Using Imagery to Simplify Perceptual Abstraction in Reinforcement Learning Agents

Samuel Wintermute

University of Michigan
2260 Hayward St.
Ann Arbor, MI 48109-2121
swinterm@umich.edu

Abstract

In this paper, we consider the problem of reinforcement learning in spatial tasks. These tasks have many states that can be aggregated together to improve learning efficiency. In an agent, this aggregation can take the form of selecting appropriate perceptual processes to arrive at a qualitative abstraction of the underlying continuous state. However, for arbitrary problems, an agent is unlikely to have the perceptual processes necessary to discriminate all relevant states in terms of such an abstraction.

To help compensate for this, reinforcement learning can be integrated with an imagery system, where simple models of physical processes are applied within a low-level perceptual representation to predict the state resulting from an action. Rather than abstracting the current state, abstraction can be applied to the predicted next state. Formally, it is shown that this integration broadens the class of perceptual abstraction methods that can be used while preserving the underlying problem. Empirically, it is shown that this approach can be used in complex domains, and can be beneficial even when formal requirements are not met.

Introduction

Many problems that an intelligent agent might address have spatial characteristics. Here, we are interested in solving spatial problems through reinforcement learning (RL, Sutton & Barto, 1998). RL is a methodology for solving problems modeled as Markov Decision Processes (MDPs), where the agent learns a policy that maps states to actions. The number of states in an MDP has a large influence on the agent's performance. Spatial data is inherently continuous, but directly learning a policy in terms of continuous states is intractable for many problems, as the number of states the agent encounters can be infinite. To decrease the size of the state space, states that are functionally equivalent can be aggregated. Here, aggregate states will be called *abstract* states, and the original states *concrete* states.

For an agent receiving low-level perceptual data, often a library of perceptual abstraction procedures are available, such as the ability to identify objects and simple qualitative properties of how they relate, which can be used to compose state abstractions (e.g., Stober & Kuipers, 2008). For example, in the blocks world, predicates such as $on(A,B)$ can be composed from object identities and primitive topology and direction information (Wintermute, 2009). An abstract state composed of these predicates implicitly aggregates many concrete states where properties such as the shape and position of the blocks differ in unimportant ways.

However, there is a fundamental problem with this approach: it is unlikely that there exists a set of perceptual abstraction methods that can work well in all spatial problems (this is related to the poverty conjecture of Forbus et al. (1991)). Even if a human is designing an abstraction for a single problem, it is often difficult to do so while perfectly preserving the underlying problem. Both of these situations have the same implication: the best available state-space abstraction may be imperfect.

Although abstraction in spatial problems can be difficult, movements through space are often simple and easy to locally predict in terms of concrete states. Research in bimodal spatial reasoning, also known as *imagery*, pursues task-independent means for this sort of prediction (e.g., Wintermute & Laird, 2008). An imagery system maintains representations of a problem state in terms of both concrete information (such as a spatial representation) and abstract information, where perceptual processes derive the abstract representation from the concrete representation. Predictions can be made in terms of the concrete representation, and perceptual processes can then be applied to the "imagined" concrete state to determine the resulting abstract state. It has previously been identified that imagery can be useful to compensate for difficulties in perceptual abstraction (Wintermute, 2009a; Wintermute & Laird, 2009). Here, we formally examine why that approach is useful and how imagery can be tightly integrated with RL.

The technique introduced here, Reinforcement Learning using Abstraction and Imagery (ReLAI), works in conjunction with an existing RL algorithm, Q-learning. Where conventional Q-learning learns the value of each action in each state, with ReLAI it learns the value of state-action pairs sharing common predictions.

Formally, ReLAI is a technique for aggregating state-action (s, a) pairs in the table of values learned by Q-learning. The aggregate (or *category*) that an (s, a) pair belongs to is determined by the predicted next abstract state resulting from action a in concrete state s. If abstract states are related in the right way to the concrete state space (as will be discussed), and predictions are accurate, Q-learning will converge to the optimal policy, potentially much faster than it would without aggregation.

Conventionally, abstraction in RL has been used as a means to model reduction, where a new MDP is generated by aggregating states together (Li et al., 2006). However, with ReLAI, abstract states are used to categorize (s, a) pairs in the original MDP, not to generate a new MDP.

In this paper, (s, a)-aggregation in general is first examined, showing sufficient conditions for convergence to the optimal policy. Then, this approach is compared to MDP reduction, showing that (s, a)-aggregation can result in a more compact learning problem. ReLAI is then considered as a special case of this aggregation, and the convergence criteria are translated to requirements on how abstract states must relate to concrete states. It is then shown that the ability to use ReLAI instead of simple state aggregation increases the class of perceptual abstraction mechanisms that can be used by an agent. Finally, ReLAI is shown to provide benefits even when formal requirements are not met. An agent playing the game Frogger II is demonstrated. Given a set of mechanisms for perception (from the pixel-level on up), imagery, and abstraction, ReLAI outperforms state aggregation in this domain.

In short, compared to the alternative of simply using perceptual abstractions as problem states, integrating an imagery component with RL via ReLAI formally increases the class of abstraction mechanisms an agent can use, and provides substantial empirical benefits within tasks.

State-Action Aggregation in RL

ReLAI is a technique for aggregating state-action pairs in a Q-learning agent. In this section, we consider this concept separately, as background for the rest of the paper. Basic definitions and notation will follow Sutton & Barto (1998). At every time step, the agent observes a state $\mathrm{s_t}$, and selects an action $\mathrm{a_t}$. The environment then transitions and provides the agent with a reward $\mathrm{r_{t+1}}$ at the next time step.

Previous work has shown that Q-learning with state-action aggregation converges to the optimal Q^* function when all (s, a) pairs in the same category have the same Q^* value (Goetschalckx, 2009). Prior systems have used function approximation schemes that can be formulated as (s, a)-aggregation, however, there is little other theoretical work considering this aggregation as an exact method.

Properties of the function that assigns categories will be examined here. This function, $C(s, a)$, returns a symbol representing the category of (s, a). An appropriate function only assigns two (s, a) pairs to the same category if their respective Q^* values are the same, and will hence allow Q-learning to converge.

Here, we will show sufficient conditions such that a function C is appropriate. These conditions are that, for all states and actions, the reward received for a transition is independent of (s, a), given $C(s, a)$, and that categories of actions in a given state are independent of the (s, a) pair that led to that state, given the category of that pair. That is, these equations must be true:

$$Pr\{r_{t+1} = r | s_t, a_t, C(s_t, a_t)\} = Pr\{r_{t+1} = r | C(s_t, a_t)\} \quad (1)$$

$$Pr\{C(s_{t+1}, a) = x | s_t, a_t, C(s_t, a_t)\} = Pr\{C(s_{t+1}, a) = x | C(s_t, a_t)\} \quad (2)$$

$Q^*(s, a)$ is defined as $\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a \, max_{a'} \, Q^*(s', a')$.

Here, $\mathcal{P}_{\mathrm{ss'}}^{\mathrm{a}}$ represents state transition probability for action a in state s going to state s′, and $\mathcal{R}_{\mathrm{s}}^{\mathrm{a}}$ represents expected reward for a in s. In the case where Equations 1 and 2 are true, similar quantities can be defined with respect to categories. The expected reward for executing an action in a given category will be called $\mathcal{R}_{\mathrm{c}}$. Due to Equation 1, this is equal to $\mathcal{R}_{\mathrm{s}}^{\mathrm{a}}$ for all (s, a) in category c.

For the state transition component, an additional concept is needed. A block (written b) is a set of states that have the same categories for all actions. All states are in some block, and $B(s)$ signifies the block of state s. More formally:

$$(\forall_a [C(s_1, a) = C(s_2, a)]) \Leftrightarrow [B(s_1) = B(s_2)]$$

The function $C_b(b, a)$ returns the category of action a in block b.

Equation 2 requires that the categories of actions in a state s' be independent of the previous (s, a), given $C(s, a)$. The block of s' is determined by the action categories, so it must then also be independent of (s, a), given $C(s, a)$. This means that all (s, a) pairs in a common category must share a common probability of transitioning to a given block. $\mathcal{P}_{sb'}^a$ is defined as the probability of (s, a) transitioning to a state in block b', and $\mathcal{P}_{cb'}$ as the probability of transitioning to block b' with an action in category c, which must be equal to $\mathcal{P}_{sb'}^a$ for all (s, a) in c.[1]

With these concepts, category values can be defined:

$$V(c) = \mathcal{R}_c + \gamma \sum_{b'} \mathcal{P}_{cb'} \, max_{a'} \, V\big(C_b(b', a')\big)$$

It will now be shown that when the conditions are met, $\forall_{s,a}, V(C(s, a)) = Q^*(s, a)$.

[1] In the general case, where equations (1) and (2) are not necessarily true, similar quantities to R_s^a and $P_{cb'}$ could be defined, but they would be dependent on the policy, since (s, a) pairs that are in the same category but have different expected rewards or block transition probabilities could be sampled at different rates depending on the policy.

Since for all $(s,a) \in c$, $\mathcal{R}_c = \mathcal{R}_s^a$ and $\mathcal{P}_{cb'} = \mathcal{P}_{sb'}^a$,
$V(C(s,a)) = \mathcal{R}_s^a + \gamma \sum_{b'} \mathcal{P}_{sb'}^a \max_{a'} V(C_b(b',a'))$

The probability of transitioning to a block is the sum of the individual probabilities of all states in that block, so
$V(C(s,a)) = \mathcal{R}_s^a + \gamma \sum_{b'} \sum_{s' \in b'} \mathcal{P}_{ss'}^a \max_{a'} V(C_b(b',a'))$

All states in a block have the same categories, so:
$V(C(s,a)) = \mathcal{R}_s^a + \gamma \sum_{b'} \sum_{s' \in b'} \mathcal{P}_{ss'}^a \max_{a'} V(C(s',a'))$

After combining sums, we have:
$V(C(s,a)) = \mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a \max_{a'} V(C(s',a'))$

This is the same set of equations defined by $Q^*(s,a)$, simply renamed to $V(C(s,a))$, so the two formulations are equivalent. Since for all s and a $V(C(s,a)) = Q^*(s,a)$, all (s,a) in the same category must have the same Q^* value. Therefore, if C is such that Equations 1 and 2 hold, Q-learning aggregating (s,a) pairs according to C will converge to Q^*, resulting in a learned policy as good as what would have been learned without aggregation.

State-Action Aggregation vs. MDP Minimization

Figure 1 shows a simple problem that illustrates the benefits of state-action aggregation. The agent starts in some cell of a grid. There are two actions available, n, and e, which move the agent to the adjacent cell to the North or East. In each cell, there is an object of a particular type (**A** or **B**) that affects the reward the agent receives when it enters that cell. Each type causes a specific reward if the agent transitions to a containing cell—**A** causes a reward of -10 and **B** +1. In addition, **B** causes the episode to terminate. The grid is of infinite size, and the objects are randomly distributed within it in a fixed proportion (one-third **B**, two-thirds **A**). However, the entire layout of the world is inaccessible to the agent, as it has a small sensory horizon and can only perceive the cell it is in, along with the adjacent cells to the North and East.

This is an episodic task, and the object layout changes between episodes (sampled according to the given distribution). Figure 1 shows one possible state, viewed as a map and as an MDP. The letters in the MDP states correspond to the visible objects to the North, in the current cell, and to the East respectively. The figure shows the transitions from the state **AAB**. For example, moving East from **AAB** will always result in some state **?B?**, where the two **?** values—the North and East objects in the new state—are **A** or **B** with the appropriate probabilities. Note that the map and accompanying description contain useful information missing from the MDP. In particular, it is apparent that the only relevant information for deciding which action to take is the contents of the cell in that direction. An agent viewing the problem as an MDP does not know that the object in the cell it is moving to is more important than the other objects in other cells in determining the action value.

Consider Q-learning in a generalized version of this domain with N object types. $N-1$ of these types cause some negative reward (particular to each type) and do not terminate, while one causes +1 and terminates. If the perceptual information available is simply used as a state (as on the right of Figure 1), there are N^3 states. Since there are two actions, such an agent must learn $2N^3$ Q-values.

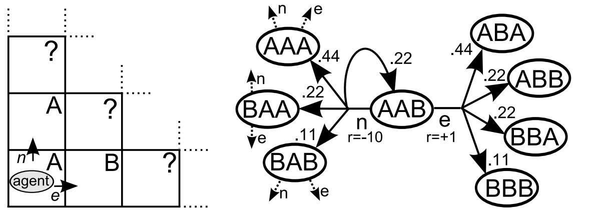

Figure 1: Two-action grid world problem. On the left, a map view; on the right, a partial MDP representation.

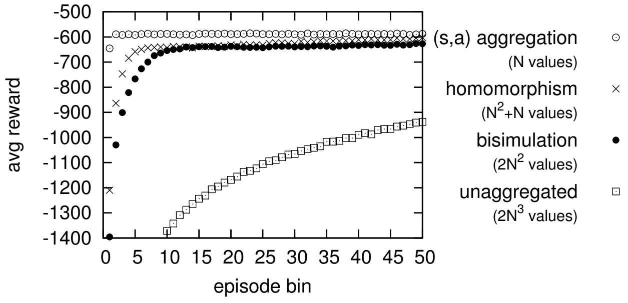

Figure 2: Performance of MDP minimization techniques vs. (s,a)-aggregation in a simple problem.

However, this representation makes unnecessary distinctions between states and actions. The MDP can be minimized by grouping together equivalent states and/or actions, generating a smaller MDP. Two possible formalizations of "equivalent" here are equivalency under stochastic bisimulation and under homomorphism (Givan et al., 2003; Ravindran & Barto, 2002). If either of these formalisms are used to map together states and/or actions within a single MDP, a minimal MDP can be iteratively generated by combining equivalent states.

In this domain, MDP minimization using bisimulation would result in states which only encode the contents of the neighboring cells, ignoring the current cell, resulting in $2N^2$ Q-values to learn. Homomorphism allows actions to be mapped together: for example, the state with **A** North and **B** East is equivalent to that with **B** North and **A** East, with action e in the first equivalent to n in the second. This reduces the values to learn to $N^2 + N$.

As an alternative to MDP minimization, state-action aggregation can be used. Here, the categorization function can aggregate (s,a) pairs based on the object in the destination cell (e.g., all (s,a) pairs arriving at an **A** are aggregated). Since there are N objects, there are then N (aggregate) Q-values to learn. This aggregation function fulfills equations (1) and (2), so Q^* will be learned.

Figure 2 shows data demonstrating the effects of these techniques. Here, $N = 26$ types were used. One type caused a +1 reward and termination, and was present in 15/115 of the cells. The remaining 25 (non-terminal) types had rewards of either -10, -50, -100, -800, or -1000, and were distributed with proportions of 2/115, 3/115, 4/115, 5/115, or 6/115. One type for each combination of these parameters was present. Q-learning with epsilon-greedy exploration[2] was used (parameters were $\alpha = 0.3, \varepsilon = 0.1, \gamma = 0.9$). 2500 trials of 5000 episodes each were run, and bins of 100 adjacent episodes were combined and averaged across all trials.

As is clear from these data, compared to MDP minimization, (s, a)-aggregation can lead to a more compact problem, resulting in much faster learning.

State-Action Aggregation via Prediction

The goal of the previous sections was to introduce the concept of state-action aggregation in reinforcement learning. Here, the special case of ReLAI is considered: where the category of an (s, a) pair is determined by the predicted next abstract state resulting from action a in concrete state s.[3]

The example in the previous section can be viewed in these terms. The MDP in Figure 1 shows concrete states of the problem, which encode the identities of all three perceived objects. Predictions are made in terms of abstract states which encode the object in the current cell where the agent is (ignoring the adjacent cells). Based on abstract predictions, aggregates are formed: for example, the agent considers all (s, a) pairs equivalent where the predicted next abstract state encodes an **A** in the agent's cell.

ReLAI predictions are made by an algorithm that maps (s, a) pairs to successor abstract states. In the example problem, this is simple to implement: given a concrete state, such as **AAB**, the predicted next abstract state encodes the first object (**A**) when the action is n, and the third object (**B**) when the action is e. The prediction algorithm instantiates a deterministic mapping, however, the underlying problem may be probabilistic, as it is here.

Although this scheme uses an abstract state space, abstract states are never seen as "states" by the RL algorithm. States that are useful when used with prediction are not necessarily as useful without prediction. In the example problem, an agent learning directly in terms of abstract states encoding only the object in the current cell would do no better than random.

How, then, should these abstract states relate to the underlying concrete states? Since Equations 1 and 2 are sufficient for appropriate (s, a) aggregation, and thereby allow Q-learning to converge, we will re-examine them in the situation where the category of (s, a) is a correct prediction of the abstract state that will follow from it.

The function $A(s)$ represents the abstract state of concrete state s. Assume that predictions are correct: in all cases $C(s_t, a)$ is what $A(s_{t+1})$ would equal if action a were to be taken.[4]

Under this assumption, Equation 1 holds if and only if the reward received for a transition is always independent of (s, a), given the next abstract state:

$$Pr\{r_{t+1} = r | s_t, a_t, A(s_{t+1})\} = Pr\{r_{t+1} = r | A(s_{t+1})\} \quad (3)$$

Similarly, under this assumption Equation 2 holds if and only if the abstract state that would result from taking some action a in s_t (which is $C(s_t, a)$) is independent of the previous state and action (s_{t-1}, a_{t-1}), given $A(s_t)$. This implies that the next abstract state must be independent of the previous (s, a) pair, given the current abstract state and action:

$$Pr\{A(s_{t+1}) = x | s_{t-1}, a_{t-1}, A(s_t), a_t\} = Pr\{A(s_{t+1}) = x | A(s_t), a_t\} \quad (4)$$

Equations 3 and 4, along with prediction correctness, can then be regarded as sufficient conditions for convergence with ReLAI[5]. This means that ReLAI can use abstraction functions where $A(s_{t+1})$ is not independent of s_t given $A(s_t)$, but is independent of s_{t-1}. This stands in contrast to MDP reduction using bisimulation or homomorphism, where $A(s_{t+1})$ must always be independent of s_t given $A(s_t)$.

Relaxing Requirements on Perceptual Abstraction

In the example problem, ReLAI predictions were generated by a simple algorithm that derived $A(s')$ directly from (s, a) pairs. However, as will be examined in detail in the next section, in an imagery system, an alternative is to predict $A(s')$ by first predicting s', and then applying a perceptual abstraction to that prediction. Since the final output is an abstract state, the prediction of s' can be arbitrarily wrong, as long as it lies within the same abstract state as the true s'.

A major goal of this work is to simplify the problem of perceptual abstraction in RL agents by increasing the class of abstraction functions that can be used. As an alternative to ReLAI, abstract perceptions can be directly used as states in Q-learning, instantiating state aggregation. Li et al. (2006) recently presented a comprehensive theory of state aggregation, describing five aggregation classes of increasing generality, and grouping aggregation techniques into those classes. Of those classes, the most general for

[2] The convergence proof of Goetschalckx (2009) is for learning under a fixed policy, but it could likely be extended to epsilon greedy.

[3] An alternate way of aggregating (s, a) pairs is to use afterstates (Sutton and Barto, 1998). This is done in game playing: the effects of the agent's actions are simulated, and $C(s, a)$ is the state of the game after the agent has played, but before the opponent has. Equations (1) and (2) can be applied in this context, and provide sufficient conditions that an afterstate formulation will work.

[4] This means that the agent correctly predicts all actions it takes, and $C(s_t, a_t) = A(s_{t+1})$ at all times, but also that the agent correctly predicts actions it does not actually take.

[5] Note, however, that Equation 4 does not strictly imply Equation 2. Equation 4 only covers actions the agent actually takes, not all *possible* actions as Equation 2 requires. There might be some abstraction function that, when used with a particular policy, meets Equation 4 for the actions taken, but would not have for other actions. However, that possibility will not be considered here.

which Q-learning convergence is guaranteed is called Q^*-*irrelevant*. Here, the only requirement is that all concrete states in the same abstract state have the same Q^* value for all actions. This category covers many state aggregation techniques, including bisimulation-based MDP reduction.[6]

However, ReLAI allows convergence with abstraction functions that are not Q^*-irrelevant. For instance, in the problem in Figure 1, the abstract states used with ReLAI (which simply encode the object in the same cell as the agent) group many concrete states together where Q^* values differ. Any agent architecture supporting ReLAI can also easily instantiate state aggregation (by simply not making predictions), so ReLAI increases the class of perceptual abstraction functions usable by an agent. Conversely, given a limited library of abstraction mechanisms available in an architecture, more problems can be covered if ReLAI can be used than if it cannot.[7]

Using ReLAI in Complex Problems

The previous section shows that ReLAI can be guaranteed to work with a wider class of abstraction functions than simple state aggregation. A broader goal of this work is building a general-purpose cognitive architecture capable of solving complex problems. In this section, ReLAI is examined with respect to that goal.

Inspired by other work using arcade games as a source of AI problems (e.g. Agre & Chapman, 1987; Diuk et al., 2008), the problem here uses the game Frogger II for the Atari 2600 (Figure 3). The actual game is used (run in an emulator) – it has not been reimplemented.

The agent does not play the whole game, but has a simpler goal of navigating the frog (bottom center of the Figure) to the top of the screen, without colliding with any of the moving obstacles or leaving the area shown. Without considering the rest of the game, this problem is still very difficult. The frog has five actions: move in four directions, or do nothing. There is a slow current in the water pulling the frog to the right, so inaction still results in motion.

The position of the frog is discrete in the vertical direction (there are 9 rows to move through), but many horizontal positions are possible due to the current. Most of the obstacles move continuously at uniform speed to the right or the left, although some move vertically or diagonally. Obstacles are constantly appearing and disappearing at the edges of the screen. This is an episodic task, and the initial state of the game differs across episodes (the obstacles start in different positions).

A reward function similar to that of the game score has been implemented: there is a reward of 1000 for winning (reaching the top row), and -1000 for losing (colliding with an obstacle or leaving the area). There is a reward of 10 for

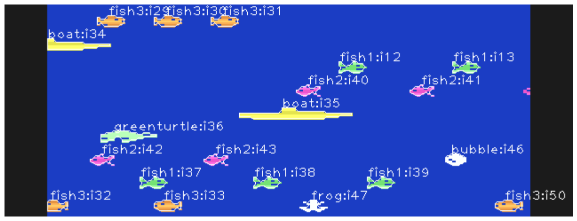

Figure 3: Frogger II ((c) 1984, Parker Bros.), with labels.

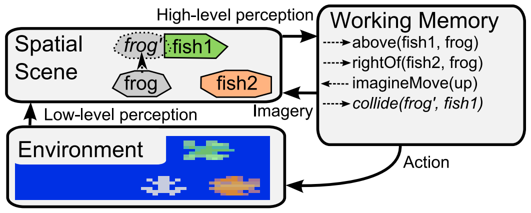

Figure 4: Imagery in the Soar/SVS architecture.

moving up, and -10 for moving down. At every time step, there is also a reward of -1 to encourage short solutions.

Our solution to this problem is implemented using the Soar cognitive architecture (Laird, 2008), which has recently been augmented with specialized processing for spatial information (Wintermute, 2009b). Figure 4 shows the relevant parts of the architecture. A low-level perception system segments, tracks, and labels the objects perceived. Such a system has been built for emulated Atari games (its labels are shown in Figure 3). This information is passed to the Spatial Scene, which represents objects geometrically in Euclidean space. High-level perceptual processes can act on this representation, to extract qualitative properties of the objects and their relations to one another into Soar's symbolic working memory, where they are accessible to Soar's decision process.

In addition, *imagery* can be used. Here, commands are formulated in working memory describing changes in the scene, and the scene is then updated. This includes the capability to simulate motion (Wintermute & Laird, 2008). In this domain, all motion is assumed to be linear translation at a constant velocity, which is a close approximation of the actual dynamics. The agent can track and project forward the motion of the obstacles, and has knowledge about how its own actions move the frog.[8]

In Figure 4, the frog and two fish have been identified by low-level perception. High-level perception has extracted qualitative information about relative object positions. A command has also been formulated to imagine the effect of moving the frog up. The imagery system has responded by creating a new object at the location the frog would move to. When high-level perception is then applied to the modified (imagined) scene, the agent infers that the imagined frog collides with a fish.

[6] Homomorphism-based MDP reduction is not included in the analysis, since it does more than aggregate states, but has the similar property that Q^* values are preserved in the new MDP.

[7] Of course, it is possible that state-aggregation formalizations broader than Q^*-irrelevance could be discovered that also guarantee convergence, in which case ReLAI would have to be compared against those as well.

[8] Note that this allows local predictions of the sort needed by ReLAI to be made, but does not approach being a full model of the world, and is far from sufficient for model-based RL techniques to be used. Object appearances and disappearances are not modeled, randomness is approximated deterministically, and rewards are not modeled.

```
for each episode
 for each step in the episode
  perceive the concrete state $s$ and any reward,
  store $s$ in the spatial scene
  for each action $a$
   use imagery to simulate $a$ in the spatial
   scene, along with any environmental changes
   apply high-level perception to the imagined
   scene, derive the next abstract state $A(s')$
   lookup the learned value of $a$ in $s$ based on
   the category of $(s,a)$, which is $A(s')$

  given the current action values and the
  reward, apply a Q-learning update to the
  category of the previous action (if any)
  choose an action using epsilon-greedy policy
 repeat until $s$ is terminal
repeat for all episodes
```

Figure 5: ReLAI algorithm as instantiated in the architecture.

In this architecture, the spatial scene is a concrete representation of the problem, and imagery provides the capability to simulate future problem states with a high degree of short-term accuracy. The architecture provides a library of atomic high-level perceptual operations that can be composed together using task knowledge to instantiate an abstraction function. Soar's RL functionality, operating over working memory, can then be used to learn a policy using abstract perceptual information. The abstraction can be used directly to learn in terms of abstract states (state aggregation), or can be combined with imagery-based predictions to instantiate ReLAI.

However, in complex domains like Frogger II, it is unlikely that such a perceptual abstraction function can be instantiated that exactly meets the convergence requirements of state aggregation or ReLAI. Instead, the agent designer must come up with a reasonably good abstraction and hope for the best. The hypothesis here is that, given a perceptual abstraction, the relaxed formal requirements of ReLAI can translate to better empirical performance compared to state aggregation when the precise requirements of neither technique are met.

To test this, an abstraction function for Frogger was created. The abstract perceptions encode the following information in working memory:

- the vertical position of the frog: one of the 9 rows
- a rough horizontal discretization of the frog's position into left, middle or right regions
- a determination as to whether or not the frog currently collides with an obstacle

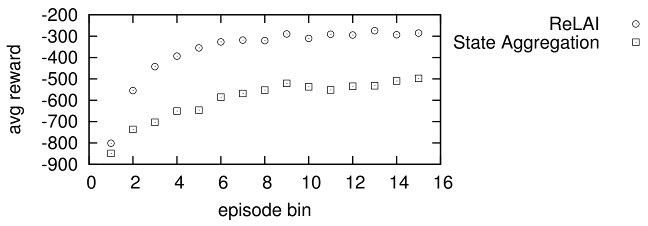

Figure 6: Frogger II performance.

- a determination as to whether or not an obstacle (or screen edge) is adjacent to the frog in each of the four directions.[9]

As a state representation, this abstraction loses potentially useful information, and is not Markovian (since the agent could make better decisions by remembering where it has seen obstacles). However, it is compact, and just as important, it can be composed from the simple perceptual operations available in the architecture.

The same perceptual abstraction function is used in both a state-aggregation agent and a ReLAI agent. Both agents choose an action every 15 game frames (four per second). The ReLAI algorithm as used in Soar is shown in Figure 5. Here, action categories are determined by using imagery to project forward the motion of the obstacles near the frog along with the effect of the action on the frog, and applying the abstraction to that imagined state. In addition to abstract perceptions, in this domain the ReLAI agent also encodes the proposed action as part of the abstract state. This is because perceptions about the next state alone cannot capture the immediate reward for the transition, as moving up or down a row effects reward (not just being in a particular row). However, the last action taken is not useful as part of the other agent's state, so it is not included there.

For ReLAI, the requirement that the abstraction captures immediate reward (Equation 3) is met, and the requirement that predictions are accurate comes close to being met, only missing a few cases where moving objects do not follow a constant velocity or disappear unexpectedly. The requirement on state independence (Equation 4) is not met: $A(s_{t+1})$ is not strictly independent of s_{t-1}, given $A(s_t)$, so convergence to Q^* isn't guaranteed. However, unlike state aggregation, ReLAI is robust to abstractions where $A(s_{t+1})$ *is* dependent on s_t given $A(s_t)$, which can be beneficial.

For example, the ReLAI agent can base its action choice on a precise prediction of whether or not it will collide with an obstacle in the new state $A(s_{t+1})$, where the other agent can only base its decisions on $A(s_t)$, which includes information (obstacle adjacency) that can only roughly predict future collisions between moving objects. The concrete state s_t contains enough information to predict collisions in the next state almost exactly, but this information is only useful to the ReLAI agent.

[9] An "adjacent" obstacle is one that intersects a rectangular region starting at the frog's bounding box and projecting in the appropriate direction by 10 pixels (about the same as the inter-row distance).

Experiments were run using the actual (emulated) game. Q-learning with epsilon-greedy exploration was used (parameters were $\alpha = 0.3, \varepsilon = 0.1, \gamma = 0.9$). 30 trials of 6,000 episodes each were run in each condition. Figure 6 shows the results. Here, groups of 400 adjacent episodes were binned together; the results are averaged across all episodes in the bin and across all trials (each point represents 12,000 games). The graphed results do not show the ability of the agents to play the game well: epsilon-greedy exploration meant the agent acted randomly 10% of the time (often with fatal results), and some of the randomly-chosen start states were unwinnable. These factors contributed to high variability in the data, necessitating the averaging of many games per data point.

To examine the final policy, 700 games were run in each condition using the final policies, but without exploration and with unwinnable games filtered out. Of these, the state aggregation agent received an average reward of -66 and won 45% of the games, while the ReLAI agent received an average reward of 439 and won 70% of the games.

The ReLAI agent clearly outperforms the state-aggregation agent: it learns a better policy, and learns it faster. The use of simple, local predictions based in imagery led to much better performance than without, using the same perception system.

As Atari graphics are simple, the perception system can be configured to work in many games. Agents for two other games (Space Invaders and Fast Eddie) have been implemented, with similar results to what was achieved in Frogger. As task independence has been a priority of the design of Soar's imagery components, no architectural modification was necessary to address these games.

Conclusion

A technique for integrating reinforcement learning and imagery, ReLAI, has been introduced. It has been shown that ReLAI learns faster and allows a broader class of perceptual abstraction functions to be used compared to standard RL state aggregation.

ReLAI has been instantiated in the Soar cognitive architecture, integrated with a low-level interface to an Atari video game emulator, and was used successfully to play the game Frogger II. Its performance was compared to that of a similar agent using the same perceptual operations, but without imagery, and it was shown to learn a better policy, faster.

At a broad level, this work serves to demonstrate the benefits of multiple representations and imagery for AI systems. It has previously been identified that imagery can ease the problem of abstraction in particular problems, such as in motion planning (Wintermute, 2009a) and in versions of the blocks world (Wintermute & Laird, 2009). However, the formal results above allow this work to apply much more widely. In that way, it is step toward a better understanding of the relationship between perception, internal representation, decision making, and learning.

Acknowledgments

John Laird provided guidance and support on this project, and helped edit the paper. I thank Erik Talvitie and Satinder Singh for useful feedback on the theoretical aspects of the project, and Yongjia Wang, Joseph Xu, and the anonymous reviewers for helpful comments on the paper. Development of the emulator interface was greatly aided by code provided by members of the RL3 lab at Rutgers. This research was funded by a grant from US Army TARDEC.

References

Agre, P. & Chapman, D. (1987). Pengi: An implementation of a theory of activity. *Proceedings of AAAI-87*

Diuk, C., Cohen, A., & Littman, M.L. (2008). An object-oriented representation for efficient reinforcement learning. *Proceedings of ICML-08*

Forbus, K.D., Nielsen, P., & Faltings, B. (1991). Qualitative spatial reasoning: the CLOCK project. *Artificial Intelligence* 51(1-3)

Givan, R., Dean, T., & Greig M. (2003). Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence* 147(1)

Goetschalckx, R. (2009). On the Use of Domain Knowledge in Reinforcement Learning. PhD Thesis, Katholieke Universiteit Leuven

Laird, J.E. (2008). Extending the Soar Cognitive Architecture. *Proceedings of AGI-08*

Li, L., Walsh, T.J., & Littman, M.L. (2006). Towards a unified theory of state abstraction for MDPs. *Proceedings of ISAIM-06*

Ravindran, B. & Barto, A.G. (2002). Model minimization in hierarchical reinforcement learning. *Proceedings of SARA 2002*

Stober, J., & Kuipers, B. (2008), From pixels to policies: A bootstrapping agent. *Proceedings of ICDL-08*

Sutton, R.S. & Barto. A.G. (1998). Reinforcement Learning: An Introduction. MIT Press: Cambridge

Wintermute, S. (2009a). Integrating Reasoning and Action through Simulation. *Proceedings of AGI-09*

Wintermute, S. (2009b). An Overview of Spatial Processing in Soar/SVS. Report CCA-TR-2009-01, U. Michigan Center for Cognitive Architecture

Wintermute, S. & Laird, J.E. (2008). Bimodal Spatial Reasoning with Continuous Motion. *Proceedings of AAAI-08*

Wintermute, S. & Laird, J.E. (2009). Imagery as Compensation for an Imperfect Abstract Problem Representation. *Proceedings of CogSci-09*