Symbol Generation and Grounding for Reinforcement Learning Agents Using Affordances and Dictionary Compression

Marcus Oladell and Manfred Huber
Department of Computer Science and Engineering
The University of Texas at Arlington
Arlington, TX 76019-0015, USA
marcus.oladell@mavs.uta.edu, huber@cse.uta.edu

Abstract

One of the challenges for artificial agents is managing the complexity of their environment as they learn tasks especially if they are grounded in the physical world. A scalable solution to address the state explosion problem is thus a prerequisite of physically grounded, agent-based systems. This paper presents a framework for developing grounded, symbolic representations aimed at scaling subsequent learning as well as forming a basis for symbolic reasoning. These symbols partition the environment so the agent need only consider an abstract view of the original space when learning new tasks and allows it to apply acquired symbols to novel situations.

Introduction

The gap between the capabilities of biological and artificial learning systems is vast. Two aspects that contribute to this are knowledge transfer and complexity management. Biological systems form abstract, task appropriate representations of the environment. One theory for such concepts is affordance theory (Gibson and Spelke 1983) which postulates that infants and other biological systems learn action specific concepts, which simplify reasoning and acting. This idea has received significant attention from affordance-based robotics (Rome, Hertzberg, and Dorffner 2006).

Our approach to representation abstraction is to build symbolic concepts that are semantically grounded in the environment aimed at generating an abstract feature set on which the agent can learn. The power of symbolic concepts stems from their ability to yield a compact representation of important aspects of the environment. Knowledge grounded in such concepts is often more general and applicable in new situations, allowing transfer of strategies to novel domains. Symbolic concepts grounded in the agent's capabilities also potentially form an efficient, expressive basis for communication. While this paper focuses on representational compression, future work will also explore communication.

Knowledge transfer has been a topic of much research (Taylor and Stone 2005; Marthi et al. 2005; Marx et al. 2005). Recent work has looked at skill hierarchies which facilitate increasingly complex behavior (Asadi and Huber 2007). This approach is centered on the learning framework with only ancillary concern for representation.

Work by (Taylor, Kuhlmann, and Stone 2008) focuses on physical knowledge in games. Feature sets are mapped from one game to others. This transfer is useful but limited.

In contrast to previous work, we focus on a representational framework for grounded, functional symbols which allow compression of raw features. The foundation of this approach are policies, affordances, and goals. We use the option framework (Sutton, Precup, and Singh 1999) to learn policies, from which affordances and goals are constructed.

Since a symbol represents a single, abstract feature, the dictionary is the set of acquired symbols and an instance of an abstract feature set. In this context, managing representational complexity is an issue of managing dictionary size.

Affordances and Goals

Affordances and goals, along with corresponding policies, are at the heart of symbolic representation. They facilitate the creation of grounded, abstract features.

Functionality and Structure

Affordances and goals are represented by classifiers or functions. They map a state region to a single, abstract feature or to a probability distribution, forming an abstract feature set. Decision trees are used here because they are transparent, allowing easy analysis. However, the algorithm is independent of the affordance/goal implementation. Future research may require a different implementation but at this point, the transparency of decision trees make them the best choice.

Conceptual Roles

Affordances and goals are here structurally similar but functionally distinct. A goal is an *indicator* of success, allowing the agent to determine if a policy's objective is fulfilled. In contrast, affordances are *predictors* of success, indicating the probability that a policy will succeed from a given state.

Each symbol, or abstract feature, is grounded in the affordance or goal used to generate it. This provides a mapping between symbolic representation and semantic meaning.

Methodology

Figure 1 represents our approach to symbol formation.

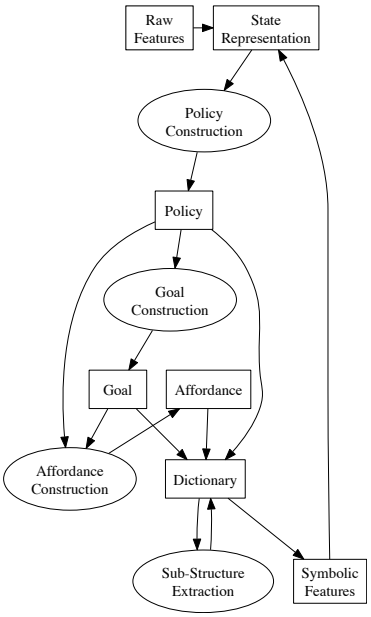

Figure 1: Symbol Generation. Processes are represented as ovals, while data are represented as boxes

Policies are constructed on the current state representation which initially consists of raw features. Once built, it is used as input to the goal and affordance construction processes. A policy goal is constructed first before a corresponding affordance. Each affordance and goal results in the creation of a new symbol which is appended to the agent's dictionary.

A separate process examines the dictionary for common sub-structures. Extracting such structures and creating corresponding symbols results in dictionary compression. Also, the new symbols are not bound to a policy but represent environment features which, together with affordance and goal symbols, generate new, abstract state representations.

Underlying Policy and System Model

The initial set of policies are constructed on the raw feature space using Q-Learning on an MDP. Different tasks are here learned by changing the reward function, *R*. Once initial policies are learned, these are leveraged as potential decisions (options) using the formulation of SMDPs (Sutton, Precup, and Singh 1999) where the state is represented by a set of *N* features, each representing a variable that can take on *n* values. Together they form the state space, *S*. This flexible definition can accommodate raw and abstract features.

$$S = \bigtimes_i F_i \quad , \quad F_i \in \{f_{1i}, f_{2i}, ..., f_{ni}\}$$

Policy Construction

Once a new task is presented, a goal-based option is learned on the current representation. Here, the option, O_π, is a policy, π, which defines a mapping from states to actions, the initialization states, S_I, where the policy can be initiated, the termination states, S_T, which signal the end of policy execution and the goal states, S_g, which is a subset of S_T.

$$O_\pi = \langle \pi, S_I, S_T, S_g \rangle \quad , \quad S_g \subset S_T$$

Goal and Affordance Construction

For each policy, a goal and affordance are built as decision trees. A goal of policy π *indicates* the likelihood that a state meets the goal criteria of the policy:

$$g_\pi : s_i \rightarrow \sum_{s_g \in S_g} Pr(s_t = s_g | s_i, \pi)$$

where $Pr(s'|s, \pi)$ indicates the probability that policy π initiated in state s terminates in state s'.

Discovering goal states is non-trivial (Digney 1996; McGovern and Barto 2001). For this paper we assume S_g is known. Given this, goal classifier training data is generated by executing a policy, π, and labeling the resulting terminal states, s_t, according to their membership in S_g. Once a goal is constructed, an affordance is built which provides a mapping from the feature set to the probability of policy success.

$$A_\pi : s_i \rightarrow \mathbb{R} \quad , \quad s_i \rightarrow \sum_{s_g \in S_g} Pr(s_g | s_i, \pi)$$

Once constructed, affordances and goals act as symbols, representing abstract state features. The symbolic feature set allows the agent to reason without considering the raw features and forms the foundation of the agent's vocabulary.

The result of an affordance or goal can be discrete or continuous. Our current implementation constructs affordances and goals as discrete classifiers. However, symbol learning can also use continuous features (Papudesi and Huber 2006), and future work will include continuous symbols.

Structure Extraction and Dictionary Compression

There is a correlation between dictionary size and complexity of the feature set. Our approach is to find common sub-structures that occur multiple times within the dictionary.

Representing symbols as decision trees imposes restrictions on potential sub-structures. First, any sub-structure node must maintain its edges since we can not ignore potential outcomes. Second, only non-leaf nodes are included to facilitate use as a parameter. Categories in leaf nodes become edge labels, leaving the construct intact and allowing leaf nodes to be appended for use as an independent symbol.

There are many sub-structure identification approaches (Holder, Cook, and Djoko 1994; Kuramochi and Karypis 2001). Given the small size of affordances and goals in this paper, we use a brute force method. However, future work will integrate a more efficient algorithm. The proposed framework is indifferent to the method, provided the resulting structure meets the requirements. Once a sub-structure has been found, its impact on dictionary size is evaluated.

Using decision trees, a symbol, Φ, is formulated in graph notation. The size of a vertex, v_Φ, and an edge, e_Φ, are fixed:

$$G_\Phi = (V_\Phi, E_\Phi) \ , \ N_{V_\Phi} = size(v_\Phi) \ , \ N_{E_\Phi} = size(e_\Phi)$$

Using this, we can calculate the size of each symbol, Φ and, by extension, the size of a dictionary, *D*.

$$size(\Phi) = N_{V_\Phi} * |V_\Phi| + N_{E_\Phi} * |E_\Phi|$$

$$size(D) = \sum_{\Phi \in D} size(\Phi)$$

With the dictionary size we can calculate the impact of a sub-structure. *D'* is the dictionary after extraction of sub-structure Φ_{ss}. The number of occurrences of Φ_{ss} is $N_{\Phi_{ss}}$.

$$size(D') = size(D) - size(\Phi_{ss}) * (N_{\Phi_{ss}} - 1)$$

The greater the difference between the size of *D* and *D'*, the better the compression impact of Φ_{ss}.

Gripper Robot Example This example shows a simple instance of sub-structure extraction. Our agent is a robot arm with a camera. The environment contains dice which vary in size and color and always have one side facing up.

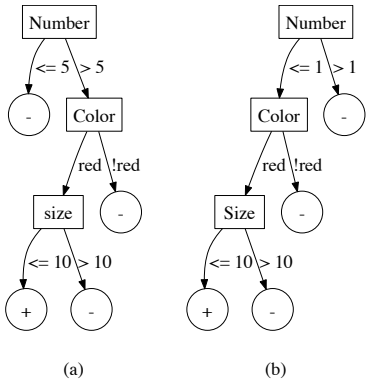

Figure 2: Affordances for Dispose Policies. (a) *6Disposable (Number, Color, Size)*, (b) *1Disposable(Number, Color, Size)*

The agent knows two policies, *Dispose6* and *Dispose1*, with affordances shown in Figure 2. There is a hidden correlation between size and color and the agent's ability to lift it. If size is no more than 10 and color is red, the die is "light". The underlying correlation is encoded in the affordance.

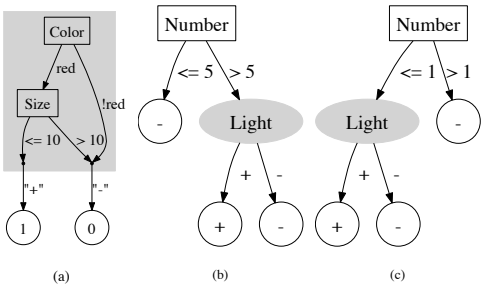

Figure 3: Dictionary Following Sub-Structure Extraction. (a) "Light", (b) Disposable6(Number, "Light"), (c) Disposable1(Number, "Light")

The dictionary compression yields the symbols shown in Figure 3. A sub-structure, labeled *"Light"*, was extracted and *6Disposable* and *1Disposable* became parameterized symbols. The new symbol, *"Light"*, has some noteworthy properties. The output from ¬red and > 10 is combined as they yield the same results. The original structure is maintained to facilitate use in a context where ¬red and > 10 differ. When used as an independent symbol, however, the external results are compressed, resulting in a binary feature.

While the symbol here is binary, this is not a restriction and symbols, in general, can be discrete or continuous.

Linguistic Aspects of Symbolic Representations Affordances, goals and policies represent potential means of symbol grounding where new symbols generated via sub-structure extraction are grounded in raw features. Policies here are action phrases. Affordances and goals are similar to noun phrases with an implicit subject. Given a dictionary as described above, we can construct a sentence that takes the form of an *if-then* statement: $A_\pi \wedge \pi \rightarrow g_\pi$.

Parameterization yields shorter sentences as the sentence is constructed using symbolic rather than raw features.

$$6Disposable(N, C, S) \wedge Dispose6 \rightarrow 6Disposed$$

$$6Disposable(N, "Light") \wedge Dispose6 \rightarrow 6Disposed$$

As shown in Figure 2, the uncompressed version of *6Disposable* requires three features. After compression, as shown in Figure 3, *6Disposable* requires only two parameters.

Shorter sentences improve reasoning as it implies less complexity. Moreover, it has potential benefits for communication as the number of bits required for transfer is reduced.

Implementation

The approach is demonstrated in a simulation environment with a robot arm and camera. Learning tasks revolve around lift, move, and drop and the SMDP framework (Sutton, Precup, and Singh 1999) is used to build policies using Q-Learning. Policies can then be used in higher level tasks.

As the agent learns new policies, affordances and goals are generated and added to the dictionary. After each addition, the best common sub-structure is extracted, compressing the dictionary and generating a new, abstract feature.

Raw Feature Set

Table 1 shows the raw feature set. Although the environment has an arbitrary number of objects, agent perception is limited to three. This limitation will be lifted in future work.

Attribute	Description	Visual?
hue	object color	yes
type	visual categorization	yes
stack	object top of stack?	yes
weight	object liftable?	no
at	agent currently at object?	yes

Attribute	Description
selected	which object is currently selected?
held	which object is currently held?
finished	has the agent finished?

Table 1: Object Features (Top), Global Features (bottom)

Primitive Actions and Policy Construction

Table 2 lists the agent's primitive actions from which we constructed three policy classes: *lift*, *stack*, and *clean*.

Within each policy class, we learn three policies: one general instance targeting the first lift-able object, one targeting a red object, and one targeting objects of type 1.

For each policy, affordance and goal classifiers are built. Goal states are assumed to be known.

Action	Description
find	Reveals the next visible object
select	Selects the next visible object
goto	Move to selected object
pickup	Pickup the object at the current location
drop	Drop the currently held object
finish	Terminate learning task

Table 2: Primitive Actions

Training data for an affordance is generated using the policy and goal classifier by sampling terminal states of policy executions from random starting points, resulting in a mapping from state to the probability of successful execution. The implementation limits affordances to three classes, *yes*, *no* and *maybe*, the last representing nondeterministic outcomes. Future work will use a more detailed representation.

Dictionary Compression Results

We compressed the dictionary as shown in Table 3.

	Bytes	Nodes + Edges
D	221296	2506
D'	194707	2089
Φ_{ss}	8864	140

Table 3: Compression Results ($N_{\Phi_{ss}} = 3$)

Symbol extraction yielded a reduction of 12% in dictionary size and 16% in node/edge count. Extracted symbols were typically around 100 elements.

Most Frequent Sub-Structure In addition to dictionary compression, we also looked at sub-structures in the context of object generalization. Figure 4 shows two very similar structures. Sub-structure detection treats them as different due to the difference in object id. To address this, we altered the algorithm to treat all object ids as being the same object.

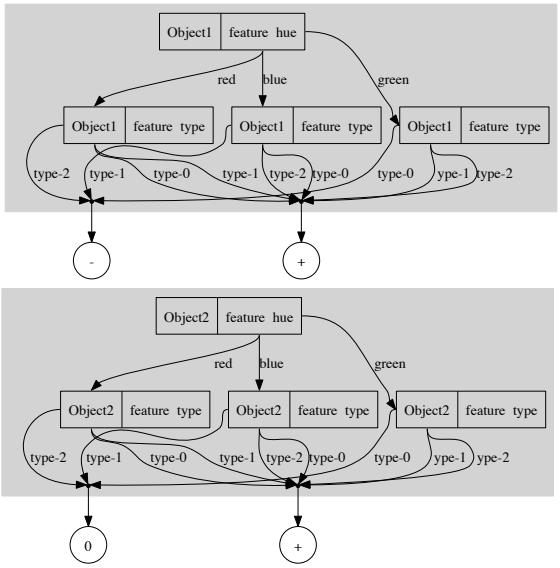

Figure 4: "Light" for Object 1 (top) and Object 2 (bottom)

Table 4 shows significantly improved results with this approach. This indicates that parameterized sub-structures can yield improved compression and symbolic features.

	Bytes	Nodes + Edges
D	221296	2506
D'	170374	1696
Φ_{ss}	944	16

Table 4: Compression for Generalized Object ($N_{\Phi_{ss}} = 54$)

Conclusion

As the complexity of learning tasks increase, the size of the representational space grows. This paper proposes an approach to managing representational complexity using a symbolic feature representation generated via policies, affordances and goals. Affordances and goals ground the symbols that constitute the agents dictionary and generate an abstract feature set for subsequent learning. Dictionary size is managed through sub-structure extraction, which removes redundancy. Extracted symbols form new abstract features.

References

Asadi, M., Huber, M. 2007. Effective control knowledge transfer through learning skill and representation hierarchies. *Proc. IJCAI*, 2054–2059.

Digney, B. 1996. Emergent hierarchical control structures: Learning reactive / hierarchical relationships in reinforcement environments. *Proc. SAB*.

Gibson, E., Spelke, E. 1983. *The Development of Perception*. Wiley.

Holder, L.B., Cook, D.J., Djoko, S. 1994. Substructure Discovery in the SUBDUE System. *Proc. AAAI Workshop on Knowledge Discovery in Databases*.

Kuramochi, M., Karypis, G. 2001. Frequent subgraph discovery. *Proc. ICDM*, 313-320.

Marthi, B., Russell, S., Latham, D., Guestrin, C. 2005. Concurrent hierarchical reinforcement learning. *Proc. AAAI*,

Marx, Z., Rosenstein, M., Kaelbling, L., Dietterich, T. 2005. Transfer learning with an ensemble of background tasks. *NIPS 2005 Workshop on Transfer Learning*.

McGovern, A., Barto, A. 2001. Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. *Proc. ICML*, 361–368.

Papudesi, V., Huber M. 2006. Learning Behaviorally Grounded State Representations for Reinforcement Learning Agents. *Proc. EpiRob*.

Rome, E., Hertzberg, J., Dorffner, G. 2006. *Towards Affordance-Based Robot Control*. Springer.

Sutton, R.S., Precup, D., Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112(1-2):181.

Taylor, M.E., Stone, P. 2005. Behavior transfer for value-function-based reinforcement learning. *Proc. AAMAS*, 53.

Taylor, M., Kuhlmann, G., Stone, P. 2008. Autonomous transfer for reinforcement learning. *Proc. AAMAS*, 283–290.