Center for Cognitive Architecture
University of Michigan
2260 Hayward Ave
Ann Arbor, Michigan 48109-2121

TECHNICAL REPORT
CCA-TR-2007-02

Investigating Transfer Learning in the Urban Combat Testbed

Investigators

Nicholas A. Gorski
John E. Laird

10 September 2007

Abstract

We have developed a complex agent capable of transferring learned declarative knowledge between tasks in order to improve performance in a rich domain developed to test transfer performance (the Urban Combat Testbed). We present and discuss our experiments and results which demonstrate successful transfer learning. To better understand the underlying sources of transfer in this domain, we empirically evaluate the types of knowledge that contribute to the observed improvement in performance due to transfer, and discuss the implications of our analysis for general transfer learning research.

Table of Contents

1. Introduction

In the field of machine learning, the study of transfer learning has recently received widespread attention. Machine learning techniques making use of transfer have the advantage of applying previously learned knowledge in order to bootstrap learning and improve performance in a new task, whereas approaches not making use of transferred knowledge must start learning from scratch on each new task. Thus, the study of transfer learning is important in order to improve the real-world performance of existing machine learning techniques.

However, the study of knowledge transfer is typically limited to simple domains, often toy problems designed to explicate the underlying mechanisms of a given algorithm rather than study the transfer performance of a particular approach. When knowledge transfer is investigated in complex domains, the underlying sources of knowledge that give rise to improvements in performance are not typically identified.

Furthermore, in literature discussing transfer learning it is typically sufficient to demonstrate that an approach accomplishes transfer rather than demonstrate to which degree transfer is achieved; metrics for measuring the degree of transfer performance or that allow the fair comparison of transfer performance across learning methods and domains are not yet well agreed upon in settings involving complex intelligent agents.

In order to help categorize the different types of transfer learning that can be demonstrated, DARPA (2005) identified ten distinct levels of transfer, each detailing a specific relationship between a pairing of tasks. Each level can describe either the type of internal reasoning that an agent must be capable of in order to demonstrate an improvement in performance due to transfer, or a transformation of the problem structure itself. Using the latter interpretation, these levels are currently being used in several domains within the DARPA Transfer Learning program, such as the Urban Combat Testbed (UCT).

The motivation for our research was to both demonstrate different types of transfer learning, as well as to understand the sources of transfer performance in our domain. This involved identifying what is learned in the UCT as well as constraints imposed on agent design both by the UCT and our research program's goals. Given an agent that was able to accomplish transfer in the UCT, it also involved performing an empirical analysis of the types of knowledge that led to improvements in performance due to transfer.

We begin by describing the UCT and details of the domain that are relevant to our research. We then present our transfer learning agent, UCTBot, and describe its implementation. Given our domain and an agent to perform in it, we describe the specifics of how transfer learning is studied in the UCT. We then describe our methodology, results, and conclusions, and conclude this document with a discussion of related work.

2. A Motivating Domain: The Urban Combat Testbed

The UCT (University of Texas at Arlington, 2006) is a software suite consisting of a collection of scenarios designed to test transfer along with the game engine that supports their execution, implemented as a first-person shooter video game built using the open-source Quake III Arena engine.

The UCT scenarios are a collection of tasks specifically designed to test transfer performance in each of the first eight DARPA levels. Five different scenarios have been implemented at each level of transfer, each testing that level's behavior using a different configuration of components. A scenario consists of a pairing between two tasks, the source and target, related to each other in such a manner that an agent reasoning over knowledge gained in the source task can solve the target with a better score than an agent not exposed to the source task. For more details regarding the design and implementation of the scenarios, see Youngblood, Shapiro and Holder (2006), UT Arlington (2006), and Shapiro, Fawcett and Langley (2006).

2.1. UCT Details

The goal of each scenario is to find a single improvised explosive device (IED) as quickly as possible and defuse it, which reduces to the familiar "first-to-flag" mission common to the first-person shooter game genre. The primary challenge in all scenarios is to find a path to the goal while navigating around obstacles and interacting with objects.

Whereas a human player perceives the UCT environment visually via a computer monitor, an agent's perceptual state is a collection of symbolic and numeric information. Space in the domain is partitioned into a set of convex polyhedrons, called *regions*; the agent can perceive the coordinates defining the region that it is currently in, the unique symbol identifying its current region, and the available *gateways* connecting regions. The topology remains constant within a scenario, but can change from scenario to scenario within a level. While the UCT attempts to provide the same types of information to an agent that a human would perceive, it provides topological information to the agent in way that is very different from how a human would perceive it.

Gateways are perceived as convex polygons which are the intersection of two regions, and consist of a set of vertices as well as the symbol identifying the destination region. Crossing a gateway requires one of three possible actions: walking, jumping, or ducking. Additionally, some gateways are impassible. All gateways are unidirectional (i.e. a gateway connecting region A to region B does not imply a gateway from B to A)

The agent also perceives obstacles contained within in its current region. Obstacles may be surmounted by jumping, ducking, climbing, punching or shooting. Like gateways, some obstacles are insurmountable and must be avoided entirely.

Although obstacles are essentially topological information, they are differentiated from a task's topology by their perceptual presentation to the agent. Topological knowledge is presented as static knowledge and encoded as topology by the agent's percepts; obstacles, on

the other hand, can be presented both as static (e.g. a tall wall) and dynamic (e.g. a window that can be broken) and are encoded in the perceptual interface specifically so that an agent may know that it is different from other topological knowledge, and must thus be reasoned about differently.

Additionally, each obstacle is annotated with a set of tags drawn from the Cyc knowledge-base (Matuszek et al., 2006), such as "#$Obstacle-TallWRTHuman", "#$HighChainLinkFence", and "#$LowSturdyOutdoorWall". In the UCT, agents are currently prohibited from using provided semantic knowledge to infer meaning from these tags; rather, they are used to find similarities and differences between objects in the domain. For example, if an agent learns that it cannot jump over an obstacle tagged with "#$Obstacle-TallWRTHuman", it may assume that another, different obstacle also tagged as "#$Obstacle-TallWRTHuman" should not be jumped over either. In the future, however, these annotations may facilitate more complex inference.

Doors, buttons, keys, ammunition, and some types of obstacles are perceived as dynamic entities. Along with the IED, agents can perceive these dynamic entities whenever they are visible from its current position (calculated by ray casting), and is similar to human perception of these objects (although it is calculated without respect to the agent's orientation).

Also provided to a player is the general location of the IED in the current task. For agents, this is communicated by the symbol of the region containing the IED; for humans, this information takes the form of an image depicting a top-down view of the IED and its immediate surroundings. If the agent has perceived the region denoted by the given symbol previously, it is able to recall the coordinates of that region and possibly plan a route to it as well; otherwise it must explore in order to discover the location of the indicated region.

Performance in the UCT is measured by the score, which is the number of seconds spent in the task plus penalties. Damage from electrical barriers is 100 points per second, while damage from drowning is 30 points per second. Additionally, each gunshot fired results in a penalty of 1 point. Players in the UCT are limited to 300 seconds of real-time per task; note that due to damage penalties, the score can potentially be much greater than 300 points at the conclusion of a task.

2.2. Learned and Provided Knowledge in the UCT

The scenario designers specify that certain knowledge would be provided to agents, while agents are required to learn other types of knowledge (Shapiro, Fawcett and Langley, 2006). Controlling the types of knowledge that would be learned in the domain allows investigators to test for specific types of transfer capabilities.

In the UCT, it is assumed that agents are provided with the necessary procedural knowledge for both reasoning over learned knowledge and performing in the environment. Certain background knowledge for each task is also provided, in order to limit the amount of search that each task requires. This includes the symbol representing the region containing the IED,

whether an agent must enter a building in order to find the IED, and if necessary the symbol representing the building containing the IED.

Knowledge that must be learned relates to:

- the topology of each task,
- the locations and properties of static and dynamic obstacles and objects,
- and the action models of gateways, obstacles and objects.

The scenarios were designed such that for each level of transfer learning tested, the transformation specifying the reconfiguration of objects and obstacles between the source and target tasks is representative accurately reflects the level of transfer being tested (rather than the alternative interpretation of the transfer learning levels: that they specify an internal reasoning capability possessed by the agent).

3. A Soar Agent to Perform in the UCT: UCTBot

In order to examine transfer in the UCT, we required an agent capable of performing in the domain. The UCT is a rich, complex environment containing a large continuous state space as well as a large noisily-deterministic action space. We used the Soar cognitive architecture in order to create an agent capable of performing in such an environment.

Soar is a general cognitive architecture that incorporates many of the mechanisms believed to underlie human intelligence (Lehman, Laird and Rosenbloom, 2006). In Soar, all declarative knowledge is stored in working memory, and production rules that test and elaborate upon this knowledge are stored in a long-term procedural memory. These production rules are responsible for proposing, retracting and selecting operators, which in turn can add and remove working memory elements or take actions in the environment. Inconsistent or incomplete knowledge leads to an impasse, creating a sub-goal in which additional reasoning can take place, facilitating the hierarchical decomposition of agent behavior.

We implemented an agent designed to perform in the UCT using Soar, named *UCTBot*. UCTBot's behavior is decomposed into a hierarchy of actions and follows the Soar convention of pursuing a single top-level goal at a time, creating sub-goals as necessary when impasses in the agent's internal reasoning are reached. We were able to learn from previous work implementing agents using Soar to operate in first-person shooter video games such as Unreal Tournament (Wray et al., 2004), Haunt (Magerko, 2006; Magerko et al., 2004), as well as a preliminary version of the UCT (Gorski and Laird, 2006). UCTBot is a sizable agent consisting of over 600 rules, more than 530 of which were written specifically for the work presented in this document.

3.1. Design Constraints

In order to create an intelligent agent to act in the UCT, we identified the following design constraints to guide our development:

1. The agent should not change between tasks; rather, the exact same agent should achieve transfer on all levels without any modifications to source code. This preserves generality in the agent's reasoning and behavior, which was an important goal of our research program.
2. Any constraint required by the designers of the UCT scenarios (such as requiring agents to learn action models over gateways and obstacles) must be honored. In general, the agent must follow the spirit of the testbed, which facilitates fair comparisons between agents.
3. The UCT presents each task only once, which means that the agent must learn sufficiently in a single example of the source task to achieve significant initial transfer in the target (and thus demonstrate *one-shot* transfer performance).
4. The agent's performance must be optimized for both the transfer and non-transfer conditions in order to permit a fair comparison between conditions; while agents are not aware of whether the current task is a source or target, performance must be optimized for all levels of available knowledge.

5. The agent may only use percepts provided by the various UCT processes. This allows for fair comparisons to be made to other agents using the UCT.
6. To achieve optimal performance, the agent must perform in real-time.
7. The agent should be explicit about what is being transferred between tasks, so that types of knowledge contributing to transfer can be easily studied.
8. The agent must be robust over taking actions in the UCT (which are deterministic but noisy) and not crash or perform undesirable behaviors.

Some of these constraints were imposed by the testbed, while we identified others. Given these constraints, we implemented UCTBot as an agent capable of acting within the domain.

3.2. Details of the UCTBot Implementation

The agent's primary behavior is movement through the domain; this movement is controlled by two top-level operators, **locate-ied** and **goto-ied**. The **locate-ied** operator is selected when the agent is searching the problem-space for the IED; **goto-ied** is selected when the agent has either observed the IED or has knowledge pertaining to the area containing the IED (which could have either been learned in the current task or transferred from a previous task). While both operators control movement and rely on many of the same sub-operators, they represent conceptually different goals. Figure 1 provides a partial outline of the operator decomposition of our agent.

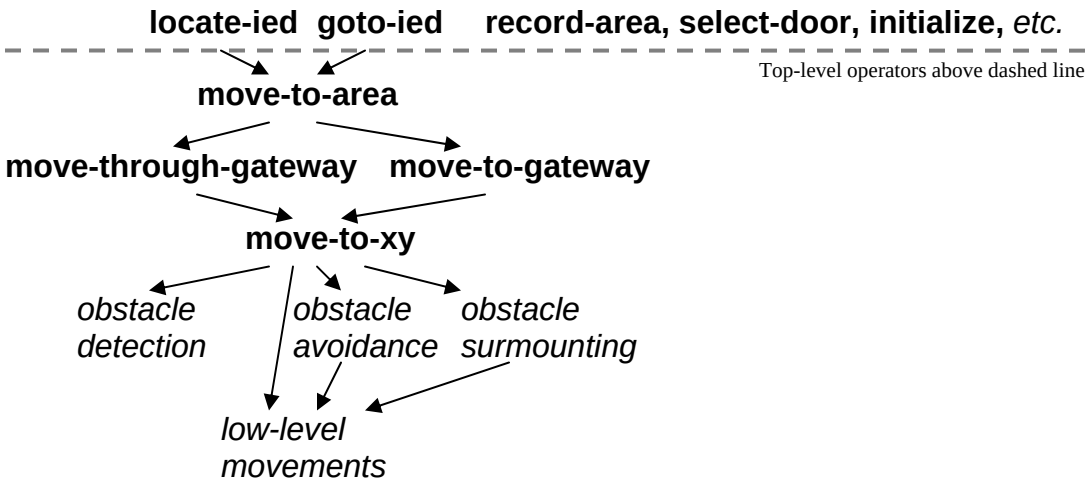

Figure 1: Hierarchical decomposition of important operators in the UCTBot.

As mentioned previously, space in the UCT is partitioned into regions (which are convex polyhedrons). As the spatial reasoning capabilities in UCTBot are limited, UCTBot projects these regions to a 2-dimensional plane; these projections are guaranteed by the testbed to be

convex polygons and are called *areas*.[1] Although the agent reasons over space in two dimensions, it is fully capable of navigating within the multi-story buildings of the UCT.

Inter-area movement is controlled by the **move-to-area** operator, which controls the agent's movement from one area to an adjacent area. This behavior is decomposed into two operators: **move-to-gateway** and **move-through-gateway**. The **move-to-gateway** operator controls aligning the agent within a gateway, while **move-through-gateway** controls moving the agent through a gateway into an adjacent area. Each of these operators rely on the **move-to-xy** operator to control low-level movement, which in turn is decomposed into the lower-level action primitives executed in the environment, such as turning, thrusting, strafing, jumping etc. Other intra-area movement also uses **move-to-xy**.

When UCTBot desires to move to an area that is not adjacent to its current area, it initiates a model-based look-ahead search using the default Selection rules (Laird, 2006). In this search, the agent moves from area to area in its internal model, using all available information about the goal area to heuristically guide its search through its topological model. In order to balance exploration of the map with the agent's exploitation of its available knowledge, the agent will move to any unrecorded area that is expanded in the course of its internal search.[2] In this manner, the agent may continue to explore even when a known (or transferred) route is available; however, often this exploration will lead to a better route to the goal area. When the agent reaches a goal area in its internal search, either the area containing the IED or an unrecorded area, a result is returned from the sub-goal to resolve the impasse, and the agent initiates the first action that led to that resolution.

When performing in successive tasks, the agent's internal map grows to a considerable size, and at times the agent's internal search through its internal map can require hundreds of decision cycles. Optimization of the rules used in the internal search combined with the real-time performance of Soar on the high-end machines used in these simulations results in good performance of the system. When the agent has a large internal map, the agent's internal search can potentially account for several seconds in a given task; when the agent has a small internal map relative to the topology of the task, the internal search accounts for less than a second. As the internal search is used more often when the location of the IED is known, time spent performing internal search would serve to negatively impact transfer performance. Importantly, the time spent performing an internal search is much less than time spent navigating in the domain. See section 3.3 for a more detailed explanation of UCTBot's path planning strategy.

An agent that is able to robustly plan movement over and around obstacles would require significant spatial reasoning capabilities. Instead, UCTBot uses a reactive approach to

[1] Explicitly, *regions* refer to the set of perceptual information provided by the testbed and correspond to 3-dimensional convex polyhedrons; *areas* refer to the 2-dimensional projections, and are the conceptual representation of space used within UCTBot.

[2] An agent initiates an internal search with a specific goal in mind. While it may seem that it is being sidetracked by continuing exploration even when it knows a route to the IED, it is logical to visit unrecorded areas that are nearer to the IED rather than traverse a known route that is likely suboptimal. In the UCT domain, the majority of areas are not dead-ends; furthermore, empirical evidence strongly suggests that this is a good strategy, as seen later in this document.

surmounting and avoiding obstacles. The agent moves through the world and detects when its movement is impeded; it then reasons over its available knowledge to determine which obstacle or gateway is blocking its movement.

Upon encountering a blocking obstacle, the agent checks its knowledge base to see if that type of obstacle has been previously encountered. If the blocking obstacle (or gateway) has not been previously encountered, it begins a sequential series of actions and records their results in its knowledge base. If it has been previously encountered, it retrieves the correct action from its knowledge base and begins executing it. This knowledge may indicate that the agent has no action available to surmount the obstacle, in which case it attempts to avoid it (possibly encountering a different obstacle in the process). Information contained in the knowledge base is updated if UCTBot later determines that it is inaccurate, which can be caused by noisy perceptual information and faulty reasoning. All of these behaviors take place within the **move-to-xy** sub-goal.

Knowledge regarding the topology is stored by the top-level operator **record-area**, which stores information about the current area to the map knowledge base. Other top-level operators maintain other declarative knowledge, such as the location of doors, keys, ammunition and the like.

3.3. Path Planning Implementation

The primary task in the UCT is navigation, and thus the UCTBot's ability to plan paths is the most essential component of its behavior. As mentioned previously, the UCTBot must robustly find efficient routes with varying amounts of topological knowledge (from no knowledge to complete knowledge and all degrees in between). Our approach, which utilizes a look-ahead search without caching, is not optimal but does generate behavior that is reasonably intelligent and satisfies our design constraints (section 3.1).

UCTBot's path planning strategy is an optimistic internal look-ahead search. When it needs a path, it considers all of the traversable gateways leading out of its current area (where traversable means that UCTBot has not yet found it to be impassible). The distance between each gateway and the goal is calculated, and UCTBot conceptually moves itself to the region on the other side of the gateway. It continues this search until it reaches the goal, an unexplored area, or a dead-end. If UCTBot finds itself to be (conceptually) in a dead-end region, or one containing no gateways to regions that are not already on the path, then it marks the gateway that led to that region as a failure and considers the next nearest gateway leading out of the previous region. UCTBot terminates the search and begins following the found path when it finds either the goal region or an unexplored region. UCTBot's search is optimistic, then, in that it explores previously unvisited areas that are nearer to the goal region than other, previously explored alternatives that may guarantee a path. Due to wide, highly-connected areas in the UCT, this is a good strategy for that domain.

Consider the hypothetical configuration of areas in Figure 2. UCTBot initially can move to either two known areas or two unexplored areas. The gateway nearest to the goal leads to area A, so UCTBot conceptually moves to area A. There, it can explore either the known area to the South or the unexplored gateway to area B. UCTBot optimistically chooses to

explore area B, in the hopes that it will find a shorter path to the goal. It thus returns the partial path as success, and physically moves to region A. Once it reaches a different area (in this case, region A), it begins a new look-ahead search (as it does not cache paths). In its new search, it would decide to explore region B and then physically move there. Once in region B, the agent would find the gateway that leads directly to the goal and move immediately to the goal area.

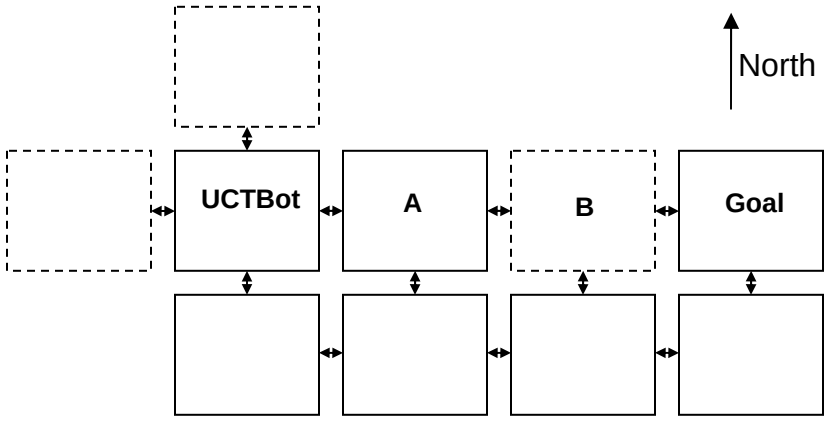

Figure 2: Hypothetical UCT topology. Arrows represent bidirectional gateways, and unexplored areas are outlined with a dashed line.

Note that although areas in the UCT vary in size, UCTBot does not weight areas by size or distance required to traverse an area. While our approach does not guarantee a shortest path, the topology of the scenarios is such that UCTBot almost always finds a near-optimal path. In order for UCTBot's search algorithm to return sub-optimal paths when searching over complete topological knowledge, regions must be connected in such a manner that a visited region leads to a suboptimal path due to a maze-like configuration. In the UCT, however, regions typically have high degrees of connectivity and few dead-ends or corridors, which means that UCTBot typically finds near-optimal paths.

Figure 3 shows a configuration of areas for which UCTBot would find a poor path. UCTBot would first explore the area to the South of its starting location, and then follow a long path to the goal region through a series of areas with only two gateways. The optimal path, however, is to initially move West and then follow the shorter path to the goal.

One of our design constraints (section 3.1) is that UCTBot must be robust over the noisily-deterministic actions of the UCT. Due to the narrowness of many of the regions in the UCT, oftentimes UCTBot would intend to move to one region only to find itself in a completely new, unexpected region. UCTBot's path planning behavior is robust over these frequent occurrences as it does not cache paths but rather initiates a new look-ahead search every time it finds itself in a new area. This phenomena led in part to our choice of path planning strategies.

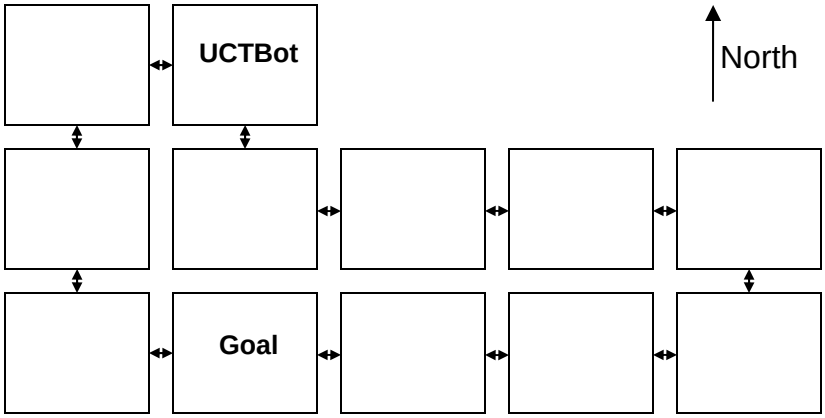

Figure 3: A hypothetical configuration of regions in which UCTBot would perform suboptimally.

Although not currently implemented in UCTBot, our path planning approach would be improved by adding a caching mechanism to the look-ahead search. Soar's general learning mechanism, *chunking* (discussed in section 9.2 below; see also Lehman, Laird and Rosenbloom, 2006), could be applied in order to compile much of its search into a single production, saving decision cycles and thus execution time during its search. Similarly, one could cache paths as declarative structures in working memory (for an example of this approach, see Gorski and Laird, 2006). We found that UCTBot's look-ahead search accounted for a very small portion of its overall performance in the domain and thus such optimization was unnecessary for our (pragmatic) purposes.

4. Transfer Learning in the UCT

Having described the UCT and an agent designed to perform in the testbed, it is useful to consider the possible sources of transfer within the domain. Shapiro, Fawcett and Langley (2006) identify the two sources of knowledge transfer within UCT: the topology of each scenario and the action models required to surmount obstacles and gateways. Intuitively, this knowledge corresponds to what agents are required to learn in the UCT.

Agents must also reason over transferred knowledge. Shapiro, Fawcett and Langley (2006) state that the scenarios "must evoke the type of transfer called for at each level in an intuitive way" (p. 4). The authors' claim is that while the knowledge being transferred may take the form of either procedural or declarative knowledge, the complex transfer behavior is captured by the reasoning that an agent must perform over this transferred knowledge. However, we argue in this document that agents employing simple reasoning strategies and transferring declarative knowledge without any transformation can achieve satisfactory transfer performance in the UCT.

4.1. Transfer Learning Strategy using Soar

In a Soar agent, the transferred knowledge discussed above is represented as structured, symbolic, declarative knowledge. As UCTBot explores a scenario, it learns information relating to regions, their contents, and their connectivity and stores it in working memory. As the Soar agent learns action models over gateways and obstacles, these structures are further annotated with symbolic knowledge indicating the correct action to take when encountering each item. This collection of declarative knowledge is then be transferred from task to task.

As the agent explores a task, it accumulates knowledge relating to that scenario's topology as well as its action models. It uses a look-ahead search, guided by several heuristics, to find routes through the environment as needed. When the agent encounters an obstacle, it retrieves the necessary action to surmount it; likewise, when an agent attempts to cross a gateway, it retrieves the action that will be necessary. The agent reasons over this accumulated knowledge in the exact same manner regardless of whether it was accumulated in the current task or was transferred from a previous task.

Although the implicit claim of the scenarios in the UCT is that they require specific reasoning capabilities, we made a conscious decision not to enable our agent with these behaviors. Our hypothesis was that since scenarios in the UCT involved only the structural reconfiguration of components, an agent lacking complex reasoning skills might still demonstrate satisfactory transfer performance. Were our agent unable to demonstrate transfer, we would then revisit our design decision and perhaps enable it with more advanced reasoning behaviors.

Earlier, we identified three different types of knowledge that must be learned in the UCT (section 2.2). In the UCT, two types of knowledge remain constant between the source and target tasks (and thus transfer perfectly): the topology of the domain, and gateway/obstacle

action models. The third type, the locations of static and dynamic obstacles and objects, changes between tasks. UCTBot transfers the first two types of knowledge but does not transfer the locations of obstacles between tasks, as obstacle configurations can change dramatically between source and transfer tasks.

Were the agent to be endowed with more advanced reasoning behaviors (see above), we would consider transferring obstacle locations. For a simple agent, it is unclear how such dynamic information could lead to successful transfer, given our first design constraint (section 3.1).

4.2. Research Questions

Our primary research goal is to demonstrate that agent performance improves for the transfer condition. Our primary hypothesis is that an agent with simple reasoning mechanisms will be capable of demonstrating transfer performance in the UCT scenarios.

It is also an open question as to which type of knowledge will contribute the most transfer performance. We hypothesize that time spent navigating through the environment will dominate time spent exploring action models of gateways and obstacles, thus contributing more to transfer performance. It remains to be seen how the topology is most useful to agents: for route finding, or for guiding the agent through unexplored terrain when it has already seen a distant goal region and thus can use its coordinates to guide its search.

5. Methodology

The UCT contains forty different scenarios designed to test a variety of types of transfer learning. These scenarios are grouped according to the DARPA transfer levels, with five scenarios testing each of the first eight levels (levels nine and ten have not yet been implemented).

We evaluated the UCTBot's transfer performance using the UCT scenarios. Each scenario consists of two tasks, a source and a target, where the structures of the two tasks are related according to the definition of the DARPA transfer level that they are designed to test. To evaluate transfer performance, the agent was presented with the source task followed by the target task. After completing the source task, the agent had learned knowledge, some of which could apply to the target task. In contrast, an agent operating under control conditions was only presented with the target task and had no prior knowledge regarding either task (Figure 4). After gathering data for both the transfer and control conditions, the transfer performance of the UCTBot was evaluated. UCTBot performs no transformation over transferred knowledge (it transfers perfectly).

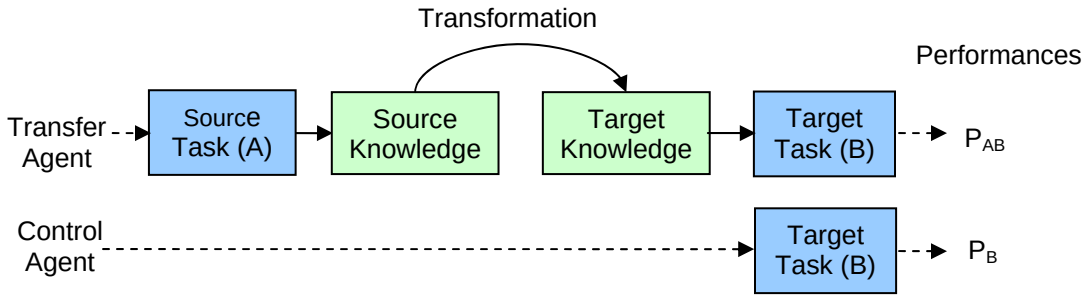

Figure 4: Methodology for comparison between transfer and control conditions.

Note that although each level that was tested had five scenarios implemented in the UCT, some of those scenarios had features that made it impossible for our agent to solve. In those cases, we evaluated our agent on the scenarios that our agent was able to solve –four of the five except for level seven in which three of five were evaluated.

To measure the transfer performance of the UCTBot at a given level, multiple trials of each scenario were performed and the mean performance for each of the transfer and control conditions was taken. The Calibrated Transfer Ratio (CTR; discussed below) was then measured for each scenario and the mean CTR was measured for each level, giving an indication as to the transfer performance that the agent achieved for each of the different DARPA levels of transfer.

5.1. Sources of Transfer

In order to more clearly understand how the UCTBot accomplishes transfer in the testbed, we investigated how different types of knowledge contribute to transfer. We earlier identified three different types of knowledge that must be learned in the domain, and discussed that we

would only be investigating the transfer of two of those categories: topological knowledge of the task and action models over gateways and obstacles (section 4.1).

Our agent makes use of topological knowledge in two different ways. During path planning, the agent examines the connectivity of regions that it has knolwedge of in order to find paths to some remote desired region. This desired region can either be an unexplored region that the agent has observed a gateway leading to, or the region containing the goal (given to the agent at the beginning of the scenario by the testbed).

We thus experimented with removing different pieces of knowledge from the information that a trained agent transferred between the source and target tasks (see Figure 4). We experimented with three combinations, allowing the agent to transfer only

- action models,
- all topological knowledge,
- and action models and the coordinates of the goal region that could be used to direct the agent's path planning search.

Although it would have been best to explore transferring every combination of knowledge types, we were limited by the amount of real time that was required to gather results.

5.2. Transfer Learning Metrics

In order to evaluate transfer performance, it is necessary to compare the performances obtained by both transfer and control conditions. One common approach is to take a ratio of performance using transferred knowledge to performance without transferred knowledge, known as a *transfer ratio* (Gorski and Laird, 2006; Mehta et al., 2005; Morrison et al., 2006). While such ratios are informative, they cannot be fairly compared across tasks or domains due to their inability to control for task- and domain-dependent scaling factors. This would render any cross-task comparisons within the UCT as invalid.

In order to allow for valid comparisons across tasks (and domains), it is possible to calibrate a ratio of transfer and control performances with the optimal performance. This new metric then measures the ratios of how close two performances came within optimal, rather than simply the ratio of two performances. The Calibrated Transfer Ratio (CTR) is thus defined as

$$1 - \frac{P^* - P_{AB}}{P^* - P_B}$$

where P^* is the optimal performance on a given task and P_{AB} and P_B are the performances for the transfer and control conditions respectively. The CTR is interpreted as the percentage of possible improvement that was achieved due to the change in conditions (in our case, transferred knowledge from the source task, although the CTR can be used to measure changes in performances from sources other than transferred knowledge).

While the CTR requires some measure of optimal performance for a given task, optimal performances are unknown for the UCT scenarios. Therefore, we measured optimality using

two different approaches: using human experts and using agent experts. For the former, a human expert was trained on a task and then performed the task several times. The best score achieved was recorded as the *human-optimal* score. For the latter, an agent was given perfect knowledge of the task and then performed several times. The best time that the agent was able to achieve over multiple trials (either with perfect knowledge, incomplete knowledge, or no transferred knowledge) was recorded as *agent-optimal*.

In our case, these two measures of optimality are different because the agent was not optimized for low-level control and navigation. Additionally, artifacts of the domain representation and search algorithm can combine to create undesirable behaviors. This means that UCTBot could never be expected to achieve human-optimal levels of performance. While it seems intuitive that the human-expert measure of optimality would result in fairer cross-agent comparisons, using agent-specific measures of optimality allows agents to control for artifacts in their implementations, and thus more accurately compare the improvement in a task due to transfer. Therefore, it is prudent to report results using both measures of optimality.

6. Transfer Learning Results and Discussion

Figure 5 shows the results of our agent performing in the UCT across all eight levels of transfer learning. As the CTR is interpreted as the percentage of available improvement that was achieved (section 5.2), higher values of CTR indicate better transfer performance.

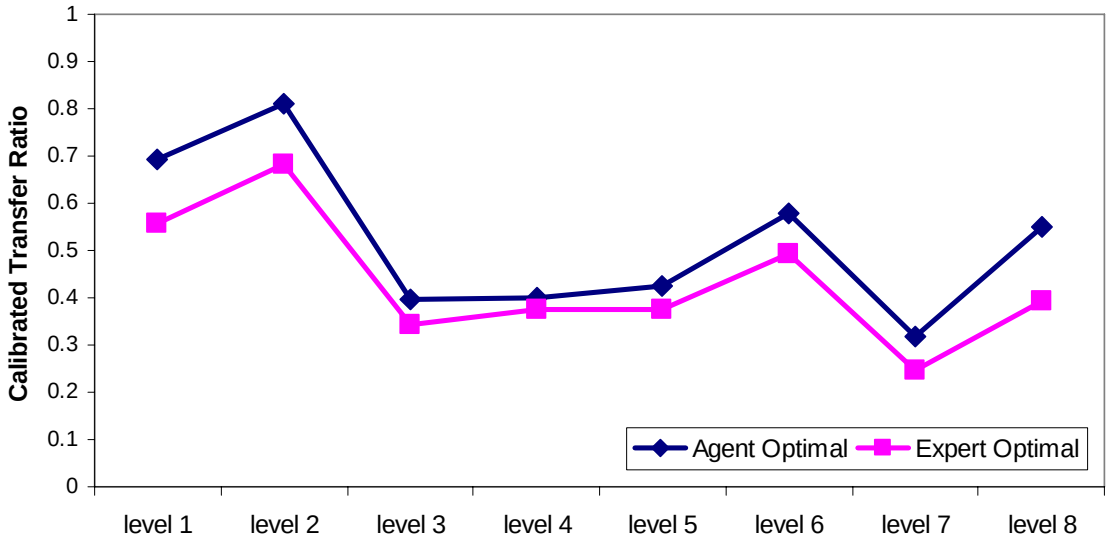

Figure 5: Summarized results of the UCTBot for eight levels of transfer in the UCT. Each point represents the average CTR for all scenarios evaluated at given level for a particular measure of optimality.

These results demonstrate that the UCTBot does indeed benefit from transferred knowledge. Although transfer performance varies between the DARPA transfer levels, it captures 32-81% of the available improvement in performance when calibrated with agent optimal, and between 25-68% of the available improvement in performance when calibrated with human expert optimal.

The agent demonstrates the best transfer performance in levels 1 and 2. As mentioned previously, UCTBot's behavior was not optimized for low-level navigation and control, which includes obstacle detection and avoidance routines. In levels 1 and 2, there are relatively few obstacles, and UCTBot is able to navigate through the tasks with minimal mistakes. However, in later levels some scenarios involve many different obstacles spaced near each other, which confuses UCTBot's simple navigation behavior and leads to slower overall times, regardless of knowledge transferred.

While the performance improvement due to transfer is better on some levels and worse on others, one can't conclude that it is more difficult to perform transfer on those levels in the general case, only that those levels are more difficult on average for the UCTBot. As mentioned above, these difficulties typically lie in the configuration of components and the low-level navigation that causes tasks in the UCTBot's reasoning.

It is clear that for our data both measures of optimality result in qualitatively similar CTRs across all 8 levels. This indicates that evaluating our transfer performing results in this domain with either measure of optimality will result in qualitatively similar values. As the agent optimal measure differed from the human expert optimal measure only in that it controlled for artifacts in the agent's behavior, we expected to see similar ratios in the analysis.

What do these results actually tell us? Our approach was for UCTBot to transfer learned topological knowledge and action models. As is clear from the results, transferring these two types of knowledge and not performing any complex internal reasoning over them is sufficient to achieve improvements in performance. An alternate approach, on the other hand, would be to perform complex reasoning such as structure mapping or analogical reasoning; however, our results indicate that in the UCT a much better approach is to focus purely on optimizing low-level navigation routines.

The results for individual scenarios are presented in Table 1, using both human expert and agent measures of optimality. By examining CTRs for individual scenarios, we can more accurately determine the source of transfer in the UCT.

UCTBot performs worst (on average) in the level 7 scenarios, as seen in Figure 5. As seen in Table 1, the agent observes slight negative transfer in the 4^{th} scenario of level 7 – that is, the agent performs slightly worse with transferred knowledge than without. In this particular scenario, the entrances to the building containing the goal location have very high visibility and map knowledge learned in the source task does not help improve performance (for a description of the scenario, see Youngblood, Shapiro and Holder, 2006; UT Arlington, 2006). This scenario suffers from a ceiling effect as the UCTBot can solve it very well both with and without transferred knowledge. In turn, poor transfer performance on this particular scenario leads to poor average performance for level 7 when compared to the other levels. Also note that UCTBot was only evaluated on 3 level 7 scenarios, as opposed to 4 scenarios for all other levels, which means that its poor relative performance weighed more heavily in the average than a single poor performance in another level would have.

UCTBot showed the best transfer performance in level 6 scenario 5. This scenario involved an electrical barrier which dramatically increases the agents score when touched. Under the control condition, the agent would often spend time navigating the area containing the electrical barrier, which would lead to the agent accumulating large penalties. However, in the transfer condition the agent would typically know the location of the goal and navigate directly to it, minimizing the amount of contact it would have with the barrier.

In this case, then, our agent thus observes good transfer performance because of its naivety. Had the agent been provided with procedural knowledge that it should avoid contact with damage-causing barriers, its performance for the control condition would improve and thus decrease overall transfer performance. This demonstrates that sometimes transfer performance can be directly caused by a lack of knowledge or capabilities in the agent, a pitfall that should be avoided in more comprehensive transfer learning evaluations.

Table 1: UCTBot's transfer performance by scenario.

Level	Scenario	CTR (expert opt.)	CTR (agent opt.)
1	1	0.52	0.69
1	2	0.62	0.79
1	3	0.46	0.55
1	5	0.64	0.75
2	1	0.77	0.94
2	2	0.71	0.76
2	4	0.68	0.82
2	5	0.56	0.73
3	1	0.35	0.42
3	2	0.50	0.59
3	3	0.09	0.10
3	5	0.43	0.47
4	1	0.63	0.67
4	2	0.29	0.33
4	3	0.13	0.15
4	5	0.44	0.44
5	1	0.23	0.26
5	2	0.46	0.52
5	4	0.53	0.59
5	5	0.28	0.33
6	1	0.32	0.48
6	2	0.41	0.47
6	3	0.36	0.47
6	5	0.89	0.90
7	1	0.30	0.39
7	3	0.46	0.58
7	4	-0.01	-0.02
8	1	0.71	0.80
8	3	0.32	0.73
8	4	0.48	0.54
8	5	0.06	0.13

In level 3 scenario 3, the agent also shows relatively poor transfer performance. This scenario involved a relatively large number of obstacles, very poor visibility of the goal, and was configured such that only two small, hard-to-find areas would eventually lead to the goal region. In this scenario, the agent's transfer performance was hindered by a floor effect – the scenario was so difficult that the agent rarely solved it in either the transfer or control conditions and thus observed little transfer because it so rarely solved the task.

The small amount of transfer that was observed is due to the fact that the agent was able to explore a large amount of the domain in the source task and thus was more likely to explore one of the two small "bottleneck" areas that would eventually lead to the goal. Note that level 4 scenario 3 has a very similar configuration of components, and in this scenario the agent's performances also suffer from a floor effect.

7. Sources of Transfer in the UCT

The results of our investigation into the sources of transfer in the UCT are summarized in Figure 6.

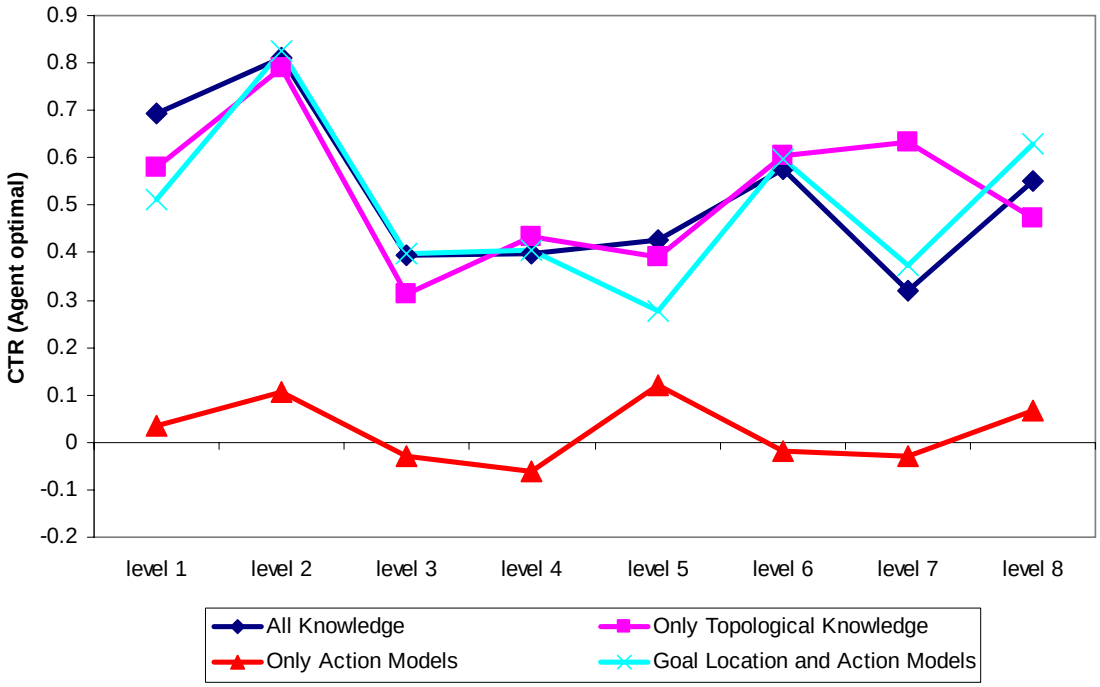

Figure 6: Summarized results of the UCTBot performing in the UCT when transferring only specific types of knowledge.

When transferring only action models for gateways and obstacles, the agent demonstrated very small improvements in performance (and on half of the levels showed a small decrease in performance). It can thus be concluded that learned action models had a small, if not negligible, impact on the UCTBot's transfer performance.

When transferring the other three types of knowledge (all, topological, and goal location with action models), the agent performed similarly for the most part. As transferring only the coordinates of the goal location is a very small subset of the agent's topological knowledge, it indicates that this particular piece of knowledge was very important to the agent's ability to transfer in the UCT domain.

All tasks in the domain depend upon the agent being able to navigate as quickly as possible to a goal location, and thus the actual coordinates identifying this goal location are the most important information for solving the tasks quickly. This observation is surprising, however, in that it suggests that the agent is able to find a path to the remote goal location equally well regardless of the extent of its topological knowledge, indicating that the topological knowledge relating to regions and their connectivity is not very important in the domain to

finding good (quick) paths. While we had expected the goal location to be an important piece of transferred knowledge, we had also expected other topological knowledge to contribute significantly to transfer performance.

When transferring only topological knowledge, UCTBot appears to do much better on level 7 than when it transfers all knowledge. As shown in Table 2, UCTBot achieved higher transfer performance when transferring only topological knowledge than when transferring any other tested combination of knowledge in all three level 7 scenarios. This indicates that the agent either learned inaccurate action models, inaccurately used the learned action models, or some combination thereof.

Table 2: UCTBot's transfer performance by scenario when transferring only specific types of knowledge between source and target tasks. All results are CTR, calibrated with agent optimal scores.

Level	Scenario	All Knowledge	Topological Knowledge	Action Models	Goal Location and Action Models
1	1	0.69	0.52	0.01	0.50
1	2	0.79	0.63	-0.02	0.46
1	3	0.55	0.50	-0.01	0.50
1	5	0.75	0.67	0.15	0.59
2	1	0.94	0.91	0.20	0.93
2	2	0.76	0.65	0.20	0.82
2	4	0.82	0.89	0.04	0.83
2	5	0.73	0.72	-0.01	0.73
3	1	0.42	0.42	-0.53	0.88
3	2	0.59	0.20	0.00	0.45
3	3	0.10	-0.04	0.13	-0.02
3	5	0.47	0.67	0.29	0.29
4	1	0.67	0.47	-0.16	0.53
4	2	0.33	0.28	-0.05	0.42
4	3	0.15	0.18	0.04	-0.04
4	5	0.44	0.80	-0.06	0.71
5	1	0.26	0.11	0.27	0.52
5	2	0.52	0.64	-0.23	-0.17
5	4	0.59	0.59	0.06	-0.87
5	5	0.33	0.23	0.38	0.04
6	1	0.48	0.46	-0.09	0.60
6	2	0.47	0.56	0.18	0.54
6	3	0.47	0.50	-0.03	0.32
6	5	0.90	0.90	-0.13	0.93
7	1	0.39	0.66	-0.03	0.13
7	3	0.58	0.76	0.05	0.74
7	4	-0.02	0.48	-0.11	0.26
8	1	0.80	0.67	0.25	0.78
8	3	0.73	0.70	0.08	0.76
8	4	0.54	0.31	0.06	0.64
8	5	0.13	0.22	-0.12	0.33

8. Conclusions

Using the Soar cognitive architecture, we implemented an intelligent agent that is capable of improving performance in a target task through the transfer of knowledge learned in a different, but related, source task. Along the way, we've reached several conclusions.

In the UCT, it is sufficient to transfer two types of learned knowledge (topological and action models) in order to improve performance in later tasks. Furthermore, it is sufficient to transfer only topological knowledge. And although computational resources prevented us from gathering data to conclude this, we are confident that transferring only the coordinates of the goal would be sufficient knowledge to observe a high degree of transfer performance in the UCT.

In general, we can conclude from this investigation that topological knowledge is much more important to achieving transfer in the UCT than knowledge of action models, and specifically, knowledge of the location of the goal is very important to solving tasks in the UCT as it is necessary for the agent to navigate quickly to the goal. For the most part, the goal location is more important than the connectivity of regions both because the agent can quickly find which potential paths are blocked and navigate around as well as because obstacles are sparsely distributed in the environment and much of the space in the UCT can be traversed..

Along these lines, when investigating transfer learning in complex domains it is prudent to investigate which types of knowledge are contributing to improvements in performance and how they are used. While agents may be allowed to use many types of knowledge in the domain, it may turn out that (as in our case) one particular piece of knowledge is more important than all other types.

As such, when designing and developing complex domains it is important to both thoughtfully consider how agents will be interacting with and reasoning in the domain as well as concurrently design and develop agents to perform in the domain. By performing exploratory studies early on, the path of domain development can more easily be corrected if necessary given the results of those exploratory studies.

When investigating transfer learning, it is important to strive to create agents that perform as well as possible for both transfer and control conditions. It is easy to fall into a trap of devoting the majority of energy to the optimization of agents under the transfer condition, but neglect the development of agents that perform well under the control condition. In such a case, the agent will appear to improve performance dramatically when transferring knowledge, but in reality much of the performance is due to poor agent development.

Given this, a more comprehensive (and fairer) study involving our agent would require that our agent was capable of better low-level navigation between regions and across the domain in general. For example, an agent able to make use of the SRS, a complex spatial reasoning system for Soar agents (Wintermute and Laird, 2007), would be better able to navigate through the UCT domain and we hypothesize that it would approach human-expert optimal performance scores.

9. Related Work

Transfer has been studied in many subfields of machine learning; for example, it has been investigated in the supervised learning setting (Caruana, 1997; Baxter, 1997; Thrun and Pratt, 1998), semi-supervised learning (Ando and Zhang, 2005), reinforcement learning, and learning using cognitive architectures amongst others. Our work, which involves transferring knowledge in an intelligent agent using a cognitive architecture, is part of the latter category; related to it is the study of transfer in reinforcement learning where an intelligent agent additionally receives explicit reward signals from the environment.

9.1. Transfer in Reinforcement Learning

Much recent work has been published regarding the application of reinforcement learning methods to demonstrate transfer; we briefly detail a few of those approaches here. Transfer in reinforcement learning typically involves one or more of the following approaches: transferring learned value functions, transferring learned policies, or transferring learned model parameters. Taylor and Stone (2005) demonstrated that by providing a mapping from one state-action space to another, it is possible to transfer Q-values for temporal difference methods and improve learning, while Banerjee, Kuhlmann and Stone (2006) used given value-function features to transfer Q-values between various 2-player alternate move complete information games. As for transferring policies, Soni and Singh (2006) applied homomorphisms to specify compact transformations between state spaces in order to transfer learned policies as options; Torrey et al. (2006) used a relational reinforcement learning approach and given mappings to transfer skills consisting of first-order rules. Gorski and Laird (2006) noted that the simplest reinforcement learning approach (i.e. learning simple navigational knowledge) does not perform well in the one-shot scenario presentation required in the UCT. For a brief but more comprehensive overview of transfer learning using reinforcement learning, see Stone (2007).

9.2. Knowledge Transfer using Cognitive Architectures

A central goal of research in cognitive architectures is fostering the creation of intelligent agents that have the same capabilities as those underlying human cognition (Langley, Laird and Rogers, 2005). As there is evidence for the human capability to transfer knowledge (see below), it follows that these capabilities should also be explored using cognitive architectures. We begin by briefly discussing the psychological research suggesting human transfer capabilities, and then review research using three cognitive architectures (Soar, ICARUS and Companions) to investigate knowledge transfer.

Psychologists have studied transfer in human learning for over a century, especially in educational settings, and have found evidence of transfer in human cognition across related tasks (Anderson, 2005, pp. 304-311). There is support for the theory that the human capability to transfer knowledge is performed over abstractions of *identical elements* (Thorndike, 1903; Newell, 1990), essentially specialized procedural knowledge; other work has examined the transfer of declarative knowledge structures in human cognition (Gagne

and White, 1978; Brooks and Dansereau, 1987). Similarly, research using cognitive architectures has focused on transferring both procedural knowledge (as in Soar and ICARUS, discussed below) as well as declarative knowledge structures (Soar and Companions). It should be noted that there is scant evidence of *far* transfer in human cognition; that is, the transfer of more complicated strategies or behaviors from one task to a distantly related task (Anderson, 2005). For example, Kotovsky, Hayes and Simon (1985) found little evidence of transfer between problems with isomorphic representations, indicating that after subjects had learned one of the isomorphs they were unable to transfer procedural knowledge to the new, related problem.

The Soar cognitive architecture (discussed in more detail in section 4) is a production-based system which contains a learning mechanism called *Chunking* (Lehman, Laird and Rosenbloom, 2006). Chunking compiles a sequence of rules into a single production, eliminating all internal reasoning that those rules would otherwise require. Laird et al. (1986) explored transferring these *chunks* (compiled rules) between related problems in the 8-puzzle domain. By making use of symmetries in the problem space, the agent was able to relate previously learned chunks to the current situation, thereby improving performance.

ICARUS is a cognitive architecture in which skills and structural knowledge are stored in separate memories (Langley, Choi and Rogers, 2005). Skills are hierarchically organized and are associated with preconditions, effects, and either sub-tasks or actions. Central to the ICARUS architecture is its commitment to backward-chaining planning. Nejati, Langley and Konik (2006) have demonstrated that skills can be learned through observation in two simple domains, Blocks World and Depots. Once learned, these skills can be transferred to new problems, creating the potential for transfer. Similar work demonstrated successful transfer in the Urban Combat Testbed (Choi et al., 2007).

The Companion Cognitive Systems is a cognitive architecture inspired by the analogical aspects of human cognition (Forbus and Hinrichs, 2006). In Companions, structural knowledge elements are compared for similarities in both structure and semantic content. Klenk and Forbus (2007) have successfully applied the analogical reasoning capabilities of Companions to AP physics problems. In this work, the system was presented with a set of problems, their solutions, and the correct steps taken to arrive at the solutions. Using this knowledge, it was able to relate new, unsolved problems to those previously seen through analogy. To solve a problem, the system examines candidate solutions that it has already seen for similarities, and then attempts to validate problem-solving steps in the current problem by examining that step in the context of the previously taken step. Using this mechanism, their system was able to solve problems demonstrating parameterization, restructuring and restyling (DARPA transfer levels 1, 3, and 5 respectively).

Also using the Companions system, Hinrichs, Nichols and Forbus (2006) have begun to explore using analogy to plan and learn in Freeciv, an open-source turn-based strategy game. This work attempts to increase the performance of a Hierarchical Task Network planner by relating action preconditions to sub-goal conditions by analogy

9.3. Transfer Learning in the UCT

Previously, Gorski and Laird (2006) demonstrated that an agent using Soar was capable of transfer in a preliminary version of the UCT. This work compared the transfer performance of three learning mechanisms: a memory-based approach, search-based approach, and a reinforcement learning approach. The memory-based approach transferred topological knowledge, the search-based approach transferred both topological knowledge as well as chunks, and the Reinforcement Learning agent transferred operator Q-values. While the work was preliminary in nature, it demonstrated significant initial transfer at the first DARPA level for all three approaches and the second level for the memory- and search-based approaches. One conclusion from the work noted that the look-ahead search approach was most applicable to the UCT domain due to its real-time performance and low memory requirements. While not mentioned in this previous work, the search-based approach is even more appropriate in the current testbed as it is the most robust when reasoning over partial topological knowledge.

As mentioned above, Choi et al. (2007) successfully demonstrated transfer learning in the UCT using the Icarus architecture. Their approach involves the transfer of hierarchically-decomposed structural skills. Asadi and Huber (2007) have also investigated the transfer of learned hierarchical skills in the UCT, but not on the same tasks as used in our investigation or other work in the testbed.

Acknowledgments

The authors kindly thank Larry Holder and Michael Youngblood for their contributions to the Urban Combat Testbed domain. We also thank Dongkyu Choi, Tom Fawcett, Tolga Konik, Pat Langley, Yaxin Liu, Cynthia Matuszek, Dan Shapiro, and Blake Shepard for their thoughtful insights, assistance, and contributions to the Transfer Learning program.

This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and managed by the Air Force Research Laboratory (AFRL) under contract FA8750-05-2-0283. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of DARPA, AFRL, or the United States Government.

References

Anderson, J.R. (2005). *Cognitive Psychology and Its* Implications (6th ed.). New York: Worth Publishers.

Ando, R.K. & Zhang, T. (2005). A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. *Journal of Machine Learning Research, 6*, 1817-1853.

Asadi, M. & Huber, M. (2007). Effective Control Knowledge Transfer Through Learning Skill and Representation Hierarchies. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. Hyderabad, India.

Banerjee, B., Kuhlmann, G., & Stone, P. (2006). Value Function Transfer for General Game Playing. *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*. Pittsburgh, PA.

Baxter, J. A Bayesian/Information Theoretic Model of Learning to Learn via Multiple Task Sampling. *Machine Learning, 28*, 7-39.

Brooks, L.W., & Dansereau, D.F. (1987). Transfer of Information: An Instructional Perspective. In S.M. Cormier & J.D. Hagman (Eds.), *Transfer of Learning: Contemporary Research and Applications* (pp. 121-150). San Diego: Academic Press.

Caruana, R. (1997). Multitask Learning. *Machine Learning, 28*, 41-75.

Choi, D., Konik, T., Nejati, N., Park, C. & Langley, P. (2007). Structural Transfer of Cognitive Skills. *Proceedings of the 8th International Conference on Cognitive Modeling*. Ann Arbor, MI.

DARPA (2005). *Transfer Learning* (BAA 05-29).

Forbus, K. & Hinrichs, T. (2006). Companion Cognitive Systems: A Step towards Human-Level AI. *AI Magazine*, 27(2).

Gagne, R.M., & White, R.T. (1978). Memory Structures and Learning Outcomes. *Review of Educational Research, 48*, 187-222.

Gorski, N.A. & Laird, J.E. (2006). Experiments in Transfer Across Multiple Learning Mechanisms. *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*. Pittsburgh, PA.

Hinrichs, T.R., Nichols, N.D. & Forbus, K. (2006). Using Qualitative Reasoning in Learning Strategy Games: A Preliminary Report. *20th International Workshop on Qualitative Reasoning*. Hanover, NH.

Klenk, M. & Forbus, K. (2007). Measuring the Level of Transfer Learning by an AP Physics Problem-solver. *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*. Vancouver, B.C., Canada.

Kotovsky, K., Hayes, J.R. & Simon, H.A. (1985). Why Are Some Problems Hard? Evidence From Tower of Hanoi. *Cognitive Psychology, 17,* 284-294.

Laird, J.E. (2006). The Soar 8 Tutorial (Part 5). Retrieved September 9, 2006 from http://sitemaker.umich.edu/soar/documentation_and_links

Laird, J.E., Rosenbloom, P.S., & Newell, A. (1986). Chunking in Soar: The Anatomy of a General Learning Mechanism. *Machine Learning, 1,* 11-46.

Langley, P., Choi, D. & Rogers, S. (2005). Interleaving Learning, Problem-Solving, and Execution in the ICARUS Architecture. Technical Report, Computational Learning Laboratory, CSLI, Stanford, CA.

Langley, P., Laird, J.E., & Rogers, S. (2005). *Cognitive Architectures: Research Issues and Challenges*. Unpublished Manuscript, Center for the Study of Language and Information, Stanford, CA.

Lehman, J.F., Laird, J., & Rosenbloom, P. (2006) A Gentle Introduction to Soar, an Architecture for Human Cognition: 2006 Update. Retrieved September 7, 2006 from http://ai.eecs.umich.edu/soar/sitemaker/docs/misc/GentleIntroduction-2006.pdf

Magerko, B. (2006). *Player Modeling in the Interactive Drama Architecture*. Unpublished doctoral thesis, University of Michigan, Ann Arbor, MI.

Magerko, B., Laird, J.E., Assanie, M., Kerfoot, A., & Stokes, D. (2004) AI Characters and Directors for Interactive Computer Games. *Proceedings of the 2004 Innovative Applications of Artificial Intelligence Conference*. San Jose, CA.

Matuszek, C., Cabral, J., Witbrock, M., & DeOliveira, J. (2006). An Introduction to the Syntax and Content of Cyc. *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*. Stanford, CA.

Mehta, N., Natarajan, S., Tadepalli, P., & Fern, A. (2005). Transfer in Variable-Reward Hierarchical Reinforcement Learning. *Proceedings of the NIPS 2005 Workshop on Inductive Transfer: 10 Years Later*.

Morrison, C.T., Chang, Y., Cohen, P.R., & Moody, J. (2006). Experimental State Splitting for Transfer Learning. *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning.*

Nejati, N., Langley, P. & Konik, T. (2006). Learning Hierarchical Task Networks by Observation. *Proceedings of the 23rd International Conference on Machine Learning*. Pittsburgh, PA.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Shapiro, D., Fawcett, T., & Langley, P. (2006). *Internal Evaluation Plan – Urban Combat*. Unpublished Manuscript, Institute for the Study of Learning and Expertise, Palo Alto, CA.

Soni, V. & Singh, S. (2006). Using Homomorphisms to Transfer Options across Continuous Reinforcement Learning Domains. *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Boston, MA.

Stone, P. (2007). Learning and Multiagent Reasoning for Autonomous Agents. *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. Hyderabad, India.

Taylor, M.E. & Stone, P. (2005). Behavior Transfer for Value-Function-Based Reinforcement Learning. *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*. Utrecht, Kingdom of the Netherlands.

Thorndike, E.L. (1903). *Educational Psychology*. New York: Lemcke & Buechner.

Thrun, S. & Pratt, L. (Eds.). (1998). *Learning to Learn.* Boston: Kluwer.

Torrey, L., Shavlik, J., Walker, T., & Maclin, R. (2006). Relational Skill Transfer via Advice Taking. *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*. Pittsburgh, PA.

University of Texas at Arlington CSE AI Research Group (2006). *The Urban Combat Testbed*. Retrieved August 27, 2006 from http://gameairesearch.uta.edu/

Wintermute, S. & Laird, J.E. (2007). Predicate Projection in a Bimodal Spatial Reasoning System. *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*. Vancouver, B.C., Canada.

Wray, R.E., Laird, J.E., Nuxoll, A., Stokes, D., & Kerfoot, A. (2004) Synthetic Adversaries for Urban Combat Training. *Proceedings of the 2004 Innovative Applications of Artificial Intelligence Conference*. San Jose, CA.

Youngblood, G.M., Shapiro, D., & Holder, L. (2006) Urban Combat Testbed Transfer Learning Scenarios. Unpublished Manuscript, University of Texas at Arlington, Arlington, TX.