Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Learning to Generalize and Reuse Skills Using Approximate Partial Policy Homomorphisms

Srividhya Rajendran
Department of Computer Science and Engineering
The University of Texas at Arlington
P.O. Box 19015, Arlington, Texas-76019
srividhya.rajendran@mavs.uta.edu

Manfred Huber
Department of Computer Science and Engineering
The University of Texas at Arlington
P.O. Box 19015, Arlington, Texas-76019
huber@cse.uta.edu

Abstract—**A reinforcement learning (RL) agent that performs successfully in a complex and dynamic environment has to continuously learn and adapt to perform new tasks. This necessitates for them to not only extract control and representation knowledge from the tasks learned, but also to reuse the extracted knowledge to learn new tasks. This paper presents a new method to extract this control and representational knowledge. Here we present a policy generalization approach that uses the novel concept of policy homomorphism to derive these abstractions. The paper further extends the policy homomorphism framework to an approximate policy. The extension allows policy generalization framework to efficiently address more realistic tasks and environments in non-deterministic domains. The approximate policy homomorphism derives an abstract policy for a set of similar tasks (a *task type*) from a set of basic policies learned for previously seen task instances. The resulting generalized policy is then applied in new contexts to address new instances of related tasks. The approach also allows to identify similar tasks based on the functional characteristics of the corresponding skills and provides a means of transferring the learned knowledge to new situations without the need for complete knowledge of the state space and the system dynamics in the new environment.**

We demonstrate the working of policy abstraction using approximate policy homomorphism and illustrate policy reuse to learn new tasks in novel situations using a set of grid world examples.

***Keywords*—Transfer Learning, Policy Homomorphism, Reinforcement Learning**

I. Introduction

A life-long learning agent that performs tasks in the real world needs to continuously learn new tasks. In addition, these agents need to adapt what they have learned to perform successfully in a complex and dynamic environment. While traditional RL agents have the capability to learn new tasks, they face significant challenges in terms of scaling to complex domains because the policies learned by these agents do frequently not transfer and therefore the knowledge gained from learning to perform a given task in a specific situation cannot be used to learn related new tasks in novel scenarios. The main reason for this is that the policies learned are usually directly tied to the perceptual representation of the environment and as a result the policies become useless as soon as the environment or the way the perceptual information is represented changes. Another issue with traditional RL agents is that they often use the raw perceptual information to learn policies. These percepts, however, produce a huge amount of data and processing and basing decisions upon this perceptual information can easily become a computationally intractable problem in complex environments. Furthermore, traditional RL agents generally make decisions as to what action they need to perform at each point in time. However, reasoning about actions at this scale and performing them in real time can become impossible as the complexity of the tasks these agents are learning increases.

Biological systems face similar issues [8][9], still they learn to perform increasingly complex tasks in the real world. They do this by successfully learning to only process the relevant information for the task at hand while ignoring the irrelevant aspects of the environment. Furthermore, they also learn throughout their development to build reusable abstractions that contain the knowledge gained from learning and performing tasks, and to use this knowledge to learn and perform new, increasingly complex tasks.

RL agents need to have similar capabilities that would not only allow them to autonomously identify similar tasks, but also to extract knowledge from the learned policies of these tasks, and to reuse them to learn new tasks. Further, they need methods that allow them to make decisions at a higher level of abstraction and techniques that can extract information relevant for task completion while ignoring the task-irrelevant aspects of the environment.

The work presented in this paper introduces a new approach to policy generalization and abstraction of control knowledge and their reuse, using the new framework of policy homomorphism. Our approach to life-long learning uses the policy homomorphism to form abstract skill and representational abstractions. To achieve this, the policy homomorphism framework uses a set of policies from similar tasks to extract a general policy for a corresponding *task type*, where a *task type* is defined by the maximal set of policy instances for which a general homomorphic policy can be found. The abstracted policy is here defined by two sets of functions where the first maps the individual states of each homomorphic policy to an abstract state of the general policy, thus allowing to identify situations in which the abstracted general policy is applicable. The second set of functions maps individual actions from each base policy to specific actions of the abstract policy, thereby identifying the action that the RL

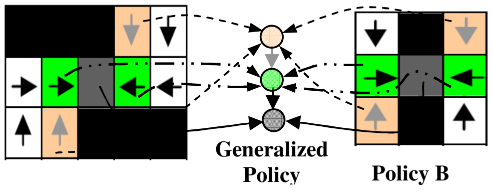

Figure 1. Policy Mappings From Base Policies A and B

agent needs to perform from each state of the general policy. The abstracted general policies are then reused to learn new tasks by adding each of them as one of the actions that the RL agent can choose to perform. Fig. 1 shows an example of a generalized policy derived from a set of two homomorphic policies as well as the corresponding state and action mappings.

In this paper we focus on the formalism of policy homomorphism and of policy reuse and extend our previous framework [1] of partial policy homomorphism to approximate partial policy homomorphism in order for it to be more efficient and applicable in non-deterministic environments and real world scenarios. The working of the policy homomorphism framework to abstract policies and their reuse to learn new tasks is demonstrated using examples in non-deterministic grid worlds.

II. Related Work

Over the years there has been a significant amount of research related to abstraction in the context of decision making and learning. Most of this research can be loosely divided into temporal abstraction and spatial abstraction. Sutton Precup and Singh [12] developed a temporal abstraction method that allows the agent to reason at a higher level of abstraction by representing the system as a Semi-Markov Decision Problem (SMDP). Here higher level actions, referred to as options, take multiple time steps to complete and are temporal extensions of lower level actions. Once a higher level action is chosen it follows the policy of the option until it terminates. Although this work allows for accelerated learning and scaling to more complex tasks, it does not in itself provide a mechanism to automatically find a useful set of options and to reduce the complexity of the state space. To address this and to provide a general learning system, hierarchical reinforcement learning techniques have been proposed [2][3][4][5][7]. These techniques aim at autonomously forming skill and representational abstractions. However, although these techniques can abstract useful sets of options that can be reused to allow faster learning times and to learn complex tasks, the extracted options are generally context specific and can thus still only be applied to tasks in the same environment.

Similar to temporal abstraction, the issue of spatial abstraction is very important to learn to successfully perform in a complex and dynamic environment. Among others, Stone and Jong [6] have recently tried to address this issue. They present an approach that autonomously aggregates states based on policy irrelevance of state attributes. Though their approach allows to successfully abstract a task-specific state representation, the need for carefully engineered initial state variables limits the applicability of their approach to a narrowly defined environment. A totally different approach to state space abstraction was presented by Ravindran and Barto[10][11]. They develop a formal framework based on MDP homomorphism that extracts a smaller state space from the original one by exploiting its symmetries and redundancies. They further extend their method to SMDPs by defining SMDP homomorphism which, in addition to state space compression provides a framework to identify situations in which policies can be transferred. However, their condition requires that the abstracted SMDP and the core MDP have the same properties and identical transition structure under all possible policies, limiting its applicability to perfectly symmetric or identical environments. An extension to this approach was proposed by Wolf and Barto[14]. In their work they identify object types based on MDP homomorphisms. Their approach considers two objects to be of the same type in the presence of additional state properties if the behaviors of all possible policies in their context are homomorphic and achieve the same result in terms of the chosen property. The limitation of this approach is that it identifies two objects as similar only if all functions of these objects are similar and further, due to the requirement of a local MDP homomorphism, both these objects have to be in the same environment. As a result of this these objects become dissimilar if the environment changes.

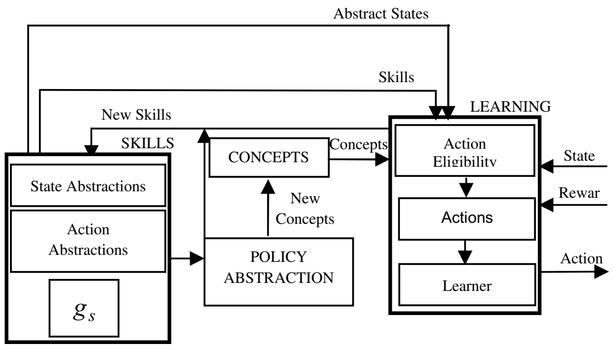

Figure 2. Learning Architecture of RL Agent

To address some of the limitation of these abstraction approaches and to allow for the efficient reuse of learned control knowledge, this paper presents a framework of approximate policy homomorphisms which allows for a more effective abstraction of general skills and representations in terms of *task types*. This abstraction framework is further integrated into a learning architecture which allows for the efficient acquisition, management, and reuse of the abstract skills and concepts.

III. Learning Architecture

The learning architecture is the core part of the RL agent. Fig. 2 shows our RL agent's learning architecture which the agent uses not only to learn skills and representational abstractions in the form of a general policy and corresponding representational concepts but also to reuse the abstracted general policy to learn new tasks. The learning architecture shown in Fig. 2 is made up of skills, concepts, learning, and policy abstraction components. The agent with this architecture starts out by learning basic policies for tasks by interpreting the state of the environment and interacting with it using its primitive actions. As a result of performing each action, the agent receives reinforcement from the environment. This is used by the agent to learn a policy for the task and the policy learned for the specific task instance is stored in the skills memory. As the number of basic policies learned increases, the agent uses the policy abstraction component of the learning architecture to extract a general policy from a set of policies for similar task. The policy abstraction component uses the policy homomorphism framework to extract reusable skills represented as general policies and corresponding representational concepts. These reusable skills and concepts are stored by the agent in its skills and concepts memory and can subsequently be used as higher level actions that the agent can choose to perform. This, in turn, allows the agent reuse the extracted control knowledge to learn related tasks in novel environments.

A. Skills and Concepts Memory

The skills and concepts components serve mainly as repositories for the learned skills and concepts. These learned skills and concepts are then used by the learning component to learn new, more complex tasks and concepts.

B. Learning Component

The learning component of the RL agent's learning framework learns policies to perform tasks successfully. At each time *t* the agent perceives the state s_t of the environment and chooses to perform action a_t from the set of admissible primitive actions. As a result the environment reaches state s_{t+1} and the agent receives a reward r_t. The agent uses this piece of information to learn a policy that maximizes the expected reward using Q learning. Each time a state–action pair is visited its value is updated using Q value update equation:

$$Q(s_t,a_t)' = Q(s_t,a_t) + \alpha(r_t + \gamma(\max_{a' \in A} Q(s_t,a')) - Q(s_t,a_t))$$

where $0 \le \alpha \le 1$ is the learning rate and $0 \le \gamma \le 1$ is a constant that represents the relative value of delayed versus immediate rewards. Policies that choose actions only from the set of primitive actions are called basic policies. Once the agent learns a basic policy, it stores this policy in the skill memory which is in turn used by the policy abstraction component to abstract a general policy for a given task type. These abstracted general policies are known as the higher level actions or options which allow the agent to reuse the knowledge gained from previous experiences of a given task type to learn policies for related tasks in novel situations. Each option is only admissible in specific situations, identified by the abstracted state representations of the general policy corresponding to the given option. All options may take multiple time steps to complete. A lower level action is a special option that takes one time step to complete. To learn a policy for a new task reusing the knowledge of the past experiences available, the agent at time step *t* perceives the state s_t of the environment and calculates the set of admissible options from the current state, and chooses an option o_t. Once an option is chosen the agent continues to choose lower level actions based on the general policy corresponding to this option o_t until the policy terminates or the agent reaches a state s_{t+k} from where o_t is no longer available. As a result, the environment gives the agent reward r. The agent uses this to learn a policy for this new task by iteratively updating the values of state-action pairs each time they are visited using the SMDP value function update equation:

$$Q_O^*(s_t,o_t) = E\left[r + \gamma^k \max_{o' \in O_{s_{t+k}}} Q_O^*(s_{t+k}, o' \mid \varepsilon(o_t, s_t, t))\right]$$

where k is the number of time steps between initiation of option o_t at state s_t and its termination at s_{t+k}, and $\varepsilon(o_t, s_t, t)$ is the event of option o_t initiated at time *t* in state s_t.

C. Policy Abstraction Component

The policy abstraction component of the learning framework extracts a general policy for a *task type* from a set of situation specific policies. To do this, the policy abstraction unit uses the policy homomorphism framework to autonomously identify policies of previously learned instances of similar tasks and to construct a general policy from them. The abstracted general policy is represented in terms of a function $\overline{g_s}$ that encodes the state-action mapping for the abstract policy and two sets of functions, $h(s), g_s(a)$, that map the individual states of the specific policy instances to unique states of the abstract policy and that map the actions from specific policy instances to abstract actions in the general policy, respectively. During reuse, the state mapping functions allow to identify the situations in which the corresponding general policy is applicable and the action mappings along with function $\overline{g_s}$ gives the agent information about what action the agent needs to perform in a specific situation while the agent is executing the option corresponding to the generalized policy.

IV. POLICY HOMOMORPHISM

To derive the formalism of a policy homomorphism we formulate our problem as a finite Markov Decision Process.

A Finite Markov Decision Process is a tuple $\langle S, A, T, R \rangle$ where $S = S_a \cup S_{na}$ is a finite set of states, with $S_a \subseteq S$ and $S_{na} \subseteq S$ representing the sets of absorbing and non absorbing states, respectively, and $S_a \cap S_{na} = \phi$. A is a finite set of actions, $T = S \times A \times S \to [0,1]$ is the transition probability function, and R is the expected reward function.

For the experiments presented in this paper, the agent uses Q-learning to learn a policy for the given tasks. The only policies considered in our framework are the goal based policies as goal of a task defines the objective of a task. Thus it important to capture and generalize goal based policies of similar *task types* to learn policies of related tasks in novel environments.

Definition 1: A Goal Based Policy is a tuple $\langle \pi, S_I, S_T, S_g \rangle$ where $\pi(s,a): S_\pi \times A \to [0,1]$ is a mapping from states in S_π to probabilities of selecting actions in A. $S_\pi = S_I \cup S_T$ is the state set on which policy π is defined and $S_I \subseteq S_{na}$,and $S_g \subseteq S_T$ are the sets of initiation states, termination states, and goal states for policy π, respectively, with $(S_a \cap S_\pi) \subseteq S_T$. □

Once the agent learns a set of basic policies they are generalized using the policy homomorphism framework.

Definition 2: A Policy Homomorphism $f : \pi \to \pi'$ is a surjection from base policy π to an abstract policy π', where f is defined by a tuple of surjections, ($h : S_\pi \to S'_\pi$, $g_s : A_\pi \to A'_\pi$ and function $\overline{g_S} = A_\pi \times S_\pi \to [0,1]$ over the state and action sets for policy π , $S_\pi \subseteq S$ and $A_\pi = \{a \in A | \exists s \in S_\pi : \pi(s,a) > 0\}$, such that the following properties hold:

1. For each state-action pair (s,a): $\pi'(h(s), g_s(a)) = \overline{g_s}(s,a)$
2. For each state pair (s_i, s_j) :

$$\sum_{b \in A'_\pi} \pi'(h(s_i), b) T'(h(s_i), b, h(s_j)) = \sum_{a \in A_\pi} \pi(s_i, a) T(s_i, a, s_j)$$

where T and T′ are the transition probabilities in the base and in the abstract policy, respectively. □

A complete policy homomorphism requires that every state in a given policy be mapped onto a particular state in the abstract policy. As a result, it does not allow abstraction of policies that might be partially homomorphic. Another issue with an absolute policy homomorphism is that it requires the probability of transitions into a state s' from state s by taking an action a in the base policy to be exactly equal to the probability of transitions into $h(s')$ from state $h(s)$ by taking an action $g_s(a)$ in the abstract policy. However, having two or more goal based policies with equal transition probabilities is very unlikely in the real world, significantly reducing the applicability and thus impact of absolute policy homomorphisms. In order to make the policy homomorphism framework applicable in real world tasks we extend it to an approximate partial policy homomorphism that addresses both the limitations that exist in a absolute policy homomorphism.

Definition 3: An Approximate Partial Policy Homomorphism $f : \pi_p \to \pi'_p$ is a surjection from a partial base policy π_p to an abstract policy π'_p where f is defined by a tuple of surjections ($h : S_{\pi_p} \to S'_{\pi_p}$, $g_s : A_{\pi_p} \to A'_{\pi_p}$)and functions $\overline{g_s} = A_{\pi_p} \times S_{\pi_p} \to [0,1]$ over the state and action sets of the partial policy π_p , $S_{\pi_p} \subseteq S_\pi$ and $A_{\pi_p} = \{a \in A | \exists s \in S_{\pi_p} : h(s) \notin S'_T \wedge \pi(s,a) > 0\}$, such that the following properties hold:

1. For all state action pairs $(s,a) : \pi'_p(h(s), g_s(a)) = \overline{g_s}(s,a)$
2. for each state pair (s_i, s_j) with $s_i \in S_{\pi_p}, h(s_i) \notin S'_T$:

$$\left| \sum_{b \in A'_{\pi_p}} \pi'(h(s_i), b) T'(h(s_i), b, h(s_j)) - \sum_{a \in A_{\pi_p}} \pi(s_i, a) T(s_i, a, s_j) \right| \le \varepsilon$$

where T and T′ are the transition probabilities for the base policy and the abstract partial policy, respectively. □

Both absolute and approximate partial policy homomorphisms can be applied to derive abstract policies. However, to ensure that the general policy captures the objective of the underlying policies, the approach presented here limits the application of policy homomorphisms to goal based policies. To do this we define a goal based policy homomorphism which is then applied in conjunction with the absolute or approximate partial homomorphism definition to extract an abstract policy from a set of basic policies.

Definition 4: A Goal Based Policy Homomorphism $f : \pi \to \pi'$ is a policy homomorphism that fulfills the following additional properties:

1. All states that are goal states in the base policy are present as goal states, S'_{π_g} in the abstract policy π' and $S'_{\pi_g} = \bigcup_{s \in S_{\pi_g}} h(s)$.
2. $S'_{\pi_g} \cap (\bigcup_{s \in (S_{\pi_p} - S_{\pi_g})} h(s)) = \phi$.
3. All non-goal states in policy π' are either terminal states or have a non-zero probability to lead to a goal state under policy π'.□

By utilizing this framework, the learning and abstraction approach extracts a maximal abstract policy that is homomorphic to all policy instances of a given *task type*. The framework uses a greedy algorithm and decision tree learners to extract the abstract policy and corresponding surjective mappings from the base set of policy instances to make the extraction procedure efficient. Table 1. shows the algorithm used to autonomously extract a general policy. To achieve this the policy abstraction algorithm starts at the goal state of each policy and builds a set of functions that map each state of the policy. The result of the algorithm is a general goal based policy π' and general mapping function $f_G(s,G) = (h_G, g_{s_G})$ such that $h'(G,s) = h_G(s)$, $g'_S(G,s,a) = g_{s_g}(a)$ where G is the goal state or states in the policy π.

These general mapping functions allow states in new policy instances to be mapped to states in the general policy without the need to map the entire partial policy and thus without the need for a complete state space model for the new environment. The algorithm uses a greedy method to build the functions. This is aimed at keeping the method tractable and continues until it reaches a point where the addition of more

TABLE I. POLICY ABSTRACTION ALGORITHM

```
1. Initialization t=0,
       $h' = \{\}, g'_s = \{\}, P = \{\pi_{p_1}, \pi_{p_2}, \pi_{p_3} \cdots, \pi_{p_n}\}$
       $S'^{(0)}_{\pi_p} = s'_g ; \forall i : S^{(0)}_{\pi_{p_i}} = s_{g_i}, h'(s_{g_i}, s_{g_i}) = s'_g$
2. Increment t
3. $\forall i : \pi_{p_i} \in P$ find $s_i^{(t)} \in (S_{\pi_{p_i}} - S^{(t-1)}_{\pi_{p_i}})$ such that:
    • $s_i^{(t)}$ is a predecessor of a state in $S^{(t-1)}_{\pi_{p_i}}$
      $\exists s \in S^{(t-1)}_{\pi_{p_i}}, a :$
      $(T(s_i^t, a, s) > 0.0) \wedge (\pi_i(s_i^t, a) > 0.0)$
      $\left| T(s_i^t, a, s) - T'(h(s_i^t), g_s(a), h(s)) \right| \leq \varepsilon$
    • There is an abstract state $s'^{(t)}$ such that
      o $h'(s_{g_i}, s_i^{(t)}) = h^{(t)}(s_i^{(t)}) = s'^{(t)}$
      o $g'_s(s_{g_i}, s_i^{(t)}, b) = g_{s_i}(b)$
      o $(h_i, g_{s_i})$ is an approximate partial policy
        homomorphism for the partial policy $\pi_{p_i}$ with
        state set $S^{(t-1)}_{\pi_{p_i}} \cup s_i^{(t)}$
    • $S^{(t)}_{\pi_{p_i}} = S^{(t-1)}_{\pi_{p_i}} \cup s_i^{(t)}, S'^{(t)}_{\pi_p} = S'^{(t-1)}_{\pi_p} \cup s'^{(t)}$.
4. If step 3 is successful, then: Goto 2. Else: Stop
```

states no longer increases the expected utility of the general policy.

V. EXPERIMENTS AND RESULTS

We demonstrate the working of the approach presented using examples in a non deterministic grid world domain. The experiments are divided into three phases. In the first phase we learn a set of "Reach Door" policies to reach specific doorways using a set of non deterministic grid worlds with rooms connected by doorways. The agent then uses these specific "Reach Door" policies to abstract a general policy using the approximate partial policy homomorphism framework in the second phase. In the final phase of the experiments we demonstrate how the learned generalized policy can be reused to learn similar tasks in novel environments and situations.

Learning Basic Policies: In this phase of the experiments the agent learns a set of basic "Reach Door" policies to reach specific doorways. Fig 3. shows the set of grid worlds used by the RL agent to learn to reach specific doorways. The action space of a RL agent in this phase of the experiments consists of FORWARD, TURN-LEFT, and TURN-RIGHT actions. The grid world domain is probabilistic, i.e. when the agent picks a lower level action like FORWARD then the agent successfully reaches the grid location in front of it 80% of the time and results in the agent's orientation being changed 20 % of the time. Similarly, if the agent chooses to execute the other lower level actions TURN-LEFT or TURN-RIGHT then the agent successfully turns in the specified direction 80 % of the time, 10% of the time it reaches the grid location in front of it and another 10 % of the time it results in the agent turning in the opposite direction to the specified action. All three actions have a cost of -0.25. The agent receives a reward of +100 when it reaches the goal doorway. The agent's state space for this set of experiments consists of the agent's x, y location, its orientation, and the 4 nearest doorways' x, y locations. To learn a policy to the specified doorway the agent starts at a random empty location within the grid world and explores the world. To do this the agent uses the Boltzmann "soft-max" distribution and reduces the temperature variable continuously until the exploration drops to 10%. This level of exploration is maintained to enable the agent to learn a globally optimal policy. During each step of learning the agent uses Q-learning to update the Q value for the state action pair visited. As a result of learning in both worlds with each of the doorways as the goal doorway at some point in time during the learning phase, the agent learns a total of 5 basic policies to reach a specified doorway within these grid worlds.

Extraction of Generalized Policy: In this phase of the experiments the agent used the learned policies to extract a general policy. To do this the agent first enhances the state representation within the learned policies by adding new state attributes. The new state attributes that are added are formed by applying various logical, relational, and arithmetic operators on an attribute or a set of attributes of the original state representation. This step is aimed at capturing information that is not explicit in the original state attributes but may be important to capture the non contextual information relevant for successfully completion of a task type. The policy

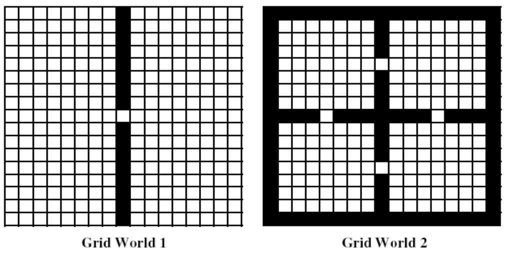

Figure 3. Grid Worlds used to learn basic "Reach Door" policies

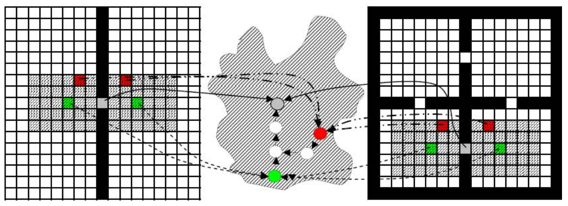

Figure 4. Generalized policy (middle) with partial mapping to Grid World 1 (left) and Grid World 2 (right). Shaded regions indicate the scope of the generalized partial policy

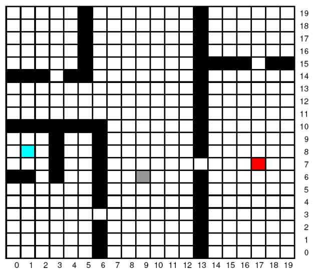

Figure 5. Grid world domain used for Cleaning Task

abstraction algorithm uses this state representation to extract a general mapping function without the need of a complex function approximator. Table. 1 shows the steps of the policy abstraction algorithm. Fig. 4 shows the area covered by the extracted general policy in both the grid worlds, along with a subset of the state and action mappings from the original policy to the general policy.

Reuse of Generalized Policy: To demonstrate the reuse of the abstracted general policy to successfully learn new tasks, the RL agent learns a cleaning task in a novel grid world environment. Fig. 5 shows the grid world environment used for learning the cleaning task. In this task the agent has to learn a policy to pick up the "Blue" (at row:8, col:1) and "Red"(at row:7, col:17) objects and drop them in the "Grey" colored trash can(at row:6, col:9). To learn this task the agent uses the extracted general "Reach Door" policy as a higher level action. Besides this action, the agent's action list consists of the primitive actions FORWARD, TURN-LEFT, TURN-RIGHT, PICK-DROP. The grid world domain is probabilistic and behaves in same ways to that of grid worlds in previous section. Action PICK-DROP always succeeds with a probability of 1. The agent incurs a cost of -0.25 for performing each single step action and receives a reward of +100 when it drops each object in the trash can. The agent starts at a random start location within the grid world and uses SMDP learning to learn an optimal policy for the cleaning task. Fig 6. shows the learning curve for the cleaning task with and without using the generalized policy. Each curve is an average of 30 runs and the performance is presented in terms of the average reward per step (where a step corresponds to the execution time of a primitive action) that the agent would receive under the policy learned at that point. The vertical confidence intervals indicate one standard deviation in each direction. The learning curves show that there is a significant improvement in the time it takes for the agent to learn a policy to reach the goal state when using the generalized policy. These experiments successfully demonstrate the applicability and usefulness of policy generalization using partial policy homomorphism.

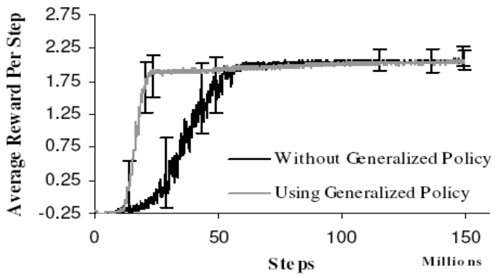

Figure 6. Learning Curve for Cleaning Task with and without using Generalized Policy

VI. CONCLUSION AND FUTURE WORK

This paper proposes a new approach that allows autonomous abstraction of important skills and concepts for task completion. We defined an absolute and an approximate partial policy homomophism framework which the agent uses to abstract a general policy from a set of policy instances. The abstracted policy is then used to address related tasks in novel situations and environments. The experiments in this paper demonstrate that reuse of the knowledge gained in the form of the general policy and the corresponding learned mapping functions reduces the time it takes to learn new tasks. In addition to improving learning time, this framework also promises to allow the agent to compress the state space by abstracting information important for task completion.

REFERENCES

[1] S. Rajendran and Huber M, "Generalizing and Categoriazation Skills in Reinfocement Learning Agent Using Partial Policy Homomorphisms", In the Proc. of FLAIRS, 2009, In Press.

[2] Asadi, M., and Huber M., "Effective Control Knowledge Transfer Through Learning Skill and Representation Hierarchies", In the Proceedings of IJCAI, 2007, pp. 2054-2059.

[3] Bakker B., and Schmidhuber J., "Hierarchical Reinforcement Learning, based on Subgoal Discovery and Subpolicy Specialization", In the Proc. of IAS, vol. 8, 2004, pp 438-445.

[4] Barto, A.G., and Mahadevan, S., "Recent Advances in Hierarchical Reinforcement Learning", DEDS, Vol.13(4), pp 341-379, 2003.

[5] Bulitko V., Sturtevant N., and Kazakevich M., "Speeding Up Learning in Real-time search via Automatic State Abstraction", In the Proc. of AAAI, pp 1349-1354. 2005.

[6] Jong, N. K., and Stone, P., "State Abstraction Discovery from Irrelevant State Variables", In the Proc. of IJCAI, pp 752-757, 2005.

[7] Konidaris G, and Barto A. "Building Portable Options: Skill Transfer in Reinforcement Learning", In the Proc. of IJCAI 2007, pp 895-900.

[8] Lakoff, G., "Women, Fire, and Dangerous Things", University of Chicago Press, 1987.

[9] Mandler, J. M., "How to build a baby: II. Conceptual, primitives ", Psychological Review, vol. 99(4), pp 587-604, 1992.

[10] Ravindran, B., and Barto, A. G., "Model minimization in hierarchical reinforcement learning". In the Proc. of SARA, LNCS (2371), pp 196-211, 2002.

[11] Ravindran, B., and Barto, A., "SMDP Homomorphisms: An Algebraic Approach to Abstraction in Semi-Markov Decision Processes", In the Proc of IJCAI, pp 1011-1016, 2003.

[12] Sutton, R., Precup, D., and Singh, S., "Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning", AI vol. 112, pp 181–211, 1999.

[13] Watkins, C. J. C. H., "Learning from delayed rewards", Ph.D. thesis, Psychology Dept., Cambridge University, 1989.

[14] Wolfe, A. P., and Barto, A.G., "Defining Object Types and Options Using MDP Homomorphisms", In Proc. of ICML Workshop on Structural Knowledge Transfer for ML, 2006.