visit **www.rlang.ai**

A Declarative Language for Expressing Prior Knowledge for Reinforcement Learning

Rafael Rodriguez-Sanchez Benjamin Spiegel Jennifer Wang Roma Patel Stefanie Tellex George Konidaris
Department of Computer Science, Brown University

Motivation

- ➔ Learning tabula rasa is hard and requires an unreasonable number of samples.
- ➔ When teaching humans a new task, it is natural to lend advice to speed-up learning. This advice typically contains partial information about goal states, rewards, best actions, relevant intermediate tasks, etc.
- ➔ We sought to formalize advice-giving so that humans can easily provide meaningful guidance to RL agents.

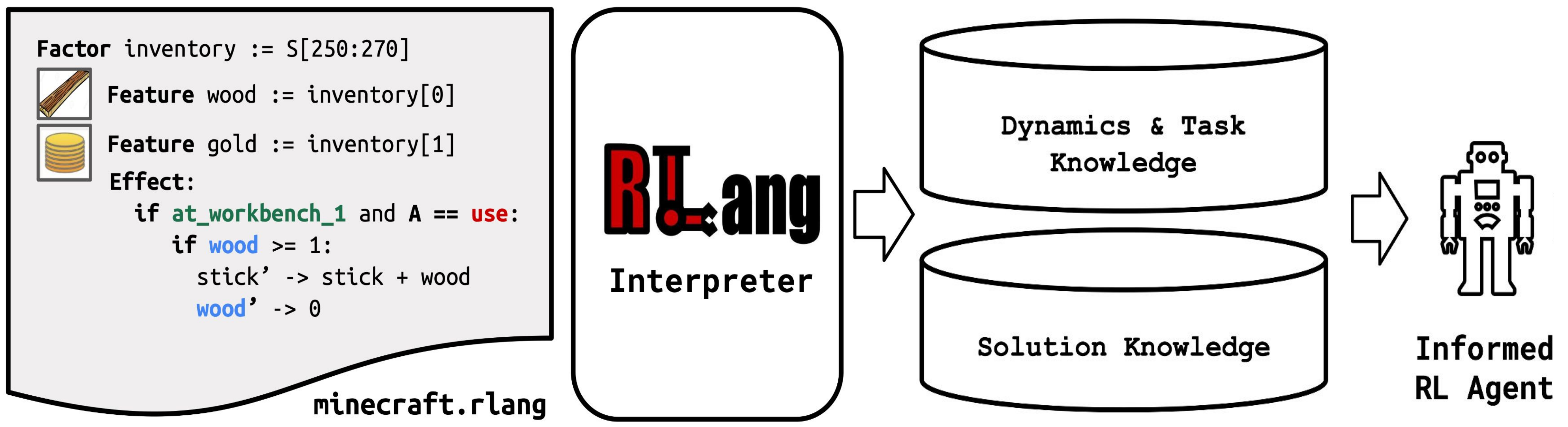
What can we express with RLang?

- ➔ RLang provides a formal, unambiguous, and unifying framework for expressing task-specific information.
- ➔ RLang provides syntax to specify information about an MDP's:
  - ◆ **Model**: Rewards and Dynamics,
  - ◆ **Solution**: Policy Hints and Policy Priors,
  - ◆ **Abstractions and Features**: Subpolicies (as Options) and State Features.

What's next?

- ➔ RLang unifies the varied features of other DSLs previously proposed; thus, it can more effectively be used as a universal store of symbol-oriented knowledge for RL agents in natural language grounding research;
- ➔ RLang enables research on **neuro-symbolic RL**; agents can reason and behave using both symbolic *and* latent knowledge/policies.
- ➔ RLang enables research in general ***informed* RL** methods.

RLang proposes a unified system for providing RL agents with task-specific, grounded advice that helps them learn faster than tabula rasa



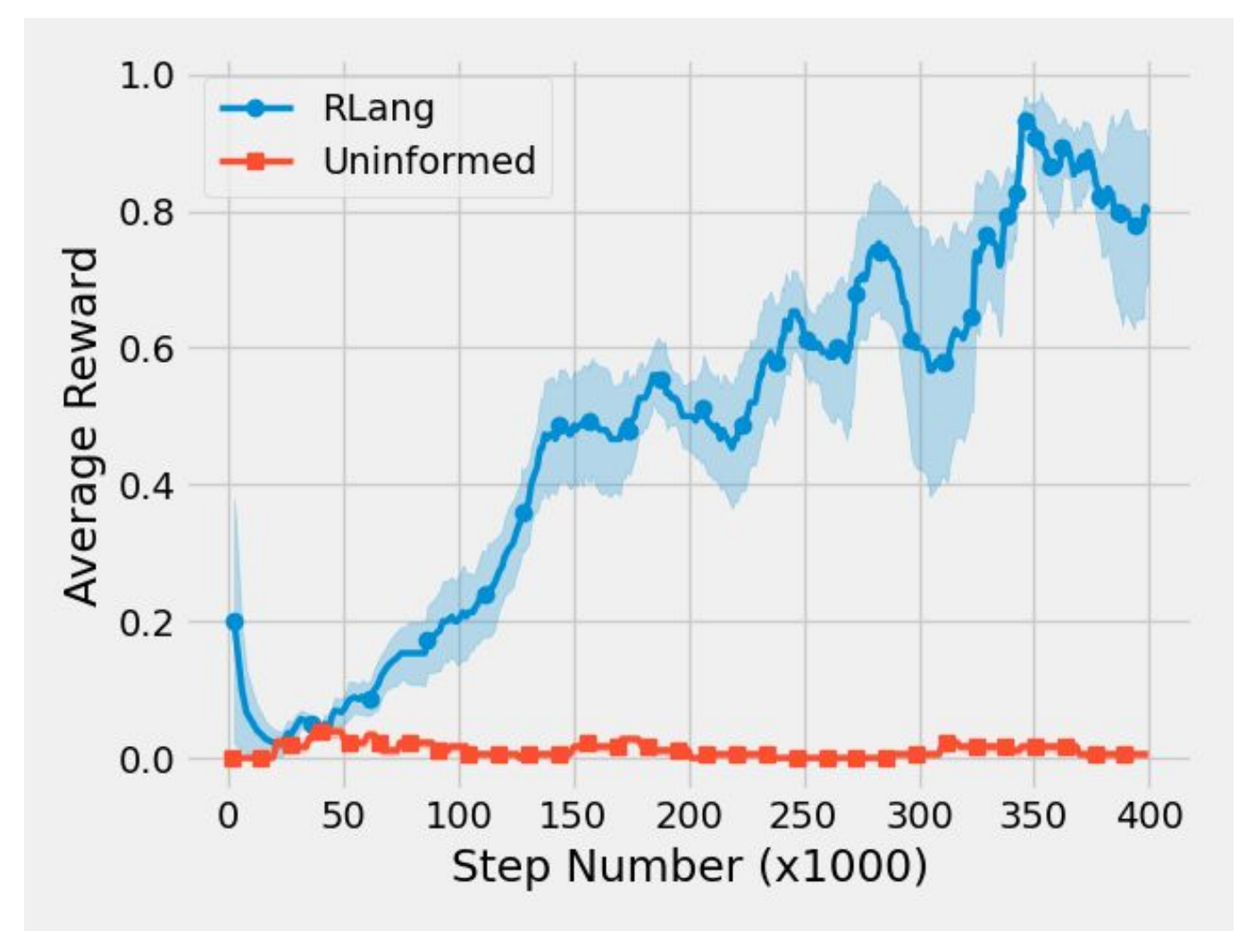
```
Factor inventory := S[250:270]
Feature wood := inventory[0]
Feature gold := inventory[1]
Effect:
  if at_workbench_1 and A == use:
    if wood >= 1:
      stick' -> stick + wood
      wood' -> 0
```

minecraft.rlang

Demonstrations

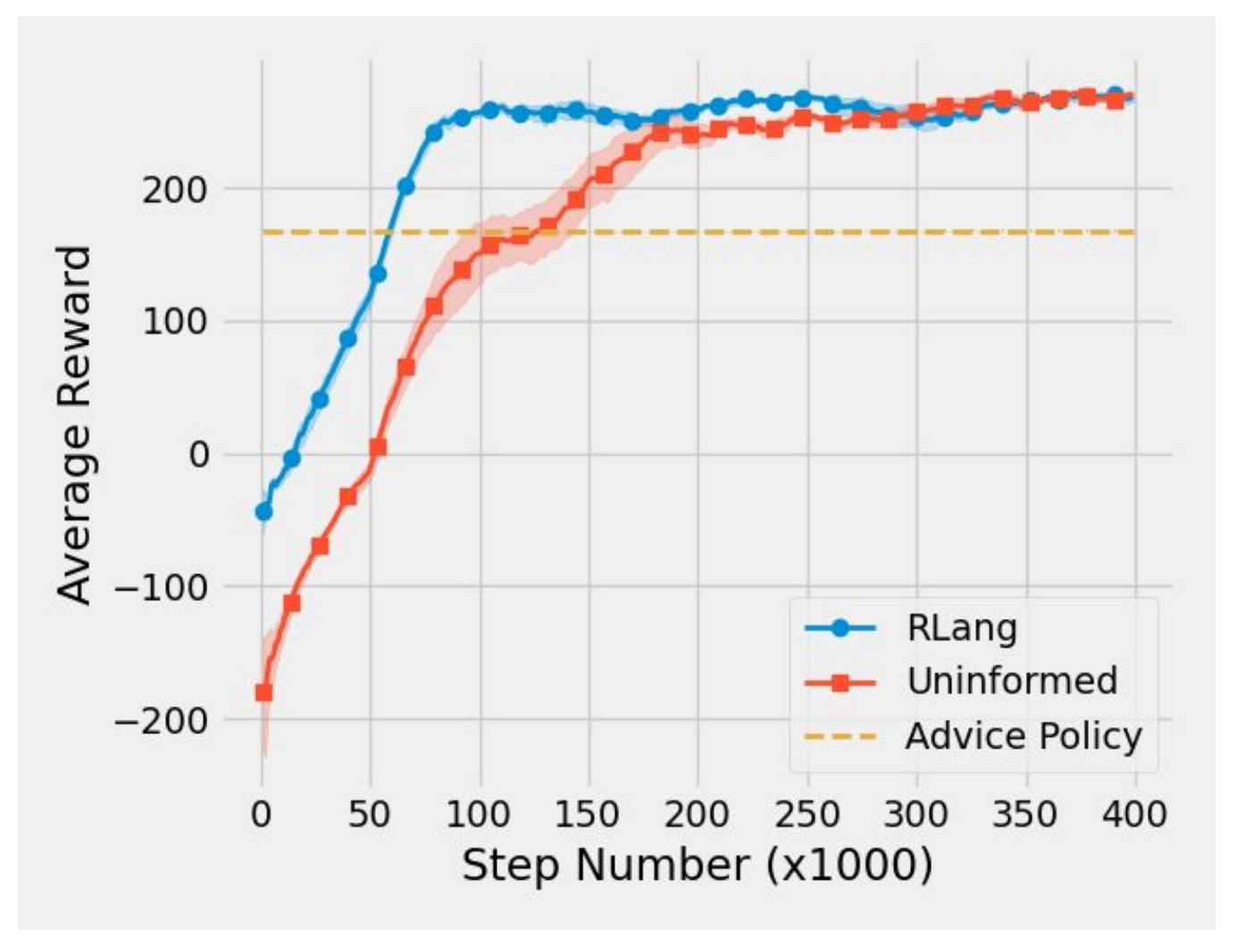
2D-Minecraft: Hierarchical Structure

```
Option go_to_workshop_0:
  init(any):
    Execute go_to_workshop_0_learnable_policy
  until(at_workshop_0)
Option go_to_workshop_1:
  init(any):
    Execute go_to_workshop_1_learnable_policy
  until(at_workshop_1)
Option get_wood:
  init(there_is_wood):
    Execute get_wood_learnable_policy
  until delta_wood >= 1
Option build_plank:
  init(wood >= 1 and at_workshop_1):
    Execute use
  until (delta_plank >= 1)
Option build_stick:
  init (wood >= 1 and at_workshop_1)
    Execute use
  until (delta_stick >= 1)
Option build_ladder:
  init (stick >= 1 and plank >= 1)
    Execute use
  until (delta_ladder >= 1)
```



Lunarlander: Policy Prior

```
Policy land:
  if (left_leg_in_contact == 1.0) or (right_leg_in_contact == 1.0)
    if (velocity_y/2 * -1.0) > 0.05:
      Execute main_engine
    else:
      Execute do_nothing
  elif remaining_hover > remaining_angle and remaining_hover > -1
 * remaining_angle and remaining_hover > 0.05:
    Execute main_engine
  elif remaining_angle < -0.05:
    Execute right_thruster
  elif remaining_angle > 0.05:
    Execute left_thruster
  else:
    Execute do_nothing
```


Average Reward
Step Number (x1000)
RLang
Uninformed
Advice Policy