Appearing in the *Proceedings of the 23th International Florida Artificial Intelligence Research Society Conference (FLAIRS 2010)*, Daytona Beach, FL

Improving Structural Knowledge Transfer with Parametric Adaptation

Tolga Könik, Kamal Ali, Dan Shapiro
konik@stanford.edu, kamal3@yahoo.com,
daniel.g.shapiro@gmail.com
Cognitive Systems Lab, CSLI
Stanford University, Stanford, CA 94305

Nan Li, David J. Stracuzzi
{Nan.Li.3 | david.stracuzzi}@asu.edu
School of Computing and Informatics
Arizona State University, Tempe AZ 85287

Abstract

Transfer of learned knowledge from one task to another offers an opportunity to reduce development cost of knowledge-based systems by reusing existing knowledge in novel situations. However, minor differences in the initial and target environments can reduce the effectiveness of the system substantially. In previous work, we presented a system that acquired procedural knowledge of American football from video footage, and then applied it to controlling players in a simulated environment. In this paper, we extend that system by adding the ability to adapt the transferred procedures to better fit the simulator. We show that even when the transferred structural knowledge provides a quality starting point for performance in the game environment, a simple parameter optimization technique can significantly improve its performance and utility.

Introduction

Research into transfer of learning considers the problem of applying knowledge obtained in one context to improve performance in another. This include the initial acquisition of knowledge in a source context, mapping of the learned knowledge into a target context, and adaptation of the transferred knowledge to improve system performance in the target. Transfer offers the potential to greatly speed up learning through reuse of related knowledge. More broadly, it offers the long-term promise of replacing a one-off software development model with a less expensive process of transfer, adaptation, and reuse.

While much work in transfer of learning has focused on the reuse of classification knowledge, procedural knowledge transfer concerns the reuse of executable skills, such as game playing strategies or problem solving behaviors. The difficulty of the transfer task scales with the difference between the source and target contexts, so research often concerns transfer among distinct tasks in the same domain (Choi et al. 2007; Hinrichs and Forbus 2007) or analogous tasks in related domains (Taylor and Stone 2005; Könik et al. 2009).

This paper concerns transfer of procedural knowledge between two families of tasks in a context that makes contact with the physical world: from play recognition in American football videos into play execution in a simulated American football game. The work presented here begins with an existing transfer system described by Li et al. (2009b) that observes football videos, extracts procedural knowledge structures, and then uses the learned procedures to execute plays in a simulated game environment. We extend this system by adding a parametric learning component that adapts the transferred structural knowledge to the dynamics of the target domain by playing games in the target domain and adjusting numerical parameters to improve performance.

Although the initial work by Li et al. (2009b) demonstrated the feasibility of structural transfer and demonstrated qualitative similarity between the behavior in observed videos and simulated environment, performance in the simulator suffered because the transferred knowledge was not adapted to the dynamics of the target environment. The work reported here addresses this limitation with a parametric learning component that adapts the transferred structural knowledge to the dynamics of the target domain by playing games in the target domain and adjusting numerical parameters to improve performance. We claim that even simple parametric adaptation substantially increases the performance and utility of transferred structural knowledge and we support this claim with experiments. We also argue that structural knowledge transfer simplifies target play optimization by shaping the parameter space. This is based on the observation that the transferred knowledge structures provide a starting point and boundaries for the parameter estimation algorithm.

The Transfer Task

This work investigates procedural knowledge transfer in the context of American football. The rules of the game are complex, but we focus here on individual plays rather than the entire game. This eliminates questions of long-term strategy and play selection, and focuses attention on the execution of specific sequences. A play begins with the ball placed near the center of the field horizontally, and at the location of the end of the previous play vertically. The team in possession of the ball (offense) attempts to move the ball into the opposition's territory in an effort to score points by reaching the end of the field. We consider here only passing plays, in which one player (the quarterback) attempts to throw the ball down-field to a receiver. The opposing team

Figure 1: A partial video frame from the source domain.

Figure 2: An initial play state in the target domain.

(defense) attempts to prevent any forward progress by the offense by interfering with the pass. In the following, we provide details on the source and target tasks, and introduce the relevant elements of American football as needed.

The Source Task: Play Recognition

The raw data in our experiments consists of preprocessed (Hess and Fern 2009) video footage that shows individual plays executed by the Oregon State University football team as recorded by an overhead camera. Figure 1 shows a single frame from one of the plays. Given the preprocessed video, the system described by Li et al. (2009b) first performs play recognition to determine what happened during the play. The system then extracts a hierarchy of executable procedures that represent the observed behaviors employed by all of the offensive players. The next section provides a more detailed review of the transfer system.

The Target Task: Simulated Play Execution

After observing the plays and acquiring procedures for reproducing the behaviors of the offensive players, the task then becomes one of executing the same plays in the Rush 2008 football simulator[1]. Figure 2 shows an initial configuration of players in the simulator. The goal here is for the offense to move the ball down-field as far as possible by executing the observed plays. The system controls the offensive players in Rush by providing commands that control them on a tick by tick basis. Rush then automates the defensive players using built-in scripts and controllers. The simulator returns the outcome of the play in terms of yards gained or lost. The extended transfer system that we describe here attempts to improve upon the procedures learned by the original system by adapting parameters embedded within them to the simulated environment.

Rush differs from the observed college football setting in several ways. First, Rush uses eight player teams as opposed to eleven, and relies on simplified physics (no momentum, for example). Likewise, the physical characteristics of the simulated players, such as speed and ability, differ from the players observed in the videos. The defensive strategies that Rush applies during simulation also differ from the those observed in the videos. As a consequence, plays that worked well for Oregon State team may not work well in Rush. Nevertheless, the plays and player behaviors recognized from the source task provide a useful starting point for adaptation.

[1] http://www.knexusresearch.com/projects/rush/

The Transfer System

Our system follows a sequence of three steps to acquire and transfer knowledge from videos into execution of plays in a simulator. These include low level processing of videos, construction of executable procedures by explaining observed videos, and adaptation of those procedures to the target domain by experimentation and parameter learning. We discuss each of these steps and knowledge representation they use.

Representation and Execution of Procedural Knowledge

We use knowledge representation and performance components of the ICARUS agent architecture (Langley and Choi 2006) both in the source recognition and target execution tasks. In the source task, ICARUS uses conceptual background knowledge about football to observe plays and extract executable procedural knowledge that encode strategies for football plays. In the target, ICARUS executes the learned procedural structures to control individual players in the Rush simulator, and uses feedback from the simulator to adjust parameters on the learned procedures with the objective of improving performance.

Representation. ICARUS represents *concepts* in terms of first-order logic rules similar to Horn clauses (Table 1). Primitive concepts consist of tests on raw perceptual data. When matched, they assert the relation given by the head of the rule. For example, *near* becomes true when two perceived agents are within a certain distance of one-another. Non-primitive concepts reference other relations. For example, the notion of an open recipient (for a pass) is defined as an offensive receiver who is not near a defensive player.

ICARUS represents procedures (called *skills*) with hierarchical structures. Each skill has a head that describes a *goal*, which is an abstract concept that is expected to be satisfied at the end of skill execution, and a body that describes meth-

Table 1: Sample concepts in the American football domain.

```
((near ?agent1 ?agent2)
 :percepts  ((agent ?agent1 x ?x1 y ?y1)
 (agent ?agent2 x ?x2 y ?y2)
 :tests ((<= (sqrt (+ (expt (- ?x1 ?x2) 2)
                      (expt (- ?y1 ?y2) 2))) *threshold*)

((covered-recipient ?agent1 ?agent2)
 :percepts  ((agent ?agent1 team offense recipient t)
             (agent ?agent2 team defense))
 :relations ((near ?agent1 ?agent2)))

((open-recipient ?agent)
 :percepts  ((agent ?agent team offense recipient t))
 :relations ((not (covered-recipient ?agent ?a2))))
```

ods for achieving the goal of the skill. The body of primitive skills refer directly to executable procedural attachments via an `actions` field, while non-primitive skills reference other skills via an ordered set of `subgoals`. Moreover, each skill has a start field that determines when a skill is applicable. In Table 2, the non-primitive skill for controlling the Quarterback specifies subgoals for completing a pass that will be reactively executed as they are enabled by game conditions.

Inference and Execution. ICARUS operates in distinct cognitive cycles. At each cycle, it first uses its conceptual knowledge, by forward chaining on facts asserted in the global *perceptual buffer* to infer their transitive closure. It asserts these inferences into a short-term belief memory. Next, it considers the top level goal for each agent. If the concept corresponding to the goal is not satisfied in the current state, ICARUS considers decomposing that goal into subgoals using applicable skills. If ICARUS selects a non-primitive skill, it considers the subgoals in order to select the first unsatisfied subgoal and tries to achieve that subgoal using other skills. The decomposition continues until the system forms a path of skills starting at the top level goal and ending at a primitive skill such that the start conditions of all the skills in the path are satisfied with consistent variable bindings.

At this point, ICARUS instantiates the variables in the primitive skill and records the action it will need to execute. ICARUS applies this process for each top-level goal (one for each offensive player), and then executes the resulting actions in parallel. This high-level loop of perception, inference, action selection, and execution repeats every cognitive cycle. When a subgoal at any point in the hierarchy is satisfied, ICARUS considers the next subgoal of that skill. This process continues at all levels of the hierarchy until all top level goals are satisfied.

As an example, consider the first skill in Table 2. This skill requires three sub-skills to be executed in sequence. Note that the execution of any of sub-skill may consume several time cycles, for example, if the QB has to wait for a receiver to become clear.

Extraction of Structural Knowledge

Preprocessing of Real Football Videos. As input of our transfer system, we converted preprocessed versions of the source videos (Hess and Fern 2009) into a sequence of ICARUS perceptions for the architecture to observe. Specifically, the perceptual representation includes: (1) the 2D field

Table 2: Sample skills in the American football domain.

```
((pass-completed ?passer ?receiver ?n-steps)
   :start ((snap-completed passer ?ball))
   :subgoals
          ((dropped-back ?passer ?n-steps)
           (scrambled ?passer ?ball)
           (pass-completed ?passer ?receiver)))

(pass-completed ?passer ?receiver-role)
  :start    ((possession ?passer ?ball))
  :actions  ((*throw ?passer ?receiver)))
```

coordinates of each player at each video frame, (2) player labels describing the functional role of each player (e.g. quarterback, running back, etc), and (3) activity labels describing the low-level activity (such as running or blocking) of each player throughout the play. We used the activity labels and player labels generated by Hess, Fern, and Mortenson (2007) to produce the perception sequence for the play.

Extraction of Procedural Knowledge. The next step is to construct skills that explain the behavior of the offensive players in the videos. In previous work, Li et al. (2009b) used ICARUS to extract skills from football videos using a learning technique (Li et al. 2009a) they integrated in the architecture. The input of this system consists of a goal, a set of concepts sufficient for interpreting the observed agents behavior, a set of low-level methods available in the environment, and a sequence of observed perceptual states. The output is a set of executable skills hierarchies, one for each offensive player, consistent with the behavior of the observed players.

The algorithm runs in three steps. First, the system observes the video of the game and infers beliefs about each state, storing the results in belief memory. Next, the agent explains how the goal was achieved using its background knowledge. Then, it parses the observation sequence based on the times when these subbeliefs became true. This explanation process continues recursively until the agent builds explanations that show what sequence of events and/or known actions led the agent to achieve its goal. Finally, the explanation is used to determine the start conditions and subgoals of the skill hierarchy.

Mapping executable knowledge to target domain. The mapping algorithm translates generalized source behavior represented with ICARUS skills to the target environment. Since in this research, we have selected the same basic vocabulary (i.e., the player and ball location) for the source and target domains, the same conceptual knowledge base can be used to interpret both real video data and Rush events and therefore no representation mapping (Könik et al. 2009) between source and target symbols is required. The only remaining problem is to map the source players to their target counterparts since Rush employs fewer players with different requisites and a different field size.

Since the source involves eleven players but the target only has eight, the mapping algorithm selects and eliminates three players. It retains any player that touches the ball, always including the center, the quarterback and one receiver. Per the rules of Rush football, it must also retain a minimum of three guards on the line of scrimmage. A source play contains two guards on either side of the center. The algorithm keeps the two guards that flank the center, but eliminates

the two outermost guards, leaving 9 players on the offensive team. Thus the problem comes down to eliminating one eligible receiver among the three or four such receivers in typical source plays. Since the algorithm retains the player that catches the ball, it randomly eliminates one of the remaining eligible receivers. Finally, the initial positions extracted from videos are scaled according to new field size in Rush.

Parametric Adaptation in the Target

The parameter learner accepts as input ICARUS skills with some constant values marked as parameters and a reward function, executes the skills using different parameter values to evaluate the reward they return, and updates the skill parameter values to increase predicted future rewards.

In our current implementation, the parameter learner is an external wrapper around the ICARUS architecture. However, in future, we want to make parameter learning part of the architecture, functioning in conjunction with other architectural processes. Therefore, our algorithm improves the parameters incrementally when the agent experiences reward, without depending on the history of the behavior except some summary statistics that are updated incrementally.

In our transfer framework, the arguments that occur in the top level goal of each player agent are selected as parameters and the reward function is the yardage gained at the end of each play. Our system differentiates between ordinal and nominal parameters that have ordered and categorical values respectively. For example the parameterized top level goal of the quarterback `(pass-completed QB *receiver *qbDropSteps)` can be achieved using the first skill in Table 2 for specific parameter values for the nominal parameter `*receiver` (the player the quarterback is going to throw the ball), and the ordinal parameter `*qbDropSteps` (number of steps the quarterback runs back once he receives the ball). We use the convention that the first argument (here, QB) denotes the player the goal belongs to and that argument is not parameterized. Our current implementation assumes a method for specifying the legal range and step size for ordinal parameters and legal values for nominal parameters.

Our system start optimizing the parameters of a skill set by starting with their initial values and perturbing those values one parameter at a time. It executes the skills to perform the task (a play in our case) for each new parameter setting a few times and returns an average reward. The system selects which parameter to perturb randomly, but the probability of choosing a parameter is proportional on expected reward improvement. More precisely, the probability of choosing a parameter depends on the average reward improvement historically observed when perturbing that parameter.We use the Laplace notion to assign an initial reward improvement credit to each parameter so that previously unselected parameters have some non-zero probability of being selected for perturbation.

The parameter learner maintains a *personal best*, a record of the highest reward that has ever been received and the complete parameter setting that produced that reward. It also records how long it has been since it has set a new personal best. Using that information, the longer the system has gone without setting a new personal best, the greater the perturbations it is willing to make on the parameter values.

When an ordered parameter p_i is selected for perturbation, if this is the first time this parameter has been chosen since a personal best was set, then the *magnitude* of the perturbation is set to the step-size of the parameter (the intuition is that the step-size specifies the minimum amount of change in the parameter that is worth thinking about for reward maximization). A direction (one of "increasing" or "decreasing") is chosen at random. The magnitude together with the direction of perturbation is added to the previous value of this parameter to produce the new value. If this parameter is chosen again (on another iteration) in between setting a personal best, then the step size on each such occasion is increased by s_i. Thus, the agent searches further and wider until it sets a new personal best score. If it reaches the limit (minimum or maximum) of the parameter, it resumes its search by setting the step size down to its initial value (s_i) again. If a significant improvement in reward is experienced after perturbation of a parameter, then that same parameter is locked in for the next iteration - the previously used step-size and magnitude are retained and the agent simply takes another step in that good direction in parameter space.

If a nominal parameter is chosen, the agent chooses a value at random, with probability depending upon how much reward was historically obtained when choosing that value. More precisely, the probability of a value being chosen is computed by taking the mean reward associated with that value and subtracting its standard-deviation (variability)^2. Only 3 running totals per value need be kept in memory to support online computation of mean and standard-deviation, therefore the current memory requirement is linear to the number of parameters and independent of the length of the history making our algorithm incremental.

Transfer Experiments

In this section we describe the methodology and results of our transfer experiments. Our goal is to show that our transfer system creates agents with good performance and parameter learning contributes to the effectiveness of the transfer system.

Methodology

In this paper, our focus is the case where initially our system does not possess any executable target knowledge and all such knowledge must be transferred from the source domain. This scenario does not permit a methodology that compares a transfer agent against a non-transfer agent because the non-transfer agent would always achieve zero performance. Instead, we show that our system achieves positive transfer (positive average yardage), parameter learning in the target improves upon skills structurally transferred from source, and the performance of the transferred knowledge is comparable to the performance of build-in Rush agents.

To measure the effect of target learning on transfer, we compared the performance of agents generated by three

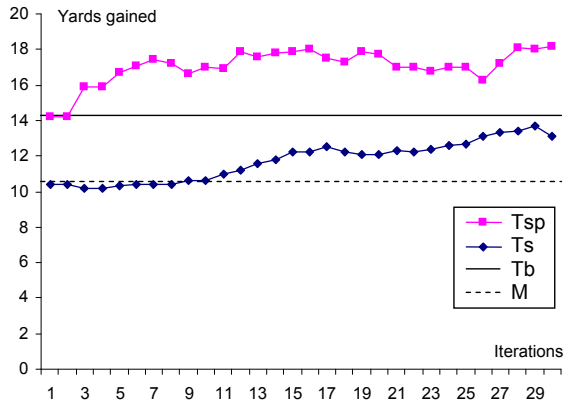

Figure 3: Comparison of transfer agents on play13.

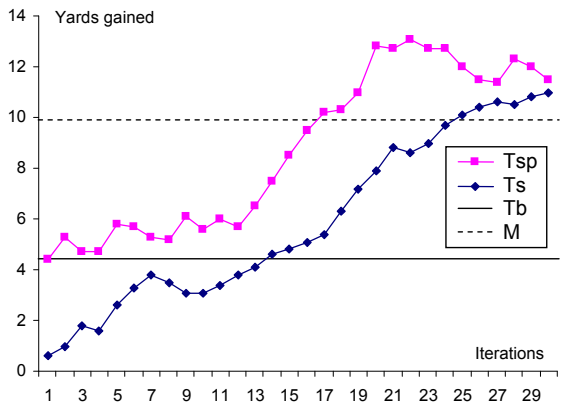

Figure 4: Comparison of transfer agents on play63.

transfer systems against hand-coded agents. The first system is a baseline system (Tb) that transfers structure and parameters from the source but does not apply target learning, the second system applies parameter learning starting with structures and parameters transferred from source (Tsp), and the third system employs target learning only on transferred structures using random initial parameters (Ts). We compared the performance of the agents generated by these systems against manually coded agents (M) written in Rush script language designed to generate behavior similar to the input videos.

We randomly chose two plays to which we applied the source recognition and target learning. For each agent in each play, we performed five repetitions - the learning curves in Figures 3 and 4 are averages over these five repetitions (each learning curve took 1-2 days of CPU runtime (˜2Ghz) to produce, which limited how many plays we could use for our experiments).

Results

In both scenarios, the final agents created by transfer systems that employ learning (Tsp, Ts) gained significant yardage, and after learning, they outperformed manually coded agents. In Play 63 (Figure 4), target learning significantly improved the performance of transferred agents (Tsp, Ts) and they outperformed the agents of the transfer system that does not employ target learning (Tb). The effect of target learning was also positive but not as dramatic in Play13, because the transferred agents were already performing at the level of hand-coded agents or better.

Our experiments show that both structural and parametric transfer contribute to the performance of transferred agents. In our system, structural transfer is crucial because the parameters are variables of the transferred structures and they are meaningless in the absence of those structures. Our results confirm that transferring parameters in addition to structures is also useful because starting with good parameter values can help the learning algorithm to converge faster or to avoid local minimas. For example, in Figure 4, we observe that Tsp converges faster than Ts and in Figure 3, Ts does not catchup with Tsp. These observations confirm the added value of transferring parameters.

Related Work

Research on procedural knowledge transfer is rare in comparison with work in a classification setting, but it shares an evolution towards producing transfer between increasingly distant source and target tasks. This work is often conducted in a reinforcement learning framework, where the object of transfer is a value function (Taylor and Stone 2005). This form of transfer requires auxiliary mappings between domain states and actions, and the technology for acquiring those mappings can become quite involved as the gap between source and target environments grows. For example, Liu and Stone (Liu and Stone 2006) employ a structure mapping engine to find analogies among qualitative dynamic Bayes nets expressing soccer keepaway tasks, while Kuhlmann and Stone (Kuhlmann and Stone 2007) develop a graph-based method for identifying large scale structures (e.g., rectilinear grids) in previously encountered games. Torrey et al. (2006) transfer situation-action rules obtained by ILP techniques (again, among keepaway tasks), but employ them to provide advice for constructing a value function.

Research on transfer in the context of cognitive architectures tends to communicate more structured skills. For example, Gorski and Laird (2006) port executable SOAR rules representing topological knowledge (maps, locations, and routes) among path finding tasks. As in the reinforcement context, this form of transfer requires sophisticated machinery to bridge distinct source and target domains. For example, Hinrichs and Forbus (2007) use analogical mapping to transfer case-based decision rules among Freeciv tasks, while Könik et al. (2009) develop a goal-directed analogical reasoning algorithm to transfer skills between grid-based games that lack shared symbols and subgoal structure.

Our work on reusing recognition knowledge for play design continues this trend towards disparate source and tar-

get tasks. However, in contrast with the above, we bridge distinct purposes versus representations. This shifts emphasis from the mapping technologies that enable transfer onto the power of the skill representation itself. Like Choi et al. (2007) who argue that the generalization inherent in ICARUS skills naturally supports performance in similar tasks, we argue that the structure of such skills natively biases learning after transfer. That benefit can be exploited by simple methods, such as plan critics that date back to Hacker (Sussman 1975) and NOAH (Sacerdoti 1977).

Summary

This paper has presented a system that accomplishes procedural knowledge transfer between distinct families of tasks; from recognition of American football plays into play design for a different variant of the game, beginning with raw video footage. The work has the character of an application study, in that it involves significant integration of multiple technologies.

Our experimental results demonstrate first that the task is feasible, and second that the transfer is strong, it generates comparable results to hand-coded agents. Moreover, even simple parametric target learning can significantly improve the performance and usefulness of transferred structures.

Since our transfer system is based on a general cognitive architecture, we are able to make a clear distinction between domain dependent knowledge and domain independent mechanisms. All components of our transfer system except low level video preprocessing, is domains independent and could in principle be applied to different problems by changing only background knowledge.

Our current work expands on this theme. We are working on structural learning, which would further adapt transferred knowledge to the target domain. Other fruitful future directions include synthesis of new plays by merging knowledge extracted from multiple source plays, and learning play selection based on capabilities of the offensive and defensive teams. Although our work focuses on a single application domain, it shows promise of transferring data from real world to generate software agents automatically.

Acknowledgements

We thank Alan Fern, Robin Hess and Jervis Pinto for providing us the results of their work on preprocessing the raw videos. We would also like to thank David Aha, Julie Fitzgerald, Negin Nejati and Pat Langley. This paper reports research sponsored by DARPA under agreement FA8750-05-2-0283. The U. S. Government may reproduce and distribute reprints for Governmental purposes notwithstanding any copyrights. The authors' views and conclusions should not be interpreted as representing official policies or endorsements, expressed or implied, of DARPA or the Government.

References

Choi, D.; Konik, T.; Nejati, N.; Park, C.; and Langley, P. 2007. Structural transfer of cognitive skills. In *In Proceedings of the ICCM*.

Gorski, N., and Laird, J. 2006. Experiments in transfer across multiple learning mechanisms. In *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*.

Hess, R., and Fern, A. 2009. Discriminatively trained particle filters for complex multi-object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Hess, R.; Fern, A.; and Mortenson, E. 2007. Mixture-of-parts pictorial structures for objects with variable part sets. In *Proceedings of the Eleventh IEEE International Conference on Computer Vision*. Rio de Janeiro, Brazil: IEEE Press.

Hinrichs, T., and Forbus, K. 2007. Analogical learning in a turn-based strategy game. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*.

Könik, T.; ORorke, P.; Shapiro, D.; Choi, D.; Nejati, N.; and Langley, P. 2009. Skill transfer through goal-driven representation mapping. *Cognitive Systems Research* 10(3):270–285.

Kuhlmann, G., and Stone, P. 2007. Graph-based domain mapping for transfer learning in general games. In *Proceedings of The Eighteenth European Conference on Machine Learning*.

Langley, P., and Choi, D. 2006. A unified cognitive architecture for physical agents. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Boston: AAAI Press.

Li, N.; Stracuzzi, D.; Langley, P.; and Nejati, N. 2009a. Learning hierarchical skills from problem solutions using means-ends analysis. In *Proceedings of the 31st Annual Meeting of the Cognitive Science Society*. Amsterdam: Cognitive Science Society.

Li, N.; Stracuzzi, D. J.; Cleveland, G.; Konik, T.; Shapiro, D.; Molineaux, M.; Aha, D.; and Ali, K. 2009b. Constructing game agents from video of human behavior. In *Procceedings of the fifth artificial intelligence and interactive digital entertainment conference*, 64–69. Menlo Park, CA: AAAI Press.

Liu, Y., and Stone, P. 2006. Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*.

Sacerdoti, E. 1977. *A Structure for Plans and Behaviour*. Amsterdam: Elsevier.

Sussman, G. 1975. *A Computer Model of Skill Acquisition*, volume 1 of *Artificial Intelligence Series*. New York: American Elsevier.

Taylor, M., and Stone, P. 2005. Behavior transfer for value-function-based reinforcement learning. In *Proceedings of AAMAS*.

Torrey, L.; Shavlik, J.; Walker, T.; and Maclin, R. 2006. Skill acquisition via transfer learning and advice taking. In *Proceedings of the European Conference on Machine Learning*.