# Reward Machines for Deep RL in Noisy and Uncertain Environments


Andrew C. Li[1,2] Zizhao Chen[3†] Toryn Q. Klassen[1,2,4] Pashootan Vaezipoor[2,5] Rodrigo Toro Icarte[6,7] Sheila A. McIlraith[1,2,4]

[1]University of Toronto [2]Vector Institute [3]Cornell University [4]Schwartz Reisman Institute [5]Georgian.io [6]Pontificia Universidad Católica de Chile [7]Centro Nacional de Inteligencia Artificial

[†]Work done while at the University of Toronto


UNIVERSITY OF TORONTO

VECTOR INSTITUTE

Paper Code

## TL;DR

**We show how to leverage formal reward function specifications (e.g. Reward Machines, LTL) in RL environments where key properties/events are uncertain.**

## Example: Gold Mining Robot

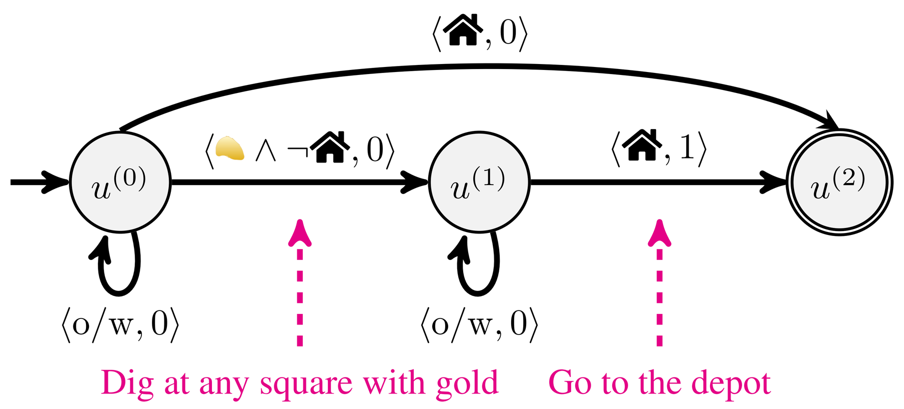
**Robot's task**: **dig at any square with gold, then go to the depot.**
**Uncertain property:** the robot cannot reliably detect gold because it looks like fool's gold.

How can the agent reliably solve the task?

*(Each grid square is labelled with the agent's probabilistic belief it yields gold.)*

### Gridworld


### Reward Machine




## Motivation

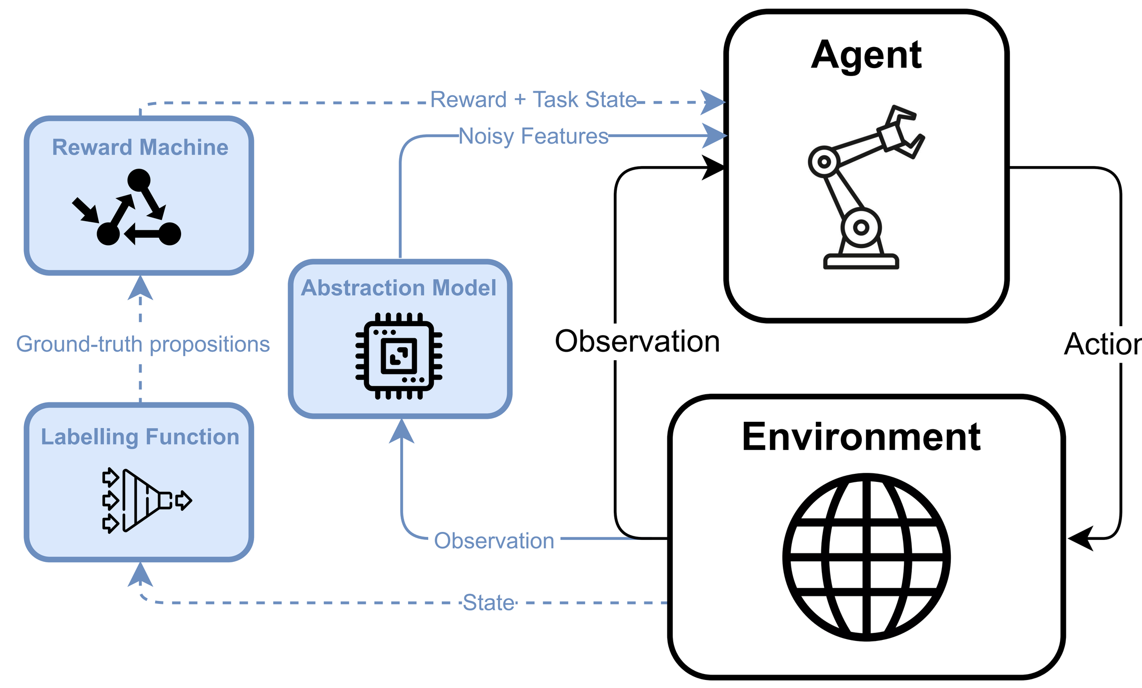
- ✔ Formal specifications expose reward function structure to enable sample-efficient RL.
- 🔍 Current methods often assume an oracle detects key properties/events in the environment (a **labelling function**).
- ⚠ In the real world, **key properties/events can be uncertain** due to partial observability, sensor noise, or classifier error.

## Contributions

1. A **deep RL framework for Reward Machines** (RMs) where the evaluation of the symbolic vocabulary is uncertain.
2. A **suite of RL algorithms** that exploit RM structure and can be deployed without an oracle labelling function.
3. An **analysis** showing serious **pitfalls** when naively addressing uncertainty (and how to overcome these pitfalls!).

- 💡 Our framework extends RMs to **POMDP** environments.
- 💡 We show how noisy **sensors** and **vision-language models** can be incorporated into an RM framework.

## Framework




**Key Concepts**

1. The reward function is defined by an RM over propositional symbols $\mathcal{AP}$ (e.g. ⟨, 🏠⟩).
2. The **ground-truth interpretation of $\mathcal{AP}$** is a binary function of *state transitions* provided by a *hidden* labelling function $\mathcal{L} : S \times A \times S \to 2^{\mathcal{AP}}$ (**hard to obtain!**).
3. An **abstraction model** $\mathcal{M} : H \to Z$ (e.g. a VLM or sensors) is a *noisy* mapping from the *observable history* to an arbitrary feature space $Z$ (**easier to obtain!**).

**Agent's goal:** Optimize performance with respect to the ground-truth interpretation $\mathcal{L}$, while deciding actions according to a noisy abstraction model $\mathcal{M}$.

## Methods

**Question: How can we leverage high-level reward function structure using only a noisy abstraction model $\mathcal{M}$?** We consider policies decoupled into two modules:

- ⚙ **Inference:** Predict a **belief over RM states** at each step from the history with the help of the abstraction model $\mathcal{M}$.
- ⚙ **Decision making:** Output actions based on observations + RM state belief.

**Naive algorithm:** Given $\mathcal{M} : H \to 2^{\mathcal{AP}}$ (a *discrete* predictor of $\mathcal{AP}$), update the RM state by treating outputs of $\mathcal{M}$ as the ground truth.

**Pitfall:** A prediction error at time $t$ propagates to all future RM states at time $t+1, t+2, \ldots$!

**Independent Belief Updating (IBU):** Given $\mathcal{M} : H \to \Delta(2^{\mathcal{AP}})$ (a *probabilistic* predictor of $\mathcal{AP}$), maintain a *distribution* over RM states at each timestep.

**Pitfall:** Assumes outputs from $\mathcal{M}$ are independent. The inferred RM state distribution can get arbitrarily bad over time!

**Temporal Dependency Modelling (TDM):** Given $\mathcal{M} : H \to \Delta U$ (a probabilistic sequence predictor of RM states), directly output the predicted RM state distribution at each timestep.

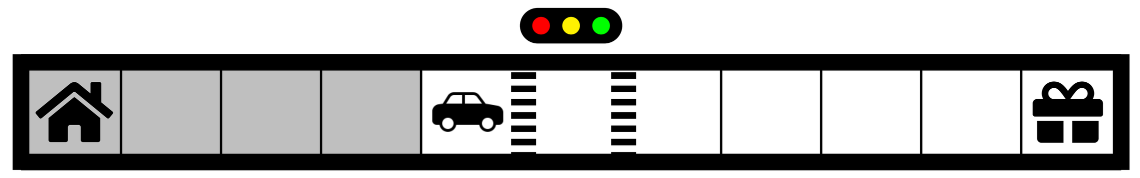
- ⚙ Can be practically implemented using RNNs, Transformers, etc.

## Experiments

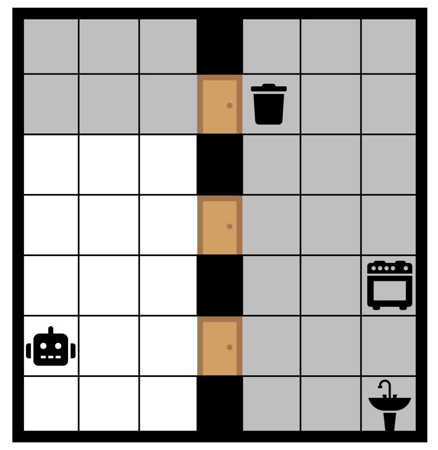
**Research Questions:**

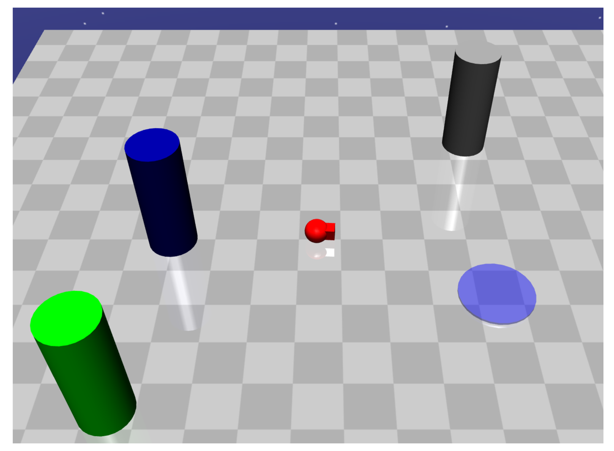
1. Which methods are robust to noisy abstraction models when applied to our RM framework?
2. Which methods improve downstream RL sample efficiency?

We target environments with **partial observability** and **high-dimensional observations**, while abstraction models include **neural network classifiers** trained from data, and **zero-shot GPT-4o**.



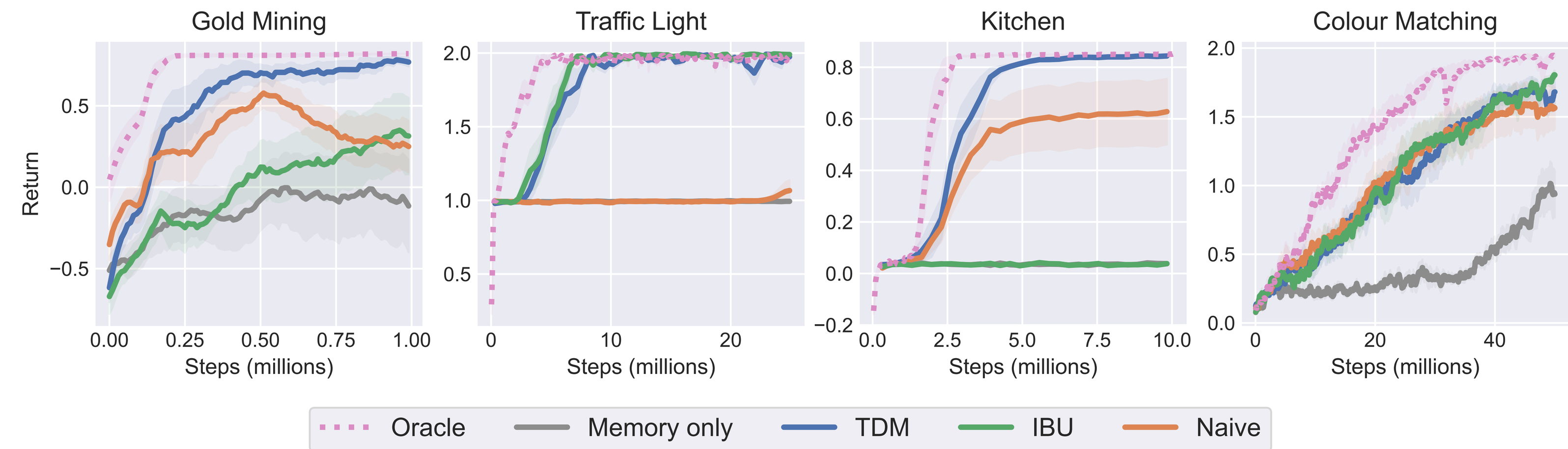
**Traffic Light** (*above*) and **Kitchen** (*below left*), are MiniGrids with image observations, where key propositions are partially observable.



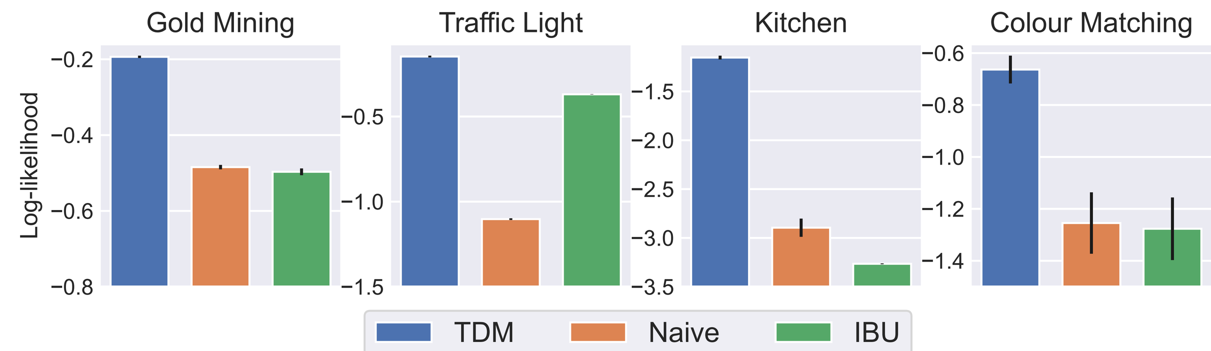


**Colour Matching** (*above right*) is a MuJoCo robotics environment where the agent must identify colour names by their RGB values to solve a sequential reach-avoid task.

### RL Curves




### RM State Belief Accuracy




- ✔ Our methods can be deployed with noisy abstraction models, instead of an oracle.
- ✔ TDM is significantly more sample efficient than Recurrent PPO!
- ⚠ RL makes little progress when task structure is not exploited.
- ⚠ Naive and IBU can lead to dangerous outcomes due to RM state modelling errors!