Characterizing MDP Abstraction Space

David Abel

This is a work in progress, please don't distribute without talking to Dave (david_abel@brown.edu)

Contents

Acknowledgements: I am grateful to the following people for many insights and discussions that underly this work: Dilip Arumugam, Kavosh Asadi, Akshay Balsubramani, Nakul Gopalan, Ellis Hershkowitz, George Konidaris, Lucas Lehnert, Michael Littman, Elizabeth Thiry, Stefanie Tellex, Carl Trimbach, and Lawson Wong.

1 Preliminaries

Through abstraction, agents form general yet concise representations of their surroundings, affording meaningful inference in diverse and complex environments. Such a tool is an essential ingredient of intelligent behavior.

Abstraction and Reinforcement Learning (RL) are natural companions; algorithms for such a broad learning paradigm naturally benefit from a core tool of intelligence. Consequently, the role of abstraction in RL has been a central focus of prior work [2, 11, 15, 10, 19, 7, 4, 9, 21, 1, 3, 13, 5, 6, 8, 12, 10, 15, 17, 18].

The goal of this document is to better understand the space of abstractions in the context of the computational learning paradigm of RL. What makes an effective abstraction? Why? What is the relationship between a representation's *generality* (how many environments it's useful for) with its *accuracy* (how much information does it preserve)?

We offer the following contributions:

- Introduce definitions for Abstractions, Partial Abstractions, State Abstractions (SA), Action Abstractions (AA), Abstraction Equivalence, and Abstraction Alignment.
- Invoke an ordering on Partial Abstractions.
- Prove that State and Action Abstractions are both Partial Abstractions and can be be made into an Abstraction.
- Prove that State and Action Abstractions are *aligned*, in the sense that they can always form the same total abstraction.
- Prove the power of different complements to State and Action Abstractions.
- Provide formal methods for evaluating partial abstractions. *(forthcoming)*
- Present a metric for quantifying the compressibility of partial and full abstractions. *(forthcoming)*

We use Markov Decision Processes (MDPs) as the central model of environments in this work. Naturally, removing the Markov assumption offers more generality but less tractability; such extensions are of interest in future work. For an overview of MDPs, see Puterman [20].

One point on notation: throughout this document we will use $\bigcirc_G$ to denote objects belonging to a *ground* MDP (the true environmental MDP) and underscore $\bigcirc_A$ to refer to objects belonging to an abstracted MDP (the agent's model of the environment).

We begin by introducing core definitions.

.....

1.1 MDP Abstractions

Let $\mathcal{M}$ denote the space of all MDPs. Consider a fixed MDP $M_G = \langle \mathcal{S}_G, \mathcal{A}_G, \mathcal{R}_G, \mathcal{T}_G, \gamma_G \rangle \in \mathcal{M}$.

An *RL agent* interacts with M_G via the repetition of the following two steps:

(1) The agent receives a state, $s \in \mathcal{S}_G$ and reward, $r \in [0, 1]$ from M_G.

(2) The agent outputs an action, $a \in \mathcal{A}_G$, which is executed in the MDP.

The RL agent's goal is to maximize expected discounted reward.

A *planning agent* is given the full description of M_G, and possibly some distribution on start states, $s_0 \sim \Pr(\mathcal{S}_G)$ and computes a policy, $\pi_G : \mathcal{S}_G \mapsto \mathcal{A}_G$. Again, the goal of planning is to compute a π_G that maximizes expected discounted reward.

We now introduce the central definition: an MDP Abstraction. Intuitively, an MDP abstraction is a method for creating a model of an environment. Later we'll evaluate the efficacy of these abstractions by investigating the degree to which their induced models enable effective decision making in the true environment.

Definition 1 (MDP Abstraction): *An* **MDP abstraction** *is a function, $\Delta : \mathcal{M} \mapsto \mathcal{M}$, that takes as input an M_G and outputs an MDP, $M_A = \langle \mathcal{S}_A, \mathcal{A}_A, \mathcal{R}_A, \mathcal{T}_A, \gamma_A \rangle \in \mathcal{M}$ such that conversion between the two MDPs' main objects (states and actions) is well defined. That is, there exist two maps:*

1. *A map from ground to abstract states: $\mathcal{S}_G \mapsto \mathcal{S}_A$.*
2. *A map from abstract actions to ground actions: $\mathcal{A}_A \mapsto \mathcal{A}_G$*

And the resulting abstraction transitions and rewards are well formed in the sense that:

$$\mathcal{R}_A : \mathcal{S}_A \times \mathcal{A}_A \mapsto [0, 1]$$

And for each abstract action, a_A, and pair of abstract states s_A and s'_A:

$$\mathcal{T}_A(s_A, a_A, s'_A) = \Pr(s'_A \mid s_A, a_A)$$

This definition accounts for applications of abstraction to both *planning* and *Reinforcement Learning*, though we pay special attention to Reinforcement Learning. For brevity, we will use the term "abstraction" to refer to MDP abstractions throughout the document.

1.1.1 RL with Abstractions

In the context of Reinforcement Learning, the two mappings accompanying the abstraction are used to convert the agent's interactions with the true ground MDP M_G into interactions with the abstract MDP M_A. The intuition is that the agent is able to maintain it's expected input/output interface with it's environment, pictured in Figure 1.

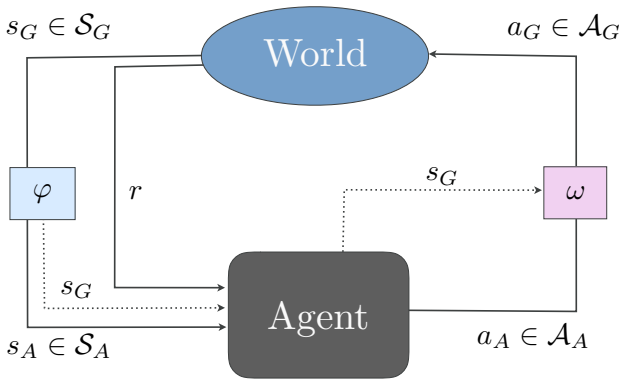


Figure 1: Abstractions and Reinforcement Learning

1.1.2 Planning with Abstractions

The solution to the planning problem given an MDP M_G is a policy, $\pi : \mathcal{S}_G \mapsto \mathcal{A}_G$. Using an abstraction, we instead compute $M_A = \Delta(M_G)$, which poses a new planning problem: compute a policy for M_A. Given the means to translate between states and actions in the ground an abstract MDP, a policy for M_A, π_A, may be used to compute a policy[1] for M_G as follows:

$$\forall_{s_G \in \mathcal{S}_G} : \pi_G(s_G) \triangleq \omega\left(\pi_A\left(\varphi(s_G)\right)\right) \tag{1}$$

Where ω takes as input an abstract action, $a_A = \pi_A\left(\varphi(s_G)\right)$ and maps it to a ground action, a_G. As we will see, this action mapping will also need to take s_G as input.

1.2 Partial Abstractions

Given above definition of an MDP abstraction, we now turn to a complementary notion: that of a *partial abstraction*. Intuitively, a partial abstraction, denoted ρ, is a function which, along with some other partial abstraction, creates an MDP abstraction:

Definition 2 (Partial MDP Abstraction): *A* **partial MDP abstraction**, *ρ_1, is a non-empty set of functions which satisfies the following two properties:*

1. *ρ_1 is not an MDP abstraction.*
2. *Pairing ρ_1 with a non-empty set, ρ_2, defines an MDP abstraction.*

For instance, consider a function that computes an abstract action space, given M_G. We need some way to inform an abstract state space, transition function, reward function, and γ. We introduce some terminology for the process of pairing a partial abstraction with its complementary constituents:

[1] Though of course, under a bad abstraction, this policy may be completely useless in M_G

Definition 3 (Completing a Partial Abstraction): *For any two non-empty sets of functions, ρ_1 and ρ_2, if $\rho_1 \cup \rho_2$ defines an MDP abstraction, we call the process of joining ρ_1 with ρ_2* **completing the partial abstraction**, *ρ_1.*

We will later see that, on their own, state and action abstractions are simply *partial abstractions*, in the sense that we also need instructions on how to form $\mathcal{T}_A$, $\mathcal{R}_A$, and γ. There are standard methods for completing both state and action abstractions, but here we instead consider all possible methods of *completing* these partial abstractions.

In light of the terminology of a *partial* abstraction, we will occasionally use the term *full* or *complete* abstraction to contrast an MDP abstraction from any partial counterparts.

1.2.1 Ordering Partial Abstractions

Consider the five different components of an MDP, displayed in Figure 2. For a partial abstraction to be completed, we need some method for computing all five of these components.

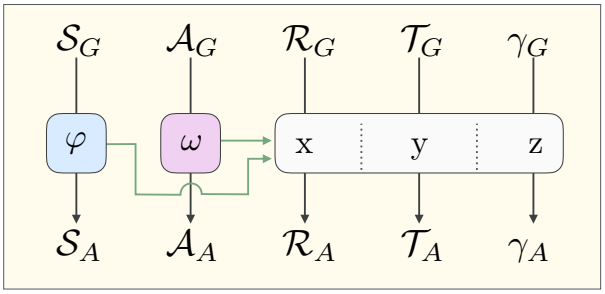


Figure 2: The Different Components of Abstraction

We now impose an ordering on partial abstractions based on the number of components defined by the partial abstraction:

Definition 4 (Level k MDP Abstraction): *A partial abstraction, ρ, is said to be a* **level k abstraction**, *denoted $\rho \in L_k$, if it computes exactly k of the components of the abstract MDP. That is, if $\rho \in L_k$, $\rho \notin L_{k-1}$.*

In principle, it is possible to have levels L_1 through L_5, where L_i indicates that i of the 5 components of $\mathcal{M}_A$ are defined. Note that partial abstractions are any of L_1 through L_4. However, for abstraction purposes, γ_A provides no attractive insights, so we focus on the four primary entities of an MDP, $\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}$. Therefore, any abstraction at L_4 is effectively a complete MDP abstraction, while any abstraction at L_1, L_2 or L_3 is a partial abstraction.

Since $\mathcal{R}_A$ and $\mathcal{T}_A$ are functions of abstract states and abstract actions, the only partial abstractions that inform one or two components of the abstract MDP (L_1 or L_2 partial abstractions) are those that inform $\mathcal{S}_A$ and $\mathcal{A}_A$. Thus, L_1 and L_2 are rather simple:

Level	Components of $\mathcal{M}_A$ Defined
L_1	$\mathcal{S}_A$ `xor` $\mathcal{A}_A$
L_2	$\mathcal{S}_A, \mathcal{A}_A$
L_3	$(\mathcal{S}_A, \mathcal{A}_A, \mathcal{R}_A)$ `xor` $(\mathcal{S}_A, \mathcal{A}_A, \mathcal{T}_A)$
L_4	$(\mathcal{S}_A, \mathcal{A}_A, \mathcal{R}_A, \mathcal{T}_A)$

We use the terminology $f \in L_1$ to denote that the partial abstraction defined by the function f is a member of L_1.

Ultimately, we will see that state and action abstractions are typically posed as complete abstractions, but there is an advantage to reducing them to their core. Namely, we can better evaluate the degrees of freedom associated with the partial abstraction, and establish concretely the sense in which the abstractions are equivalent.

We now turn to precisely this question: when are two abstractions equivalent? When are two *partial* abstractions similar or equivalent?

1.2.2 Alignment & Equivalence

Definition 5 (Equivalent Abstractions): *We say that two abstractions, Δ_1 and Δ_2, are equivalent if, for all MDPs $M \in \mathcal{M}$, $\Delta_1(M) = \Delta_2(M)$. That is, the abstractions induce the same abstract state-action space, transition function, reward function, and γ.*

With partial abstractions we have a weaker notion, that of *alignment*, which states that there is some way to complete each of the two partial abstractions to make them the same (but the means of completion could be different):

Definition 6 (Aligned Partial Abstractions): *Two partial abstractions, ρ_1 and ρ_2, are said to be* **aligned**, *denoted $\rho_1 \rightleftharpoons \rho_2$, if, for any completion of ρ_1, denoted as C_1, there exists a completion of ρ_2, denoted as C_2, such that $\rho_1 \cup C_1 = \rho \cup C_2$. That is, the full abstractions induced by $\Delta_1 = \rho_1 \cup C_1$ and $\Delta_2 = \rho_2 \cup C_2$ are equivalent.*

As we will see, it is not always the case that the alignment operator is symmetric. That is, there are cases where $\rho_1 \leftharpoons \rho_2$, but $\rho_2 \not\leftharpoons \rho_1$. Intuitively, the completions of ρ_1 may only correspond to a small subset of completions of ρ_2.

We also have a strong notion of equivalence, though as we will see, alignment is the piece we care about:

Definition 7 (Equivalent Partial Abstractions): *Two partial abstractions, ρ_1 and ρ_2, are* **equivalent** *if they are aligned, and for all completions of ρ_1, C_1, using this to complete ρ_2 results in an equivalent abstraction: $\Delta_1 = \Delta_2 = \rho_1 \cup C_1 = \rho_2 \cup C_1$*

1.2.3 Summary

We now have the relevant terminology on the table. In summary, we introduced:

- *MDP Abstraction*: a function that transforms the objects of any MDP to the objects of a smaller MDP.
- *Partial MDP Abstraction*: a collection of functions which, when coupled with another partial abstraction, forms an MDP abstraction.
- *Level k MDP Abstraction*: a level k abstraction, $\rho \in L_k$, implies that ρ can compute at most k of the four relevant components of the abstract MDP.
- *Equivalent MDP Abstractions*: we say that two abstractions are equivalent if they always induce the same abstract MDPs.
- *Aligned Partial MDP Abstraction*: two partial abstractions are aligned, $\rho_1 \rightleftharpoons \rho_2$, if there is some (possibly different) method of completing each of them that induces two equivalent abstractions.
- *Equivalent Partial MDP Abstractions*: we say two partial abstractions, ρ_1 and ρ_2 are equivalent if, for a given completion of the first that results in abstraction Δ_1, there exists a completion of ρ_2, Δ_2 such that $\Delta_1 = \Delta_2$.

We now turn to one of the central question of this document:

Is there a sense in which state abstractions and action abstractions are equivalent?

.....

2 State Abstraction

A popular approach to MDP abstraction is to compress the state space, $\mathcal{S}$, by employing state abstraction.

Definition 8 (State Abstraction): *Given a ground MDP:*

$$M_G = \langle \mathcal{S}_G, \mathcal{A}_G, \mathcal{T}_G, \mathcal{R}_G, \gamma_G \rangle \tag{2}$$

A **state abstraction** *is a function, $\varphi : \mathcal{S}_G \mapsto \mathcal{S}_A$, where $\mathcal{S}_A$ is some abstract state space such that $|\mathcal{S}_A| < |\mathcal{S}_G|$. We let SA denote the space of all possible action abstractions.*

The idea is to group states into clusters, where each cluster define a state in the abstract state space. For instance, consider the following state abstraction that reduces a six state MDP (pictured on the left) to three states in the abstract (pictured on the right):

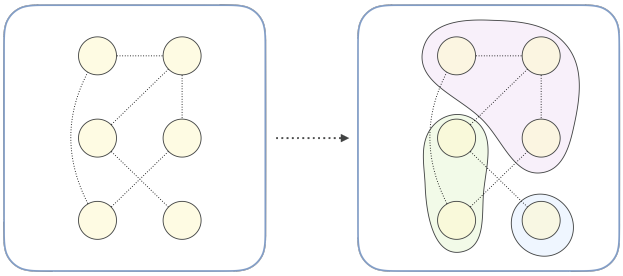


Figure 3: State Abstraction: The ground MDP is pictured on the left, and on the right, the three colored clusters define the abstract states.

Notice that a state abstraction *only* defines an abstract state space, but does not specify $\mathcal{A}_A, \mathcal{R}_A, \mathcal{T}_A$ or γ_A. Thus, it is a partial abstraction. Traditionally, state abstractions are accompanied by a weighting function, $w : \mathcal{S}_G \mapsto [0, 1]$, that dictates the abstract transition and reward function. The weighting function must satisfy:

$$\forall_{s_A \in \mathcal{S}_A} : \left(\sum_{s_G \in \varphi^{-1}(s_A)} w(s_G) \right) = 1$$

As we will see, however, there are other possible completions of the partial abstraction defined by φ that are more powerful.

3 Action Abstraction

An alternative strategy is to perform abstraction at the level of actions. The most common formalism is an Option, introduced by Sutton et al. [21].

Definition 9 (Option): *Given a ground MDP, $M_G = \langle \mathcal{S}_G, \mathcal{A}_G, \mathcal{T}_G, \mathcal{R}_G, \gamma_G \rangle$, an Option is a triple, $\langle \mathcal{I}, \beta, \pi \rangle$, where:*

- $\mathcal{I} : \mathcal{S}_G \mapsto \{0, 1\}$*: we call the initiation condition. Denotes a predicate on states, indicating in which state the Option may be initiated.*
- $\beta : \mathcal{S}_G \mapsto [0, 1]$*: we call the termination condition. Assigns a Bernoulli to every state, denoting the likelihood of termination in each state. That is, if the MDP transitions to state s_G, the option terminates with probability $\beta(s)$.*
- $\pi : \mathcal{S}_G \mapsto \mathcal{A}_G$*: Denotes a policy.*

That is, an option, o, augments a given action space $\mathcal{A}_G$ to form $\mathcal{A}_A = \mathcal{A}_G \cup \{o\}$.

An agent planning or learning with $\mathcal{A}_A$ may only execute an option o in states where $o.\mathcal{I}(s) = 1$. Then, the agent deterministically follows the policy $o.\pi$ until reaching any state s'_G such that $x \sim o.\beta(s'_G) = 1$.

Critically, we only allow for initiation conditions, termination conditions, and policies that are functions of *state*, and not of time-step. For instance, an option that always terminates after four time steps, while reasonable, is not captured by this definition. We leave considerations of this sort for future work. Here, we define a non-empty set of options to be an *action abstraction*:

Definition 10 (Action Abstraction): *Given a ground MDP M_G, a set of options, $\omega = \langle o_1, \ldots, o_\ell \rangle$, for $\ell \geq 2$, defines an* **action abstraction** *when it replaces the action space of primitive actions. We let AA denote the space of all possible action abstractions.*[a]

[a] We insist that $\ell \geq 2$, since if there is only one abstract action, the abstract MDP reduces to a Markov Chain and the agent no longer makes any decisions.

A central result from Sutton et al. [21] is as follows:

Theorem [21]: Augmenting an MDP's action space with a set of options results in a Semi-Markov Decision Process (MDP + Options = SMDP). That is:

$$M_S = \langle \mathcal{S}_G, \mathcal{A}_G \cup \omega, \mathcal{R}_{G,\omega}, \mathcal{T}_{G,\omega}, \gamma \rangle \tag{3}$$

Where $\mathcal{R}_{G,\omega}$ and $\mathcal{T}_{G,\omega}$ denote the reward and transition functions determined by the multi-time model outlined in Sutton et al. [21].

We will present an analogous result: any action abstraction is a partial abstraction, and with a trivial completion (the multi-time model and a natural state aggregation), is an MDP abstraction.

That is, *replacing* a MDP's action space with a set of options, and using the multi-time model, results in a new MDP (not an SMDP).

.....

We now introduce an important modeling concept related to Action Abstractions: the *option template*.

Definition 11 (Option Template): *An* **option template**, *denoted* τ *is a precondition for an option:*

$$\langle \mathcal{I}, \square, \square \rangle \tag{4}$$

An option template isolates a subset of the space of all possible options - namely, those that are only executable in some states, but may have different termination conditions and policies.

As we will see in the next section, option templates and state abstraction are nearly identical.

.....

4 Results

We first show that our ordering on partial abstractions also orders the space of partial abstractions by size:

Remark: $|L_1| \leq |L_2| \leq |L_3| \leq |L_4|$, where $|L_x|$ denotes how many unique partial abstractions are members of L_x.

Proof.

For any choice of k and j such that $k < j$, let ρ_k and ρ_j denote partial abstractions belonging to L_k and L_j respectively.

Observe that for any value of k, there will always exist at least one valid completion of ρ_k:

(k =1) Note that ρ_1 must define either $\mathcal{S}_A$ or $\mathcal{A}_A$ (but not both)

First suppose the former ($\mathcal{S}_A$). Then let $\mathcal{A}_A = \{a_A\}$ with mapping $\omega(a_A) = a_{G,1} \in \mathcal{A}_G$, and consider the trivial abstract deterministic transition function:

$$\forall_{s_A \in \mathcal{S}_A} : \mathcal{T}_A(s_A, a_A, s_A) = 1$$

Where $\mathcal{R}_A$ and γ_A can be defined arbitrarily.

Now suppose the latter ($\mathcal{A}_A$). Then let $\mathcal{S}_A = \mathcal{S}_G$, and consider the trivial transition function:

$$\forall_{s_A \in \mathcal{S}_A} : \mathcal{T}_A(s_A, a_A, s_A) = 1$$

where a_A is a fixed but arbitrary abstract action. Again, $\mathcal{R}_A$ and γ_A can be defined arbitrarily.

(k =2) Now ρ_2 must define $\mathcal{S}_A$ and $\mathcal{A}_A$. We find trivial completions by similar reasoning to the $k = 1$ cases.

(k =3) ρ_3 defines either $(\mathcal{S}_A, \mathcal{A}_A, \mathcal{R}_A)$ or $(\mathcal{S}_A, \mathcal{A}_A, \mathcal{T}_A)$, but not both. In the former cases, we can again define a trivial transition model, and in the latter, we can again define a trivial reward function. Any $\gamma_A \in (0, 1)$ will do.

Therefore, for $k < j$, for any $\rho_k \in L_k$ there is at least one $\rho_j \in L_j$ (though clearly there are many more). Thus, $|L_1| \leq |L_2| \leq |L_3| \leq |L_4|$. □

With this ordering in mind, we now turn our attention to state abstractions and action abstractions.

.....

Remark: Any state abstraction function is a partial abstraction of L_1, and can be trivially be completed.

Proof.

Consider a state abstraction, φ. For a given MDP, M_G, we compute an abstract state space as follows:

$$\mathcal{S}_A = \cup_{s \in \mathcal{S}_G} \varphi(s) \tag{5}$$

Further, φ does not compute $\mathcal{A}_A, \mathcal{R}_A$ or $\mathcal{T}_A$, and so $\varphi \in L_1$.

To complete φ, we need to identify a set of functions which, together with φ, form an MDP abstraction. That is, we need to define well formed $\mathcal{A}_A, \mathcal{R}_A$, and $\mathcal{T}_A$, and the following two criteria need to be satisfied:

1. We can map ground states to abstract states.
2. We can map abstract actions to ground actions.

The state abstraction φ already satisfies the first criteria.

To satisfy the second criteria, we need a mapping from abstract actions to ground actions that induce a well formed abstraction transition and reward function. In this sense, we have a huge degree of freedom: there exist *many* possible action abstractions we could adjoin to φ that lead to a valid abstract MDP.

The identity function satisfies the needed criteria - that is, the abstract action space is identical to the ground: $\mathcal{A}_A = \mathcal{A}_G$

Then, constructing $\mathcal{T}_A$ and $\mathcal{R}_A$ is done per the procedure outlined by Li et al. [16] and Abel et al. [1], which also depends on the choice of a weighting scheme:

$$\begin{aligned}
\mathcal{R}(s_A, a) &= \sum_{s_G \in \varphi^{-1}(s_A)} \mathcal{R}(s_G, a) \cdot w(s_G) \\
\mathcal{T}(s_A, a, s'_A) &= \sum_{s_G \in \varphi^{-1}(s_A)} \sum_{s'_G \in \varphi^{-1}(s'_A)} \mathcal{T}(s_G, a, s'_G) \cdot w(s_G)
\end{aligned}$$

Where:

$$\begin{aligned}
&\forall_{s_A \in \mathcal{S}_A} : \left(\sum_{s_G \in \varphi^{-1}(s_A)} w(s_G) \right) = 1 \\
&\forall_{s_G \in \mathcal{S}_G} : 0 \leq w(s_G) \leq 1
\end{aligned}$$

Thus, φ can be completed by the identity action map and any weighting function, w. □

Remark: Any action abstraction is a partial abstraction, and can be completed.

Proof.

Consider a ground MDP, M_G and an action abstraction, ω. Letting $\mathcal{A}_A = \omega$, we observe that ω is a partial abstraction, and $\omega \in L_1$.

To complete ω, we need to define $\mathcal{S}_A, \mathcal{T}_A, \mathcal{R}_A$, in a way that satisfies the two mapping criteria. Again we note a degree of freedom.

First, note that ω satisfies the action mapping: at any time step, a ground state paired with an active[a] option produces a ground action via the option's policy, $o.\pi(s_G) \mapsto s_A$.

To satisfy the state mapping criteria, we note a further degree of freedom. Here, however, our abstraction action space imposes a constraint: the abstract state space must be sufficient for distinguishing when option's are active, otherwise we destroy the action mapping from abstract to ground. One trivial valid option is to again use an identity map: $\varphi(s_G) = s_G$, so $\mathcal{S}_A = \mathcal{S}_G$. Choosing the identity maps lets each option's constituent functions apply to abstract states, too.

The transition functions and reward functions may be defined identically to the multi-time model from Sutton et al. [21] and Jong and Stone [12].

Consequently, we have constructed a function which, given M_G, outputs M_A, such that there are two mappings, φ and ω, and $\mathcal{T}_A$ and $\mathcal{R}_A$ are well defined.[b] □

[a] We're executing the option's policy or we're choosing amongst option's whose initiation condition is True.

[b] We also require that γ_A be a bit different - handled similarly to [21].

.....

One might wonder whether these completions of a state abstraction or action abstraction are unique; are there other possible completions? If there are many, how should we choose amongst them? Why use the weighting function or multi-time model?

4.1 Main Results

In this section, we'll show that $SA \rightleftharpoons AA$: that State Abstractions and Action Abstractions are aligned. More specifically, for any φ and ω, there exists a completion of φ C_φ that produces the abstraction Δ_φ and vice versa. However, the method for completing the state abstraction must be different than the weighting function discussed above. If a weighting function is used, then there are action abstractions that have no aligned state abstraction.

We first show that action abstractions can be converted from L_1 to L_2, in a result heavily aligned with the results of Konidaris et al. [14].

Remark: Any action abstraction, ω, induces a particular state abstraction φ_ω, so $\omega \cup \varphi_\omega \in L_2$. Further, this state abstraction results in the smallest abstract state space that is capable of representing option availability.

Proof.

Given a ground MDP, M_G and an action abstraction, ω.

Consider the bit string obtained by concatenating the n unique[a] initiation condition's of the set of ℓ options ($n \leq \ell$), applied to a ground state:

$$\varphi(s_G) = \mathcal{I}_1(s_G) \circ \mathcal{I}_2(s_G) \circ \ldots \circ \mathcal{I}_n(s_G) \tag{6}$$

Then each ground state is grouped into an abstract state according to which relevant initiation conditions are active in that ground state. Now, a binary string corresponds to an abstract state (for at most $2^{|\omega|}$ abstract states), producing a mapping from ground states to abstract states.

Further, this abstract state space is the smallest state space sufficient for representing when each option is available.

Suppose this is not the case: then there exists an abstract state space smaller than the one defined by Equation 6. Call this $\mathcal{S}_{small}$.

Recall that by definition of an Action Abstraction, $|\omega| \geq 2$. Then by the pigeonhole principle, there exists an abstract state $\tilde{s}$ in $\mathcal{S}_{small}$ such that two ground states s_a and s_b map to $\tilde{s}$, but have different initiation condition's active. That is:

$$\mathcal{I}_1(s_a) \circ \ldots \circ \mathcal{I}_n(s_a) \neq \mathcal{I}_1(s_b) \circ \ldots \circ \mathcal{I}_n(s_b) \tag{7}$$

Then, when the agent is in the abstract state $\tilde{s}$, there is no well formed action mapping that preserves which options are active.[b] □

[a] Two initiation conditions are equivalent if their output agrees for all states in the MDP

[b] One could assume that in this situation the agent checks its true ground state, and determines abstract action availability accordingly. However, this is suspiciously similar to augmenting the abstract state space to retain this information.

Remark: Any set of option templates, $\overline{\tau} = \{\tau_1, \ldots, \tau_\ell\}$, is an L_1 abstraction. Further, there exists a state abstraction φ that is *equivalent* to the templates.

Proof.

The above method for inducing a state abstraction from a set of options only depends on the initiation conditions. Thus, any set of options with the same initiation conditions forms

the same abstract state space, which could be analogously defined by φ. Therefore, $\overline{\tau} \in L_1$, and any completion of φ will also complete $\overline{\tau}$ and vice versa. □

Corollary: Consequently, if assigning β and π to each of the templates defines $\mathcal{T}_A$ and $\mathcal{R}_A$, this set of β's and π's also deifnes $\mathcal{T}_A$ and $\mathcal{R}_A$ for φ.

Remark: Any state abstraction induces a set of option templates.

Proof.

Consider the state abstraction φ, which maps each state in $\mathcal{S}_G$ to $\mathcal{S}_A$. Then the set $\mathcal{S}_A$ forms a set of candidate initiation conditions and termination conditions. That is we create an initiation condition $\mathcal{I}_i$ for each abstract state $s_{A,i}$ such that:

$$\forall_{s_G \in \mathcal{S}_G} : \mathcal{I}_i(s_G) \equiv (\varphi(s_G) = s_{A,i}) \tag{8}$$

We denote this set of initiation conditions, $\bar{\mathcal{I}} = \left\{\mathcal{I}_1, \ldots, \mathcal{I}_{|\mathcal{S}_A|}\right\}$.

Let each τ then be defined as any pairs of the conditions $(a, b) \in \bar{\mathcal{I}} \times \bar{\mathcal{I}}$. That is, each initial condition and terminal condition pair, $\langle \mathcal{I}_i, \mathcal{I}_j \rangle$ defines an option template, which effectively states "*activate if the agent starts in abstract state $s_{A,i}$, and terminate if the agent ends in abstract state $s_{A,j}$*". Each possible combination defines the set of option templates (every possible edge between every abstract state). □

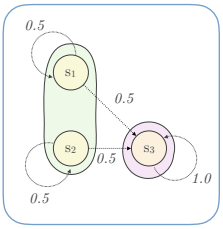
4.2 $AA \not\eqsim SA + w$

Consider a set of option templates, $\overline{\tau} = \{\tau_1, \ldots, \tau_\ell\}$. For any completion of the templates, we get the same abstract state space. However, the resulting abstractions induce different transition and reward functions. Similarly, for a state abstraction, different weighting schemes induce different abstract transitions and rewards. We now show that these traditional methods for completing State and Action abstractions are *not* aligned.

Remark: For a given set of ℓ option templates, $\overline{\tau}$ and their corresponding state abstraction, φ_τ, there exist termination conditions and policies $\pi_1, \pi_2, \ldots, \pi_\ell$ that will generate an $\mathcal{T}_A$ and $\mathcal{R}_A$ for which no weighting schema, coupled with φ_τ, can create.

Proof.

Consider a ground MDP with three ground states, s_1, s_2, s_3. Suppose s_1 and s_2 are grouped into the same abstract state, while s_3 becomes its own abstract state. That is, $\varphi(s_1) = \varphi(s_2) \neq \varphi(s_3)$:



Where the numbers along the edges denote transition probability for action a_1 (we may assume other actions exist, but are not needed for the counter example).

Consider the following option:

$$\mathcal{I}(s) = 1$$
$$\beta(s) = \begin{cases} 1 & s = s_3 \\ 0 & \text{o/w} \end{cases}$$
$$\pi(s) = a_1$$

That is, the option always executes a_1 (pictured by the dotted lines). and terminates when the agent arrives in s_3.

Suppose there is only one action in the ground MDP, a_1, with transitions as follows:

$$\mathcal{T}_G(s_i, a_1, s_j) = i \begin{bmatrix} & j & \\ 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \\ 0 & 0 & 1 \end{bmatrix}$$

Then any weighting function, w, will assign positive probability to the transition $\mathcal{T}_A(s_1, a_1, s_2)$. But this is not the case for the above option, which may set that probability to be zero. □

.....

4.3 $SA \leftrightharpoons AA$

Now we consider the converse:

Remark: Consider any state abstraction along with a completing weighting function, $\varphi \cup w$, and the set of option templates analogous to φ, denoted $\overline{\tau}$. We can assign termination conditions and policies to the templates such that the two abstractions are equivalent:

$$\varphi \cup w = \overline{\tau} \cup \overline{\beta} \cup \overline{\pi} \tag{9}$$

Proof.

> Given arbitrary $\varphi \cup w$ and $\overline{\tau}$, we assign a β and π to each $\langle \mathcal{I}, \square, \square \rangle$ that induces the equivalent MDP abstractions as follows:
>
> TODO: Finish this proof...

□

4.4 Summary

We summarize the above remarks in our main result:

Main Result: The space of state abstractions, SA, and action abstractions, AA, have the following properties:

1. All $\varphi \in SA$ are L_1 abstractions, along with their analogous option templates.
2. Without further instruction, all $\omega \in AA$ are L_1 abstractions. However, if specified, any ω can induce $\mathcal{S}_A$, too, and so $\omega \cup \overline{I}_\omega$ is an L_2 abstraction.
3. $SA \leftrightharpoons AA$: Any $\varphi \in SA$ has an aligned $\omega \in AA$, and any $\omega \in AA$ has an aligned $\varphi \in SA$.
4. Weighting functions are less powerful than options: any φ completed by a weighting function w has an aligned $\omega \in AA$, but for some $\omega \in AA$, there is *not* an aligned $\varphi \cup w$.
5. Option templates induce the smallest possible state abstraction that preserve a well formed action mapping.

5 Evaluation

We now turn to the question of *evaluating* abstractions. In particular, we seek the answer to three questions:

1. Given a state abstraction or set of initiation conditions, which set of initiation conditions/policies $\overline{\pi}$ maximizes performance on the abstract MDP?
2. Given a set of policies, which state abstraction or option templates maximizes performance on the abstract MDP?
3. Given the ground MDP, which abstraction maximizes performance on the MDP?

TODO: Focusing on this presently

6 Compression

Since $SA \subset AA$, talking about the compression achieved by any $aa \in AA$ is sufficiently general.

We capture the compression achieved by an abstraction as the reduction in the entropy rate of the stochastic Markov Chain induced by a fixed policy:

Definition 12 (Entropy Rate of a Markov Chain): *The* **Entropy Rate** *of a Markov Chain,* $\mathcal{S} = X_1, \ldots, X_n$ *is given as:*

$$H(\mathcal{S}) = \lim_{n \to \infty} \frac{1}{n} H(X_1, \ldots, X_n) \tag{10}$$

Which is equivalent to:

$$H(\mathcal{S}) = \lim_{n \to \infty} \frac{1}{n} H(X_n \mid X_{n-1}) \tag{11}$$

For a fixed policy, the MDP M collapses to a Markov Chain $\mathcal{S}$ with transition matrix dictated by π, $s_{init} \in \mathcal{S}$, and $\mathcal{T}$.

So how can we capture the entropy of the MDP, given that there are arbitrarily many policies that could be chosen?

TODO: We should really be doing this from a rate-distortion theory perspective, not Entropy Rates. TODO: Also focusing on this presently

References

[1] David Abel, D Ellis Hershkowitz, and Michael L. Littman. Near optimal behavior via approximate state abstraction. In *Proceedings of The 33rd International Conference on Machine Learning*, 2016.

[2] David Andre and Stuart J Russell. State abstraction for programmable reinforcement learning agents. In *AAAI/IAAI*, pages 119–125, 2002.

[3] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379, 2003.

[4] James C Bean, John R Birge, and Robert L Smith. Dynamic programming aggregation. *Operations Research*, 35(2):215–220, 2011.

[5] F Cao and Soumya Ray. Bayesian hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, pages 73–81, 2012.

[6] Peter Dayan and Geoffrey Hinton. Feudal Reinforcement Learning. *Advances in neural information processing systems*, pages 271–278, 1993.

[7] Thomas G Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[8] Kutluhan Erol, James Hendler, and Dana S Nau. Htn planning: Complexity and expressivity. In *AAAI*, volume 94, pages 1123–1128, 1994.

[9] Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. Hierarchical solution of markov decision processes using macro-actions. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 220–229. Morgan Kaufmann Publishers Inc., 1998.

[10] Bernhard Hengst. Discovering hierarchy in reinforcement learning with HEXQ. *Icml*, (1): 243–250, 2002.

[11] Nicholas K Jong and Peter Stone. State abstraction discovery from irrelevant state variables. In *IJCAI*, pages 752–757, 2005.

[12] Nicholas K. Jong and Peter Stone. Hierarchical model-based reinforcement learning. *Proceedings of the 25th international conference on Machine learning - ICML '08*, pages 432–439, 2008.

[13] George Konidaris. Constructing abstraction hierarchies using a skill-symbol loop. *arXiv preprint arXiv:1509.07582*, 2015.

[14] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Constructing symbolic representations for high-level planning. 2014.

[15] Tejas D Kulkarni, Karthik R Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. *ArXiv*, pages 1–13, 2016.

[16] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. In *ISAIM*, 2006.

[17] Neville Mehta, Soumya Ray, Prasad Tadepalli, and Thomas G. Dietterich. Automatic Discovery and Transfer of Task Hierarchies in Reinforcement Learning. *AI Magazine*, 32(1):35, 2011.

[18] Ronald Parr. Hierarchical control and learning for Markov decision processes. 1998.

[19] Ronald Parr and Stuart Russell. Reinforcement learning with hierarchies of machines. *Advances in neural information processing systems*, pages 1043–1049, 1998.

[20] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[21] Richard S Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1): 181–211, 1999.