Verifiable and Compositional Reinforcement Learning Systems

Cyrus Neary [1], Christos Verginis [2], Murat Cubuktepe [1], Ufuk Topcu [1]
[1] The University of Texas at Austin, USA
[2] Uppsala University, Sweden

The central question

How can we build compositional reinforcement learning systems with verifiable properties? How can we break tasks into specific requirements for subsystems?

Why build compositional RL systems?

To reduce the complexity of individual subsystems. System-level requirements may be decomposed into component level ones. Each component may be developed and tested independently, and the satisfaction of component-level requirements may be used to place assurances on the system as a whole.

A summary of the approach

We build a *high-level* system model capturing the interfaces between subsystems. The model is used to ***automatically*** synthesize subtask specifications for the *low-level* subsystems, each of which is implemented as an independent RL agent.

Novel capabilities of the framework:

1. Automatic decomposition of task specifications.
2. Targeted subsystem training to satisfy subtask specifications.
3. Iterative refinement of subtask specifications.
4. Modularity: prediction and verification in task transfer.

Problem formulation

Environment: Modeled as an unknown Markov decision process.

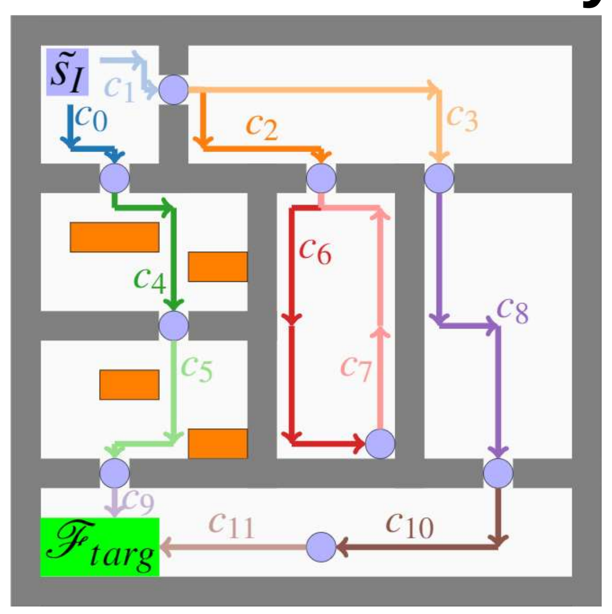
Task specification: Reach a target set of states with a specified probability of success.

Subsystems: Deploy RL-trained policies to accomplish subtasks.

Compositional system: A meta-policy that deploys subsystems in order to accomplish the task specification.

The Problem: Synthesize subtask performance requirements, train RL subsystems to satisfy them, and compute a meta-policy such that the compositional system satisfies the task specification.

An illustrative labyrinth navigation example

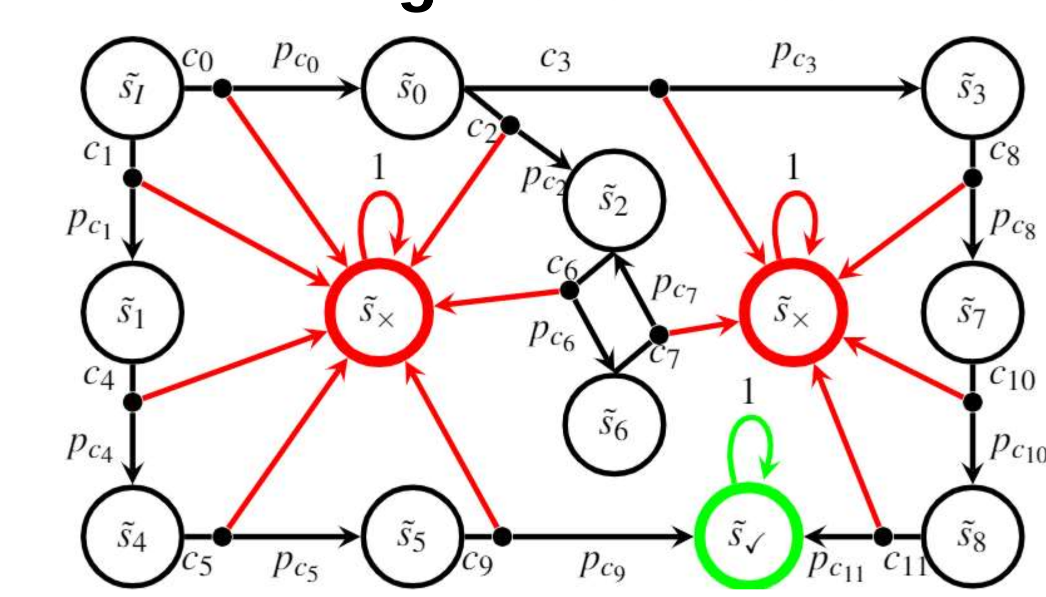


Task specification: With probability $1-\delta$, reach $\mathcal{F}_{targ}$ from initial state $\tilde{s}_I$ while avoiding unsafe lava states.

The labyrinth is broken into subsystems: each room corresponds to a subtask.

Subsystems: c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9, c_{10}, c_{11}

The high-level model



High-level states defined by subtask entrance and exit conditions.

High-level actions c_i correspond to subtasks.

Transition probabilities p_{c_i} correspond to probabilities of subtask success.

Failure states are reached upon failure of any of the subtasks.

Success states are reached upon completion of the overall task.

The high-level model is a *parametric* Markov decision process.

Relating the high-level model to the environment

If $\boxed{\text{HLM Transition Probability } p_c} \le \boxed{\text{Success probability of subsystem } c}$, for every subsystem c

Then $Prob\{\text{Reaching } \tilde{s}_\checkmark \text{ in the HLM}\} \le Prob\{\text{Compositional system completes its task in the true environment.}\}$

Automatic decomposition of task specifications

How can we use the high-level model to break task specifications into necessary requirements for the subsystems?

Interpret HLM transition probability values p_c as *subtask specifications* — required probabilities of subtask success.

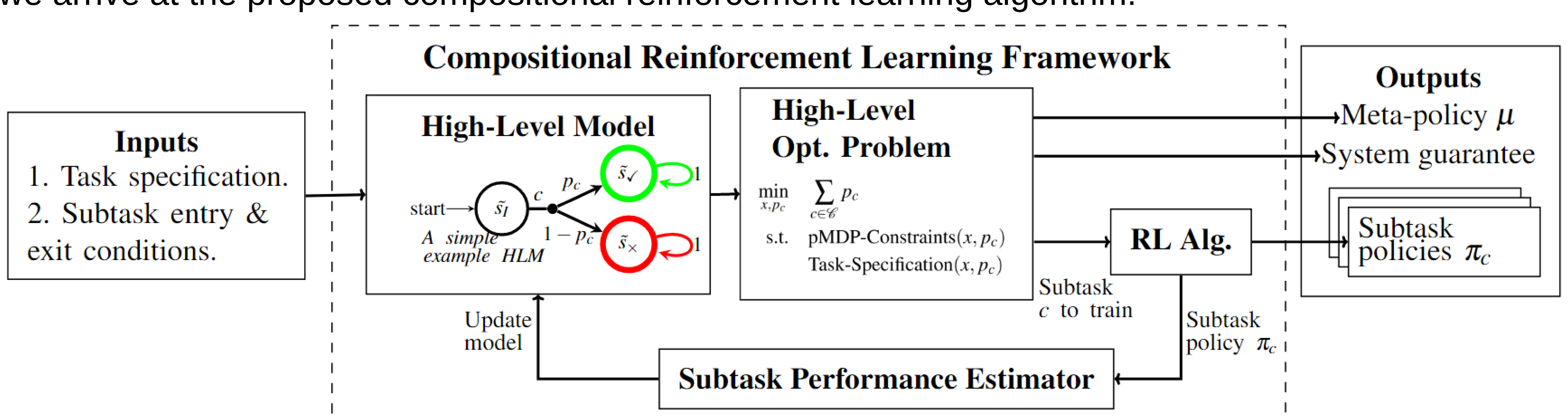
Synthesize small subtask specifications values p_c, such that a composite system exists that satisfies the task specification.

$$\min_{p_c} \sum_c p_c$$
$$\text{s.t.} \quad \text{Additional constraints}(p_c)$$
$$1-\delta \le Prob\{\text{Reaching } \tilde{s}_\checkmark \text{ in the HLM}\}$$

Encoding of the High-level model

Iterative compositional reinforcement learning

By iterating between synthesizing subtask specifications and training subsystems to satisfy them, we arrive at the proposed compositional reinforcement learning algorithm:



Numerical experiments

We tested the framework on **discrete** and **continuous** versions of the labyrinth navigation example.

The framework automatically identified two candidate routes.

Route 1: Short but risky route using c_0, c_4, c_5, c_9 to navigate past the lava to reach the goal.

Route 2: Long but reliable route using c_1, c_3, c_8, c_{10}, c_{11} to reach to the goal while avoiding the lava altogether.

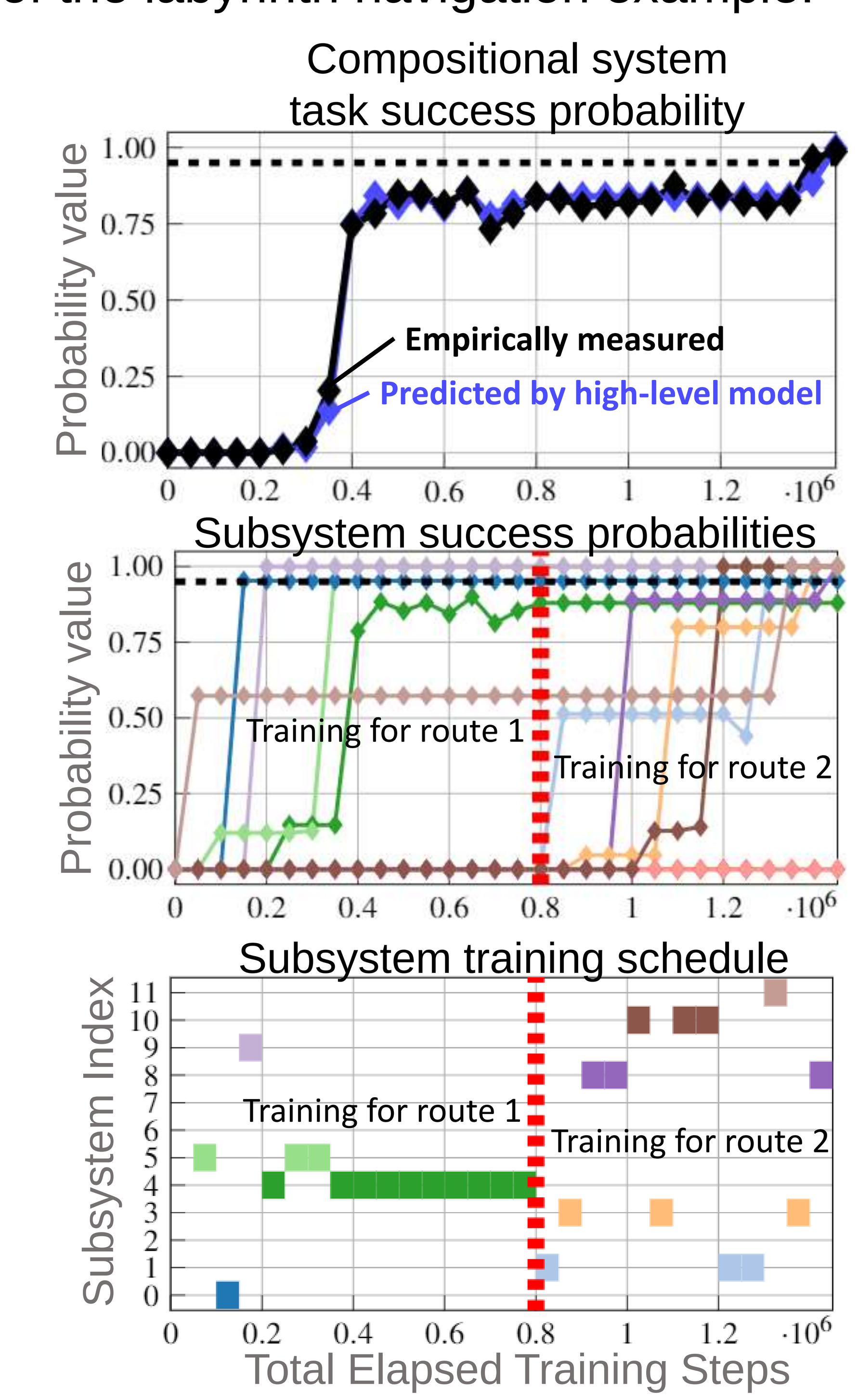
Initially, the algorithm trains the subsystems for route 1. When subsystem c_4 is unable to meet its subtask specification due to the risk posed by the lava, the algorithm automatically re-routes, and begins training the subsystems for route 2.

Once all the subsystems for route 2 meet their subtask specifications, their composition satisfies the task specification.

Automatically generated subtask specification values

Subsystem Index	0	1	2	3	4	5	6	7	8	9	10	11
Route 1	.97	.00	.00	.00	.97	1.0	.00	.00	.00	1.0	.00	.57
Route 2	.95	.99	.00	.99	.88	1.0	.00	.00	.99	1.0	.99	.99

Indicates that the subsystem is deployed by the route.



Contact: cneary@utexas.edu |