Algorithmic category theory for reinforcement learning

Tanner Duve, Theo M. Tyburn, Andrea Abeje-Stine, Onkar Kale

Adjoint School 2025 • Mentor: Georgios Bakirtzis • TA: Michail Savvas

Tue 3 Jun 2025

Reinforcement learning (RL) solves problems through reward signals

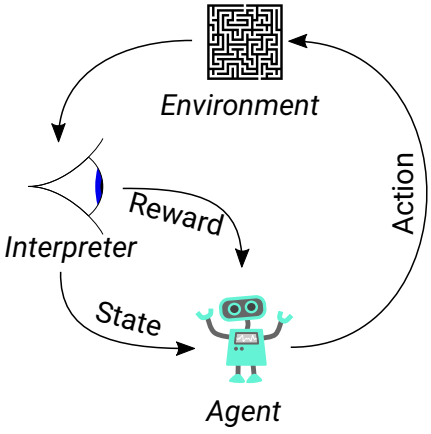
Training can happen without having a complete model of the task.

The video game Dota 2

But RL faces serious limitations in solving complex problems

Some of the limitations of RL arise from:

- reward sparsity,
- huge action spaces $\rightarrow$ sample inefficiency,
- lack of stability $\rightarrow$ poor safety guarantees,
- poor generalization, and
- poor interpretability.

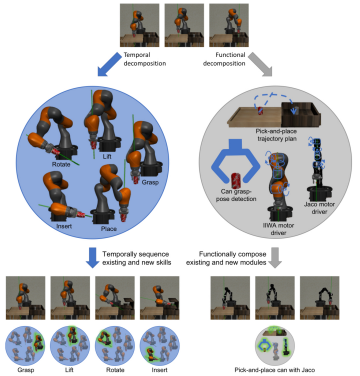


Typical framing of an RL scenario

Compositionality is being used in RL to make problems more tractable

Compositionality is being used in RL through two main paradigms:

1. **Hierarchical RL**
   models sequential tasks that can be split into independent sequential subtasks.
2. **Functional RL**
   models parallel tasks that can be split into interacting subtasks.



A robotic task decomposed hierarchically and functionally

Two major problems in RL can be solved by using compositionality

Two main problems in RL are

- **Scalability**: RL algorithms can have too complicated state and action spaces to keep track of.
- **Generalization**: RL agents may overfit to their training environments and perform poorly on new environments.

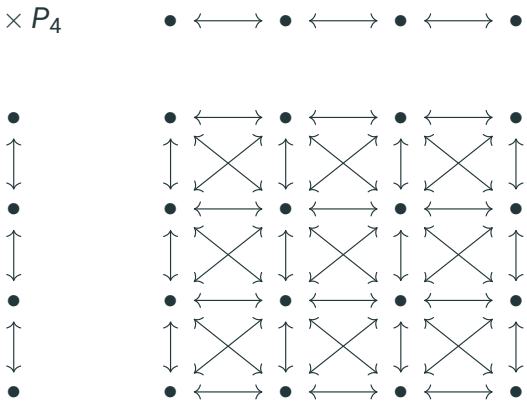
Compositionality can be used to solve both

- scalability through more efficient scaling methods and
- generalization by learning composable problems

Multi-agents scenarios suffer from both scaling and generalization issues

Multi-agent scenarios are generally modeled by taking the products of MDPs: $\mathcal{M}_1 \times \cdots \times \mathcal{M}_n$.

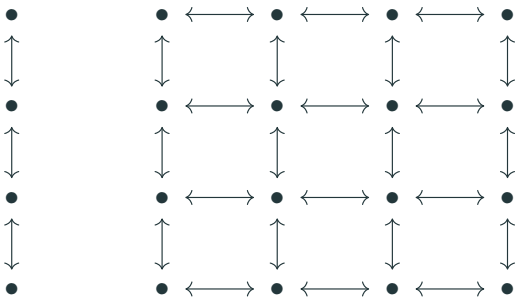
$P_4 \times P_4$



Problem: The action space $A_1 \times \cdots \times A_n$ rapidly blows up.

We want to reduce the action space to something like this

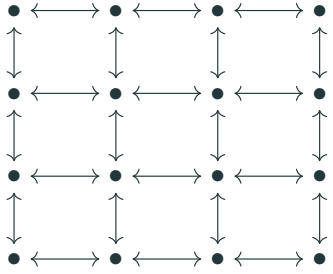
$P_4 \square P_4$



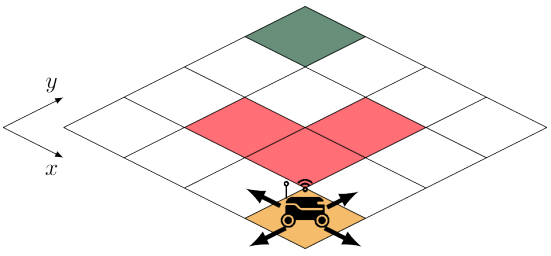
MDPs model sequential decision making under uncertainty

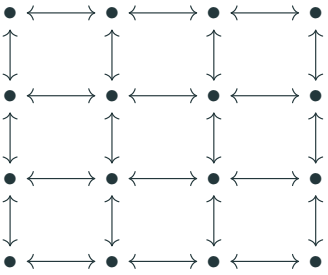
An Markov Decision Process (MDP) $\mathcal{M} = (S, A, \psi, T, R)$ is composed of the data:

- S — State space (measurable)
- A — Action space
- $\psi : A \to S$ — Action source function
- $T : A \to \mathcal{P}_S$ — Transition probability function
- $R : A \to \mathbb{R}$ — Reward function



MDPs model sequential decision making under uncertainty

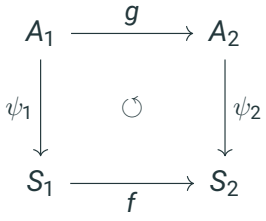


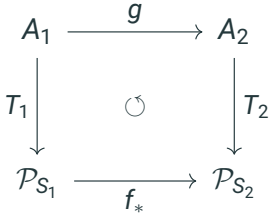


MDPs form the objects of the category **MDP**

Objects: MDPs

Morphisms:
$m = (f, g) : \mathcal{M}_1 \to \mathcal{M}_2$
$f : S_1 \to S_2$ (measurable)
$g : A_1 \to A_2$





The category **MDP** has pushouts and partial pullbacks.

The category **MDP** has a nice semantic

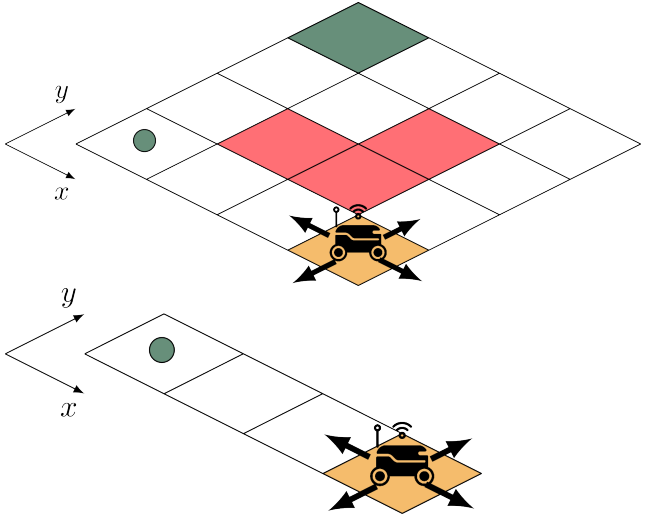
subtasks	injective morphisms
safety	punctured MDP
symmetry	quotient MDP
sequential tasks	pushouts
parallel tasks	products

We explorerd different categorical approaches of compositional RL

Through the category of **MDP** we studied

- A "box product" for MDP to potentially improve scalability
- A "twisted MDP" which is a combination of linear temporal logic specification for requirements (safety and liveness) and MDPs for behavior to potentially improve generalization
- Comonads to carry side information around in the category **MDP**

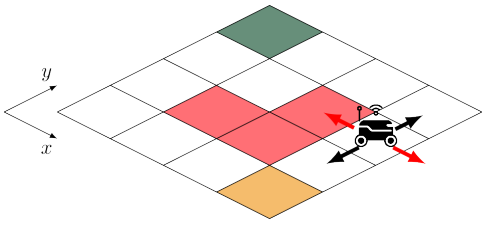
Injective maps in **MDP** model subprocess MDPs



$$\mathcal{M}_1 \xhookrightarrow{(f,g)} \mathcal{M}_2$$

Safety conditions are important for RL problems

Safe grid world: navigate to a goal state while avoiding obstacle states.

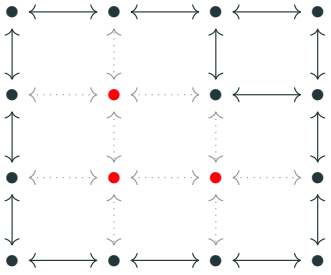


Punctured MDPs enforce safety conditions

Let $\mathcal{M} = (S, A, \psi, T)$ be an MDP,
$\mathbb{O} \subset S$.

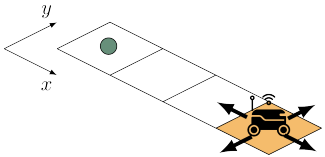
- **Punctured MDP:**
  $\mathcal{M}^\circ = (S^\circ, A^\circ, \psi^\circ, T^\circ)$ where:
  1. $S^\circ = S \backslash \mathbb{O}$.
  2. $A^\circ = A \backslash (\psi^{-1}(\mathbb{O}) \cup \mathrm{supp}_T(\mathbb{O}))$.
  3. $\psi^\circ = \psi|_{A^\circ}$.
  4. $T^\circ = T|_{A^\circ}$.

- $\mathcal{M}^\circ$ is the canonical maximal subprocess of S°.

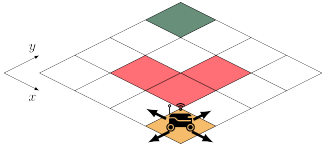


Sequential tasks can be modelled using pushouts of zig-zag diagrams

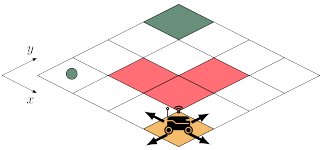
Ball



Move

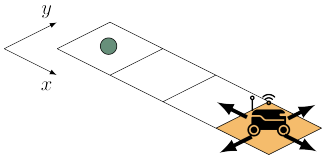


Compound

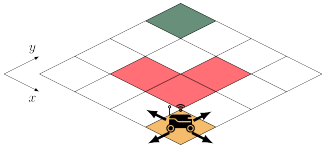


Sequential tasks can be modelled using pushouts of zig-zag diagrams

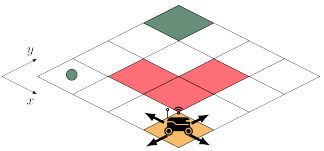
Ball

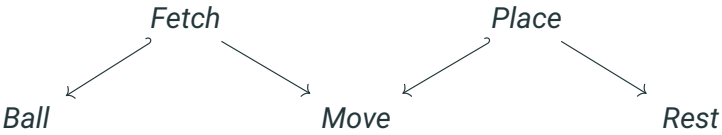


Move



Compound

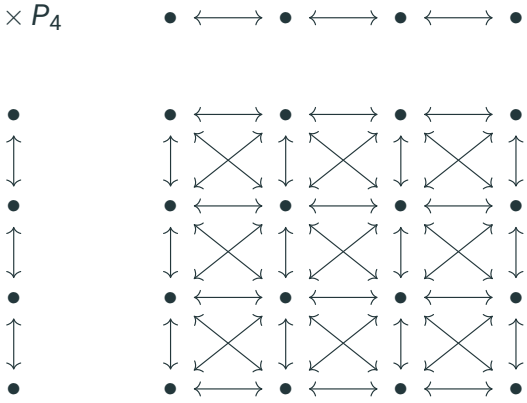




Parallel tasks can be modeled using products

To model n agents acting in parallel, we can take the cartesian product of MDPs: $\mathcal{M}_\times \cdots \times \mathcal{M}_n$.

$P_4 \times P_4$

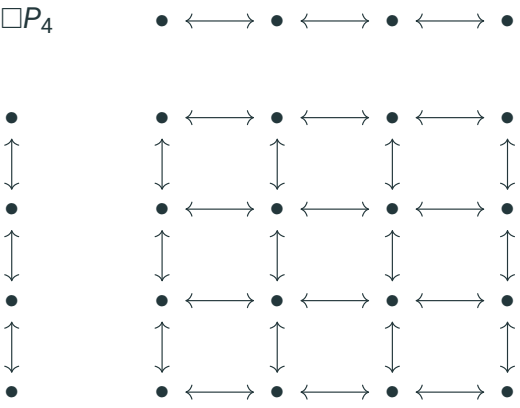


Problem: The size of the action space $A_1 \times \cdots \times A_n$ is $|A_1| \cdot \cdots \cdot |A_n|$.

We want to reduce the action space to something like this

Can we define a product $\mathcal{M}_1 \square \ldots \square \mathcal{M}_n$ where the action space has a size comparable to $|A_1| + \cdots + |A_n|$?

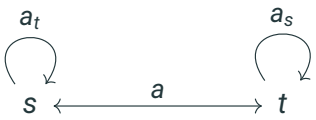
$P_4 \square P_4$



Stationary actions are the building blocks of the box product

$\mathcal{M} =$ 

$\mathcal{M}_0 =$ 

$\overline{\mathcal{M}} =$ 

MDPs admit a box product

The box product is a generalization of the box product for graphs inspired by the funny tensor product for categories.

Theorem
Given MDPs $\mathcal{M}_1$ and $\mathcal{M}_2$ we get the following pushout diagram:

$$\begin{array}{ccc} (\mathcal{M}_1)_0 \times (\mathcal{M}_2)_0 & \longrightarrow & (\mathcal{M}_1)_0 \times \overline{\mathcal{M}_2} \\ \downarrow & & \downarrow \\ \overline{\mathcal{M}_1} \times (\mathcal{M}_2)_0 & \longrightarrow & \mathcal{M}_1 \square \mathcal{M}_2 \end{array}$$

we call $\mathcal{M}_1 \square \mathcal{M}_2$ the box product of MDPs $\mathcal{M}_1$ and $\mathcal{M}_2$.

The box product has a tractable action space

1. State space $S = S_1 \times S_2$
2. Action space is given by

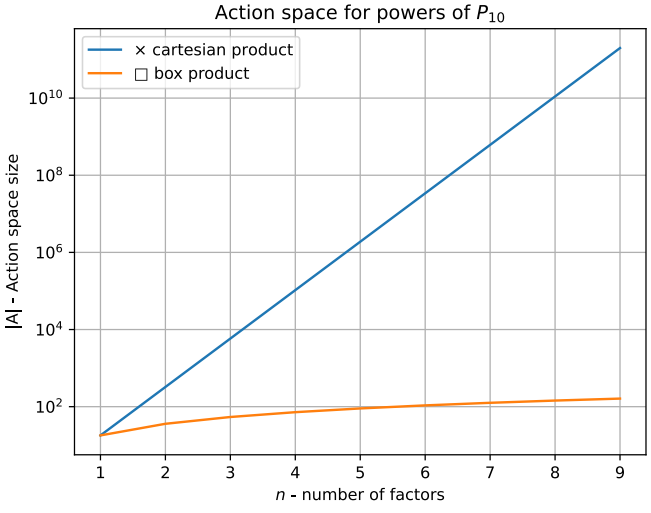
$$A = \overline{A_1} \times (A_2)_0 \amalg (A_1)_0 \times \overline{A_2} / \sim$$

   where $\sim$ identifies redundant stationary states.

For large action spaces:

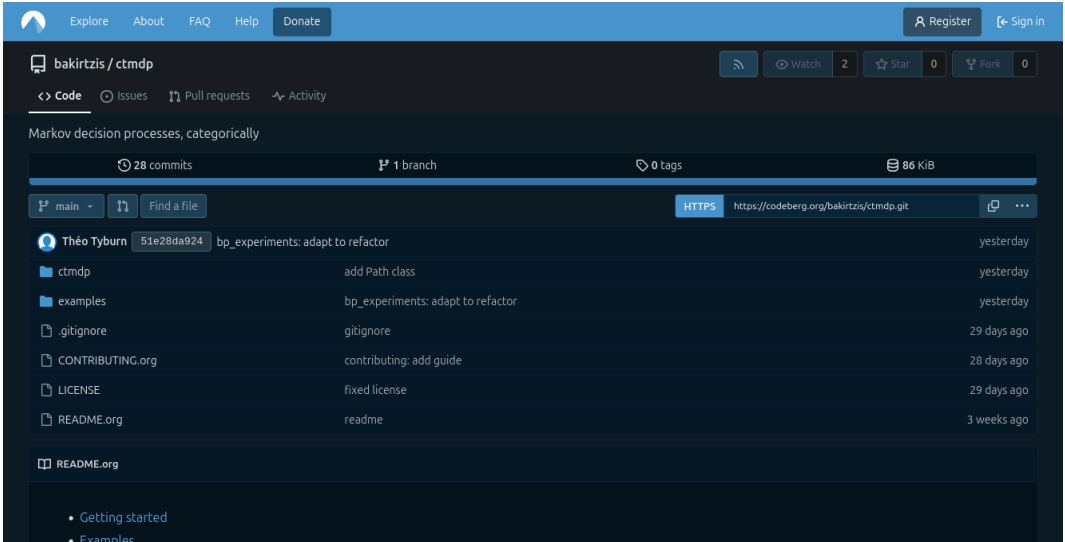
$$|S_1| \cdot |A_2| + |S_2| \cdot |A_1| \ll |A_1| \cdot |A_2|$$

The size of action space grows much slower for the box product



We wrote a python library to conduct experiments

`https://codeberg.org/bakirtzis/ctmdp`



Comonads as Side Information

Motivation.
In functional programming, **comonads** represent computations that use context.
In RL, **side information** is additional knowledge **not** encoded in the MDP

Hypothesis.
Comonads in the category **MDP** can formally model side information

Examples of side info we constructed comonads for:

- Safety constraints (e.g. unsafe region avoidance via twisted MDP)
- Symmetry structure (e.g. group actions, orbit-aware generalization)
- History context (e.g. fixed-length memory)

Towards algorithmic category theory for RL

The approach: use category theory to come up with concrete algorithms.

The hope: the category **MDP** (or an extesion of it) is the right framework to develop a theory of decomposition of RL agents that is stable under generalization.

The goal: provide RL practitioners with tools for:

- reward design,
- reusability and generalization,
- training efficiency via action space reduction.

Summary of the school

1. Problems we tried solving
    - generalization
    - scaling
2. Solution: use compositional methods
    - Sequential composition: pushouts of MDPS
    - Functional composition: box products
    - Composing side information: comonads
3. Further things we want to investigate
    - Extensions of span and cospan categories
    - Introducing error terms into MDP morphisms
    - MDPs as coalgebras
    - (Probabilistic) Linear Temporal Logic
    - Partially Observable MDPs

Sources

- *Categorical semantics of compositional reinforcement learning*, Georgios Bakirtzis, Michail Savvas, and Ufuk Topcu
- *Structure in Deep Reinforcement Learning: A Survey and Open Problems*, Aditya Mohan, Amy Zhang, Marius Lindauer
- *Categorical Semantics of motion planning*, Georgios Bakirtzis and Ufuk Topcu

THANK YOU FOR YOUR ATTENTION!