Hierarchical Reinforcement Learning

Temporal Abstraction in RL

Khimya Khetarpal

Reasoning and Learning Lab
Mila - McGill University

Consider an autonomous robot in a warehouse

Image Source: Boston Dynamics Robot Handle

Consider an autonomous robot in a warehouse

Pick up boxes

Navigate to destination

Stack boxes

Tasks at hand could be solved quickly and efficiently with ***skills***

Image Source: Boston Dynamics Robot Handle

Consider an autonomous robot in a warehouse

Pick up boxes

Scan room | Identify Objects | Find Box

Reach Box | Find Location

← ↓ → ↑

Navigate to destination

Obstacle Avoid | Obstacle Detect | Move

← ↓ → ↑

Stack boxes

Identify OBJ | Find Box | Place Box

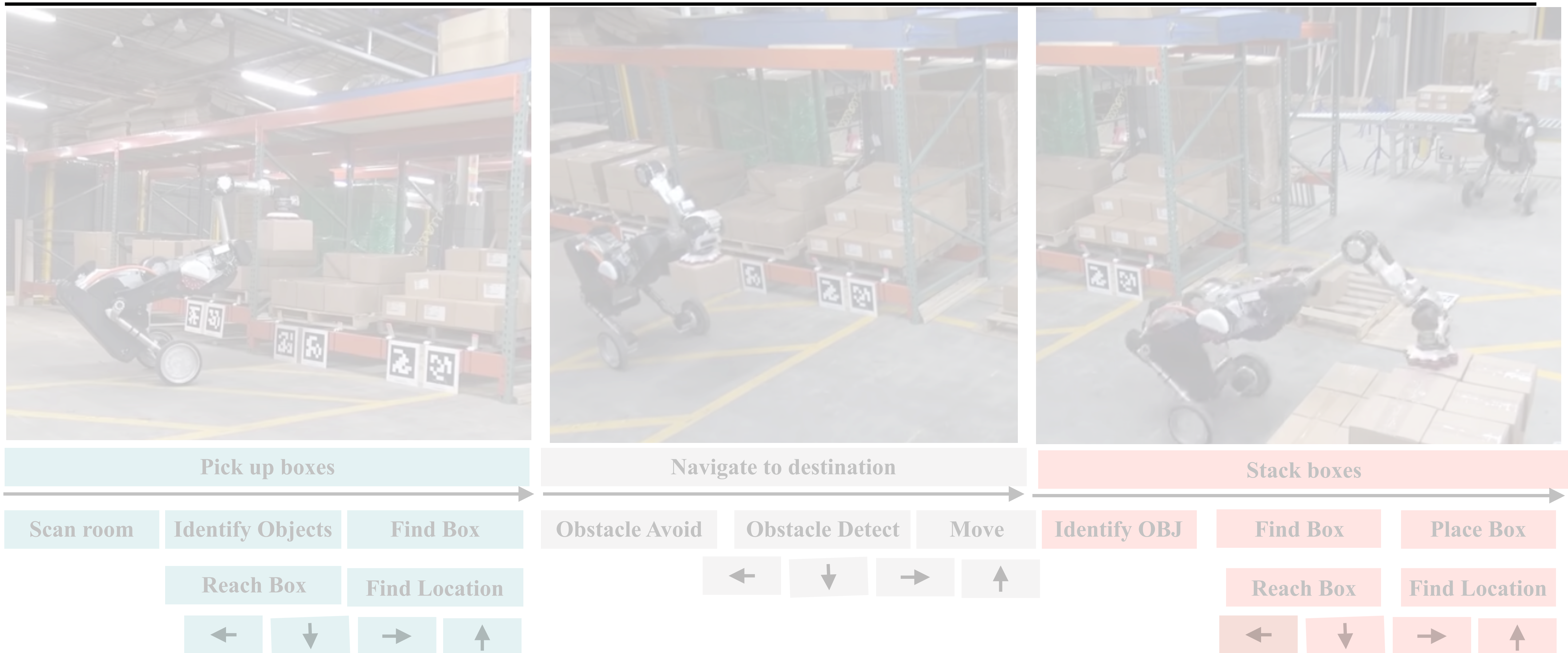
Reach Box | Find Location

← ↓ → ↑

Each ***skill*** can take ***different*** number of time steps

Image Source: Boston Dynamics Robot Handle

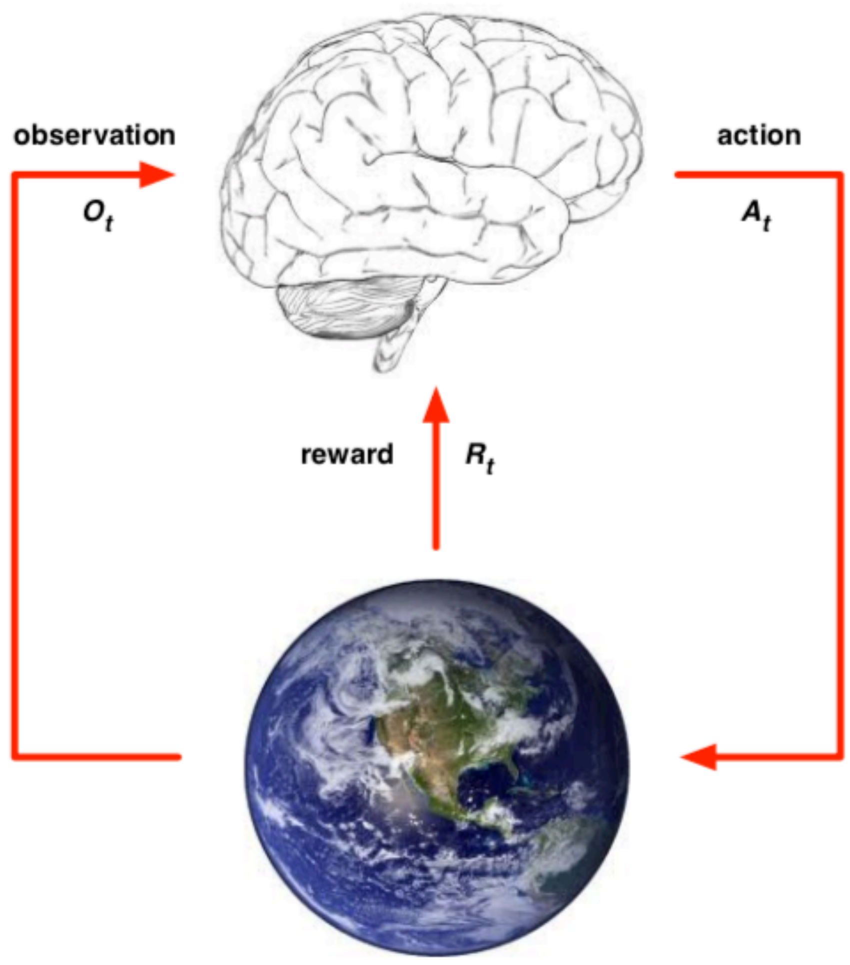
Consider an autonomous robot in a warehouse



The ability to abstract knowledge temporally over many different time scales is seamlessly integrated in human decision making!

Image Source: Boston Dynamics Robot Handle

Reinforcement Learning



At each time step, the ***agent***:

- Executes action A_t
- Receives observation O_t
- Receives reward R_t

At each time step, the ***environment***:

- Receives action
- Emits observation O_{t+1}
- Emits scalar reward R_{t+1}

Predictions : Value Functions

Policy $\pi(a \mid s)$

Value Function $$V_\pi(s) = E_\pi[R_{t+1} + \underbrace{\gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots}_{\text{Discounted Future Value}} \mid S_t = s]$$

Immediate Reward

Discount Factor

Discounted Future Value

Learning Values

Policy $\pi(a \mid s)$

Value Function $$V_\pi(s) = E_\pi[\underbrace{R_{t+1}}_{\text{Immediate Reward}} + \underbrace{\gamma}_{\text{Discount Factor}} \underbrace{R_{t+2} + \gamma^2 R_{t+3} + \ldots}_{\text{Discounted Future Value}} \mid S_t = s]$$

Temporal Difference Learning $$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

Temporal-difference error: $$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

Learning rule for parameterized value functions $$w_{t+1} = w_t + \alpha \delta_t \nabla_w V_w(S_t)$$

Why Temporal Abstraction

Planning

- Generate shorter plans
- Provides robustness to model errors
- Improves sample complexity

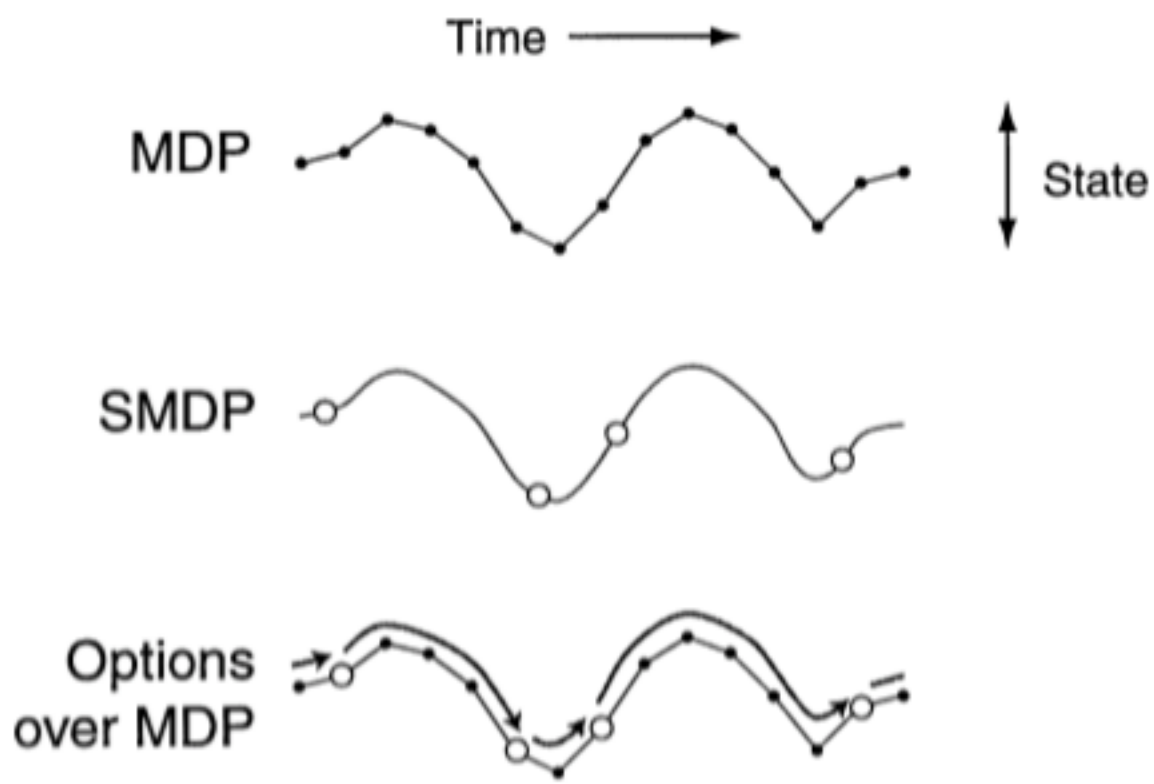
Learning

- Improve exploration by taking shortcuts in the environment
- Facilitates Off-Policy learning
- Improves efficiency/learning speed
- Helps in transfer learning

The Options Framework

The Options Framework

Options (Sutton, Precup, and Singh, 1999) formalize the idea of temporally extended actions also known as **skills.**



Sutton, Precup & Singh 1999

Options Framework

- Definition

Let S, A be the set of states and actions. A Markov option $\omega \in \Omega$ is a triple:

$$\left(\mathbf{I}_\omega \subseteq \mathbf{S}\ ,\ \pi_\omega : \mathbf{S} \times \mathbf{A} \rightarrow [\mathbf{0}, \mathbf{1}]\ ,\ \beta_\omega : \mathbf{S} \rightarrow [\mathbf{0}, \mathbf{1}]\right)$$

Initiation set **Intra option policy** **Termination condition**

- I_ω set of states aka preconditions
- $\pi_\omega(s, a)$ probability of taking an action $a \in A$ in state $s \in S$ when following the option ω
- $\beta_\omega(s)$ probability of terminating option ω upon entering state s

with a policy over options $\pi_\Omega : S \times \Omega \rightarrow [0,1]$

Sutton, Precup & Singh 1999

Options Framework

- Definition

  Let S, A be the set of states and actions. A Markov option $\omega \in \Omega$ is a triple:

$$\left(\mathbf{I}_\omega \subseteq \mathbf{S} \ , \ \pi_\omega : \mathbf{S} \times \mathbf{A} \rightarrow [\mathbf{0}, \mathbf{1}] \ , \ \beta_\omega : \mathbf{S} \rightarrow [\mathbf{0}, \mathbf{1}]\right)$$

Initiation set **Intra option policy** **Termination condition**

  - I_ω set of states aka preconditions
  - $\pi_\omega(s, a)$ probability of taking an action $a \in A$ in state $s \in S$ when following the option ω
  - $\beta_\omega(s)$ probability of terminating option ω upon entering state s

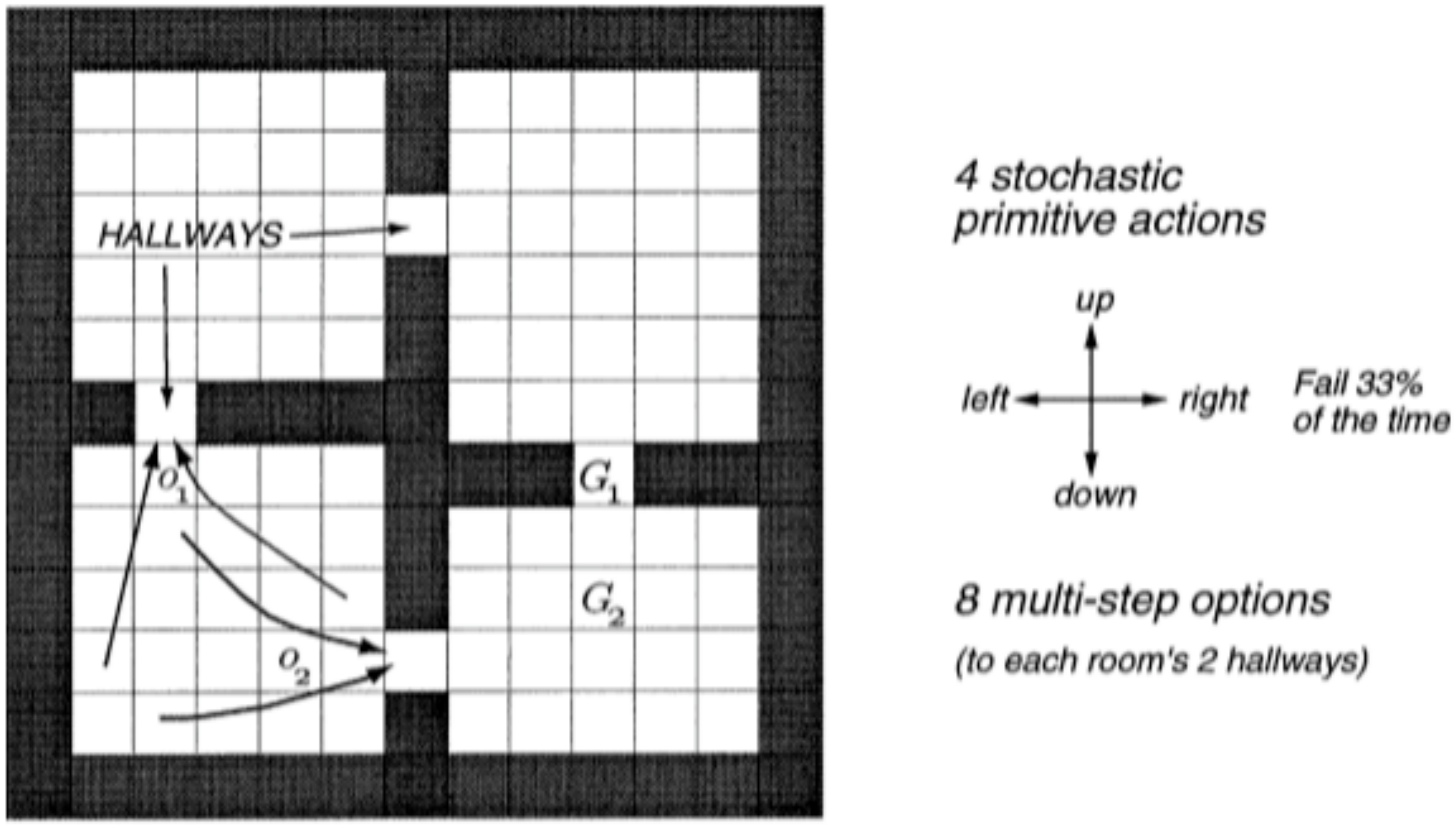
  with a policy over options $\pi_\Omega : S \times \Omega \rightarrow [0,1]$

- Example

  - Robot navigating in a house: when you come across a closed door (I_ω), open the door (π_ω), until the door has been opened (β_ω)

Sutton, Precup & Singh 1999

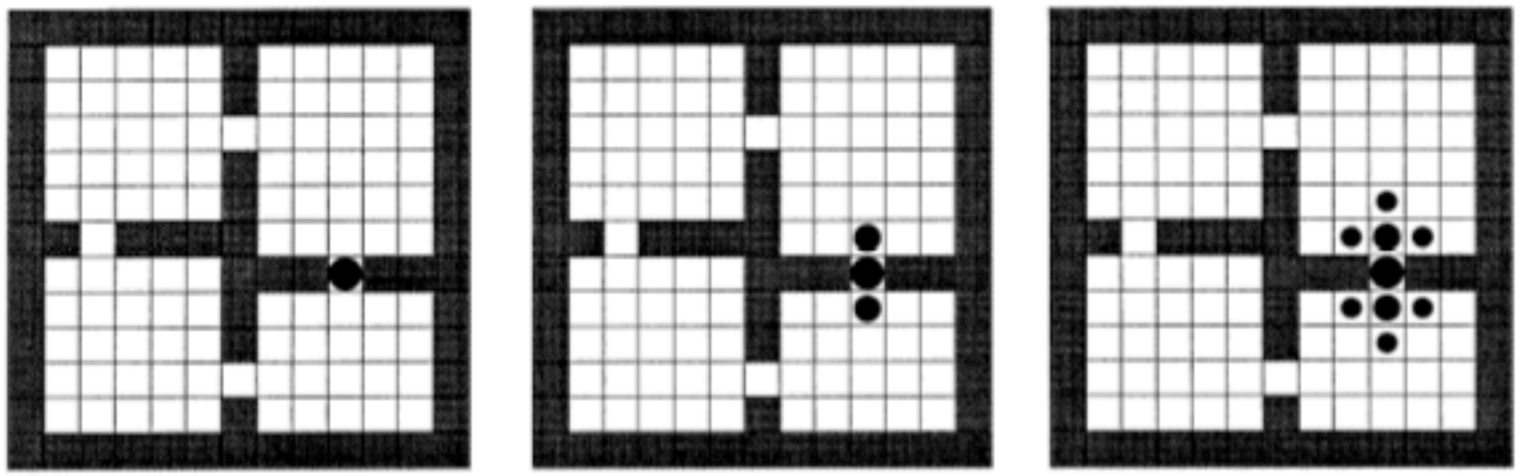
Planning with Options



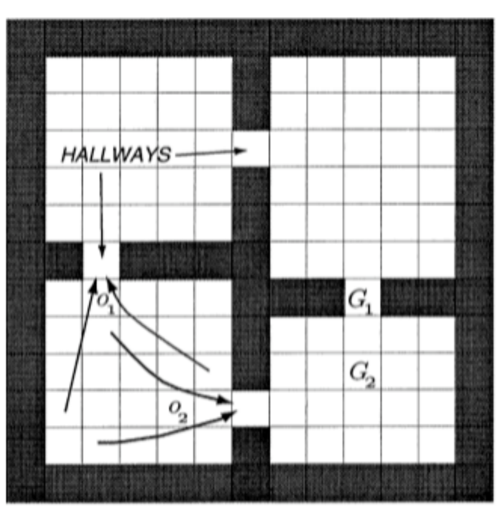
Sutton, Precup & Singh 1999

Planning with Options

Primitive actions



Initial Values   Iteration #1   Iteration #2



4 stochastic primitive actions

up
left *right* *Fail 33% of the time*
down

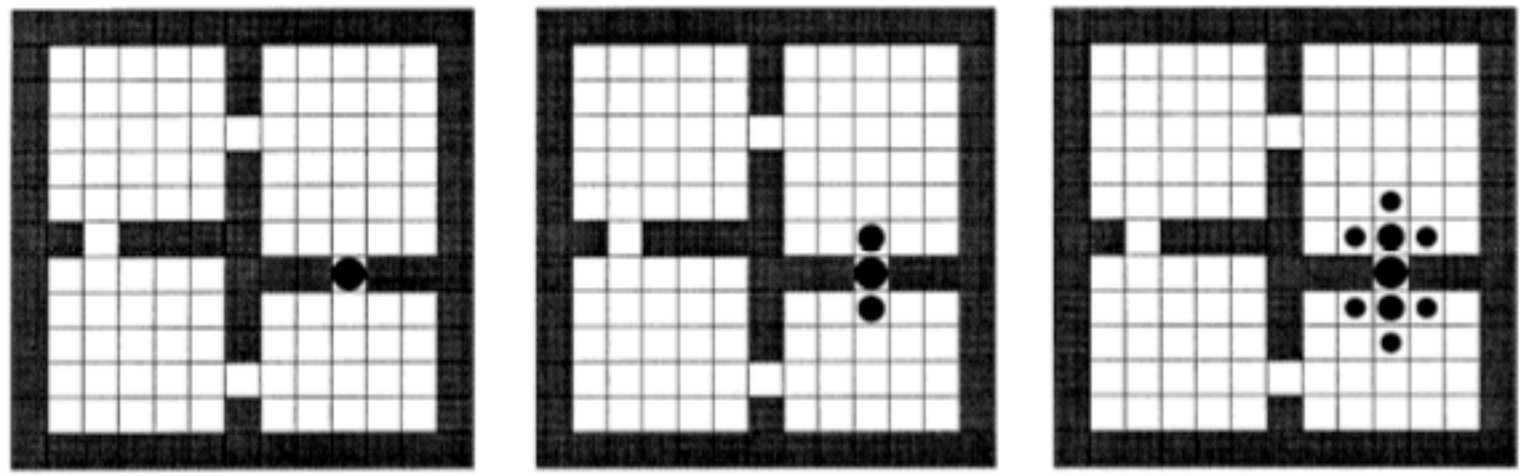
8 multi-step options

(to each room's 2 hallways)

Sutton, Precup & Singh 1999

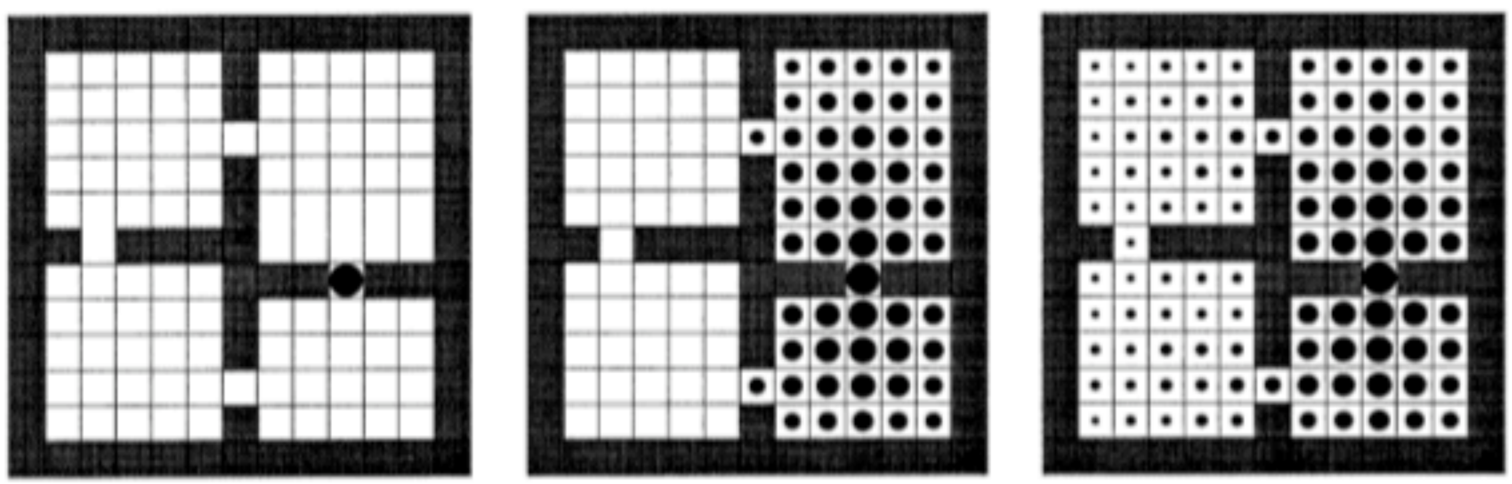
Planning with Options

Primitive actions

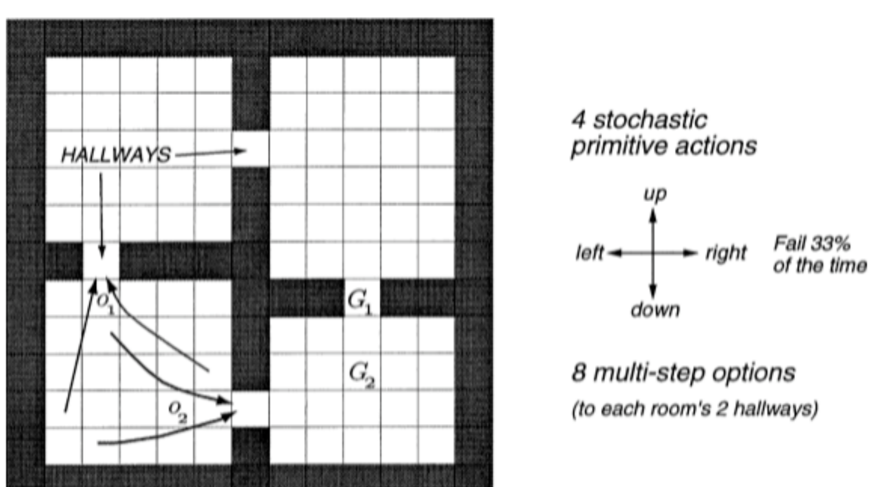


Initial Values Iteration #1 Iteration #2

Hallway Options



Initial Values Iteration #1 Iteration #2

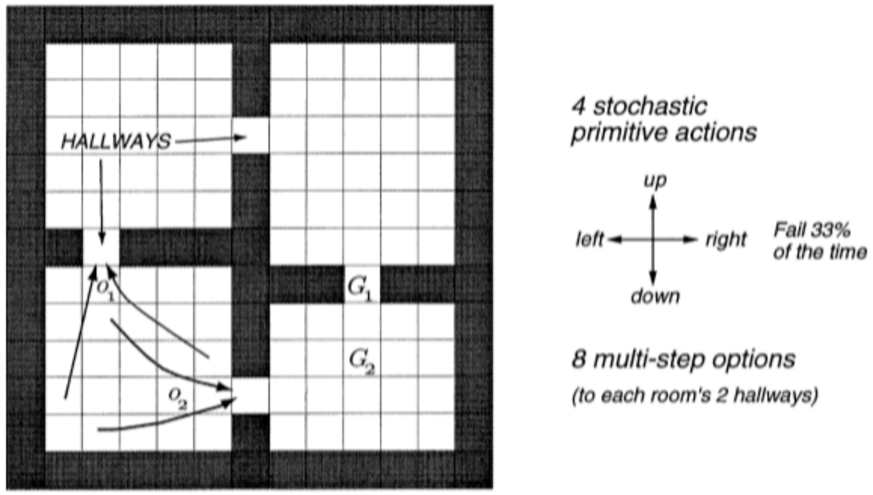


Sutton, Precup & Singh 1999

Planning with Options : Discussion

Potential Applications:

- Planning with stocks
- Planning with assets - asset management
- Clinical Domains [Y. Shahar: A framework for knowledge-based temporal abstraction]



Can we learn such temporal abstractions?

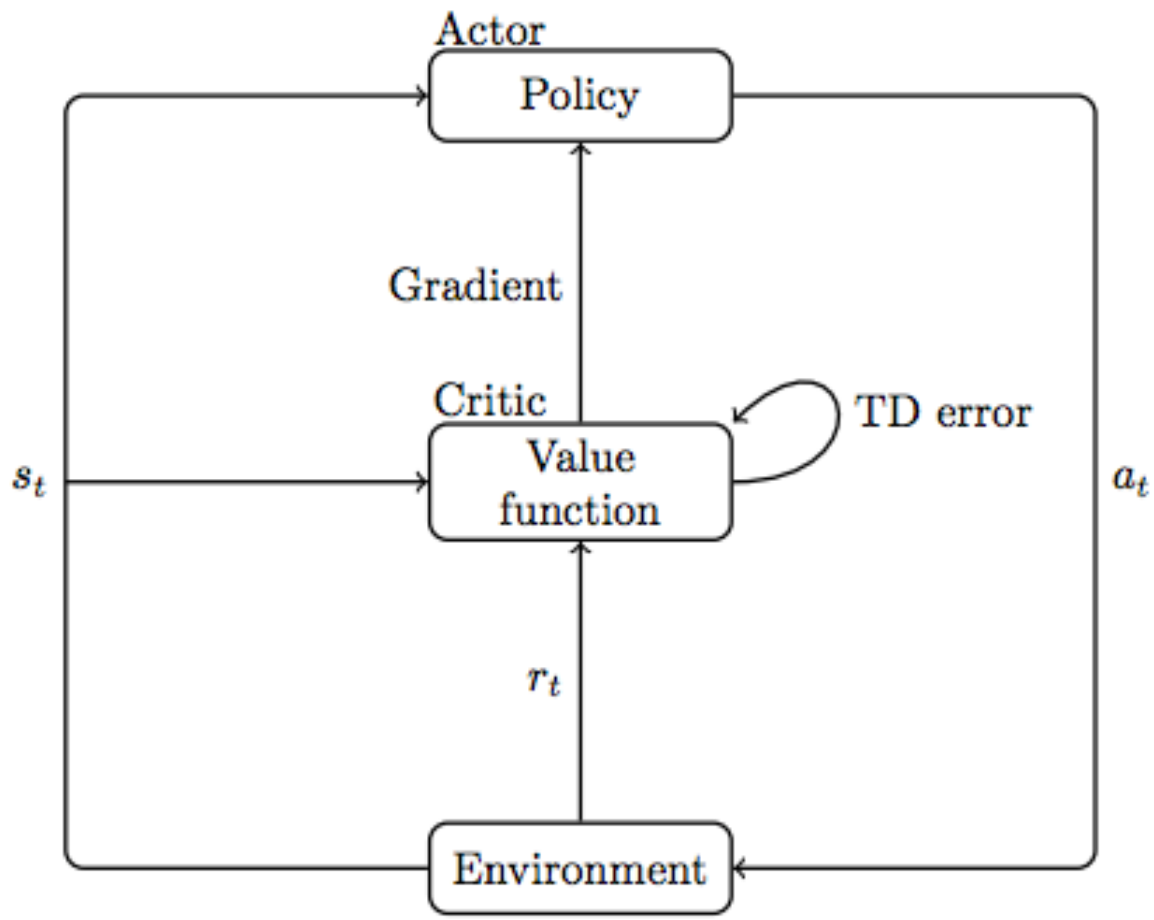
- Bacon, Harb, and Precup, 2017 proposed the option-critic framework which provides the ability to ***learn*** a set of options

- Optimize directly the discounted return, averaged over all the trajectories starting at a designated state and option

$$J = E_{\Omega,\theta,\omega}[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \,|\, s_0, \omega_0]$$

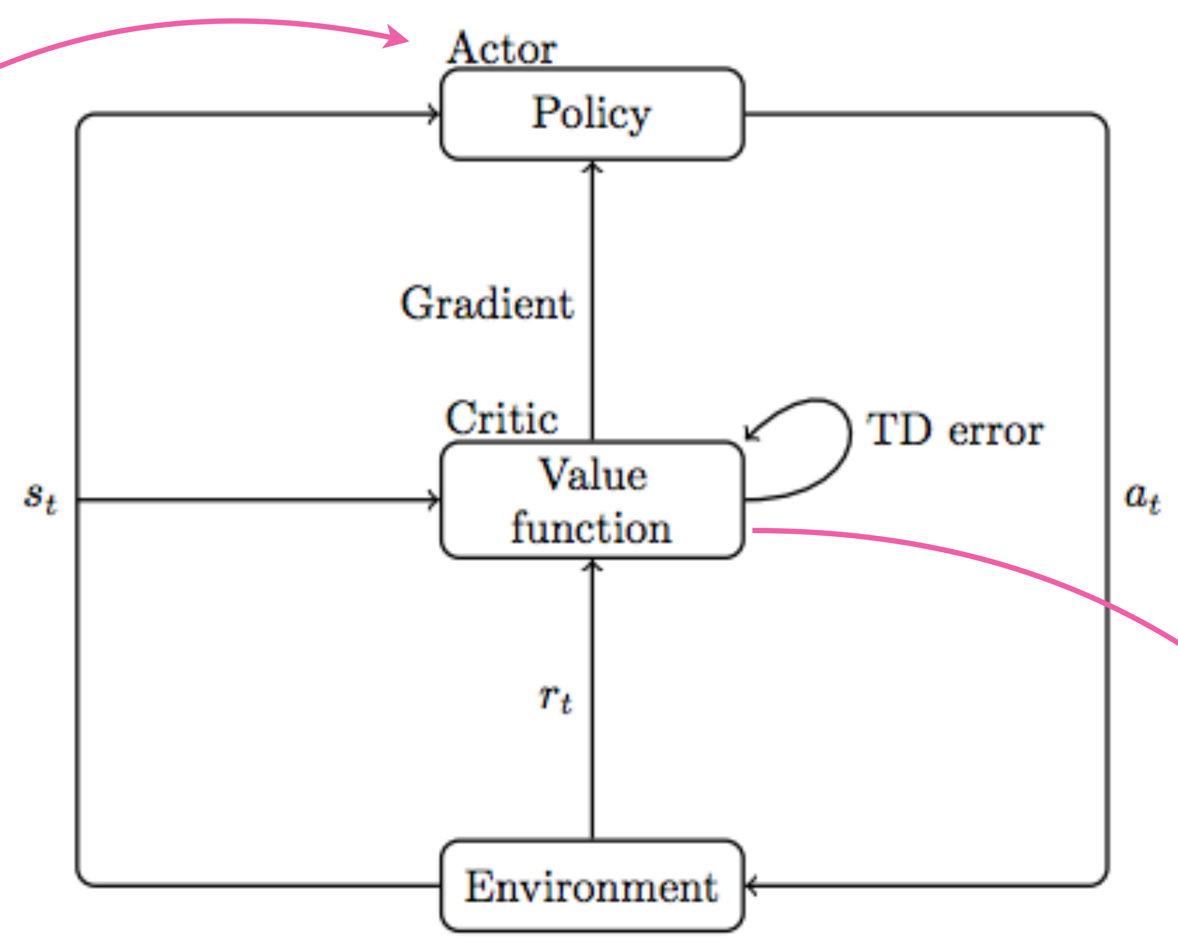
Bacon, Harb & Precup 2017

Actor-Critic Architecture



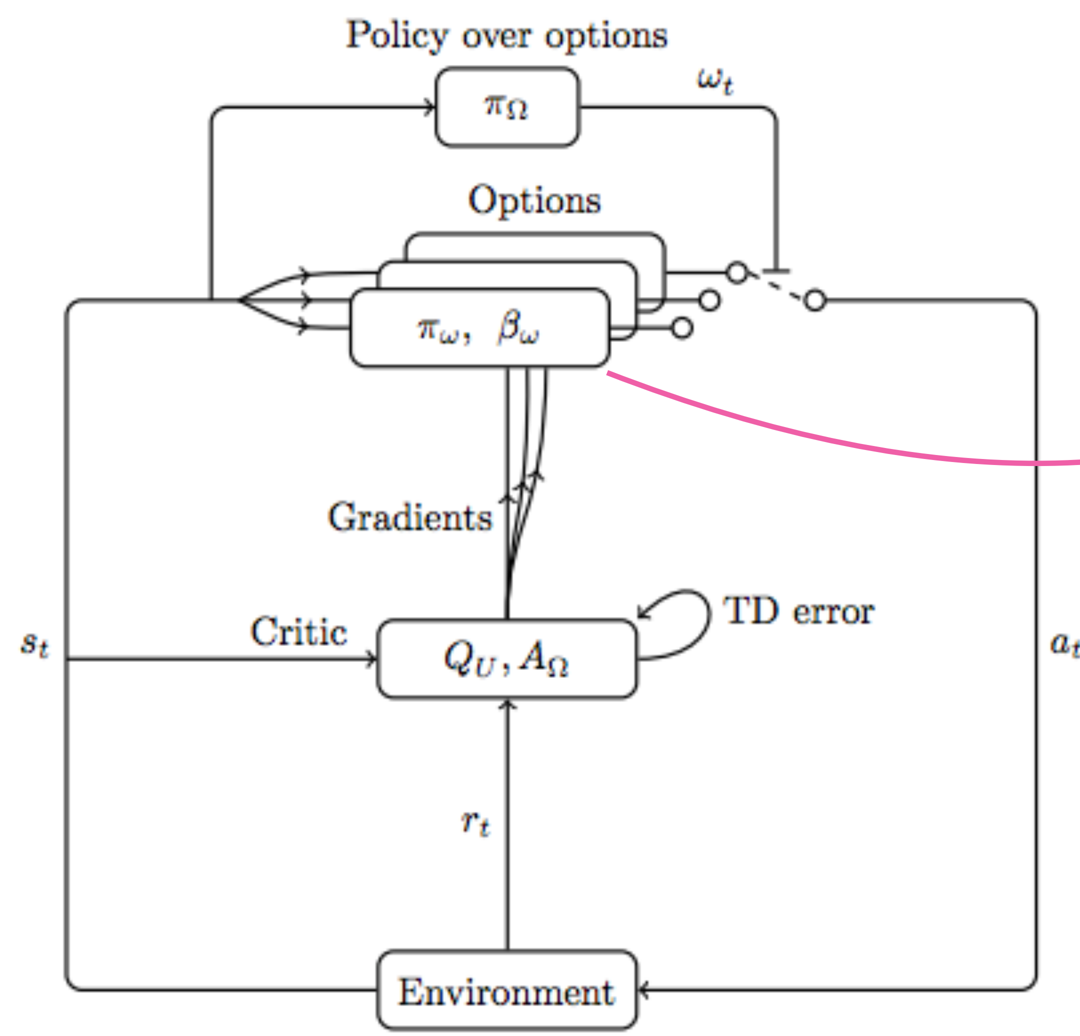
Actor-Critic Architecture

Decides how the agent acts



Provides feedback to improve the actor

Option-Critic Architecture

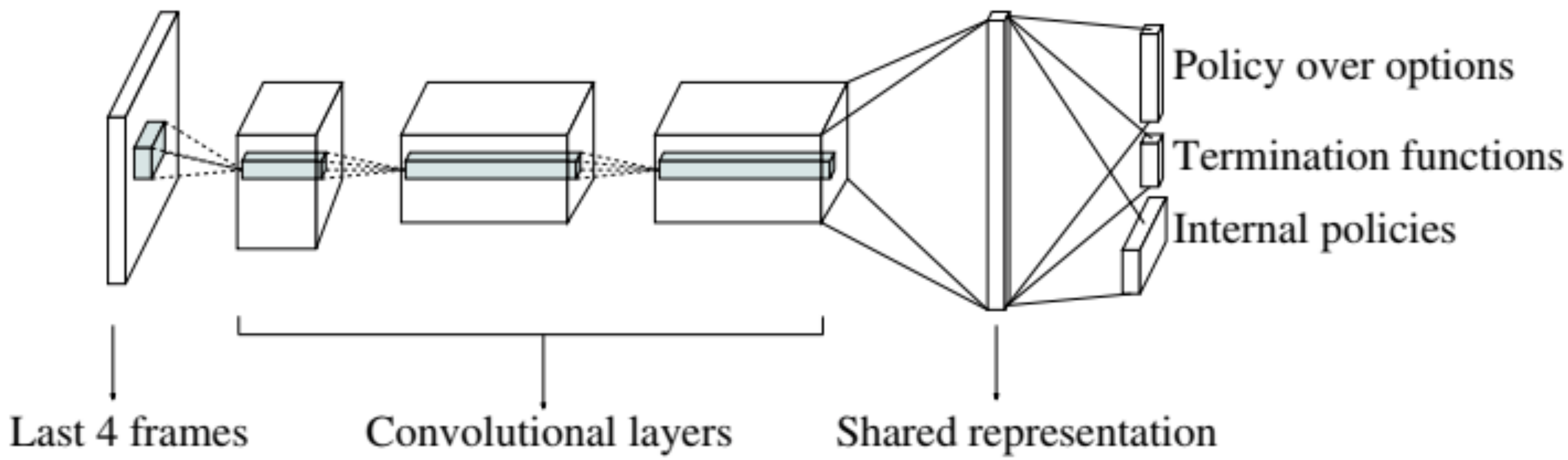


Parameterize internal policies

Parameterize termination conditions

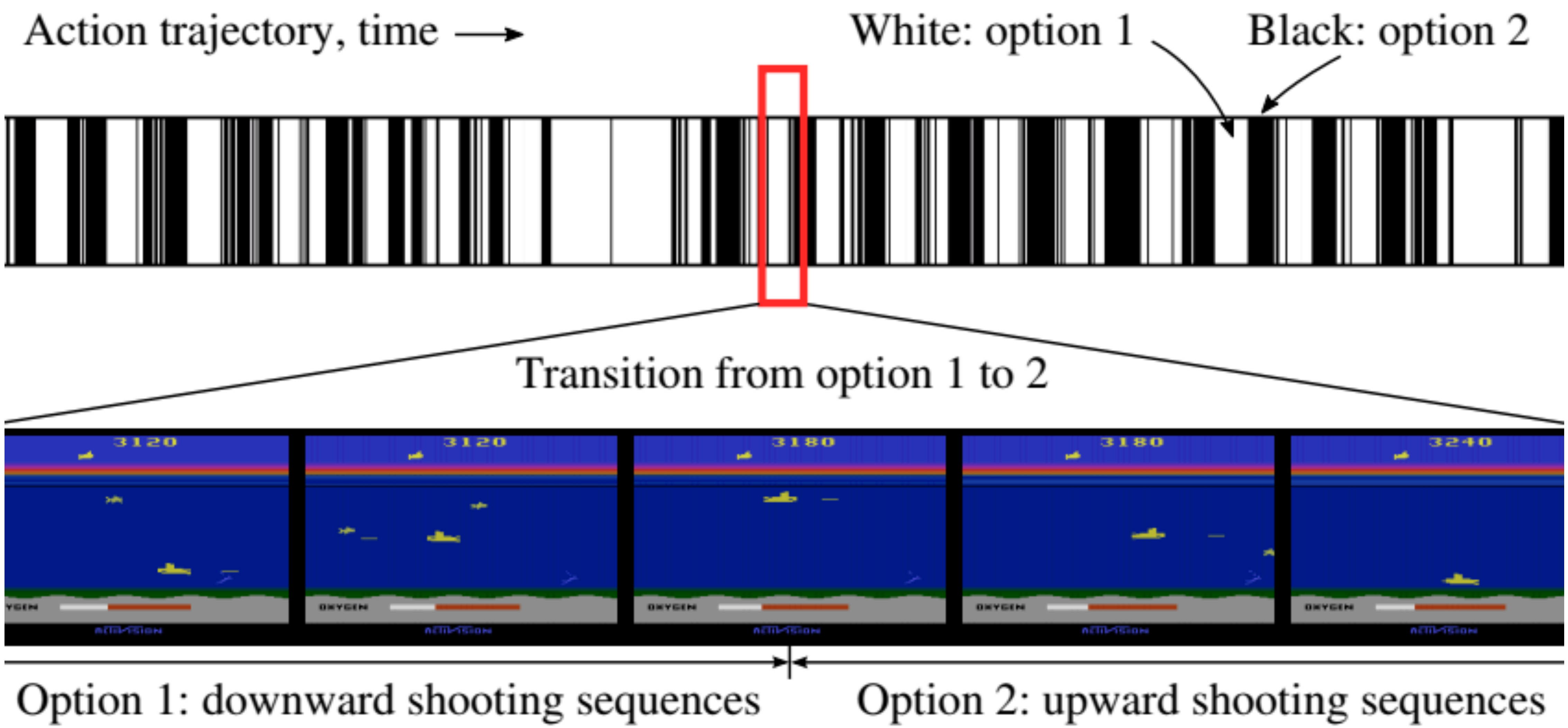
Bacon, Harb & Precup 2017

Option-Critic with Deep RL



Bacon, Harb & Precup 2017

Option-Critic with Deep RL

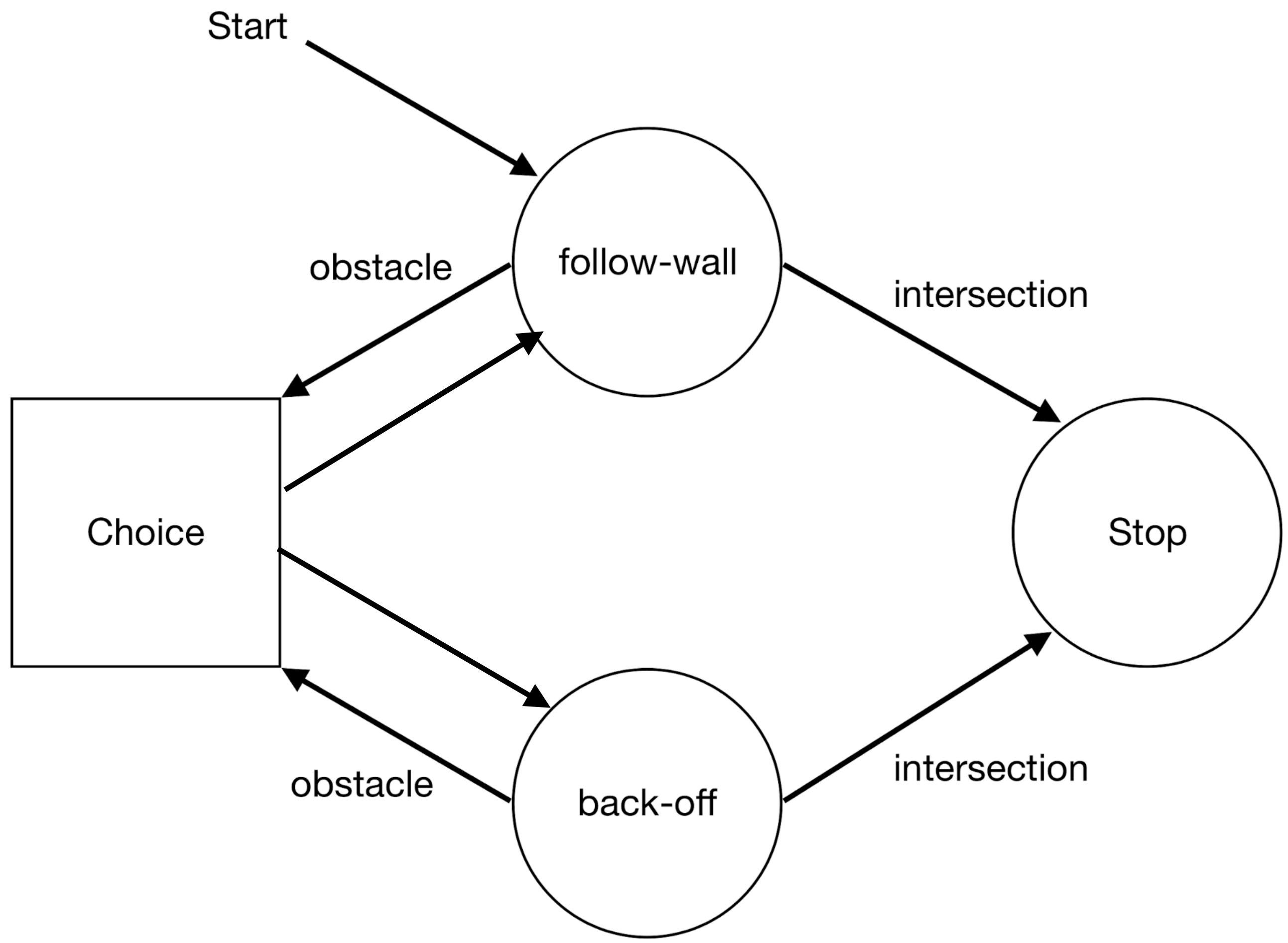


Bacon, Harb & Precup 2017

Hierarchical Abstract Machines (HAMs)

Hierarchical Abstract Machines (HAMs)

- **Key Idea:**
  - Non deterministic finite state machines
  - Transitions invoke lower level machines

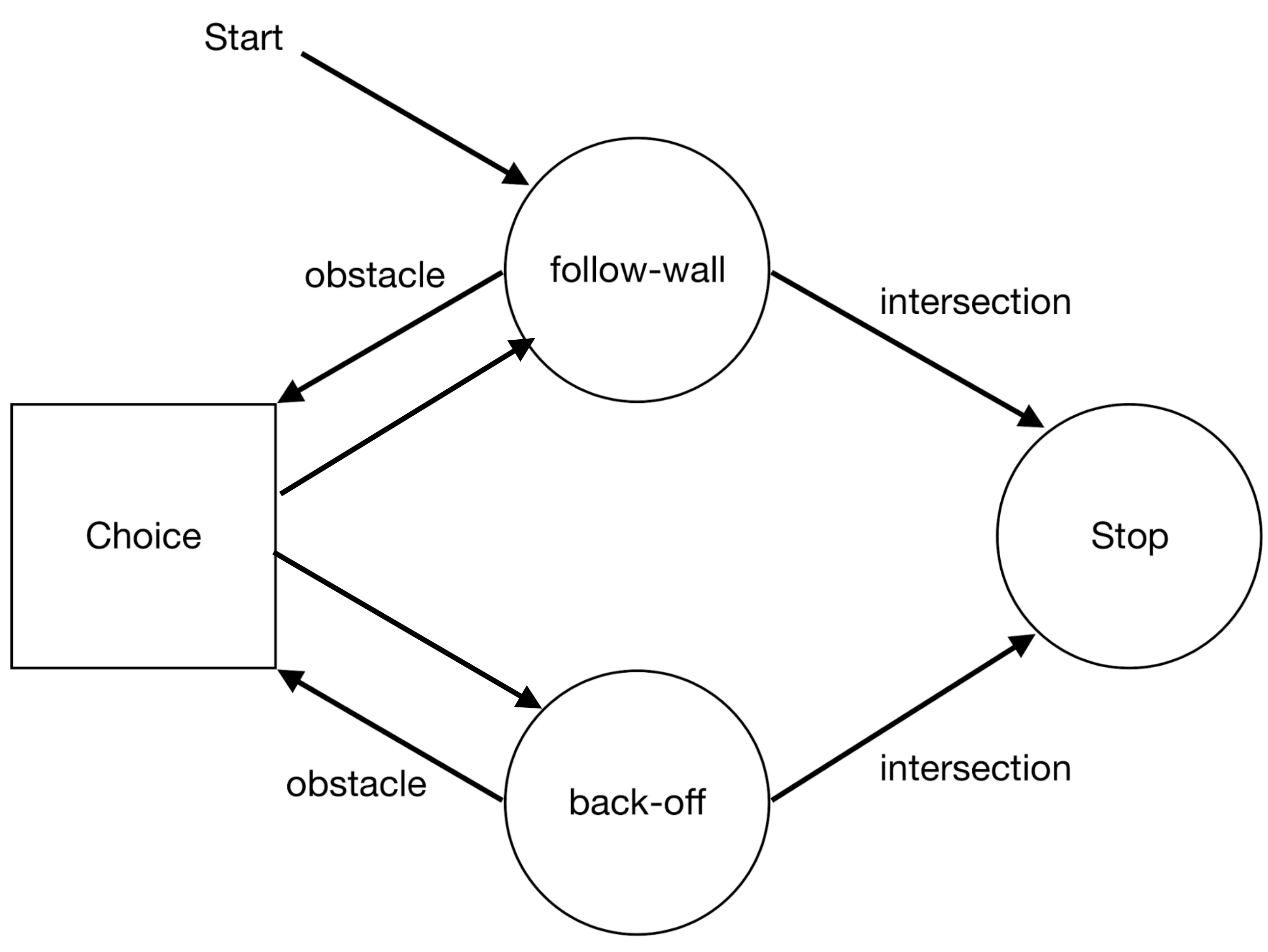


Parr & Russell, 1998

Hierarchical Abstract Machines (HAMs)

- **Key Idea:**
  - Non deterministic finite state machines
  - Transitions invoke lower level machines

- **A Machine:**
  - Is a partial policy
  - Has four states: Call/Stop/Choice/A

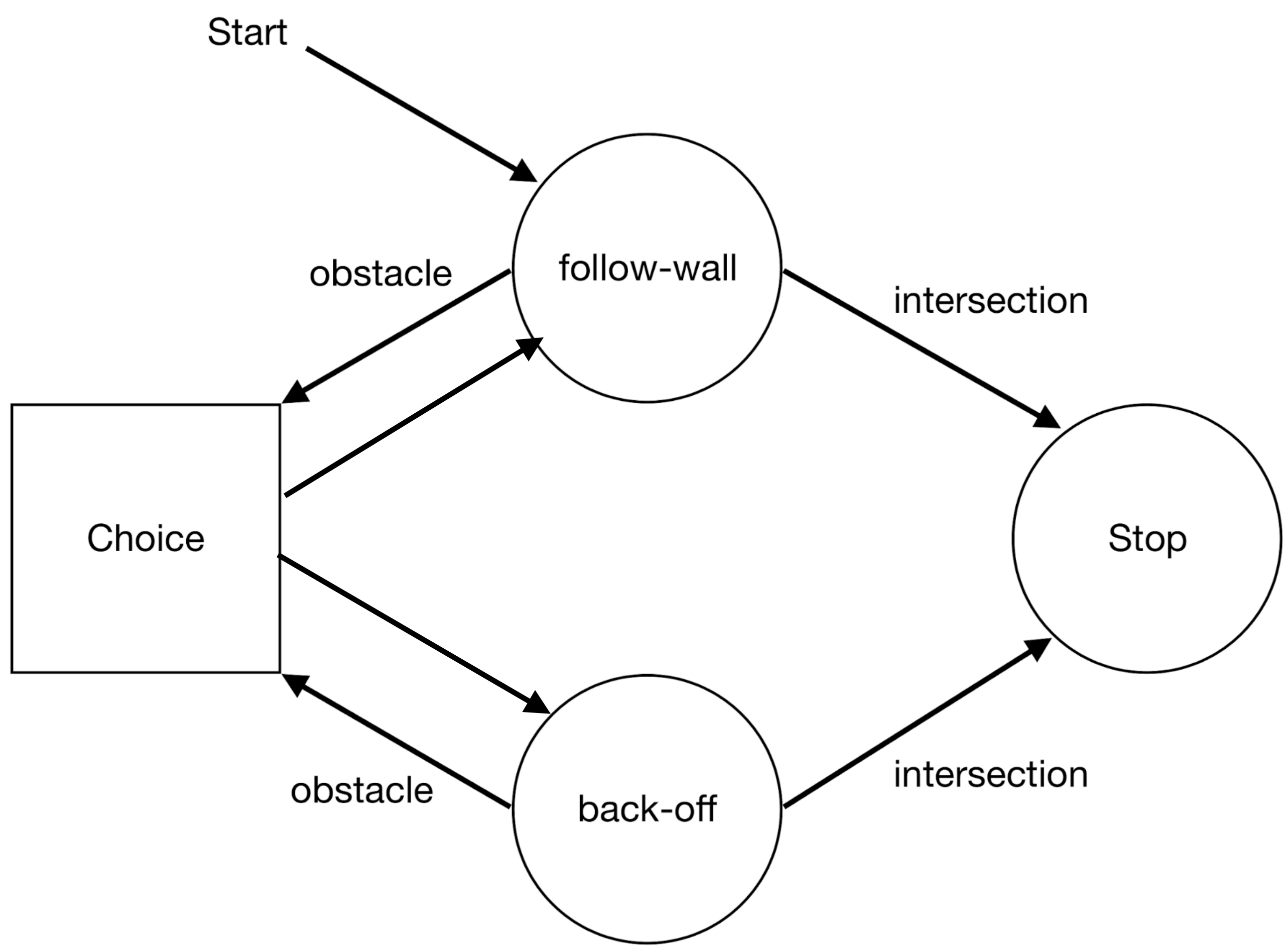


Parr & Russell, 1998

Hierarchical Abstract Machines (HAMs)

- Upon encountering an obstacle:
  - Machine enters a Choice state
  - Follow-wall Machine
  - Back-off Machine

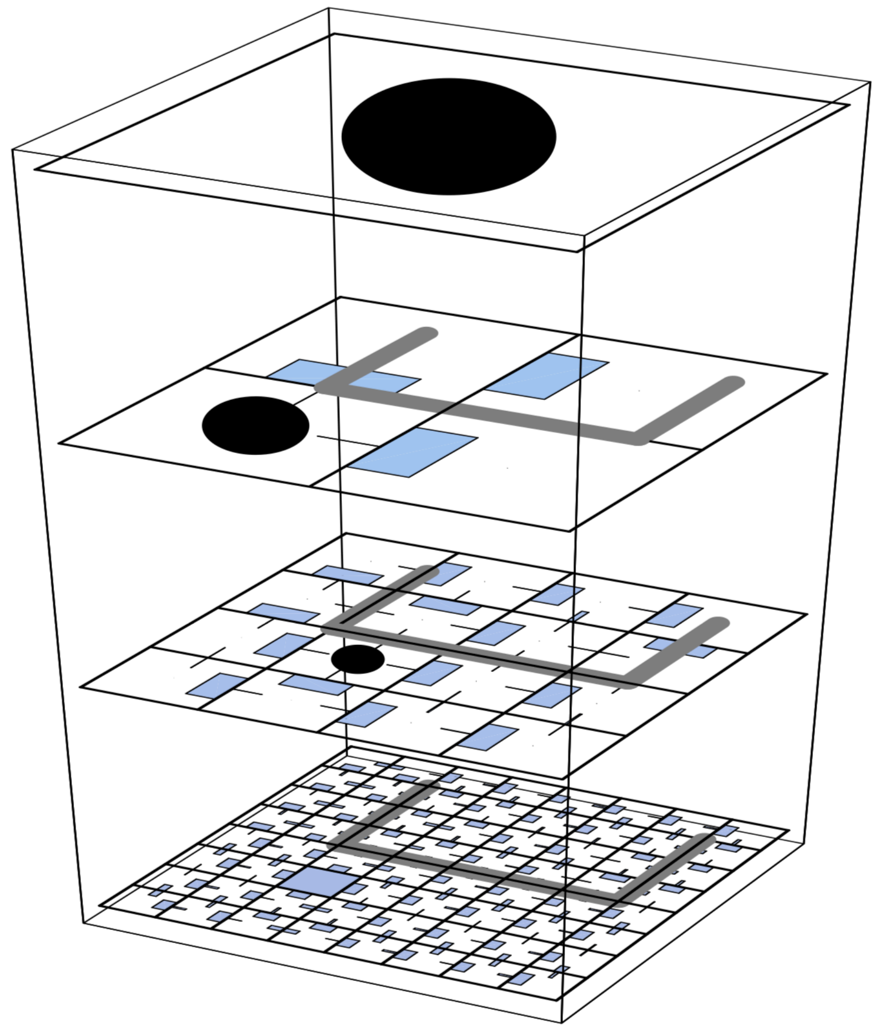
- A HAM learns a policy to decide which machine is optimal to call



Parr & Russell, 1998

Feudal Learning

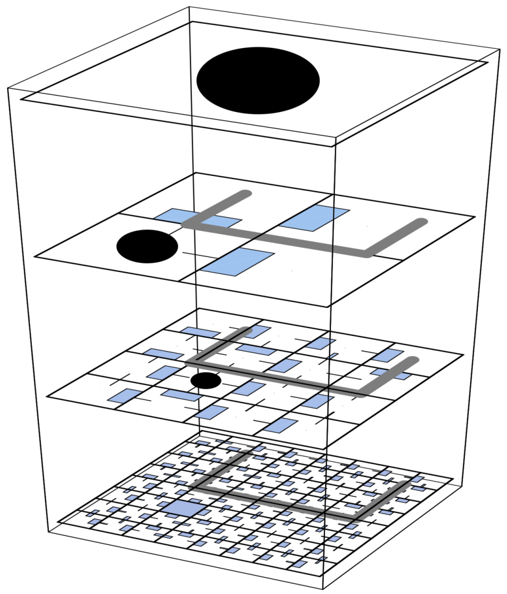
Feudal Learning



Dayan & Hinton 1993

Feudal Learning

- **Reward Hiding:**
  - The managers provide subtasks g for sub-managers
  - Managers only reward the actions if the sub-manager achieves g, irrespective of what the overall goal of the task is.
  - Low-level managers learn how to achieve low-level goals even if these do not exactly correspond together to the highest level goal.

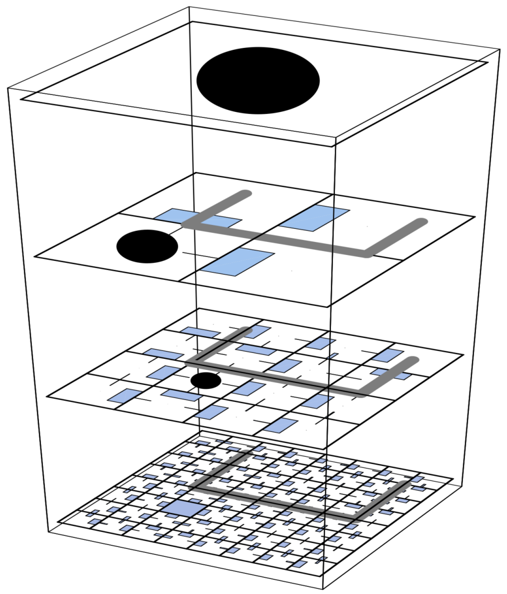


Dayan & Hinton 1993

Feudal Learning

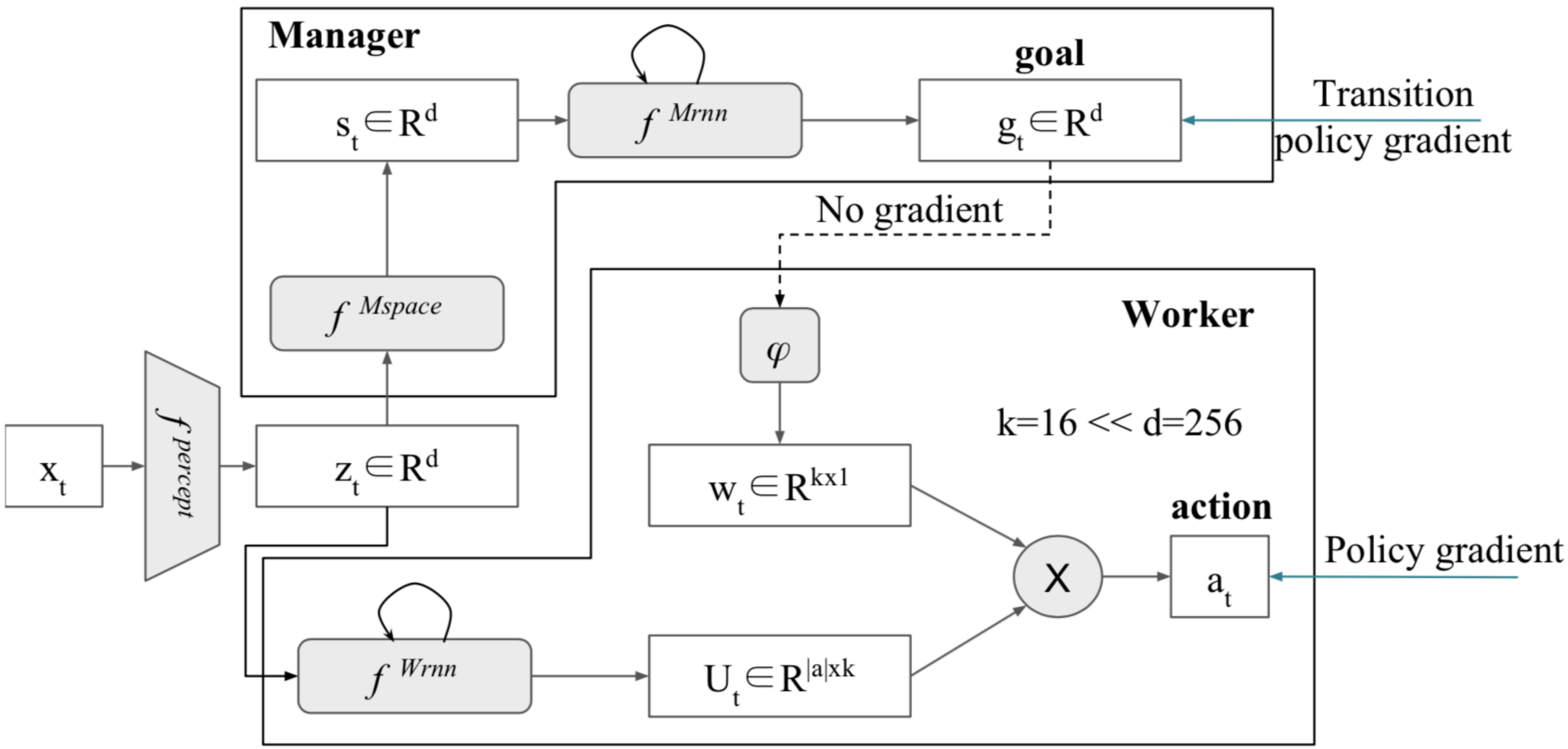
- **Reward Hiding:**
  - The managers provide subtasks g for sub-managers
  - Managers only reward the actions if the sub-manager achieves g, irrespective of what the overall goal of the task is.
  - Low-level managers learn how to achieve low-level goals even if these do not exactly correspond together to the highest level goal.

- **Information Hiding:**
  - Managers only know the state of the system at the granularity of their own choices of tasks.
  - Information is hidden both ways, upwards and downwards, in terms of the choice of sub-tasks chosen to meet the main goal.
  - Managers only reward the actions if the sub-manager achieves irrespective of what the overall goal of the task is.



Dayan & Hinton 1993

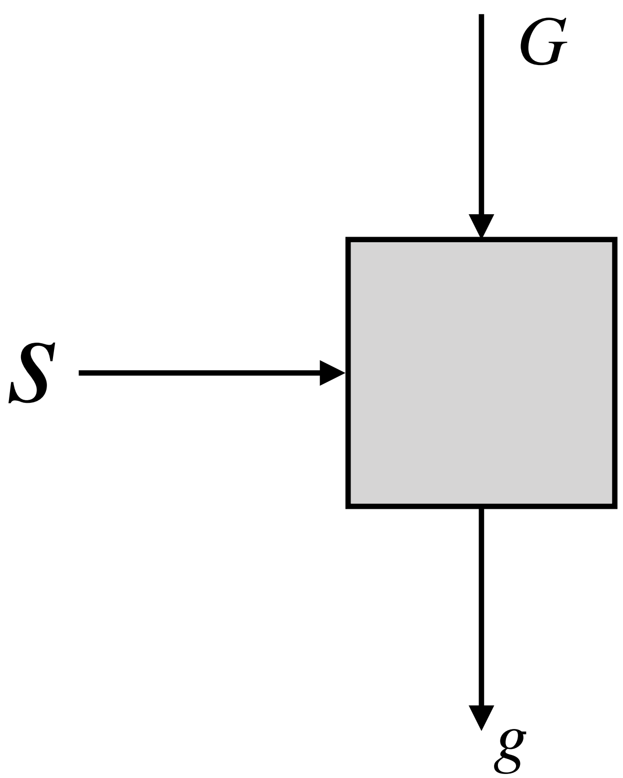
FeUdal Networks (FUN) for HRL

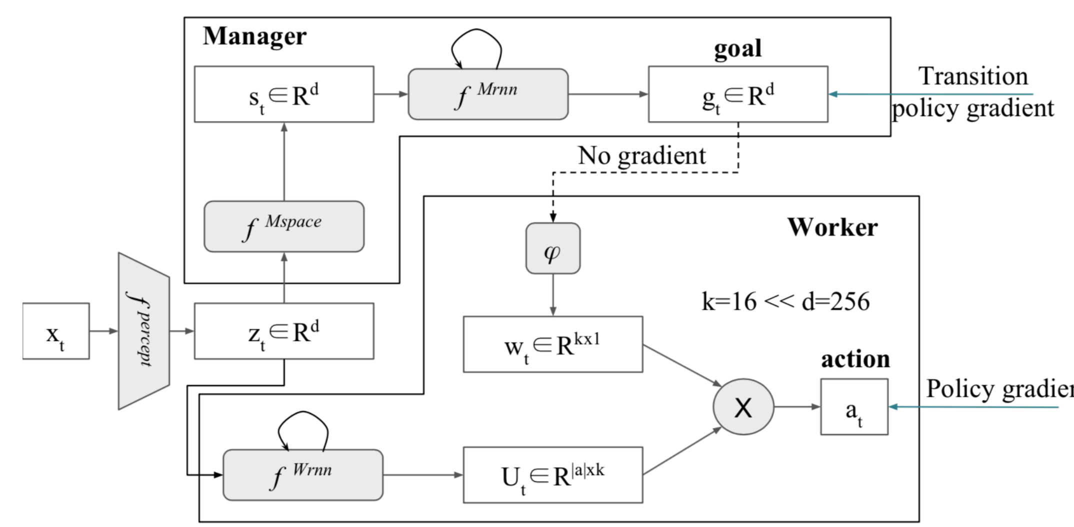


Vezhnevets et. al 2017

FeUdal Networks (FUN) for HRL

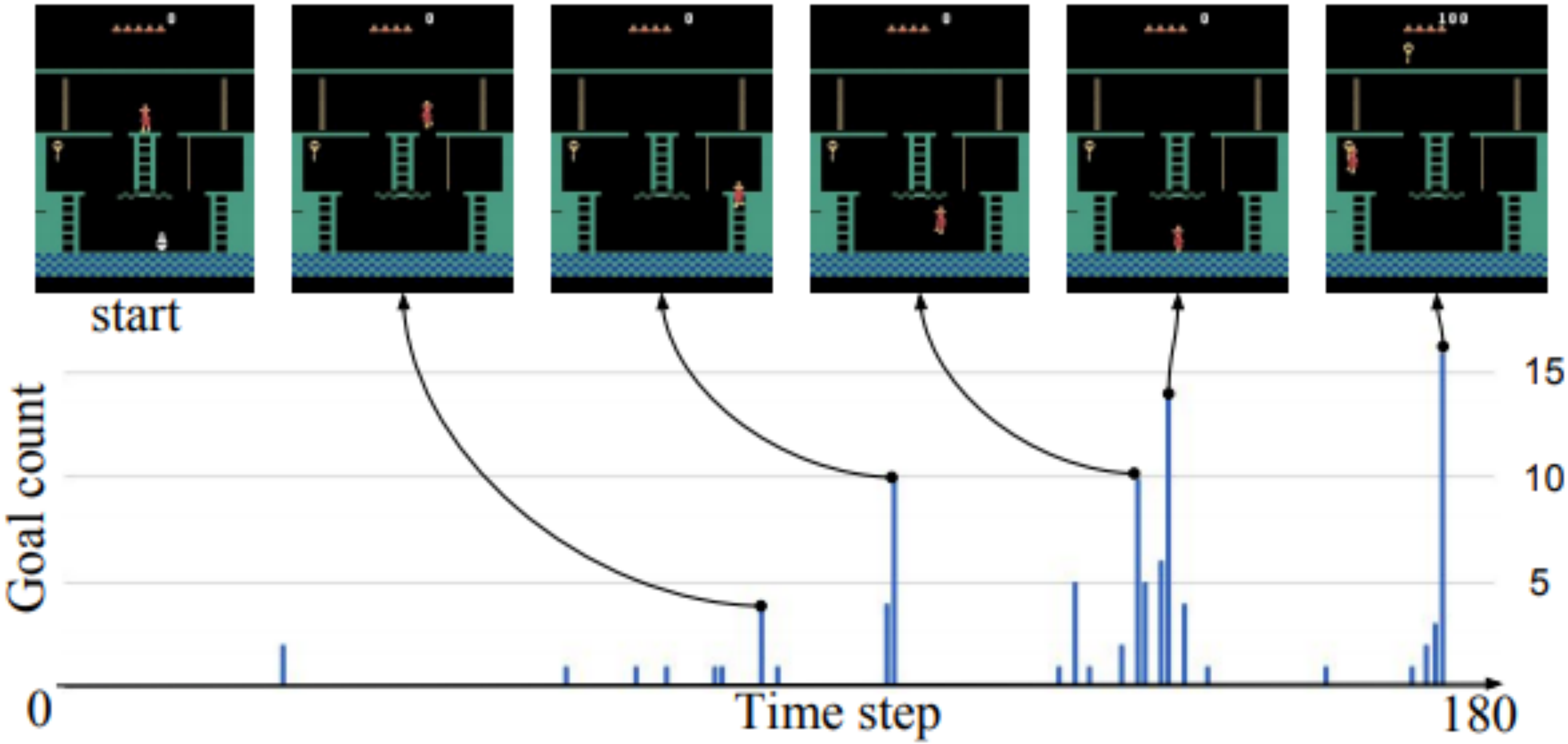
- **Key Insights:**
  - **Manager** chooses a subgoal direction that maximizes reward
  - **Worker** selects actions that maxim cosine similarity
  - FuN aims to represent sub-goals as directions in latent state space
  - Subgoals = Meaning behaviours ; Subgoals as actions





Vezhnevets et. al 2017

FeUdal Networks (FUN) for HRL



Vezhnevets et. al 2017

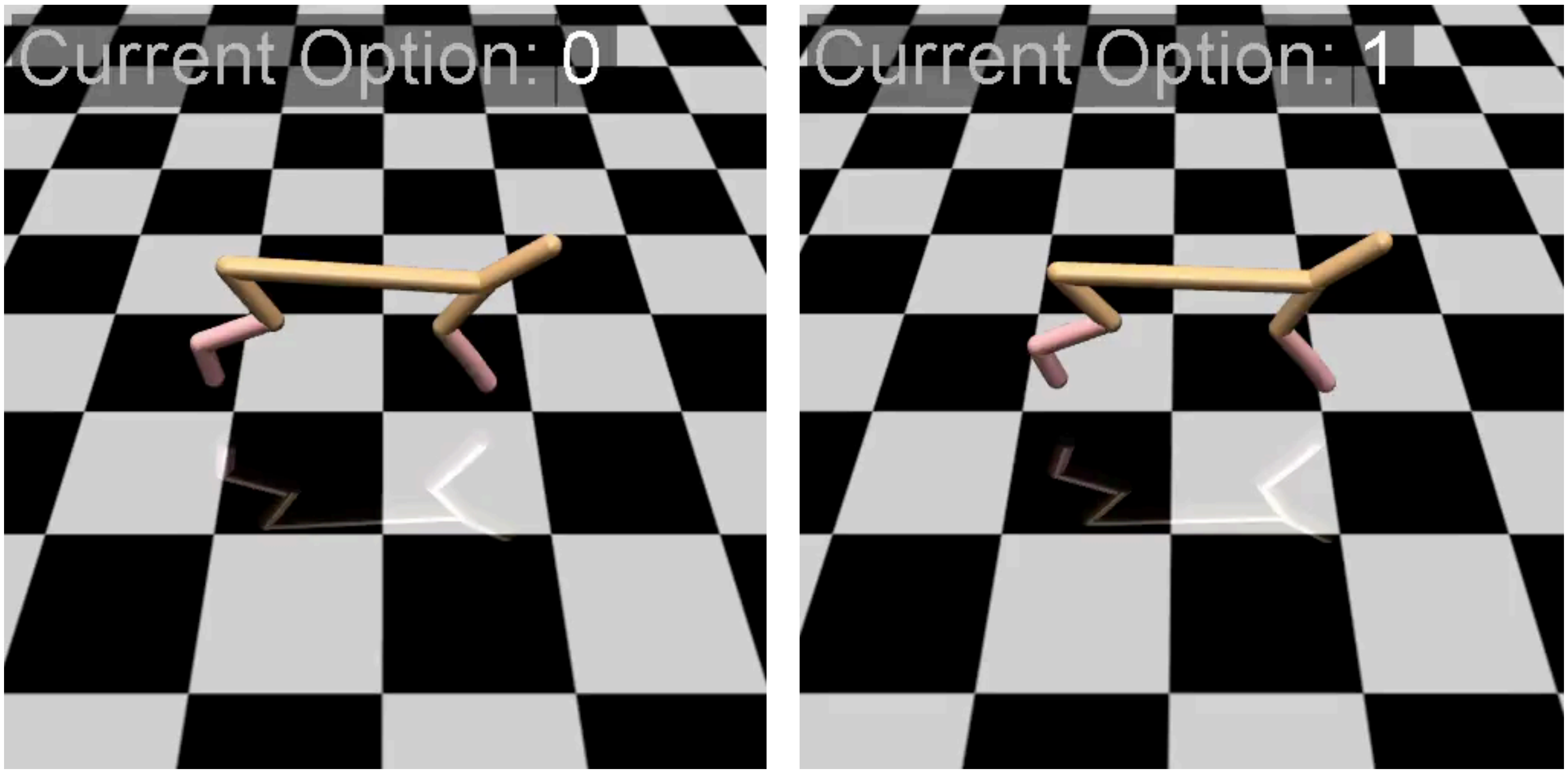
Moving towards truly scalable RL

"Stop learning tasks, start learning skills." - Satinder Singh, NeurIPS 2018

Related Literature

- MAXQ
- HIRO
- h-DQN
- Meta Learning with Shared Hierarchies
- To be completed

Demo


Current Option: 0
Current Option: 1

Questions

Extra Slides

Option-Critic

Formulation

All options are available in all states

The option value function is defined as

$$\boxed{Q_\Omega(s,\omega) = \sum_a \pi_{\omega,\theta}(a \mid s) Q_U(s,\omega,a)}$$

where $Q_U : S \times \Omega \times A \to \mathbb{R}$ is the value of executing an action in the context of a state-option pair defined as:

$$\boxed{Q_U(s,\omega,a) = r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) U(\omega, s')}$$

Option-Critic

Formulation

All options are available in all states

The option value function is defined as

$$\boxed{Q_\Omega(s, \omega) = \sum_a \pi_{\omega,\theta}(a \mid s) Q_U(s, \omega, a)}$$

where $Q_U : S \times \Omega \times A \to \mathbb{R}$ is the value of executing an action in the context of a state-option pair defined as:

$$\boxed{Q_U(s, \omega, a) = r(s, a) + \gamma \sum_{s'} P(s' \mid s, a) U(\omega, s')}$$

where $U : S \times \Omega \to \mathbb{R}$ is the option-value function upon arrival in a state:

$$\boxed{U(\omega, s') = (1 - \beta_{\omega,\nu}(s')) Q_\Omega(s', \omega) + \beta_{\omega,\nu}(s') V_\Omega(s')}$$