Symbolic Planning and Model-Free Reinforcement Learning: Training Taskable Agents

León Illanes1, Xi Yan1, Rodrigo Toro Icarte1,2, Sheila A. McIlraith1,2
1Department of Computer Science, University of Toronto 2Vector Institute

Running Example

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Agent</td>
</tr>
<tr>
<td>B</td>
<td>Furniture</td>
</tr>
<tr>
<td>C</td>
<td>Coffee machine</td>
</tr>
<tr>
<td>D</td>
<td>Office</td>
</tr>
</tbody>
</table>

A, B, C, D Marked locations

Motivation – Taskability

- Specify high-level, goal-directed tasks to an agent
- Avoid reexecution of the environment

Task examples

T1. Deliver mail to the office
T2. Deliver coffee and mail to the office
T3. Visit locations A, B, C, and D (in any order)

Possible approaches

- Model-based Reinforcement Learning
- Hierarchical Reinforcement Learning
- Reward Shaping
- Modular RL and Policy Sketches
- Structured and Decomposable Reward Functions

In this work

- Where do the options come from?
- Where do reward functions come from?
- Where do policy sketches come from?

Answer: Typically, from a human expert.

The expert has a working model of the environment in mind and chooses options, designs reward functions, or sketches policies based on that. Given a new task, most of the expert's work will need to be repeated.

Our approach: Use an explicit high-level model.

Symbolic Planning

“Planning is the art and practice of thinking before acting.”

– Patrich Haslum

State-space given by a set of state properties

- e.g., propositions
- Actions given as preconditions and effects
- Properties needed for the action to be applicable
- Properties that change after the action is applied
- Tasks are given by an initial state and a goal condition
- Solutions or plans are sequences of actions

In the example

Propositions:

- have-mail/coffee
- delivered-mail/coffee (to A/B/C/D)

Actions:

- get-mail/coffee
- deliver-mail/coffee (to A/B/C/D)

Plans

T1. (get-coffee, deliver-coffee)
T2. (get-coffee, get-mail, deliver-coffee, deliver-mail)
T3. (go-to-A, go-to-B, go-to-C, go-to-D)

Partial-Order Plans

- A collection of actions and a partial order over them
- Every strict ordering that respects the partial order is a valid sequential plan
- Well established in the Planning literature
- Some planners can produce partial-order plans
- Sequential plans can be relaxed into partial-order plans

Examples

T1. Actions: get-coffee, deliver-coffee
Order: get-coffee < deliver-coffee
T2. Actions: get-coffee, get-mail, deliver-coffee, deliver-mail
Order: get-coffee < deliver-coffee, deliver-mail < deliver-mail
T3. Actions: go-to-A, go-to-B, go-to-C, go-to-D
Order: (none)

From POP to RL

- We train a metaccontroller to execute a given POP
- The metaccontroller is trained in a standard HRL manner
- It is a-proactively restricted to only select options that advance the execution of the POP

Implementation details

- POPs are represented with Reward Machines
- Finite-state machines with transitions that match observations in the environment
- The state in the machine represents which actions in the POP have already occurred
- The transitions depend on the observed environment

Example (T2)

Can we relax the ordering constraints?

Summary

- Specify abstract state and action models
- Use them to define tasks and solve them more efficiently
- Find a family of abstract plans and train a metaccontroller to instantiate it into a single plan

Motivation – Taskability

OfficeWorld
MinecraftWorld

Discrete domains

FarmWorld
Solution quality

Continuous domain

In this work

- Where do the options come from?
- Where do reward functions come from?
- Where do policy sketches come from?

Answer: Typically, from a human expert.

The expert has a working model of the environment in mind and chooses options, designs reward functions, or sketches policies based on that. Given a new task, most of the expert's work will need to be repeated.

Our approach: Use an explicit high-level model.