Learning Behaviorally Grounded State Representations for Reinforcement Learning Agents

Vinay Papudesi **Manfred Huber**
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington TX 76019
{papudesi, huber}@cse.uta.edu

Abstract

The learning and reasoning capabilities of biological systems by far exceed those of robots and artificial agents. Part of this stems from their ability to efficiently learn behavioral skills and increasingly complex, symbolic representations that capture the important aspects of their environment. This paper presents an autonomous learning approach by which artificial reinforcement learning agents can acquire general symbolic state descriptions that are grounded in the agents' functional capabilities and take the form of behavioral *goals* and *affordances*. This action-specific *vocabulary* provides the agent with a means of representing the state of the world more concisely. Task-specific symbols are added to this vocabulary to construct an internal state space over which tasks can be represented as Markov Decision Processes. This learning framework is applied to feature-based mobile robot navigation and foraging.

1. Introduction

Biological organisms have capabilities to learn new skills and to address new tasks that far exceed those of existing computer and robot systems. We propose that part of this stems from their efficient use of initial innate structure and developmental mechanisms to construct an effective behavioral repertoire and corresponding representations over time. Behavior and representations evolve here concurrently in a number of stages (Piaget, 1952) starting from innate reflexes and relatively raw, perceptual representations of the world, towards increasingly complex skills and more symbolic representations that capture the characteristics of the world in terms of the available behavioral capabilities and the environment's corresponding *affordances* (Gibson, 1977).

For example, shortly after birth infants already exhibit an innate stepping reflex (Thelen and Smith, 1994). However, even if the muscular apparatus were sufficiently developed, this reflex would not be sufficient to permit locomotion. Throughout the first years the infant has thus to learn the skills required for competent locomotion. This development occurs in stages (Piaget, 1952) and finally results in a fully functioning locomotion system which operates largely subconsciously. Some psychologists suggests that one reason for such developmental stages might be that later stages build on the skills acquired earlier and utilize abstract representations of these skills as part of their conceptual knowledge (Bruner, 1973, Lakoff and Johnson, 1980). In the case of locomotion, for example, the infant first has to learn how to successfully coordinate the leg muscles to achieve desired leg motions. Once this is achieved, locomotion gaits can be learned by utilizing the coordination schemes acquired in the previous stage which competently handle individual muscle activations. This implies that gaits can be learned without conscious consideration of the state of the leg muscles and thus on a more abstract level. Finally, general navigation strategies can be derived at an even higher level of abstraction by utilizing the different locomotion gaits. These gaits internally handle all aspects of the legs, and navigation thus reduces to finding a traversable route. Traversability of the terrain, in turn, is an abstract concept that is specific to the available locomotion gaits and can be learned in the course of gait acquisition (Gibson, 1987). The use of learned skills and of higher-level state representations in terms of behavioral results and *affordances* (Gibson, 1977) (such as traversability) here dramatically reduces the reasoning and learning complexity and over time extends the range of tasks that can be addressed.

To provide similar capabilities for staged skill acquisition and the construction of corresponding representations in robot systems and artificial agents, the work presented here introduces a mechanism for the autonomous construction of functional representations of the agent's state space in terms of *goal* and *precondition* concepts which capture the behav-

ioral capabilities and corresponding *affordances* of the agent, resulting in a functionally grounded representation that facilitates subsequent learning of new task skills. The representations and skills are derived here to allow a reinforcement learning agent to improve its learning performance on subsequent tasks. This is achieved through the compact nature of the learned state representation which encodes only those aspects of the environmental state which have behavioral and reward implications for the specific learning agent in the context of its current behavioral capabilities. This framework for the construction of abstract state representations can form an effective foundation for a developmental learner, where new, incrementally more abstract representations are constructed as new skills are learned over the lifetime of the agent.

1.1 Efficient State Space Representations for Reinforcement Learning Agents

This paper presents a top-down approach where the functional capabilities of an agent are used to derive an abstract operational state space over which various tasks can be modeled as approximate Markov Decision Processes (MDPs). The abstract state space is represented by a vector of conditions, or state variables, that pertain to the action space of the agent and encode the different actions' goals as well as the corresponding *affordances* of the environment. By avoiding task-specific dependencies in forming these conditions, the agent is provided with a common representation which reflects its behavioral capabilities and over which a variety of tasks can be learned. Furthermore, this abstract state space need only be computed for the first task, allowing future tasks to be learned more efficiently.

Policy selection over this operational state space, however, will typically not be as optimal as over a task-specific one. For instance, if an action a in state s derives different reinforcements on different runs, this might be due to (i) state s being represented at a very coarse level and including low-level states that yield different results for action a, or (ii) action a deriving different reinforcements in state s according to some probability that is unknown to the agent.

Case (i) can be remedied by splitting state s based on the reinforcement signal. Case (ii) requires the agent to maintain the state transition probabilities for action a. If this information is not available, the agent could approximate it through its interaction with the environment. We present an approach to policy selection that modifies the factored, functionally-grounded internal state space to produce bounded optimal policies. This top-down approach accelerates policy acquisition by starting with a coarse representation and, as more task knowledge is gained, splitting sub-optimal state representations.

The basic ideas of the representation constructed here follows the tradition of interactivist approaches (Bickhard, 1996, Stojanov and Kulakov, 2003). However, rather than learning special-purpose representations for each task, it is aimed at constructing re-usable concepts that transfer knowledge to new tasks and can be subsequently refined to ensure bounded optimality. Here task behavior is learned bottom-up while representations for subsequent tasks are constructed top-down from the re-usable *goal* and *affordance* concepts as well as task-specific refinement.

A number of approaches to top-down construction of state space representations have been proposed. McCallum (McCallum, 1995) forms compact representations by selecting sensory dimensions that allow to correctly specify the value function. More closely related, Takahashi *et al* (Takahashi et al., 1996) present a technique where representation is learned in terms of action outcome predictions. The main difference is that these techniques are building task-specific from scratch while the approach presented here is aimed at constructing re-usable concepts that form initial representations for new tasks. (Takahashi and Asada, 2000) extends the latter work to address hierarchical learning where the higher level learns based on a representation consisting of the goal conditions of behavioral skills learned at the lower level. While this allows to learn a task decomposition automatically, it is still aimed at learning a single task (including a task-specific decomposition) rather than the re-use of previously learned skills and representation concepts while following a developmental trajectory.

Our approach is skeletally similar to that of Yoon *et al* (Yoon et al., 2002) in that both approaches learn rules that specify object classes and relations over which an action succeeds. However, the rules learned in Yoon *et al*'s work are task-specific, requiring them to be relearned from scratch for every new task. A task policy is then learned by iteratively selecting rules from the previously learned rule set based on their coverage. Resultantly, a "mostly optimal" policy is constructed, whose performance is improved using the ensemble method of bagging.

In contrast to these techniques, bottom-up approaches start with a large state space and attempt to combine states. They are particularly suitable when the underlying model is known, as they can reduce the state space by combining appropriate states while maintaining limits on the quality loss (Dean et al., 1997, Asadi and Huber, 2004). However, bottom-up approaches require the availability of a complete, low-level state space model which could be enormous in many practical problems, making state space reduction potentially intractable.

2. Deriving a Functionally-Grounded Internal State Space

The derivation of meaningful representations of an agent's internal state may be put as:

$$\vec{s} \leftarrow I(\vec{S}) \tag{1}$$

where $\vec{S}$ is the external world state, $\vec{s}$ is the internal state and $I(\cdot)$ represents a mapping relationship. Traditionally, these maps have been handwritten by domain experts (see, e.g., (Wang et al., 2003, Yanco and Stein, 1992)), thereby providing an explicit (flat) internal state space over which a set of tasks (as anticipated by the domain expert) can be learned efficiently. It can be argued, however, that the representations associated with such a state space are meaningful not to the agent, but to the domain expert. The dependence on domain experts for deriving the internal state space, as well as its inherent limitations — its static nature, inadaptability to environmental changes, lack of portability across heterogeneous agents — make this a difficult and time-consuming process with restrictive results.

Alternatively, the internal state space can be acquired autonomously. This requires that the agent capture some aspect of the world state while forming its internal state, thereby associating the meaning of the internal state to some generalization over world states. Cohen, Oates and Beal (Cohen et al., 2002) distinguish between a variety of meanings — formal, model-based, functional, affective, etc. The approach presented here constructs an internal state space consisting of symbols that are grounded in their functional meaning. Such a state representation may be viewed as a weaker version of Dretske's criteria for a state to be meaningful (Dretske, 1981, Dretske, 1988). In particular, it excludes the contentious requirement for the function itself to be learned (Dennett, 1998, Cohen and Litch, 1999).

In the work presented here, an internal state in such a functional state space is represented by a conjunction of symbolic state variables whose meanings are represented by classifiers over the world state that indicate particular functions that the agent provides. The set of functions used here constitutes the agent's action space or behavioral repertoire. This state representation provides the agent with the ability to predict its interaction with the environment.

Consider an agent with action space A. Conditions derived for function $a \in A$ are here task-independent and derived in terms of the outcome $o(a)$ of the action, which is provided to the agent when action a terminates. For instance, the outcome of a simple action can be either success or failure or, if information about the degree of success is available (e.g. through a measure of optimality), a better metric such as $0 \leq o(a) \leq 1$ can be provided.

Each action provides two types of conditions. To illustrate this, consider that in the external world state $\vec{S}_1$ action a transitions the agent to state $\vec{S}_1$ and achieves an outcome of o', as represented by

$$\vec{S}_1 \xrightarrow{o(a)=o'} \vec{S}_2. \tag{2}$$

We define an *indicator* of outcome o' for action a as a condition C_a^{ind} over the destination state $\vec{S}_2$:

$$C_a^{ind}(\vec{S}_2) \rightarrow o'. \tag{3}$$

Similarly, a *predictor* of outcome o' of the action is a condition C_a^{pre} defined over the source state $\vec{S}_1$ as:

$$C_a^{pre}(\vec{S}_1) \rightarrow o'. \tag{4}$$

Outcome predictors and indicators may be viewed as pre- and post-conditions that define the outcome of action a. As such, they correspond to the *goals* of the actions and the corresponding *affordances* of the environment, respectively. For example, C_a^{ind} evaluating to 1, or success, is an indicator that the current world state could be the result of a successful execution of action a. Similarly, C_a^{pre} evaluating to failure predicts that the current world state does not facilitate action a which would thus fail if executed.

We define the condition vector $\vec{C_a}$ of action a as a vector of the two types of conditions:

$$\vec{C_a} = \begin{pmatrix} C_a^{pre} \\ C_a^{ind} \end{pmatrix}. \tag{5}$$

Finally, the internal state space for an agent with an action space A of size n is defined as a vector of condition vectors for each action a_i, $i = 1 \cdots n$:

$$\vec{s} = \left(\vec{C_{a_1}}, \cdots \vec{C_{a_n}} \right). \tag{6}$$

This factored internal state space is a two-dimensional vector with a total of $2n$ ordered outcome values. The size of this state space is exponential in the size of the agent's action space. If, for example, outcomes are defined by the two discrete values of success and failure, the size of the state space has an upper bound of $|\vec{s}| = 2^{2n}$. However, the practical size of the internal state space is limited by the size of the world state space as certain conjunctions of state variables can never be encountered. Moreover, agents are usually provided with an action space that is applicable to a preconceived set of tasks in some domain, enforcing an inherent causal relationship between actions and tasks and thus resulting in far more compact state spaces.

As an example, a mobile robot in the domain of feature-based foraging is provided with a camera capable of distinguishing certain features such as color, shape, and size. We provide the agent with four actions (`find`, `goto`, `pick`, and `drop`) which are described in Table 1 along with their outcome predictors and indicators. The maximum size of this internal state space is thus 2^8 states. However, the

number of unique states encountered by the agent is far smaller (as illustrated later), and can be attributed to the similarity between several outcome predictors and indicators. For example, the outcome indicator for `find` is identical to the outcome predictor for `goto`. Similarly, the outcome indicator for `goto` is inclusive of the outcome predictor for `pick`.

2.1 Computation of Outcome Conditions

In this work, external world states are static sensory snapshots of the agent's environment taken in a discrete event dynamic system (DEDS) architecture. Changes that occur during the execution of an action are not represented and consequently a large number of properties of the external world state can be represented by a set of decision rules. In the domain of mobile robots, a typical sensory snapshot includes odometry, sonar data, gripper data and the features of an arbitrarily large set of visible objects. A large number of these variables, such as distances, are continuous, and the resulting raw state space is infinite. While the discretization of these variables, together with a limit on the number of simultaneously visible objects, can produce a finite state space, learning policies over such a space could still be computationally expensive. Furthermore, bounding the number of visible objects introduces the problem of object selection, or focus of attention, which is task-dependent and typically involves additional learning.

In this work, agents employ a collection of hierarchical rule-based classifiers learned using a genetic algorithm to create an abstract state space. Genetic algorithms apply well to problems with large state spaces due to the manner by which they select and discard rules. By directing their search away from unsuitable rules, they may be expected to learn optimal rules by natural evolution. By including some portion of random rules in every generation, the population of rules can be made more divergent and rules that represent local optima can be overcome.

2.2 Non-Deterministic Outcomes

So far, it has been assumed that outcome conditions can accurately differentiate between external world states corresponding to successful and unsuccessful outcomes. However, this is typically not the case. Consider the action `find` which causes the agent to rotate in place until an object enters its camera's field of view. If an object was visible in its initial configuration, then the `find` action would definitely succeed. However, the outcome of the action is non-deterministic given an initial configuration where no object was visible. This form of non-determinism is introduced due to limitations of the sensory capabilities of the agent. In addition to determining successful and unsuccessful outcomes of an action, we allow classifiers to categorize world states as being non-deterministic in the outcome of the action. Figure 1 illustrates the convergence of such a triple-valued outcome predictor for the action `find` to the one in Table 1. Initially the predictor over-specializes to the training data, leading to the misclassification of several states. As more training data is made available, a more general classifier is constructed which also indicates an undetermined outcome.

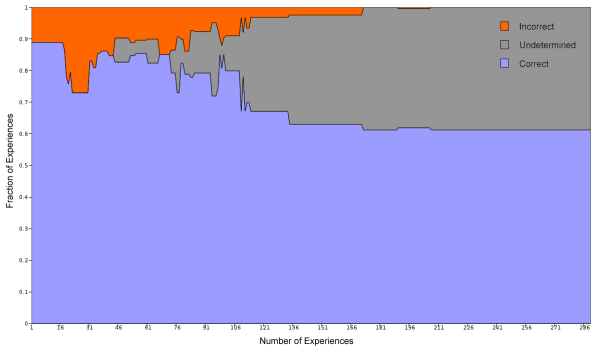

Figure 1: Convergence plot for the outcome predictor of the `find` action.

3. Policy Optimization Using Task-Specific Symbols

We have assumed that the agent is provided with a set of functions that constitute its behavioral repertoire or action space. Further, we have assumed that for each action the agent can derive its outcome, such as success or failure, independent of the task it is learning. The agent learns the outcome predictors and indicators for each action and constructs the factored internal state space $\vec{s}$ from Equation (6) wherein each outcome condition is a state variable.

Having acquired such an implicit, functionally-grounded internal state space, an agent can now learn optimal policies corresponding to a variety of tasks over this state space. The problem is treated as an MDP $M(S, A, F, R)$, where S is the set of internal states, A is the agent's action space, $F : S \times A \times S \rightarrow [0, 1]$ is a transition probability function defined as:

$$F_{pq}(a) = \Pr(X_{t+1} = q | X_t = p, U_t = a), \quad (7)$$

where the variables X_t and U_t represent the agent's state and action at time t respectively, and $R : S \times A \rightarrow \Re$ is a task-specific reward function.

Given that the agent is in some internal state, it must determine which action to take in order to perform the task optimally. This is represented by the task-specific policy $\pi : S \rightarrow A$. In this work, we use reinforcement learning (Barto et al., 1993, Kaelbling et al., 1996) to compute the optimal policy. Reinforcement learning algorithms allow an

Table 1: Action details in a feature-based mobile robot foraging domain

Action a	Description	Outcome predictor $C_a^{pre} = success$	Outcome indicator $C_a^{ind} = success$
`find`	Rotates 360^o or until an object is visible.	$visible(x)$	$visible(x)$
`goto`	Navigates to within a picking distance of the closest visible object.	$visible(x)$	$visible(x)$
`pick`	Attempts to pick the closest visible object.	$visible(x) \wedge size(x) \leq 250 \wedge distance(x) \leq 500 \vee holding(x)$	$holding(x)$
`drop`	Drops the currently held object.	$true$	$true$

agent to learn from its interaction with the environment and numerical feedback, called reinforcements, which serve as a measure of progress in completing the task. The (deterministic) policy is computed as greedy with respect to the estimated utility function:

$$\pi\left(\vec{s}\right) = \arg\max_{a \in A} Q\left(\vec{s}, a\right), \tag{8}$$

where A is the action space and the utility function $Q\left(\vec{s}, a\right)$ represents the expected discounted future return obtained by taking action a in state $\vec{s}$:

$$Q\left(\vec{s}, a\right) = R\left(\vec{s}, a\right) + \gamma \sum_{s' \vec{\in} S} F_{\vec{s}\vec{s'}}\left(a\right) Q\left(\vec{s'}, \pi(\vec{s'})\right), \tag{9}$$

where γ is the discount rate, $0 \leq \gamma < 1$.

3.1 Policy Representation

Having assumed that the agent is initially aware only of its action space, it follows that the task model M needs to be acquired. The agent interacts with its environment by executing one of its actions, and each such interaction is labeled as an experience. Let E denote the set of experiences. Each experience $e \in E$ corresponds to the execution of action $act(e) \in A$. The corresponding source and destination states are represented by $src(e), dest(e) \in S$ respectively, and the observed reinforcement is represented by $r(e) \in \Re$. Because each additional experience could change the outcome conditions, the internal source and destination state representations must be updated following each interaction.

Let E_a denote a subset of E that contains only those experiences related to action $a \in A$:

$$E_a = \{e \in E | act(e) = a\}. \tag{10}$$

Outcome conditions are learned over these action-specific sets of experiences. Further, let us partition each set E_a based on the source state $p = src(e)$, and denote each such partition by $E_{a,p}$. The state transition probability $F_{pq}\left(a\right)$ is computed from this partition as:

$$F_{pq}\left(a\right) = \frac{|\{e \in E_{a,p} | dest(e) = q\}|}{|E_{a,p}|}. \tag{11}$$

Similarly, the reward function $R\left(p, a\right)$ is computed as the average observed reinforcement in $E_{a,p}$:

$$R\left(p, a\right) = \frac{1}{|E_{a,p}|} \sum_{e \in E_{a,p}} r\left(e\right). \tag{12}$$

While the model constructed from the available experiences is initially approximate (and at worst incorrect), it can be expected to converge to the true model with the accumulation of experience.

3.2 Optimization Using Task-Specific Conditions

Not all tasks can be represented optimally over an internal state space constructed only from action goals and concepts representing *affordances*. Here, we show how task-specific conditions can be appended, thereby modifying the internal state space of the agent and allowing a wider variety of tasks to be learned within a bound of optimal.

To illustrate this, consider that in the domain of feature-based mobile robot foraging, the robot receives positive reinforcement for picking up an object. The robot's gripper can only pick up objects that are within its reach and below a certain size, and the outcome conditions for the `pick` action are learned accordingly (We will refer to objects that can and can not be picked up as small and large, respectively. Note that the agent itself is initially unable to make any such distinction.) The `goto` action, on the other hand, will succeed irrespective of the size of the object. The internal state in which an object is visible but can not be picked up (either due to the robot being too far away or the object being too large) is represented by the conjunction of the following four outcome conditions[1]:

$$C^{pre}_{\texttt{goto}} = success, C^{ind}_{\texttt{goto}} = failure,$$
$$C^{pre}_{\texttt{pick}} = failure, C^{ind}_{\texttt{pick}} = failure$$

In this state, there is no explicit distinction between small and large objects, and the optimal action is `goto`, which leads to one of two states depending on whether the target object is large or small. If the object that is reached is small,

[1]State variables related to the outcome of the other actions have not been specified as they do not affect this analysis.

the outcome predictor $C^{pre}_{\texttt{pick}}$ for the resulting state computes to success; if the object is large, it computes to failure. The other outcome conditions are identical in these two states and are:

$$C^{pre}_{\texttt{goto}} = success, C^{ind}_{\texttt{goto}} = success,$$
$$C^{ind}_{\texttt{pick}} = failure$$

As a result, the robot always attempts to reach objects, irrespective of whether or not they can be picked up. On the other hand, if the distinction between small and large objects were available in the initial state, the sub-optimal action `goto` would not have to be taken to reach large objects.

The sub-optimality in the policy can be attributed to a limitation in the underlying state space, and can be identified by its manifestation in the reward and utility functions. Our approach to removing this form of sub-optimality is through the introduction of task-specific symbols that effectively create bipartitions of existing states. Given a set of experiences $E_{a,s}$ specific to the source state s and action a, we construct a set of bipartitions $\{B_1, B_2\}$ where, for any two experiences $e_1 \in B_1$ and $e_2 \in B_2$, the following inequality holds true:

$$\left| \max_{b_1 \in A} Q\left(dest(e_1), b_1\right) - \max_{b_2 \in A} Q\left(dest(e_2), b_2\right) \right| > \epsilon \tag{13}$$

We refer to these partitions as value-interval bipartitions. Similarly, we define a set of reinforcement-interval bipartitions $\{B_1, B_2\}$ such that for any two experiences $e_1 \in B_1$ and $e_2 \in B_2$, the following inequality holds true:

$$|r(e_1) - r(e_2)| > \epsilon \tag{14}$$

The value of the bounding threshold ϵ determines both the accuracy of a bipartition as well as the number of candidate bipartitions. In our work, its value is assigned empirically, based on such factors as the task reinforcements and the discount rate.

This approach is similar to that taken by Dean *et al* (Dean et al., 1997) in the construction of bounded parameter MDPs as approximate reduced models. They define a partition as having the property of ϵ-approximate stochastic bisimulation homogeneity if similar upper bounds are enforced on the variations of the reinforcement and transition probability functions for any block of the partition.

For the purpose of measuring the suitability of a candidate bipartition, we introduce two independent metrics that the agent attempts to optimize. First, the agent prefers bipartitions for which the sum of variances is minimized. This ensures that the two data sets created by the partition individually have lower variances. The second metric is its coverage and applies to the classifier that is built over this partition. The coverage of a classifier is defined as the ratio of the number of experiences classified correctly to the total number of experiences. It is possible that the bipartition that has the minimum sum of variances is non-deterministic, and resultantly can not be classified correctly. If a classifier's coverage is below a certain threshold, the corresponding partition will not be used to split the state and as a result the agent avoids splitting states for actions whose function in that state can not be clearly determined.

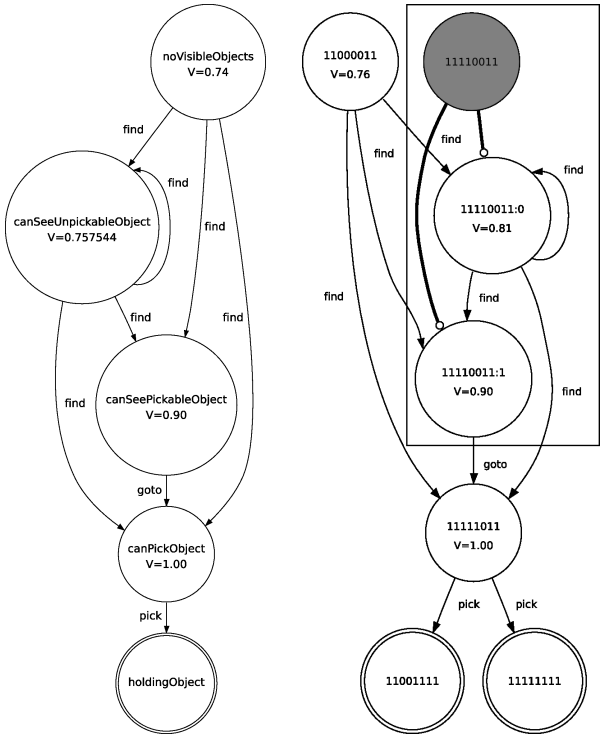

Figure 2: Hand-coded state space and policy (a), and acquired state space and policy (b) for the foraging task.

4. The Foraging Task

The task of foraging involves searching for objects, moving to them and picking them up. A reinforcement of 1 is received by the agent when an object is picked up. For simplicity we assume that (i) in its initial configuration, the robot's gripper is empty, and (ii) the structure of the environment and the placement of the objects ensure that the `find` action always succeeds. Essentially, these restrictions reduce the size of the state space so as to better illustrate the formulation of task-specific conditions.

A hand-designed expert state space for this task consisting of five states is shown in Figure 2 (a) along with the policy learned over it. For the given task and action set, this is the smallest possible state space and yields the most optimal policy.

Figure 2 (b) depicts the functionally acquired state space as well as the task-specific policy learned. States are labeled with bit strings of length eight. Each of these bits represents a state variable in the

factored state representation. The first and second bits correspond to the outcome predictor and indicator, respectively, for the action `find`. Similarly, the next three pairs of bits correspond to the outcome conditions for the actions `goto`, `pick` and `drop`.

There are two accepting states, shown in Figure 2 (b) as double-rimmed nodes. State `11000111` indicates that an object is being held while no other object is visible. State `11111111` indicates that an object is being held and another object is visible.

As described previously, the sub-optimality in this task arises from the presence of large objects that can not be picked up. This sub-optimality is removed by partitioning the source state `11110011` (represented by a shaded node in Figure 2 (b)) based on the values of the destination states that are reached through action `goto`. Following the formulation of this task-specific symbol, the source state is split into two states. State `11110011:0` indicates that a "large" object can be reached using the `goto` action, whereas state `11110011:1` indicates that a "small" object can be reached using the `goto` action. The best action in the former state is `find`, whereas the best action in the latter state is `goto`. As a result, the robot only executes `goto` if it knows that the corresponding object can subsequently be picked up using `pick`.

To evaluate the performance of the autonomously acquired representation, it is compared with a hand-designed policy. In addition, it is compared with policies learned over state spaces that are constructed by manually limiting the state attributes to the ones required and by discretizing the continuous attributes. In particular, the state spaces represent details (size, color and shape) of the object being held (if any) and details (size, color, shape and distance) of the closest visible object (if any) with the size and distance attributes discretized using four different levels of discretization (1, 10, 50 and 100 mm).

To evaluate the performance of the learned, action-specific state representation, the size of the visited parts of the state spaces as well as the obtained discounted reward are compared. Figure 3 shows how the size of the state space increases as the agent interacts with the environment. Figure 4 shows how the quality of the policies obtained over these representations improves with experience. As expected, both the hand-designed and acquired state spaces are much smaller than those obtained by the manually discretized state representations and, consequently, the task is learned faster. The sudden improvement of the policy for the acquired state space after 75 experiences is due to the discovery of the required split of state `11110011` as previously described.

The comparison also shows that the automatically learned state representation based on action goals and *affordance* concepts performs comparable to the optimal, expert-designed representation, while easily outperforming all discretizations used. This illustrates the power of state representations that are grounded in the behavioral capabilities of the agent.

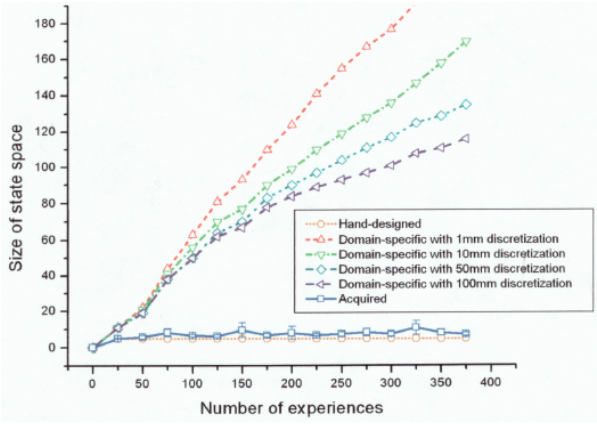

Figure 3: State space size for different representations.

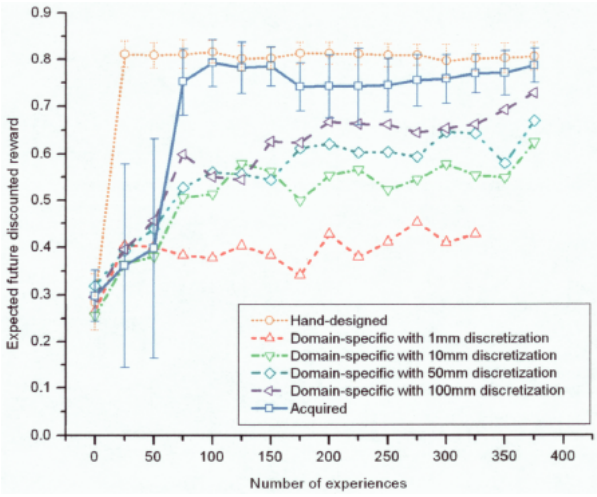

Figure 4: Policy performance for different representations.

5. Conclusions

In this paper we have described an approach that constructs a factored internal state space whose state variables correspond to the set of actions or functions that the agent provides. Such an internal state space naturally lends itself to describing state transitions triggered by the actions over which it is constructed.

To define function-oriented state variables, the agent uses information about the outcomes (or the outcome optimality) of an executed action. Using this, each action provides two state variables representing the *goal* of the action and the precondition of its success (reflecting the *affordances* of the environment in the context of the action). This collection of state variables may be viewed as a purely functional, symbolic vocabulary that can be used to reason about a variety of tasks.

During learning of a new task, this representation is extended through the addition of task-specific state variables that partition states in which there is an appreciable difference in value or immediate reinforcement. By enforcing such bounds on the reinforcement and value functions, the resultant policy is bounded in its deviation from the optimal policy. The set of such task-specific state variables may be viewed as creating additional decision points within the policy as they each pertain to a particular state at which there is an appreciable difference in the result of executing a particular action.

Since the presented approach constructs state representations that are specific to the action set available at the time of a new learning task, it provides an efficient means of transferring representational knowledge in a developmental robotics system by allowing it to generate increasingly abstract representations as new behavioral skills are learned.

We have demonstrated the application of this approach with a simple task involving foraging in the domain of mobile robots, illustrating its ability to form very compact state representations while maintaining near optimal learning performance.

Acknowledgments

This work was supported in part by DARPA FA8750-05-2-0283 and NSF EIA-0203499. The U. S. Government may reproduce and distribute reprints for Governmental purposes. The authors' views and conclusions should not be interpreted as representing official policies or endorsements, expressed or implied, of DARPA, NSF, or the Government.

References

Asadi, M. and Huber, M. (2004). State space reduction for hierarchical reinforcement learning. In *Proc. FLAIRS Conf.*

Barto, A., Bradke, S., and Singh, S. (1993). Learning to act using real-time dynamic programming. Technical report, U. Massachusetts.

Bickhard, M. (1996). The emergence of representation in autonomous embodied agents. In *AAAI Fall Symp. on Embodied Cognition and Action.*

Bruner, J. (1973). Organization of early skilled action. *Child Development*, 44:1–11.

Cohen, P. and Litch, M. (1999). What are contentful mental states? dretske's theory of mental content viewed in the light of robot learning and planning algorithms. In *IJCAI'99.*

Cohen, P., Oates, J., and Beal, C. (2002). Robots that learn meanings. In *Joint Conference of Autonomous Agents and Multiagent Systems.*

Dean, T., Givan, R., and Leach, S. (1997). Model reduction techniques for computing approximately optimal solutions for markov decision processes. In *UAI'97.*

Dennett, D. (1998). *Brainchildren—Essays on designing minds*, chapter Do it yourself understanding. MIT Press.

Dretske, F. (1981). *Knowledge and the flow of information.* MIT Press.

Dretske, F. (1988). *Explaining behavior: reasons in a world of causes.* MIT Press.

Gibson, E. J. (1987). Detection of the traversability of surfaces by crawling and walking infants. *JEXP: Human Perception and Performance*, 13:533–544.

Gibson, J. (1977). The theory of affordances. In *Perceiving, Acting and Knowing.* Erlbaum.

Kaelbling, L., Littman, M., and Moore, A. (1996). Reinforcement learning: a survey. *Artificial Intelligence Research*, 4:237–285.

Lakoff, G. and Johnson, M. (1980). *Metaphors We Live By.* University of Chicago Press.

McCallum, A. (1995). *Reinforcement Learning with Selective Perception and Hidden State.* PhD thesis, University of Rochester, Rochester, NY.

Piaget, J. (1952). *The Origins of Intelligence in Childhood.* International Universities Press.

Stojanov, G. and Kulakov, A. (2003). Interactivist approach to representation in epigenetic agents. In *Int. Workshop on Epigenetic Robotics.*

Takahashi, Y. and Asada, M. (2000). Vision-guided behavior acquisition of a mobile robot by multilayered reinforcement learning. In *IROS'00.*

Takahashi, Y., Asada, M., and Hosoda, K. (1996). Reasonable performance in less learning time by real robot based on incremental state space segmentation. In *IROS'96*, pp. 1518–1524.

Thelen, E. and Smith, L. B. (1994). *A Dynamic Systems Approach to the Development of Cognition and Action.* MIT Press.

Wang, Y., Huber, M., Papudesi, V., and Cook, D. (2003). User-guided reinforcement learning of robot assistive tasks for an intelligent environment. In *IROS'03.*

Yanco, H. and Stein, L. (1992). An adaptive communication protocol for cooperating mobile robots. In *SAB'92*, pp. 478–485.

Yoon, S., Fern, A., and Givan, B. (2002). Inductive policy selection for first-order mdps. In *UAI'02.*