Achieving Far Transfer in an Integrated Cognitive Architecture

Dan Shapiro, Tolga Könik, Paul O'Rorke

Computational Learning Laboratory, CSLI
Stanford University
Stanford, CA 94305 USA
{dgs,konik}@stanford.edu, pororke@csli.stanford.edu

Abstract

Transfer is the ability to employ knowledge acquired in one task to improve performance in another. We study transfer in the context of the ICARUS cognitive architecture, which supplies diverse capabilities for execution, inference, planning, and learning. We report on an extension to ICARUS called representation mapping that transfers structured skills and concepts between disparate tasks that may not even be expressed with the same symbol set. We show that representation mapping is naturally integrated into ICARUS' cognitive processing loop, resulting in a system that addresses a qualitatively new class of problems by considering the relevance of past experience to current goals.

Introduction

Computational systems tend to be expensive artifacts, in part because of the time and energy required to engineer their underlying knowledge. The desire to reduce this cost is the practical motivation behind a great deal of work on transfer, defined as the ability to employ knowledge acquired in one domain to improve performance in another. The long-term goal of transfer is to replace independent development efforts with an encode and reuse model.

Given this background, it is natural to study transfer in the context of agent architectures that pursue a general theory of cognition. In particular, if we can integrate the capacity for transfer into a system that performs general problem solving and execution, we will understand how to accomplish transfer across a wide variety of tasks that span perception, reasoning, and action.

We pursue transfer in the context of the ICARUS cognitive architecture (Langley & Choi, 2006), which employs hierarchical representations of concepts and skills, and provides capabilities for perception, inference, reactive execution and learning. While ICARUS offers a theory of performance and learning within a single domain, the transfer task requires knowledge acquisition, transport, and reuse between two distinct tasks. The challenge is to integrate those capabilities into ICARUS in a natural way.

Our prior work (Choi et al., 2007) demonstrated knowledge transfer among similar tasks, without new mechanism, by exploiting the generality inherent in ICARUS' acquired skills. These tasks were phrased in a single problem domain, and expressed with a common representation. In contrast, this paper reports on an extension to ICARUS that enables transfer between source and target tasks that bear little surface similarity and might not even be expressed with the same symbol set (sometimes called *far* transfer, a term we adopt here).

As before, we focus on acquiring and transferring structural knowledge in the form hierarchical skills and concepts. However, we introduce a new mechanism to perform far transfer, called representation mapping, which can be thought of as a goal-driven form of analogy; it ports skills and concepts associated with the subset of source goals that make sense in target terms.

We adapt ICARUS to support representation mapping by embedding it in the system's main cognitive processing loop. This results in an architecture that seeks opportunities for knowledge transfer each time it is exposed to a new task. This integration supports novel functionality. We claim that:

- The diverse capabilities of a cognitive architecture enable far transfer.
- ICARUS' assumptions naturally support this capacity.
- The capacity for far transfer lets the architecture solve a qualitatively new class of problems.

The following sections expand on this development and justify these claims. We begin by presenting examples of the far transfer task. Next, we discuss the mechanisms for far transfer, including an overview of ICARUS that addresses the first claim. We discuss integration issues then document the claim that far transfer produces qualitatively new functionality via a lesion experiment. We follow with a discussion of related work, and concluding remarks.

The Transfer Task

We study transfer via a collection of challenge problems phrased as source-target pairs. Here, the object is to solve the source problem, extract transportable lessons from that experience, and demonstrate that this knowledge improves the agent's ability to solve the target problem. We measure transfer by comparing the time required to solve the target problem, with and without exposure to the source.

We employ the General Game Playing (GGP) framework (Genesereth et al., 2005) to structure this task. GGP en-

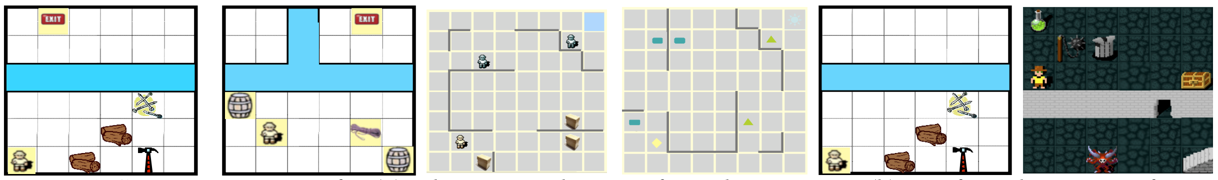

Figure 1. Source-target pairs for (a) a homeomorphic transfer task in Escape, (b) a reformulation transfer task in Wargame, and (c) a cross-domain transfer scenario from Escape to mRogue.

codes tasks (typically games) in a relational logic language that employs separate theory elements to describe legal moves, state transitions, and the initial state associated with game instances. GGP also enforces a careful evaluation model by presenting game rules at the same time as game instances. This requires agents to employ broad/general mechanisms for performance and learning, while constraining the role of background knowledge.

We study three types of far transfer tasks, characterized by the nature of the analogy within each source-target pair. *Homeomorphic* tasks admit a one-one correspondence between symbols denoting objects and/or relations, and allow elements in the source with no target corollary. Figure 1a gives an example drawn from a game called Escape, where the agent's goal is to direct the explorer to an exit. The source task requires nailing together logs to create a bridge over the river, while the target requires tying barrels together with rope. The relations for nailing and tying differ from source to target, and while the logs correspond to the barrels, the hammer has no target corollary. Note that the problem instances are distinct even given the source-target mapping. This makes transfer difficult because the agent must discover the mapping and transfer problem-solving knowledge in general form.

Reformulation scenarios consist of isomorphic problems created by systematically (but consistently) replacing all source symbols to obtain the target task. Figure 1b gives an example taken from 'Wargame', where the goal is to maneuver the soldier to the exit while avoiding/defeating enemies. The enemies actively seek the soldier and move twice as fast, but can become stuck by walls. Supply points contain weapons and ammunition. These scenarios are difficult because the agent must discover a deliberately obscured source-target relation.

Finally, *cross-domain* scenarios draw the source and target problems from different games. Figure 1c gives an Escape to 'mRogue' example (after the ancient text game), where the agent gets points for exiting the playing field, gathering treasure, and defeating monsters. These scenarios do not deliberately support analogies. However, all games occur on 2D grids and involve reaching a goal after surmounting obstacles by collecting and employing appropriate tools. Problems in this class are difficult because the agent has to identify both the symbol mapping, and the portions of the source solution that are preserved.

Adapting ICARUS to Support Far Transfer

Far transfer requires three steps: acquisition of knowledge in a source task, communication of that knowledge to a target task despite a representational divide, and reuse of that knowledge in the target context. This section discusses our technology for enabling far transfer in the context of the ICARUS architecture. We begin with a summary of the framework, focused on its key modules and their interaction, in keeping with the integration emphasis of this paper. We discuss ICARUS at length in other publications (Langley & Choi 2006).

The ICARUS Architecture

ICARUS is cognitive architecture in the tradition of work on unified theories of cognition (Newell, 1990). As such, it provides general-purpose representations and mechanisms for sensing, reasoning, and acting in a wide variety of circumstances. Figure 2 illustrates its key components. ICARUS represents knowledge in terms of hierarchical skills and concepts, which are stored in concept and belief memories. The concept memory contains structures that describe classes of environmental situations. The belief memory contains instantiations of these concepts believed to describe the current situation. The skill memory contains knowledge about how to accomplish goals in the abstract. Skills are hierarchical structures that link goals and concepts to subgoals or action.

Table 1 illustrates some of the skills and concepts required to control agent behavior in Escape. It includes a skill for accomplishing the top-level goal (called goal-1) that will cause the agent to construct a tool that will compromise an obstacle (e.g., a bridge to cross the water), bring the tool to the obstacle (which deploys the bridge in this domain), and then to proceed to the exit. Each of the subgoals is recursively defined. Here, AtExit is defined as a hierarchical concept that references the predicate 'location'. The location predicate also serves to index a skill (which moves the Explorer to that location).

ICARUS operates in distinct cycles, each of which starts by depositing observed objects into a perceptual buffer. These provide the material for conceptual inference, which adds implied ground literals to a belief memory. Next the architecture descends the skill hierarchy, starting from the highest priority top-level goal that is not satisfied. ICARUS retrieves a skill with a head that matches this goal and with conditions that match against current beliefs. The system applies this process recursively to this skill's first unsatisfied subgoal, continuing downward until it reaches a primitive skill with associated actions. ICARUS then carries out these actions in the environment, producing new perceptions that influence inference and execution on the next cycle. The architecture also includes a means-ends problem solver that deals with novel tasks for which no skills exist, as well as a learning process that stores the results of successful problem solving as new skills.

Table 1: Hierarchical skills and concepts for Escape.

```
((Goal-1   Explorer)
:start        ((obstacleType ?Obstacle ?ObstacleType)
               (location ?Obstacle ?Xo ?Yo)
               (canCompromise ?itemProperty ?ObstacleType)
               (not (destroyed ?Obstacle)))
:subgoals   ((property ?item ?itemProperty)
              ; construct an item that can overcome an obstacle.
              (holding ?item)
              (nextToExplorer ?X3 ?Xo ?Yo)
              (atExit)))
((atExit)
:relations   ((location explorer ?X ?Y)
              (location exit ?X ?Y)))
```

Details of the General Game Playing framework led us to diverge from the standard ICARUS architecture on a number of fronts. In particular, we replaced the bottom-up inference module with a top-down mechanism to handle the number of entities perceived in the environment. Because GGP provides fully modeled, deterministic domains, we replaced the reactive execution module with one that mentally simulated execution with N-step look ahead, using technology adapted from Nau et al.'s (2001) SHOP2 planner. We also replaced the ICARUS learning mechanism with a related technique (Nejati et al., 2006) that analyzes a solution trace, in this case found through lookahead search, to produce new hierarchical skills that solve the training problem and similar ones.

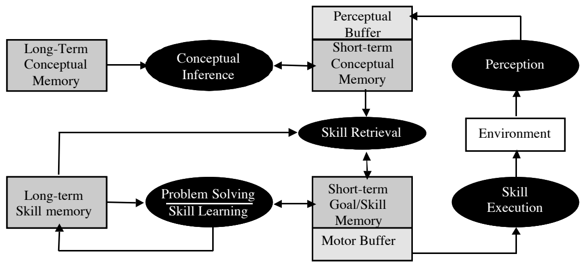

Figure 2. The ICARUS architecture.

Despite these differences, the resulting system relies on many ICARUS features to produce transfer effects, including the commitment to hierarchical knowledge, the distinction between concepts and skills, and indexing skills by the goals they achieve. The system begins by translating the GGP game specification into concepts and primitive skills, then invoking automatically generated exploration skills to search for a solution to the source problem. Upon finding one, the system acquires new hierarchical skills that generalize the solution that become the object of transfer to the target problem. After porting these concepts and skills, ICARUS uses them in the target domain, falling back on exploration behavior when guidance from source knowledge is exhausted.

In previous research, we demonstrated that ICARUS can transfer learned skills to target problems that involved the same predicates as the source. The current work introduces a mechanism for mapping representations that enables transfer into structurally similar settings that may involve different predicates. This capacity for far transfer clearly builds upon the assumptions about representation, performance, and learning described above.

Representation mapping

Transfer via representation mapping requires new process flow; it adds a comparison of source and target domains to the execution and skill learning steps described above. Representational transfer consists of two components. The *representation mapper* finds the correspondences between the source and target symbols and the *representation translator* uses those correspondences to translate source skills and concepts into skills and concepts in the target domain. We focus on the representation mapper.

The intuition behind the representation mapper is that transfer is possible if we can explain the source solution in target terms. In overview, the algorithm analyzes how the source problem is solved using source domain knowledge and replicates that same analysis from the perspective of the target domain. This process forges links between the source and target theories.

In more detail, given a pair of source and target problems, the representation mapper takes as input the *trace* of a successful source problem solution, the source and target *goals*, and the source and target *domain theories*. The solution trace consists of an initial state, a sequence of actions and the corresponding states resulting from these actions. The goals are concept instances that trigger successful termination of the tasks, and the domain theories are descriptions of domain dynamics encoded as ICARUS primitive skills (action models, including effects and preconditions) and concept definitions for percepts and abstract features of state. After analyzing the source solution in the context of source and target theories, the representation mapper outputs correspondences between the source and target predicates (and constants).

The first step of representation mapping is to analyze the source solution trace using the source domain theory to determine how the goal of the source problem is achieved. This process is similar to Nejati et al.'s (2006) analytical learning method, which explains a goal or subgoal either by decomposing its concept definition or by regressing it across a primitive skill that achieved it, producing new state descriptions that can be explained recursively.

Given an explanation of the source solution with source knowledge, the representation mapper constructs an explanation in the terminology of the target theory by asserting correspondences between the concept instances in the source explanation and target predicates as necessary to complete the derivation. The *correspondence support set* collects these assertions, and we check new assertions against this set for consistency. If no consistent correspondence exists, the search for an explanation backtracks to an earlier choice point. One correspondence often leads to another. For example, in bridging Escape and mRogue, (location explorer 1 2)

↔ (place hero ?X ?Y), implies the correspondences {location↔ place, explorer ↔ hero, 1↔?X, 2↔?Y}. This mapping includes relational symbols.

Table 2. A skill mapped from Escape to mRogue.

```
/* Source skill (learned) */
((combining ?item1 ?item2 ?combo)
 :start       ((property ?item3 hammer)
               (property ?glue  doesnail)
               (property ?item1 nailable)
               (property ?item2 nailable))
 :subgoals  ((holding ?item2)
               (holding ?glue)
               (holding ?item3)
               (holding ?item1)
               (do-combine ?item1 ?item2 ?glue ?combo)))
/* Correspondences */
      doesnail↔doestie, nailable↔tieable,
      (property ?item3 hammer) ↔nil
/* Target skill (output of mapping) */
((combining ?item1 ?item2 ?item3)
 :start       ((property ?item3 doestie)
               (property ?item1 tieable)
               (property ?item2 tieable)
               (newsymbol ?combo))
 :subgoals  ((holding ?item2)
               (holding ?item3)
               (holding ?item1)
               (do-combine ?item1 ?item2 ?item3 ?combo)))
```

Like other algorithms that find analogies, our representation mapping algorithm is guided by several constraints and heuristics. These fall into three groups; *structural*, *semantic* and *pragmatic* constraints, as described by Holyoak and Thagard (1989). *Pragmatic constraints* concern the purpose of analogy, and occupy a central role in our algorithm. Because the mapper is guided by the explanation of how the source goal is achieved, it only considers concepts relevant for the source solution and automatically abstracts away the rest. As a result, the system addresses homeomorphic tasks by removing unmapped preconditions and subgoals for source skills (implementing a form of task abstraction). Second, our algorithm uses a *structural* hard constraint that assumes a one-to-one mapping among predicates and constants found in source and target concepts. This constraint ensures that correspondence sets such as "1 ↔ ?X, 2↔?X" are disallowed. This assumption has been employed in previous systems (Falkenhainer et al., 1989; Holyoak & Thagard 1989) and has been suggested as a constraint in human analogical reasoning (Krawczyk et al., 2005). Finally, our algorithm is guided by a *semantic* constraint that prefers mapping between similar concept predicates. For a given support set *C*, we measure the degree of match between two predicates by comparing their definitions recursively, counting the shared symbols between the source and target and already constructed maps in *C*. Given multiple representation maps, the algorithm selects the one with the overall highest heuristic score.

As an illustration, consider the homeomorphic transfer scenario shown in Figure 1b. As part of skill learning, the architecture acquires the component skill for Escape shown at the top of Table 2. Next, representation mapping considers the target theory for mRogue to extract correspondences that relate properties of objects in the two domains. The representation translator completes the process by translating the source skill into target terms, as shown.

Integration

Research on cognitive architectures emphasizes economy and generality of mechanism as a necessary step en route to a general theory of cognition. This implies a desire to expand capability while minimizing the use of new representations and control structures. Integrating the capacity for far transfer required two changes to ICARUS in addition to the ones we have already described::

- We add an episodic memory to hold past experiences.
- We augment skill retrieval to seek opportunities for transfer in addition to immediately relevant skills.

In general, episodic memory supports various forms of macro and speed-up learning, as well as associative retrieval. It is also explored in other cognitive architectures, such as Soar (Laird et al., 1987). Our implementation is partial, and currently provides information necessary only for skill learning and transfer via representation mapping.

The change to skill retrieval has a more systemic effect. In addition to supporting transfer, it transforms the architecture into a cumulative learner that continuously seeks to adapt prior experience to its current context. This framework is much richer, and much more relevant to human experience, than the previous capabilities for performance and learning. This benefit is a natural consequent of supporting far transfer.

We note that our current implementation seeks the opportunity for transfer only on the first exposure to a new domain, and that we need to address many issues in cumulative learning. We expect that future versions of ICARUS will interleave skill retrieval via transfer with performance (after recognizing conditions that warrant the attempt).

Evaluation

We have made the claim that the capacity for far transfer lets the modified ICARUS solve a qualitatively new class of problems. This section justifies that claim in the context of a lesion study that determines the quantity of transfer due to (a) reuse of generalized skills without representation mapping (the lesion case) and (b) transfer of generalized skills with representation mapping (the non-lesion case). We measure transfer as the difference in agent performance on a given target problem with and without exposure to the corresponding source (normalized to facilitate comparison). We hypothesize that the ability to map skills across problem representations will qualitatively improve transfer in the non-lesion case relative to the lesion case.

In the experimental protocol, the non-transfer case (NTC) agent sees the target problem alone and must solve it by a base set of exploration skills. In contrast, the transfer case (TC) agent has the opportunity to solve the source problem, acquire knowledge from that experience, and make it available for transfer. The TC protocol involves several steps (consistent with the General Game Playing format):

1. download the source game definition and initial state
2. solve the source task via exploration,

3. learn hierarchical skills and concepts from the solution,
4. download the target game definition,
5. compute the best representation map,
6. instantiate the mapped skills and concepts in the target,
7. download initial state data for the target problem, and
8. solve the target problem using the mapped knowledge.

Only steps 7 and 8 above are timed, and reflected in the TC agent's performance score (though steps 4-6 were short by comparison). The TC and NTC agents both had access to the same exploration skills and supporting background knowledge to solve the target task. They act as a fallback for the TC agent if mapped knowledge does not suffice.

The lesion study examines transfer in 11 scenarios drawn from the problem classes discussed earlier and designed by an independent agency[1]. The first three are homeomorphic tasks (H), the next four are reformulation examples (R), and the last four are cross-domain transfer tasks. Figure 3 presents the results of the lesion study. Each data point averages across ten trials of the given target problem for the TC agent and the NTC agent, both. Values above zero indicate positive transfer (the transfer case solution is faster than the non-transfer case), values near zero indicate no transfer, and values below zero indicate negative transfer.

The results show that representation mapping (the non-lesion condition) produces positive transfer in 9 of 11 cases. The data contains some instances of very positive transfer. For example, the 0.93 score in CrossDomain-4 indicates that the agent solved the problem more than 10 times faster with transferred knowledge. The negative transfer in Wargame-R-1 is due to an incorrect representation map (found in 8 of 10 TC trials) applied to correct skills, while the effect in CrossDomain-1 is due to a partial map that creates actionable but misleading skills.

The results show that the architecture without representation mapping (the lesion condition) produces c. zero transfer in 9 of 11 cases. This effect is easy to understand. Absent a representation map, the system has no mechanism for relating source skills to target needs, so the TC and NTC agents both rely on exploratory behavior. This produces no net transfer. The main exception is Wargame H-2, where the terms that differed between source and target were bound to variables in ICARUS skills. This made source skills directly applicable for the TC agent, leading to positive transfer.

More broadly, we can draw two conclusions from the lesion study. First, representation mapping generates virtually all of the positive transfer observed in the 11 scenarios. More exactly, it provides the architecture with the capacity to exploit learned knowledge given the representational divide characteristic of far transfer tasks. Second, this effect appears robust across problem classes, as it explains all but one case of positive transfer.

[1] ICARUS was one of a set of agent architectures tested against numeric transfer goals in the DARPA Transfer Learning Program, via a formal evaluation process controlled by NRL.

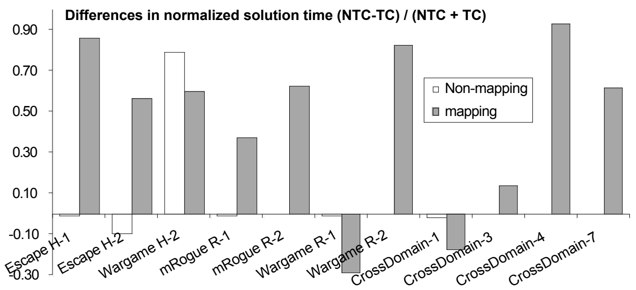

Figure 3. A lesion study showing the impact of representation mapping on transfer

Finally, we note that the scenarios were designed to afford positive transfer. This is least true in the cross-domain cases where the source-target relationship is unconstrained. However, the homeomorphic and reformulation scenarios obey what might be called a *constant content* assumption: solutions for the source *can* solve the target task, given the correct mapping among symbols. (This is evident in the reformulation case, while the transformations defining the homeomorphic class admit no new information in the target, implying source solutions are preserved). Experiments with even more disparate source and target tasks would clarify the limits of the representation mapping algorithm.

Related Work

There is an active research community studying transfer in cognitive architectures. The DARPA Transfer Learning Program has motivated a good deal of this work by the same far transfer tasks explored here. Interestingly, research on Soar and Companions has employed very different learning mechanisms (i.e., chunking vs. theory revision and directed learning) but broadly similar transfer techniques. Both map symbols via some form of structural analogy; condition and rule matching in the case of Soar (Laird et al., 1987), and the Structural Matching Engine (SME) in Companions (Klenk & Forbus, 2007). Our approach is similar to SME (Falkenhainer et al., 1989) in that it creates one to one predicate mappings. Liu and Stone (2006) also employ structural analogy to accomplish value function transfer across a representational divide.

Our approach also bears similarities to earlier work on analogical problem solving. VanLehn and Jones' (1993) CASCADE used a form of analogical search control to guide top-down chaining through an AND-OR tree. Their system did not map across representations, but it did store semi-persistent mapping that it reused later in a given problem. Jones and Langley's (2005) EUREKA also used analogical search control, in this case to direct a means-ends problem solver that spread activation through stored subproblem decompositions. This system exhibited limited ability for cross-domain transfer, but only when provided with connections between predicates.

Our research on structural analogy is distinguished by its strong emphasis on pragmatic constraints, imposed by our use of explanatory source analyses. Kedar-Cabelli's (1985) early work also employed explanation in the service of analogy, using a stored proof to show how the characteristics of objects satisfy a given purpose (e.g., a styrofoam cup lets one drink hot liquids just as does a ceramic mug with a handle). Her system introduced the use of goal regression to guide analogical reasoning, but it did not address far transfer or support representation mapping. Holyoak and Thagard (1989) advocate the use of pragmatic constraints as well, but their ACME algorithm assumes prior pragmatic values for predicate matching and leaves open the question of how they arise.

Concluding Remarks

This paper has discussed the integration of a new capability for knowledge transfer into an integrated cognitive architecture. It operates by explaining solutions found in one domain using the vocabulary native to another, where the symbols describing the domains need not be shared. We have shown that the new mechanism, called representation mapping, enables qualitatively new types of behavior, and that it is enabled, in turn, by ICARUS' assumptions about representation, performance, and learning. We demonstrated far transfer among isomorphic tasks, homeomorphic tasks, and in less constrained cross-domain scenarios. More importantly, integrating far transfer generalized the architecture's previous abilities to encompass cumulative learning and the continuous adaptation of past experience to current goals, which increases the richness of the framework.

Our integration of representation mapping with Icarus is not complete. Our future work will extend the system's ability to support more complex mappings between source and target domains. We also intend to address the tradeoff between performance and representation mapping in a real-time environment, along with retrieval of previous source experiences for a given target problem. In addition, we plan to improve our representation mapping algorithm to use feedback from applying transferred skills in the target domain. Together, these should produce a more adaptive architecture that can reuse its previously acquired knowledge in a robust way.

Acknowledgments

We thank Dongkyu Choi, Tom Fawcett, Ugur Kutur, Pat Langley, Negin Nejati, Chunki Park, and Nate Waisbrot for their contributions to this work. We gratefully acknowledge DARPA sponsorship through the Transfer Learning Program, grant FA8750-05-2-0283.

References

Choi, D., Könik, T., Nejati, N., Park, C., and Langley, P. 2007. Structural Transfer of Cognitive Skills. In *Proceedings of the Eighth International Conference on Cognitive Modeling,* 115-120. Oxford, UK:Taylor & Francis.

Falkenhainer, B., Forbus, K. D., and Gentner, D. 1989. The Structure-mapping Engine: Algorithm and Examples. *Artificial Intelligence* 41(1): 1-63.

Genesereth, M. R., Love, N., and Pell, B. 2005. General Game Playing: Overview of the AAAI Competition. *AI Magazine* 26(2): 62-72.

Holyoak, K. J. and Thagard, P. 1989. Analogical Mapping by Constraint Satisfaction. *Cognitive Science* 13: 295-355.

Jones, R. M., and Langley, P. 2005. A Constrained Architecture for Learning and Problem Solving. *Computational Intelligence* 21(4): 480-502.

Kedar-Cabelli, S. 1985. Purpose-Directed Analogy. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, 150-159. Hillsdale, N.J.: Lawrence Erlbaum.

Klenk, M. and Forbus, K. 2007. Cognitive Modeling of Analogy Events in Physics Problem Solving from Examples. In *Proceedings of the Twenty-Ninth Meeting of the Cognitive Science Society*. Nashville, TN.

Krawczyk, D. C., Holyoak, K. J., and Hummel. J. E. 2005. The One-to-one Constraint in Analogical Mapping and Inference. *Cognitive Science* 29: 29-38.

Laird, J. E., Newell, A., and Rosenbloom, P. S. 1987. Soar: An Architecture for General Intelligence. *Artificial Intelligence* 33: 1-64.

Langley, P. and Choi, D. 2006. Learning Recursive Control Programs from Problem Solving. *Journal of Machine Learning Research* 7: 493-518.

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*,421-426, Menlo Park,CA: AAAI.

Nau, D., Muñoz-Avila, H., Cao, Y., Lotem, A., and Mitchell, S. 2001. Total-Order Planning with Partially Ordered Subtasks. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 425-430. San Francisco, CA: Morgan Kaufmann.

Nejati, N., Langley, P., and Könik, T. 2006. Learning Hierarchical Task Networks by Observation. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, 665-672. New York, NY: ACM Press.

Newell, A. 1990. *Unified Theories of Cognition*. Cambridge, MA: Harvard Univ. Press.

VanLehn, K., and Jones, R. M. 1993. Integration of Analogical Search Control and Explanation-Based Learning of Correctness. In Minton, S. ed. *Machine Learning Methods for Planning*, 273-315. Los Altos, CA: Morgan Kaufmann.