# Induction and Exploitation of Subgoal Automata for Reinforcement Learning

Daniel Furelos-Blanco[1], Mark Law[1], Anders Jonsson[2], Krysia Broda[1], and Alessandra Russo[1]

[1]Imperial College London, UK [2]Universitat Pompeu Fabra, Barcelona, Spain

Imperial College London · upf. Universitat Pompeu Fabra Barcelona

## Motivation

- Exploit an abstract subgoal structure of a task.
- Temporal abstractions have been represented using *automata* across several areas, like *reinforcement learning (RL)* and *automated planning*.

### Problem

Current RL methods use *handcrafted automata*.

## Proposed Approach

### ISA (Induction of Subgoal Automata)

A method for learning and exploiting a minimal automaton from observation traces perceived by an RL agent.

- Learn an automaton whose transitions are labeled by propositional formulas representing *subgoals*.
- The *automata learning* is formulated as an *inductive logic programming* task, and *sped up* using a *symmetry breaking* mechanism.
- The automata can be exploited by RL algorithms.

## Tasks

The tasks are *episodic POMDPs* $\mathcal{M}^{\Sigma} = \langle S, S_T, S_G, \Sigma, A, p, r, \gamma, \nu \rangle$ where:

- $S$ is a set of *latent* states,
- $S_T \subseteq S$ is a set of *terminal* latent states,
- $S_G \subseteq S_T$ is a set of *goal* latent states,
- $\Sigma$ is a set of *visible* states,
- $\nu : S \to \Delta(\Sigma)$ is a mapping from latent states to probability distributions over visible states,
- $A$, $p$, $r$ and $\gamma$ are defined as for MDPs.

> - Tasks are enhanced with a set of propositions $\mathcal{O}$ called *observables*.
> - A *labeling function* $L : \Sigma \to 2^{\mathcal{O}}$ maps a state into subsets of observables called *observations*.

**Interaction** At step $t$, the agent observes a tuple $\boldsymbol{\sigma}_t = \langle \sigma_t^{\Sigma}, \sigma_t^{T}, \sigma_t^{G} \rangle$, where:

- $\sigma_t^{\Sigma} \in \Sigma$ is a visible state,
- $\sigma_t^{T} = \mathbb{I}[s_t \in S_T]$ indicates if the latent state is terminal, and
- $\sigma_t^{G} = \mathbb{I}[s_t \in S_G]$ indicates if the latent state is a goal state.

**Example** In the OfficeWorld (Toro Icarte et al., 2018), where $\mathcal{O} = \{☕, \boxtimes, o, A, B, C, D, *\}$, 'deliver coffee to the office while avoiding the $*$'.

- Latent state: $(x, y, has\_☕?)$
- Goal state: $(4, 4, \top)$
- Visible state: $(x, y)$
- Terminal states: $(4, 7, \top), (4, 7, \bot), \ldots$

**Assumptions**

1. The Markov property can be obtained through the combination of visible states and histories of observations.
2. A history of observations is sufficient to determine whether a terminal state is reached and, if so, whether it is a goal state.

## Learning Subgoal Automata from Traces

### Input

- A set of states $U \supseteq \{u_0, u_A, u_R\}$.
- A set of observables $\mathcal{O}$.
- A set of traces $\Lambda^G_{L,\mathcal{O}} \cup \Lambda^D_{L,\mathcal{O}} \cup \Lambda^I_{L,\mathcal{O}}$.
- A max. number of edges $\kappa$ between states.

### Output

The automaton's transition function such that the automaton:

- accepts all *goal traces* $\Lambda^G_{L,\mathcal{O}}$,
- rejects all *dead-end traces* $\Lambda^D_{L,\mathcal{O}}$,
- neither accepts nor rejects *incomplete traces* $\Lambda^I_{L,\mathcal{O}}$.

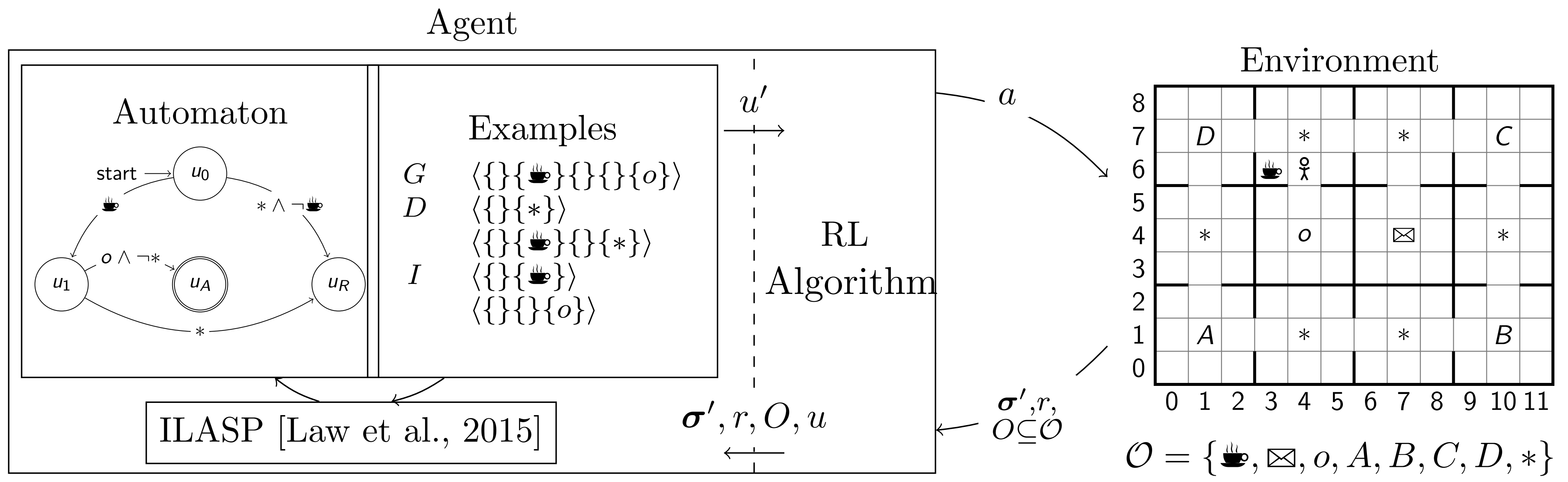
### Example (simplified)

**Transition Function:** $\texttt{ed}(\texttt{u}_0, \texttt{u}_1, 1).\ \texttt{ed}(\texttt{u}_1, \texttt{u}_\texttt{A}, 1).$

$\delta(\texttt{u}_0, \texttt{u}_1, 1, \texttt{T}) \text{ :- } \texttt{obs}(☕, \texttt{T}), \texttt{step}(\texttt{T}).$

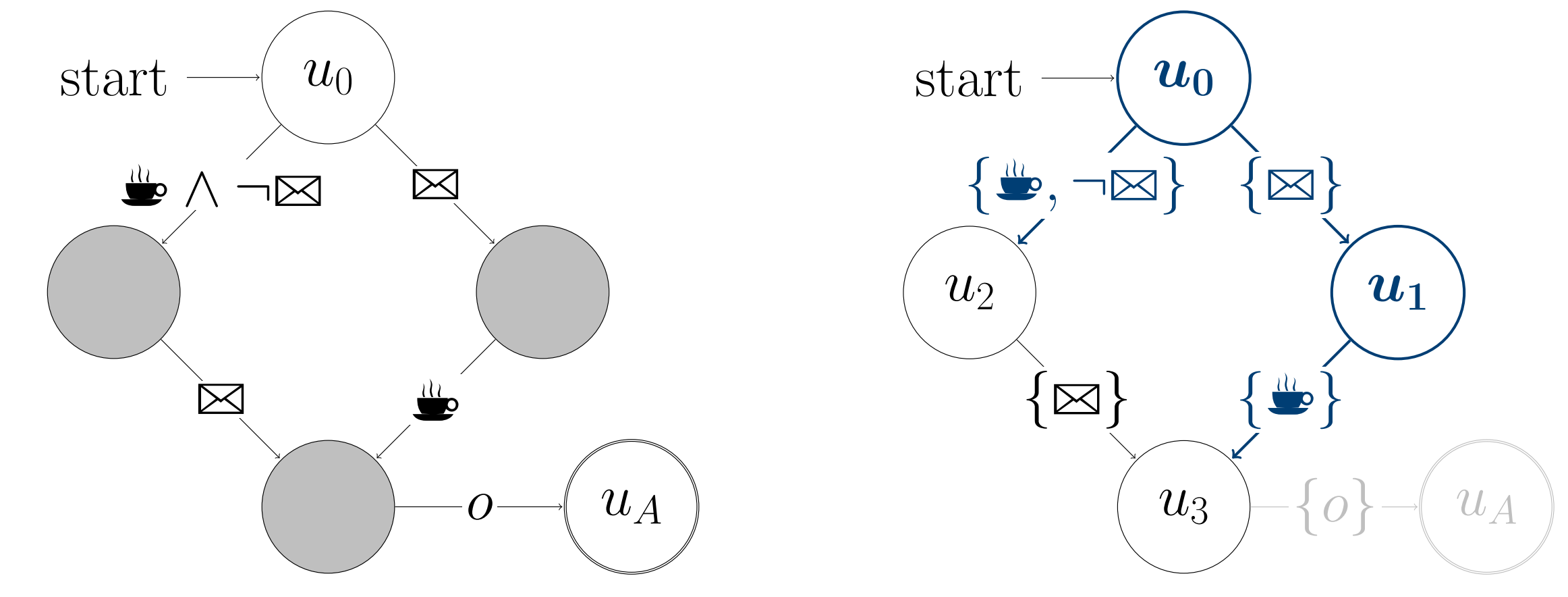
$\delta(\texttt{u}_1, \texttt{u}_\texttt{A}, 1, \texttt{T}) \text{ :- } \texttt{obs}(o, \texttt{T}), \texttt{not obs}(*, \texttt{T}), \texttt{step}(\texttt{T}).$

**Trace:** $\texttt{obs}(☕, 1).\ \texttt{obs}(o, 4).$


Agent
Automaton
start
$u_0$
☕
$* \wedge \neg$☕
$u_1$
$o \wedge \neg *$
$u_A$
$u_R$
$*$
Examples
$G$ $\langle\{\}\{☕\}\{\}\{\}\{o\}\rangle$
$D$ $\langle\{\}\{*\}\rangle$
$\langle\{\}\{☕\}\{\}\{*\}\rangle$
$I$ $\langle\{\}\{☕\}\rangle$
$\langle\{\}\{\}\{o\}\rangle$
ILASP [Law et al., 2015]
$u'$
RL Algorithm
$\boldsymbol{\sigma}', r, O, u$
$a$
$\boldsymbol{\sigma}', r, O \subseteq \mathcal{O}$
Environment
D * * C
☕
* o ✉ *
A * * B
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8
$\mathcal{O} = \{☕, \boxtimes, o, A, B, C, D, *\}$

## Symmetry Breaking

Encode rules that impose a unique *BFS traversal*.



## RL Algorithms

**Hierarchical RL with Options (Sutton et al., 1999)**

- Two decision *levels*:
  1. From a given automaton state, choose a subgoal.
  2. Given a subgoal, choose primitive actions.
- The policy can be *suboptimal*.

**QRM (Toro Icarte et al., 2018)**

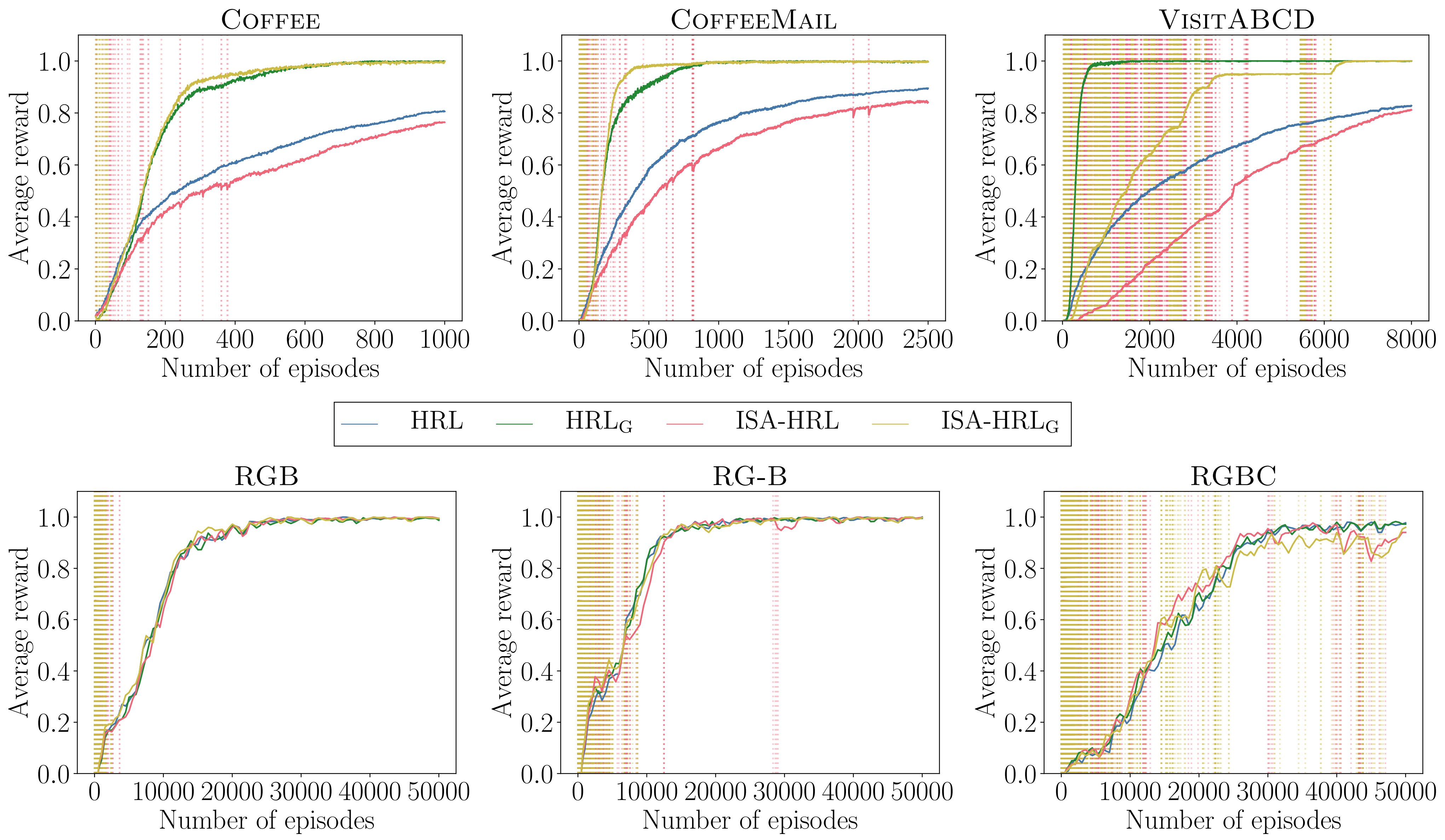
- Transform automaton into a *reward machine*.
- Learns a policy over $\Sigma \times U$.
- Guarantees *optimality* in the tabular case.

## Interleaved Learning Algorithm

RL and automata learning are *interleaved*.

- The *initial automaton* does not accept nor reject anything.
- ILASP runs when a *counterexample* is found.
- The number of states increases when the automaton learning task is UNSAT (iterative deepening).

→ A *minimal* automaton is found for a specific $\kappa$.



## Experiments

- Domains: OfficeWorld (Toro Icarte et al., 2018), CraftWorld (Andreas et al., 2017; Toro Icarte et al., 2018) and WaterWorld (Toro Icarte et al., 2018).
- Given a set of task instances, simultaneously:
  - learn a policy for each of these, and
  - an automaton that generalizes to all of them.
- Default automaton learning: symmetry breaking, acyclicity, $\kappa = 1$, trace compression.
- 20 runs for each experiment.

| | Time (s.) | # Examples | | | | Example Length |
|---|---|---|---|---|---|---|
| | | All | $G$ | $D$ | $I$ | |
| Coffee | 0.4 (0.0) | 8.7 (0.4) | 2.4 (0.1) | 3.0 (0.1) | 3.2 (0.3) | 2.8 (2.1) |
| CoffeeMail | 18.9 (3.3) | 29.0 (1.5) | 3.9 (0.3) | 9.3 (0.6) | 15.8 (1.0) | 4.0 (2.6) |
| VisitABCD | 163.2 (44.3) | 54.9 (3.8) | 1.6 (0.1) | 15.2 (0.9) | 38.1 (3.1) | 5.5 (3.1) |

Automaton learning statistics for OfficeWorld tasks (HRL$_G$)

| | Acyclic | | Cyclic | |
|---|---|---|---|---|
| | No SB | SB | No SB | SB |
| Coffee | 0.5 (0.0) | 0.4 (0.0) | 0.5 (0.0) | 0.5 (0.0) |
| CoffeeMail | 277.4 (70.2) | 18.9 (3.3) | 4204.3 (1334.4)* | 774.7 (434.4) |
| VisitABCD | 1070.0 (725.6) | 163.2 (44.3) | 3293.5 (1199.2)* | 1961.7 (1123.8) |

Total automaton learning time when symmetry breaking is disabled (No SB) and enabled (SB) using HRL$_G$. $*$ = timed out 1-10 runs.

Andreas, J., Klein, D., and Levine, S. (2017). Modular multitask reinforcement learning with policy sketches.

Law, M., Russo, A., and Broda, K. (2015). The ILASP System for Learning Answer Set Programs.

Sutton, R. S., Precup, D., and Singh, S. P. (1999). Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning.

Toro Icarte, R., Klassen, T. Q., Valenzano, R. A., and McIlraith, S. A. (2018). Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning.