta \) is the state-transition function, \( \delta : U \times 2^A \rightarrow U \), and \( \rho \) is the reward-transition function.

**Optimization Model**

- **Inputs:**
  1. A set of traces: \( T = \{ t_1, \ldots, t_n \} \) where each trace \( t_i \) is:
     \[ T_i = (o_{i1}, a_{i1}; o_{i2}, a_{i2}; \ldots; o_{ik}, a_{ik}) \]
    2. A labelling function \( L : O \times A \rightarrow 2^\mathcal{P} \).

- **Model:**
  \[
  \min_{(U, \delta, \rho, \theta)} \sum_{i \in \mathcal{P}} \sum_{t \in T_i} \log(N_{t,i} + \epsilon) \\
  \text{s.t.} (1) \quad |U| \leq n_{\text{states}} \\
  \text{s.t.} (2) \quad x_{(i)} \in L(o_{i1}, a_{i1}, o_{i2}) \\
  \text{s.t.} (3) \quad \forall i \in I, t \in T_i \cup T_j \\
  \text{s.t.} (4) \quad x_{(i)} \in L(o_{i1}, a_{i1}, o_{i2}) \\
  \text{s.t.} (5) \quad x_{(i)} \in L(o_{i1}, a_{i1}, o_{i2}) \\
  \text{s.t.} (6) \quad \forall i, t \in T_i, j \in T_j \\
  \text{s.t.} (7) \quad \forall i, t \in T_i, j \in T_j \\
  \text{s.t.} (8) \quad x_{(i)} \in L(o_{i1}, a_{i1}, o_{i2})
  \]

**Perfect Reward Machines**

Perfect RMs make the environment Markovian w.r.t. \( O \times U \), i.e.,

\[
\Pr(o_{i1}, a_{i1}; o_{i2}, a_{i2}; \ldots; o_{ik}, a_{ik}) = \Pr(o_{i1}, a_{i1}; o_{i2}, a_{i2}; \ldots; o_{ik}, a_{ik})
\]

for every possible trace \( o_{i1}, a_{i1}, o_{i2}, a_{i2}, \ldots, o_{ik}, a_{ik} \) generated by any policy.

**Properties:** Given an environment \( \mathcal{E} \):

1. If the set of belief states of \( \mathcal{E} \) is finite, then there exists a perfect RM for \( \mathcal{E} \), and (ii) Optimal policies over \( O \times U \) for perfect RMs are also optimal for \( \mathcal{E} \).

**Theorem (necessary conditions):**

When \( \mathcal{E} \rightarrow \infty \), any perfect RM is an optimal solution of LRM.

**Output:**

1. A Reward Machine: \( (U, \delta, \rho, \theta) \in \mathcal{P}_R \).
2. A set of possible next high-level observations \( N_{(i),i} \in \mathcal{P} \).

**Experimental Remarks**

- The binary classifiers were used by all the approaches.
- LRM works because Tabu search finds high-quality RMs.
- QRM pays off in domains with sparse rewards.
- Code: bitbucket.org/RToroiicarte/lrm.