

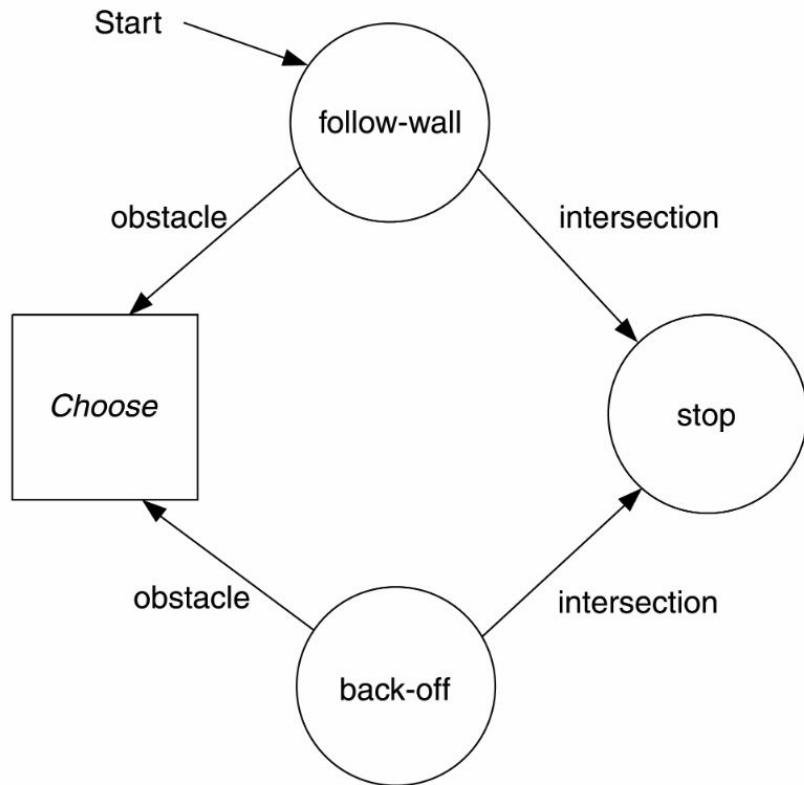
HRL: Hierarchy of Abstract Machines

B. Ravindran

HAM

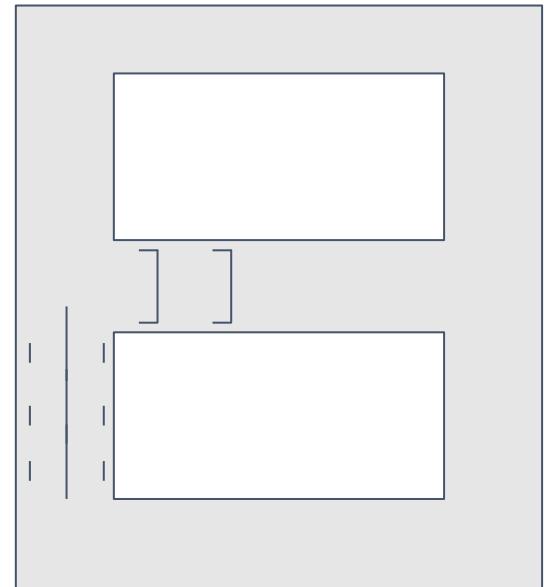
- We define a hierarchy of finite state machines.
- Each machine has 4 states:
 - Action State: takes input, gives an action.
 - Call State: takes input, calls another machine.
 - Choice State: takes input, makes a non-deterministic transition to two or more states in the same machine.
 - Stop State: stops and passes control back to the previous machine.
- For machines, Inputs are underlying states of the MDP. Outputs are primitive actions.

Example



State transition structure of a sample HAM

An example grid world with obstacles that the agent was supposed to learn to avoid.



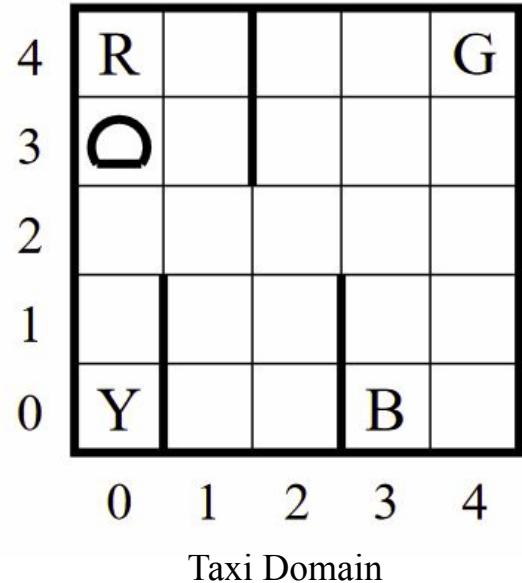
Example of an obstacle

HAM

- The SMDP for learning:
 - States: $H_i \times S_i \times M_i$
State of the base MDP S_i
A Call stack (Of machines called) H_i
Machine State M_i
 - Actions: S_m
Other machine states the agent could go to

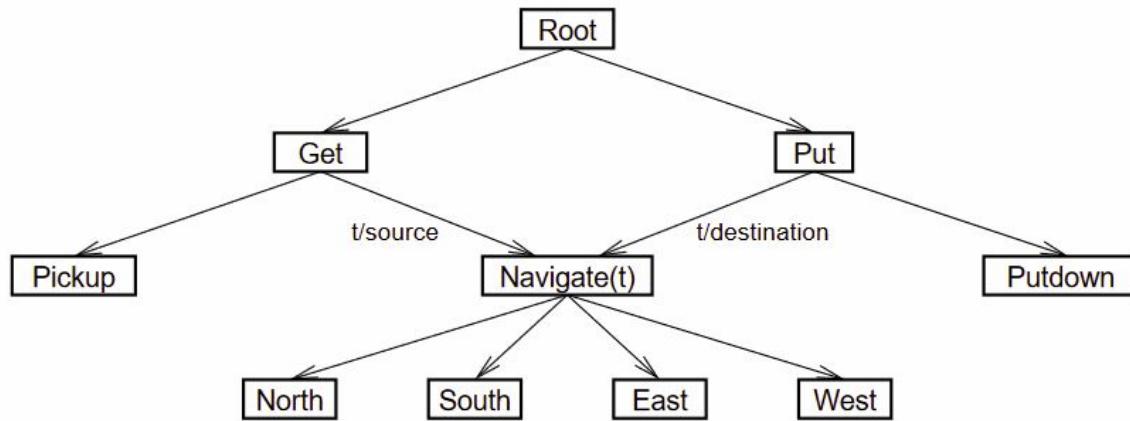
Taxi Domain (For MaxQ)

- Pick-up/drop passengers from R,G,Y & B .
- 6 primitive actions:
 - 4 navigation actions
 - pick-up action
 - drop action
- Rewards:
 - (-1) for every action
 - (+20) successful delivery
 - (-10) wrong pickup/drop execution
- States:
 - (25 squares) * (5 passenger spawns) * (4 destination)



Task Graph (For the Taxi Problem)

	R				G
3	O				
2					
1					
0	Y			B	
	0	1	2	3	4

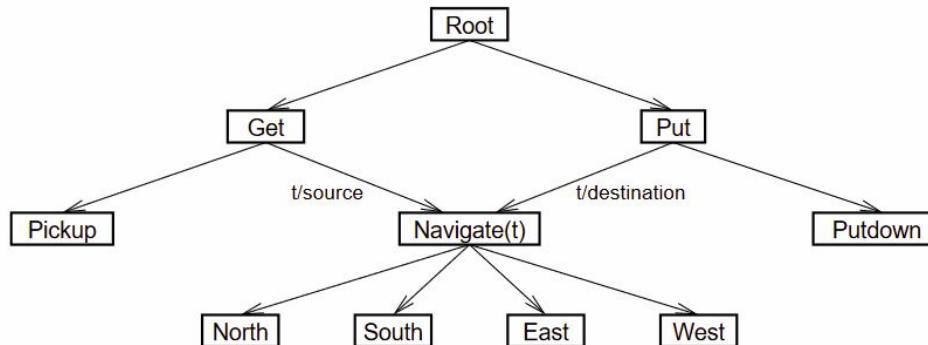


MaxQ Framework

- Let the original problem be - \mathcal{M}
- It is broken down into k sub-tasks.

$$\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_k$$

- Each sub-task, $\mathcal{M}_i = \{S_i, A_i, \bar{R}_i\}$



MaxQ Value Fn Decomposition

- For a sub-task, $\mathcal{T}_i \bigcup S_i = S$
where \mathcal{T}_i is the set of terminal states.

[NOTE: Options can have non-zero termination probability for states in initiation state as well.]

- Sub-problems are SMPDs.
We consider the SMDP corresponding to \mathcal{M}_i

$$\langle S_i, A_i, \dots \rangle$$

$$p_i(s', \tau | s, a) \sim \mathcal{M}_i$$

$$R_i(s, a, s', \tau) = E\{r_t + \gamma r_{t+1} + \dots | s_t = s, a_t = a, \pi\}$$

Value Functions & Decompositions

- Projected value function: value function considered only till completion of task ‘i’.
- Bellman equation for projected value fn:

$$v^\pi(i, s) = R^{\pi_i(s)} + \sum_{s', \tau} P_i^\pi(s', \tau | s, \pi_i(s)) \gamma^\tau v^\pi(i, s')$$

$$v^\pi(i, s) = v^\pi(\pi_i(s), s) + \sum_{s', \tau} P_i^\pi(s', \tau | s, \pi_i(s)) \gamma^\tau v^\pi(i, s')$$

$$q^\pi(i, s, a) = v^\pi(a, s) + \frac{\sum_{s', \tau} P_i^\pi(s', \tau | s, \pi_i(s)) \gamma^\tau q^\pi(i, s', \pi(s'))}{C^\pi(i, s, a)}$$

- Completion function: $C^\pi(i, s, a)$
 - Reward accumulated in sub-task ‘i’, after completing action ‘a’ from state ‘s’

$$q^\pi(i, s, a) = v^\pi(a, s) + C^\pi(i, s, a)$$

$$\begin{aligned} v^\pi(a, s) &= q^\pi(a, s, \pi_a(s)) && \text{if } a \text{ is composite} \\ &= \sum_{s'} p(s' | s, a) E[r | s, a, s'] && \text{if } a \text{ is primitive} \end{aligned}$$

- We define π as a hierarchical policy which comprises of policies followed in each sub-task.

$$\pi = \{\pi_0, \pi_1, \dots, \pi_k\}$$

$$\begin{aligned} v^\pi(0, s) &= C^\pi(0, s, a_1) + v^\pi(a_1, s) \\ &= C^\pi(0, s, a_1) + C^\pi(a_1, s, a_2) + v^\pi(a_2, s) \end{aligned}$$