



AFRL-RI-RS-TR-2010-168

TRANSFER LEARNING IN INTEGRATED COGNITIVE SYSTEMS

Institute for the Study of Learning and Expertise (ISLE)

September 2010

*Sponsored By
Defense Advanced Research Projects Agency
Darpa Order No. AQ51/00*

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

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 the Defense Advanced Research Projects Agency or the U.S. Government.

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2010-168 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/
DEBORAH A. CERINO
Work Unit Manager

/s/
MICHAEL WESSING, Acting Chief
Information & Intelligence Exploitation Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE**Form Approved**
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**1. REPORT DATE (DD-MM-YYYY)**

SEPTEMBER 2010

2. REPORT TYPE

Final

3. DATES COVERED (From - To)

September 2005 – March 2010

4. TITLE AND SUBTITLE

TRANSFER LEARNING IN INTEGRATED COGNITIVE SYSTEMS

5a. CONTRACT NUMBER

N/A

5b. GRANT NUMBER

FA8750-05-2-0283

5c. PROGRAM ELEMENT NUMBER

62304E

6. AUTHOR(S)

Daniel Shapiro, Pat Langley, David J. Stracuzzi, Dana Nau, Alan Fern, Ray Mooney, Peter Stone, and Pedro Domingos

5d. PROJECT NUMBER

TRLG

5e. TASK NUMBER

00

5f. WORK UNIT NUMBER

02

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)Institute for the Study of Learning and Expertise
2164 Staunton Court
Palo Alto, CA 94306-1438**8. PERFORMING ORGANIZATION
REPORT NUMBER**

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)Defense Advanced Research Projects Agency AFRL/RIED
3701 North Fairfax Drive 525 Brooks Road
Arlington, VA 22203-1714 Rome, NY 13441-4505**10. SPONSOR/MONITOR'S ACRONYM(S)**

N/A

**11. SPONSORING/MONITORING
AGENCY REPORT NUMBER**
AFRL-RI-RS-TR-2010-168**12. DISTRIBUTION AVAILABILITY STATEMENT**

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES**14. ABSTRACT**

The ISLE Transfer Learning project was a three year DARPA-funded research effort whose goal was to enable machines to acquire knowledge in one domain and use it to improve performance in another. This capacity is a first step of a longer term vision that employs transfer to reduce the cost of developing intelligent software, hereby replacing the construction of multiple, one-off systems, with a methodology based on the automated transfer and augmentation of capabilities encoded in an initial source. The project emphasized machine learning and artificial intelligence (AI) techniques, and focused on the acquisition and transfer of behavioral knowledge (meaning strategies for how to act together with the supporting domain knowledge) from a source task to a target task drawn from a different domain. The work addressed three component problems within transfer: knowledge acquisition in a source task, knowledge transfer to a target task, and knowledge augmentation in the target domain given the transferred wisdom.

15. SUBJECT TERMS

Machine learning, knowledge representation and reasoning, reinforcement learning, Markov Logic Networks

16. SECURITY CLASSIFICATION OF:**17. LIMITATION OF
ABSTRACT****18. NUMBER
OF PAGES****19a. NAME OF RESPONSIBLE PERSON**

Deborah A. Cerino

a. REPORT

U

b. ABSTRACT

U

c. THIS PAGE

U

UU

88

19b. TELEPHONE NUMBER (Include area code)

N/A

Table of Contents

1.	Summary and Introduction	1
2.	Technical Tasks Accomplished: Transfer Learning in Icarus	3
2.1.	Comparison of Human and Agent Transfer in the Urban Combat Domain	3
2.1.1.	Task Representation in Icarus	4
2.1.2.	Execution of Hierarchical Skills	6
2.1.3.	Learning Hierarchical Skills	7
2.1.4.	Results on Transfer in Urban Combat	7
2.2.	Achieving Far Transfer via Goal-driven Analogical Mapping.....	9
2.2.1.	Supporting Far Transfer via Representation Mapping.....	11
2.2.2.	Results on Far Transfer	13
2.3.	Transfer from Recognition into Play Improvement in American Football.....	15
2.3.1.	The College Football Video.....	15
2.3.2.	The Rush Football Simulator	16
2.3.3.	Icarus Concepts and Skills for Football	17
2.3.4.	Learning Skills from Video	18
2.3.5.	Mapping Recognized Plays into Rush	20
2.3.6.	Parametric Adaptation of Football Plays	20
2.3.7.	Results on Transfer from Recognition into Play Improvement.....	21
3.	Conclusions.....	23
4.	References.....	24
5.	List of Acronyms	25
6.	Appendix A: Learning to Annotate Video of American Football Plays.....	26
7.	Appendix B: Skill Learning and Activity Recognition for Transfer	34
8.	Appendix C: Hierarchical Task Networks in Transfer Learning.....	45
9.	Appendix D: Advances in Reinforcement Learning for Transfer	50
10.	Appendix E: Transfer Learning with Markov Logic Networks.....	57
11.	Appendix F: Deep Transfer through Markov Logic Networks.....	65

List of Figures

Figure 1. A transfer task in the Urban Combat Testbed.	4
Figure 2. A schematic of the Icarus Architecture.	6
Figure 3. Solution times in seconds for the transfer case plotted against those for non-transfer cases. Axes are inverted so that higher scores indicate better performance.	8
Figure 4. The ratio of agent to human transfer scores obtained on the same tasks. Negative ratios describe cases where human performance degrades.	9
Figure 5. Far transfer tasks set in three grid-based games.	10
Figure 6. A lesion study showing the impact of representation mapping on transfer	14
Figure 7. A typical frame from the football video.	16
Figure 8. A Rush 2008 starting formation for the offensive (bottom) and defensive (top) teams, with player/position annotations.	17
Figure 9. Diagram of play 18 with annotations for actions taken by individual players.	19
Figure 10. Average performance results when learning to improve American football plays, with and without knowledge transfer from play recognition.	22

List of Tables

Table 1. Primitive and non-primitive Icarus concepts.	5
Table 2. Primitive and non-primitive Icarus skills.....	5
Table 3. A skill mapped from Escape to mRogue.	12
Table 4. Sample concepts for the football domain.	18
Table 5. Sample skills for the football domain.	18

1. Summary and Introduction

The ISLE Transfer Learning project was a three year DARPA-funded research effort whose goal was to enable machines to acquire knowledge in one domain and use it to improve performance in another. This capacity is a first step of a longer term vision that employs transfer to reduce the cost of developing intelligent software, hereby replacing the construction of multiple, one-off systems, with a methodology based on the automated transfer and augmentation of capabilities encoded in an initial source.

The project emphasized machine learning and artificial intelligence (AI) techniques, and focused on the acquisition and transfer of behavioral knowledge (meaning strategies for how to act together with the supporting domain knowledge) from a source task to a target task drawn from a different domain. The work addressed three component problems within transfer: knowledge acquisition in a source task, knowledge transfer to a target task, and knowledge augmentation in the target domain given the transferred wisdom.

ISLE acted as prime contractor in this effort, coordinating approximately a dozen institutions to complete three annual demonstrations. A DARPA-appointed evaluation team controlled the subject of those demonstrations, the conduct of the annual experiments, and the analysis of the results with respect to predefined “go/no-go” targets. The difficulty of the demonstrations also grew as the project matured:

- In Year 1, the source and target tasks were problem-solving puzzles set within a single, real-time, 3D, 1st person shooter game.
- The Year 2 demonstration addressed six classes of transfer problems, with source and target tasks drawn from the same, or different, 2D grid-world games set in a “General Game Playing” framework.
- The Year 3 demonstration addressed transfer between families of tasks, and emphasized connection to real-world data. The source task was to recognize plays from videos of Oregon State University football games, while the target task was to redesign and improve those plays in simulation.

The project employed cognitive architectures as the primary research vehicle. These are AI systems that offer a capacity for generalized intelligence, such as Icarus, Soar, and Companions. Versions of these three systems completed the Y1 and Y2 demonstrations, while Icarus alone performed the Y3 task. Every system successfully completed its demonstration milestones.

In order to accomplish these milestones, the project advanced and utilized a number of component technologies for transfer within cognitive architectures: explanation based learning, theory formation and refinement, analogical reasoning, and hierarchical task network planning, to name a few. Some of the analogical reasoning technology was evaluated on Advanced Placement (AP) Physics problems outside of the year-end demonstrations. In addition, the project fostered independent research efforts that developed and evaluated approaches to transfer based on reinforcement learning methods, and on Markov Logic Networks (MLNs).

The project began with the broad conjecture that automated methods could utilize a small number of experiences to acquire and transfer significant structural knowledge between disparate source and target tasks. Within that context, we claimed that our methods could:

- acquire and transfer both relational and hierarchical knowledge,
- compose learned knowledge to support transfer, and
- support cumulative transfer, i.e., learn new structures in terms of one's previously acquired.

In addition, we claimed that logic-oriented frameworks would enable these capabilities, and that analogical mapping would be essential for transfer across disparate domains.

The project developed methods that demonstrated each of these claims in a research setting. More broadly, the project arrived at these conclusions:

- Explanation based learning techniques support relational and hierarchical knowledge acquisition across multiple logic-oriented domains.
- Acquired knowledge can be automatically generalized and expressed in a form that natively supports *near* transfer, i.e., use within different tasks set in the same environment/problem domain.
- Analogical representation mapping supports *far* transfer, i.e., reuse in disparate domains, providing the objects, descriptions, and/or solutions are structurally similar.
- The capacity for, and benefit of transfer is roughly proportional to the similarity between the domains.

This report summarizes the technology developed within ISLE's Transfer Learning project, with emphasis on the capabilities realized at the end of Year 3. Chapter 2 discusses work by ISLE on the Y1, Y2, and Y3 demonstrations, covering technical tasks accomplished, result, and conclusions on each task.

The Appendices to this report describe work conducted by ISLE's subcontractors. Appendices A and B examine components of the Y3 system in more detail; Appendix A discusses work on data collection and annotation of football plays via machine learning methods conducted by Oregon State University, while Appendix B discusses the belief maintenance component of Icarus developed at Arizona State University with application to football play recognition and skill learning. Appendix C discusses the capabilities to learn hierarchical task networks and combine heuristic search techniques within Hierarchical Task Network (HTN) planning developed at the University of Maryland. Appendices D through F discuss research conducted within the project but outside the demonstration objectives. In particular, Appendix D summarizes work on transfer in a reinforcement learning setting conducted at the University of Texas, while Appendices E and F describe work on transfer within the framework of Markov Logic Networks, conducted at the University of Texas and the University of Washington, respectively.

2. Technical Tasks Accomplished: Transfer Learning in Icarus

This chapter discusses integrated systems built by ISLE to address the Y1, Y2, and Y3 demonstration milestones. These systems utilized the Icarus cognitive architecture as a base while augmenting it with various additional capabilities. We document the resulting technology for transfer in separate sections devoted to each demonstration milestone.

2.1. Comparison of Human and Agent Transfer in the Urban Combat Domain

The demonstration task in the first year of the project compared transfer produced by humans against transfer produced by machines, with experiments set in a first-person perspective video game called Urban Combat. Our research focused on developing the capability for machine transfer of complex skills between tasks that involved action over time. Here, transfer primarily involves the reuse of cognitive structures, where the amount of shared structure has proven to be a good predictor for the degree of transfer in both humans and machines.

We share with many psychologists the idea that transfer is mainly a structural phenomenon, rather than a consequence of statistical summaries or value functions. This suggests that transfer is linked closely to how an agent represents knowledge in memory, how its performance methods use these structures, and how its learning elements acquire this knowledge. In this light, the Icarus cognitive architecture provides commitments to relational, hierarchical, and composable knowledge structures, and to mechanisms for using and acquiring them that provide it with basic support for effective transfer. We made the unusual claim that the architecture needs no additional mechanisms to exhibit many forms of transfer, which we examined at length in Choi, et al. (2007).

We phrased transfer tasks in the Urban Combat Testbed (UCT), a virtual 3-D environment that simulates an urban landscape, with real-time behavior and realistic dynamics. UCT contains one intelligent agent (controlled by Icarus) and no adversaries. The transfer tasks focus on navigation in the presence of physical and conceptual obstacles.

Figure 1 illustrates one such transfer task. The source problem calls on the agent to find a goal and surmount obstacles encountered en route (here, to duck under and climb over obstacles it has never seen). The target problem offers the agent the opportunity to reuse its knowledge about obstacles in a different order, assuming it is acquired and represented in a modular form. In addition, the agent can reuse learned knowledge about the map. The agent exhibits (positive) transfer if it improves its behavior in the target as a result of its exposure to the source, and zero or negative transfer if it does not.

We supply the Icarus agent with minimal background knowledge to support transfer. On initialization, it has never encountered the specific objects or operators in the domain, and it has no prior knowledge of the map. However, it is initialized with useful concepts such as a category for obstacles in general and a relation for blocked paths, plus categories for region centers and gateways (the UCT environment is divided into convex regions with passable and non-passable boundaries). The agent understands the high-level goal (e.g., to find an item), and it possesses sub goals that organize search behavior. For example, it knows to overcome an obstacle to get a clear view of the destination, and to contain exploration within the region of the goal, once seen.

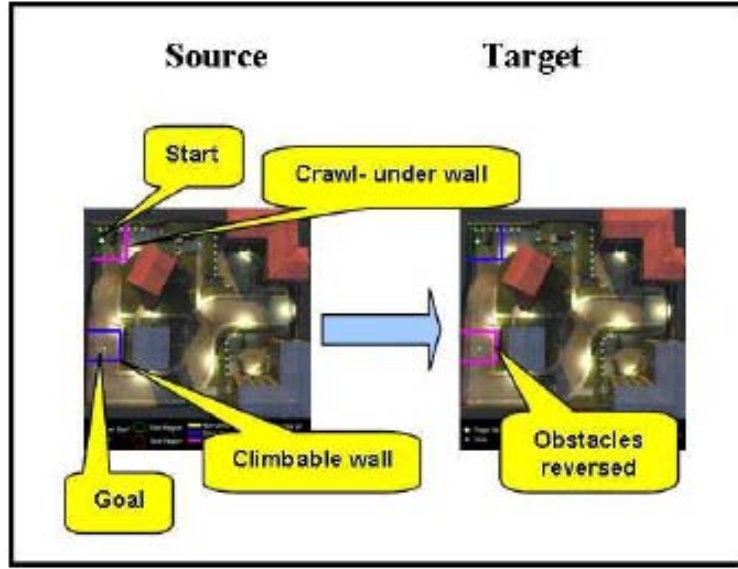


Figure 1. A transfer task in the Urban Combat Testbed.

UCT is a challenging domain for both human and artificial agents. It is partially observable because the agent can only perceive nearby objects and regions, it involves uncertain action (e.g., the agent can attempt to jump over a wall but fall backwards into a ditch), and it is real time (imposing a strong constraint on agent decision making). This complexity demands a level of robustness in the mechanisms that produce transfer.

2.1.1. Task Representation in Icarus

The Icarus architecture captures task knowledge representations for two key structures; concepts and skills. Concepts describe aspects of the agent's state, while skills are methods an agent can execute in the world to achieve goals under certain conditions. Both have a hierarchical structure, meaning that Icarus can employ multiple layers of abstraction in describing the current state and the procedures for manipulating that state, respectively.

As shown in Table 1, concepts in Icarus resemble traditional Horn clauses in first-order logic with negations. Primitive concepts like *in-region* provide state descriptions at the lowest level of abstraction using symbolic and numeric information directly available from objects the agent perceives. Higher-level concepts, such as *stopped-in-region* and *climbable-gateway*, have their basis in other concepts as well as primitive facts. The concept hierarchy provides relational, modular descriptions of the current state. It can also be used to represent a desired state, so concepts can express goals.

Icarus skills for a given domain are a specialized form of hierarchical task networks (Nau et al., 1999). A skill's head indexes it by the goal it achieves, and, since goals are naturally represented by desired concept instances, skills are linked to the concept hierarchy. Some achieve low-level concepts, while others address broad objectives. Table 2 shows some examples of skills in Icarus. While primitive skills give simple methods using basic actions executable in the world, non-primitive skills describe higher-level, complex methods with multiple ordered sub goals.

Table 1. Primitive and non-primitive Icarus concepts.

```
((in-region ?self ?region)
:percepts (self ?self region ?region))

((climbable-gateway ?gateway ?object)
:percepts (gateway ?gateway) (object ?object)
:relations ((totally-blocked-gateway ?gateway ?object)
(feature-of-object ?object CLIMBABLE)))
```

Table 2. Primitive and non-primitive Icarus skills.

```
((clear ?gateway)
:percepts ((gateway ?gateway dist1 ?dist1 angle1 ?angle1 dist2 ?dist2 angle2 ?angle2))
:start ((close-enough-to-jump-type ?gateway))
:actions ((*jump-cmd (maximum ?dist1 ?dist2))
(mid-direction ?angle1 ?angle2)))

((crossable-region ?regionB)
:percepts ((self ?self) (region ?regionB))
:start ((connected-region ?regionB ?gateway))
:subgoals ((clear ?gateway)
(in-region-able ?me ?regionA ?regionB))
:percepts ((self ?me)
(region ?regionA)
(region ?regionB))
:start ((in-region ?me ?regionA))
:subgoals ((crossable-region ?regionB)))

((in-region me region3004)
:subgoals ((in-region-able me region3003 region3004)
(in-region me region3004)))
```

Icarus' relational, hierarchical, and compos able representation of skills is crucial to its ability to transfer knowledge. In particular, the relational representation increases generality of the encoded skills, since they can apply in circumstances that are only qualitatively similar to the situations in which they were acquired. Moreover, a compos able representation lets a skill hierarchy apply as a whole in new circumstances, even if subsets of its skills are incorrect, inaccurate, or inapplicable. In these cases, failed knowledge can be patched with skills acquired from new experience or dynamically replaced with alternatives that achieve the same goal.

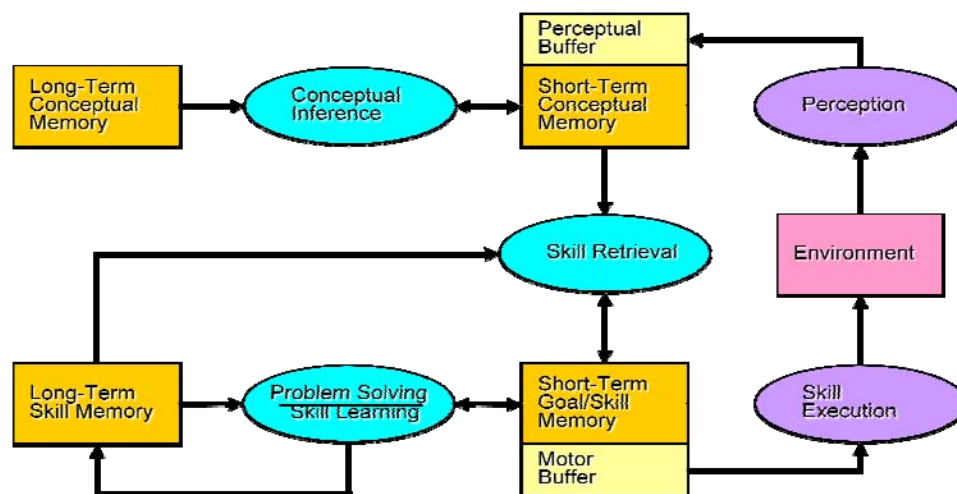


Figure 2. A schematic of the Icarus Architecture.

2.1.3. Learning Hierarchical Skills

Icarus acquires structured skills via an analytical learning mechanism that it invokes whenever the agent achieves a top-level goal. This mechanism inputs the goal plus a solution trace that achieves it, described as a sequence of observed states and selected actions. Icarus generates an explanation of how the goal was achieved by interpreting this solution trace in the context of conceptual knowledge and action models. It does so by recursively examining goals, either decomposing them into sub goals using conceptual knowledge or explaining them in terms of the effects of primitive skills. The architecture converts the resulting explanation structure into hierarchical skills and adds them to its skill memory. We have described this process elsewhere (Nejati et al., 2006) in more detail.

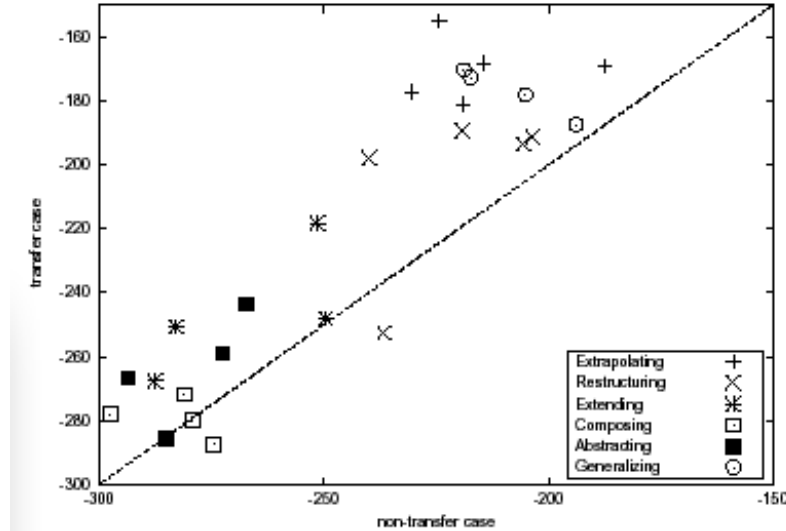
This learning mechanism facilitates transfer by associating a hierarchy of learned skills with the goals they achieve. As a result, the component skills can be used independent of the top-level goal that motivated their construction. For example, an Icarus agent tasked to enter a building may find a solution where it jumps over a fence and then enters the building, viewed as sequence of primitive skills. The analytical learner creates a new skill to climb over a fence, as well as a higher-level skill that uses it along with primitives to reach the goal from the start location. The system associates the low-level skill (for fence climbing) with the goal for reaching a parameterized location, and it considers the component whenever a fence blocks a local goal.

2.1.4. Results on Transfer in Urban Combat

We conducted two forms of evaluation to examine Icarus' account of transfer. The first measured time saved in solving a target problem given exposure to a source task, while the second compared transfer obtained with Icarus against transfer obtained by humans on the same problems. These experiments examined several classes of transfer tasks characterized by the relation between source and target problems:

- *Extrapolating* requires the agent to reuse partial solutions from a common start state in source and target problems, despite differing goals.
- *Restructuring* requires the agent to use solutions to sub problems in different orders.
- *Extending* requires repeated reuse of solutions to source sub problems in the target context.
- *Composing* requires dynamic combination of source solutions in the target task.
- *Abstracting* involves sharing hierarchical solution structure.
- *Generalizing* requires reusing acquired skills, while applying different operators to novel objects found in the target.

Figure 3 summarizes the results of the experiments on transfer within Icarus. It plots the problem solution time for the non-transfer condition against the time for the transfer condition, where each x and y value represents the average score over 20 runs of the Icarus agent, with different icons depicting the forms of source-target relationship. Entries above the diagonal line indicate that positive transfer occurred, while entries below the line reflect negative transfer. The figure shows that Icarus generally exhibits positive transfer for most problems in each type of relationship. Moreover, this transfer occurs after experience with only five source problems, meaning that the rate of learning appears roughly comparable to that observed in humans.



Domain Performance Task: Navigation time to locate IED, minimizing health penalties

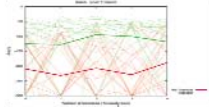
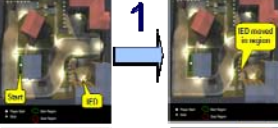
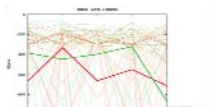
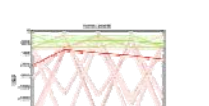
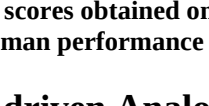
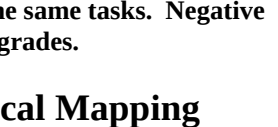
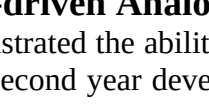
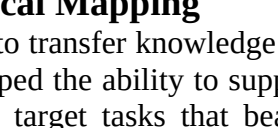
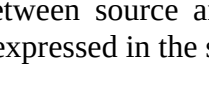
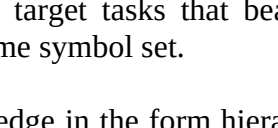
TL Level	TL Task	Agent/Human Transfer	Ave. Curves	Examples
1	<i>Parameterization</i> Vary IED location	-130.6		
2	<i>Extrapolation</i> Close off paths	-0.4		
3	<i>Restructuring</i> Rearrange obstacles	-18.8		
4	<i>Extending</i> Duplicate obstacles	2.4		
5	<i>Restyling</i> Isomorphic Problem	3.5		
6	<i>Composing</i> Concatenate tasks	0.5		
7	<i>Abstracting</i> Change a subtask	-0.6		
8	<i>Generalizing</i> Replace a subtask	2.1		

Figure 4. The ratio of agent to human transfer scores obtained on the same tasks. Negative ratios describe cases where human performance degrades.

2.2. Achieving Far Transfer via Goal-driven Analogical Mapping

While work in the first year of the project demonstrated the ability to transfer knowledge among similar tasks (called *near* transfer), work in the second year developed the ability to support *far* transfer, that is, to communicate knowledge between source and target tasks that bear little surface similarity. These tasks need not even be expressed in the same symbol set.

As before, we focus on acquiring and transferring structural knowledge in the form hierarchical Icarus skills and concepts. However, we introduce a new mechanism, called representation mapping, which can be thought of as a goal-driven form of analogy; it transfers skills and concepts associated with the subset of source goals that make sense in target terms. Shapiro et al. (2010) provides details of this process.

We employed the General Game Playing (GGP) framework (Genesereth et al., 2005) to structure this task. GGP encodes 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 examined three types of far transfer tasks within the GGP framework, 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 5a 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 replacing all source symbols to obtain the target task. Figure 5b 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 5c 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.

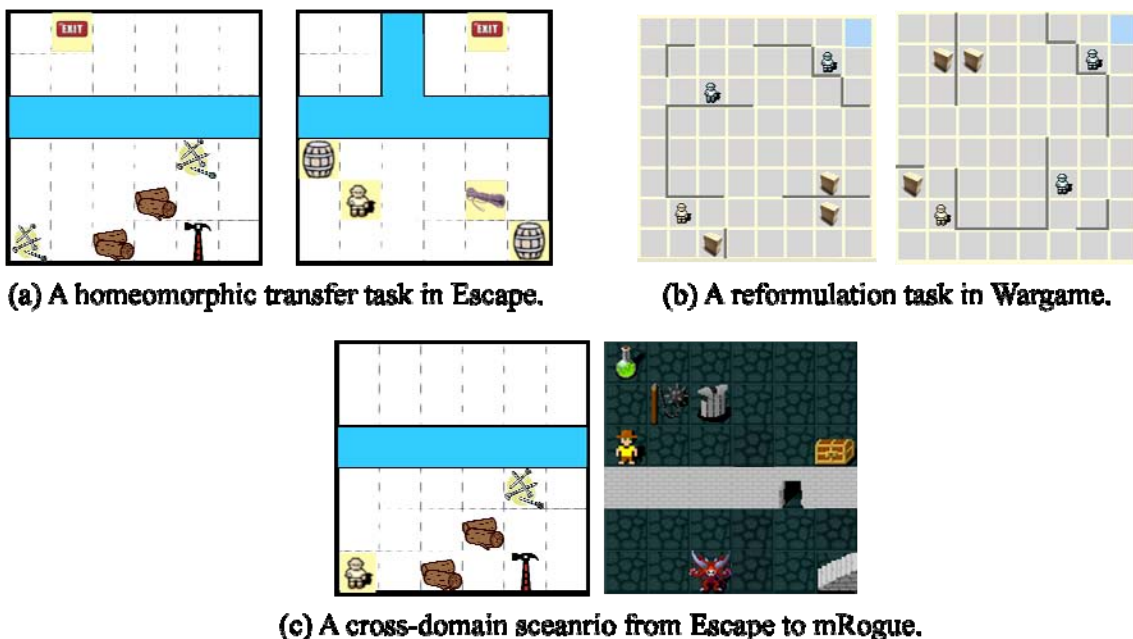


Figure 5. Far transfer tasks set in three grid-based games.

2.2.1. Supporting Far Transfer via Representation Mapping

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.

The system begins by translating a GGP game specification into Icarus 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 concepts and skills that generalize the solution, which become the object of communication/transfer to the target problem. After communicating these concepts and skills, ICARUS employs them to act in the target domain, falling back on exploration when guidance from source knowledge is exhausted.

We developed a mechanism, called representation mapping that addresses the communication step via a form of analogical reasoning. This mechanism enables transfer between structurally similar settings that may involve different predicates. Representation mapping adds a comparison of source and target domains to the execution and skill learning steps outlined above. It consists of two components. The *representation mapper* finds the correspondences between the source and target symbols, while the *representation translator* uses those correspondences to translate source skills and concepts into skills and concepts in the target domain. The representation mapper is the more significant component, and we focus on it below.

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 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 sub goal 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, $(\text{location explorer } 1 \ 2) \leftrightarrow (\text{place hero? } X \ ?Y)$, implies the correspondences $\{\text{location} \leftrightarrow \text{place}, \text{explorer} \leftrightarrow \text{hero}, 1 \leftrightarrow ?X, 2 \leftrightarrow ?Y\}$. This mapping includes relational predicates.

Table 3. 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 $\leftrightarrow$ doestie, nailable $\leftrightarrow$ tieable,
  (property ?item3 hammer)  $\leftrightarrow$  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 method is guided by several constraints and heuristics. These fall into three groups described by Holyoak and Thagard (1989) as structural, semantic and pragmatic constraints. 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 sub goals 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 “ $1 \leftrightarrow ?X$, $2 \leftrightarrow ?X$ ” 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 concepts 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 5b. As part of skill learning, the architecture acquires the component skill for Escape shown at the top of Table 3. 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.

2.2.2. Results on Far Transfer

The capacity for far transfer lets the modified Icarus solve a qualitatively new class of problems. We support that using 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. The first three are homeomorphic tasks (H), the next four are reformulation examples (R), and the last four are cross-domain transfer tasks. Figure 6 presents the results of the lesion study. Each data point averages across ten trials of the given target problem for both the TC agent and the NTC agent. 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.

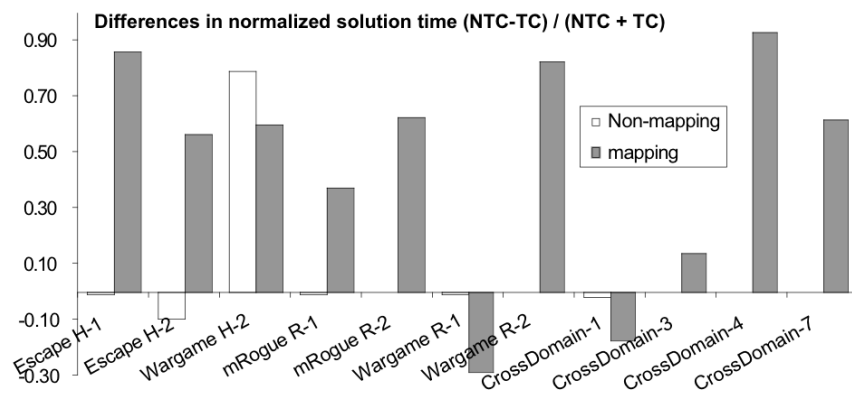


Figure 6. A lesion study showing the impact of representation mapping on transfer

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 a net 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.

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.

In summary, this work developed and demonstrated a capability for far transfer in the context of the Icarus architecture. The underlying representation mapping algorithm operates by explaining solutions found in one domain using the vocabulary native to another. With it, we are able to communicate knowledge between distinct problem domains and across a representational divide, where the symbols describing the domains need not be shared.

2.3. Transfer from Recognition into Play Improvement in American Football

While the Year 1 and Year 2 demonstration tasks intentionally used controlled settings to foster the development of transfer technology, the Year 3 task required the solution methods to make contact with real world data. This demonstration also bridged families of tasks, as it was to recognize American football plays from video and improve their performance in simulation.

We apply the Icarus cognitive architecture to this problem in the context of a larger processing pipeline. The net system takes in discrete perceptual traces generated from video footage of college football games, analyzes the traces with provided background knowledge, learns complex behaviors for individual players from that data, and employs those behaviors to drive agents in a simulated football game in conjunction with an optimization method to improve play performance. The acquired knowledge is compos able, which lets Icarus generate team behaviors not observed in the video, and is encoded, in a human interpretable format, which allows subsequent analysis and modification by human programmers. We discuss the key elements of this pipeline below.

2.3.1. The College Football Video

The raw videos used in our experiments depict individual plays as executed by the Oregon State University football team. The video was shot via a panning and zooming camera that is fixed at the top of the stadium. The video corresponds to that used by coaches, and the camera operator attempts to keep as much of the action in view as possible. A typical shot from our video is shown in Figure 7.

Icarus does not possess any specialized mechanisms for handling visual data. We therefore convert preprocessed versions of the videos into a sequence of Icarus perceptions for the architecture to observe. Specifically, the perceptual representation includes: (1) the 2D field coordinates of each player at each video frame, (2) player labels describing the functional role of each player (e.g. quarterback, and running back), and (3) activity labels describing the low-level activity (such as running or blocking) of each player throughout the play. We used the activity labels and player labels generated by Hess, et al. (2007) to produce the perception sequence for the play.



Figure 7. A typical frame from the football video.

2.3.2. The Rush Football Simulator

We employed the Rush 2008 simulator¹ as the environment for improving football plays with transferred knowledge. This version of Rush simulates an eight-player variant of American football, which typically has eleven players per team. Figure 8 shows a typical starting formation for the simulated plays. The rules and objectives in Rush are similar to those in American football. Each player is assigned a role, such as quarterback (QB) or running back (RB), and can be controlled by a set of instructions. Players are assigned a high-level goal such as *pass route cross out at yard 15* (which instructs a receiver to run 15 yards down-field, make a hard right turn, and then keep running to try to catch a pass), or assigned specific instructions such as *stride forward* on each clock tick. The hand-coded plays in our experiments use the former types of instructions to take advantage of the expertise built in to the simulator, while Icarus uses the latter form to demonstrate the architecture's ability to construct such knowledge on its own.

We instrumented Rush so that Icarus can perceive all players and objects on the field and control the actions of each offensive player on a tick-by-tick basis. Each offensive player shares perception and knowledge with the other offensive players, and carries out actions selected by Icarus. Example actions include (*throwTo <receiver>*), which instructs the QB to pass to a specific teammate, and (*stride <direction>*), which tells a player to run in one of eight directions for one clock tick. Defensive players are controlled by the simulator, which randomly selects one of several available strategies for each play.

¹ <http://rush2005.sourceforge.net/>

Table 4. Sample concepts for the football domain.

```
; Indicates that ?agent carried ?ball in the current time step
((possession ?agent ?ball)
 :percepts ((ball ?ball carriedby ?agent)))

; ?passer dropped back ?n-steps after receiving the snap
((dropped-back ?passer ?n-steps)
 :relations (((snap-completed ?passer ?ball) ?snap-start NOW)
              ((possession ?passer ?ball) ?poss-start ?poss-end)
              ((moved-distance ?passer ?n-steps S) ?mov-start ?mov-end))
 :constraints ((<= ?snap-start ?poss-start)
               (<= ?mov-end ?poss-end)
               (= ?mov-end NOW)))
```

Table 5. Sample skills for the football domain.

```
; skill for running downfield ?dist yards, then turning and
; running in ?dir until the ball is caught (by any agent)
((cross-pattern-completed ?agent ?dist ?dir)
 :start ((sccross-pattern-completed-c2 ?agent ?dir))
 :subgoals ((moved-distance-in-general-direction ?agent ?dist N)
            (moved-until-ball-caught ?agent ?dir)))

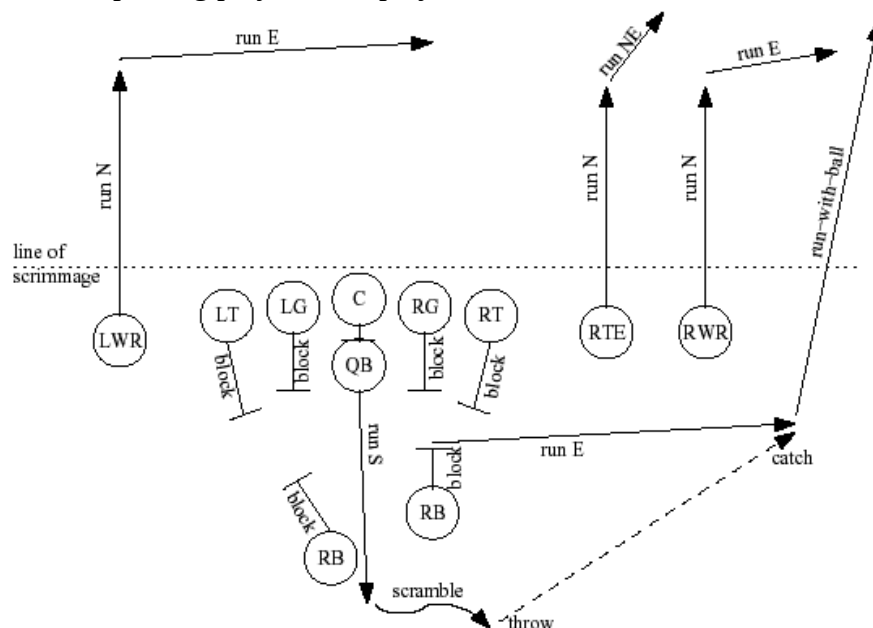
; skill for running a cross pattern, then catching and running with the ball
((cross-reception-completed ?agent ?dist ?dir)
 :subgoals ((cross-pattern-completed ?agent ?dist ?dir)
            (ran-with-ball-until-tackled ?agent ?ball)))
```

2.3.4. Learning Skills from Video

Our method for acquiring hierarchical Icarus skills from video employs an explanation-based approach. Li et al. (2009) provides details, but, in overview, the method inputs a goal, a set of concepts sufficient for interpreting the observed agent's behavior, a set of low-level methods available in the environment, and a sequence of observed perceptual states. Optionally, any known or previously acquired methods may be included.

The algorithm runs in three steps. First, the system observes the video of the game and infers beliefs about each state, storing the results in belief memory as described above. Next, the agent explains how the goal was achieved using both existing conceptual and procedural knowledge. The agent first tries to explain the trace with known skills that both achieve the goal and are consistent with the observations. It then selects one of the candidate skills and parses the observation trace into sub traces based on the times when the start conditions and sub goals were achieved.

The algorithm for acquiring skills is not equivalent to compiling information already contained in the provided concept hierarchy into an operational form. The learning mechanism must recognize which parts of a concept definition are start conditions and which are sub goals. If we simply consider all sub concepts in a definition to be sub goals, then Icarus will believe that the learned behaviors apply in many situations that they do not. The agent would take many fruitless actions as a result.



2.3.5. Mapping Recognized Plays into Rush

As noted, a team in Rush consists of eight players while the teams from the video have eleven players. To compensate, the system must map the eleven-player skills learned from the video into skills for the eight Rush players. It accomplishes this simply by dropping the goals associated with three of the players.

This process is straightforward. Rush plays typically include three offensive linemen (LG, C, and RG) along with some combination of four running backs and receivers. However, the college play shown in Figure 9 has five linemen and five backs/receivers. To map this play into Rush, the system dropped two linemen (LT and RT) and one running back (RB), all of which had top-level goals of pass-blocking for the duration of the play. Icarus used the goals for the eight remaining players to guide execution in Rush.

2.3.6. Parametric Adaptation of Football Plays

We employ a parametric learner to improve the plays recognized from video in simulation. This learner is implemented externally to Icarus. It inputs Icarus skills with certain constant values marked as parameters, plus a reward function, and then executes the skills using different parameter values to observe the reward they return. It updates the skill parameter values to increase predicted future rewards.

In our transfer framework, the arguments occur in the top level goal of each player agent. The reward function is the yardage gained at the end of each play. The system differentiates between ordinal and nominal parameters that have ordered and categorical values, respectively. For example the parameterized top level goal of the quarterback (*pass-completed QB *receiver *qbDropSteps*) can be achieved by searching over values for the nominal parameter **receiver* (the recipient of the pass), and the ordinal parameter **qbDropSteps* (the number of steps the quarterback runs back once he receives the ball). Our implementation assumes a method for specifying the legal range and step size for ordinal parameters and legal values for nominal parameters.

The optimization begins with parameter values initialized from a recognized source play. It proceeds by perturbing one parameter at a time, where that parameter is selected with a probability proportional to its expected reward improvement (specifically, the average reward improvement historically observed when perturbing that parameter). Given a vector of parameter settings, the system executes the skills a preset number of times to perform the task and measure an average reward.

Assuming the system has selected a parameter, it applies a perturbation in a random direction (increasing/decreasing) with a default step-size. If this change substantially improves reward, it selects the same parameter on the next iteration, with the same perturbation direction and step size. However, the parameter learner also maintains a record of the highest reward that it has ever received, along with the parameter vector that produced that reward. The longer the system has gone without setting a new personal best, the greater the perturbation it will apply to the current parameter value. Thus, the system searches further and wider until it sets a new personal best, or until it reaches the minimum or maximum limit of the parameter, in which case it resumes the search with the initial step size. The algorithm can be terminated at any time, at which point it returns the best parameter vector found.

2.3.7. Results on Transfer from Recognition into Play Improvement

The Year 3 demonstration task measured transfer from recognition to play improvement across a corpus of 20 videos of football passing plays. The object was to compare the benefit for play design of having seen and analyzed actual plays against the prospects for play design having seen none. As a result, the demonstration compared yardage improvement over the play design task in two conditions:

- Transfer Agent (TA): inputs to play improvement were obtained from video data.
- Non-transfer Agent (NTA): input to play improvement was specified by hand, encoding a naïve play where all eligible receivers run downfield to receive a pass.

Note that the experiments measured learning on the design task, not the impact on initial performance from exposure to the source plays. In more detail, we compared learning curves obtained from parameter adaption that measured average yardage gain as a function of experience for both the TA and NTA conditions. The protocol in the TA condition spanned presentation of the source video, meaning that any failure in the video processing, recognition, and transfer chain would result in a zero reward signal for that example.

Figure 10 illustrates the results of the Year 3 demonstration, which were collected by an independent evaluation team. The TA curve measures play improvement obtained over all 20 source videos, while the curve labeled TA on Holdout shows improvement on 6 of the 20 videos that were never made available to ISLE. The results demonstrate substantial improvements in learning via transfer. The magnitude of the effect can be measured by the regret score, which computes the area between two curves divided by the bounding box (it is a unit-less number between -100 and 100 percent). Here, regret for the TA vs. NTA agents is 61.7 with a pvalue of 0.0004, and regret for the TA on Holdout vs. NTA agents is 55.6 with a pvalue of 0.0002.

Because these results spanned all phases of video processing, recognition, skill acquisition, mapping, and play adaption, they do not clarify the source of power for the transfer effect except in very aggregate terms. We conducted a number of more specific studies to provide that clarification, which are available in separate publications (Straccuzzi et al. 2009; Könik et al. 2010). The summary is that system’s ability to acquire and transfer structured knowledge in the form of hierarchical Icarus concepts and skills is key to that benefit, as it provides an organizing framework for all future adaptation.

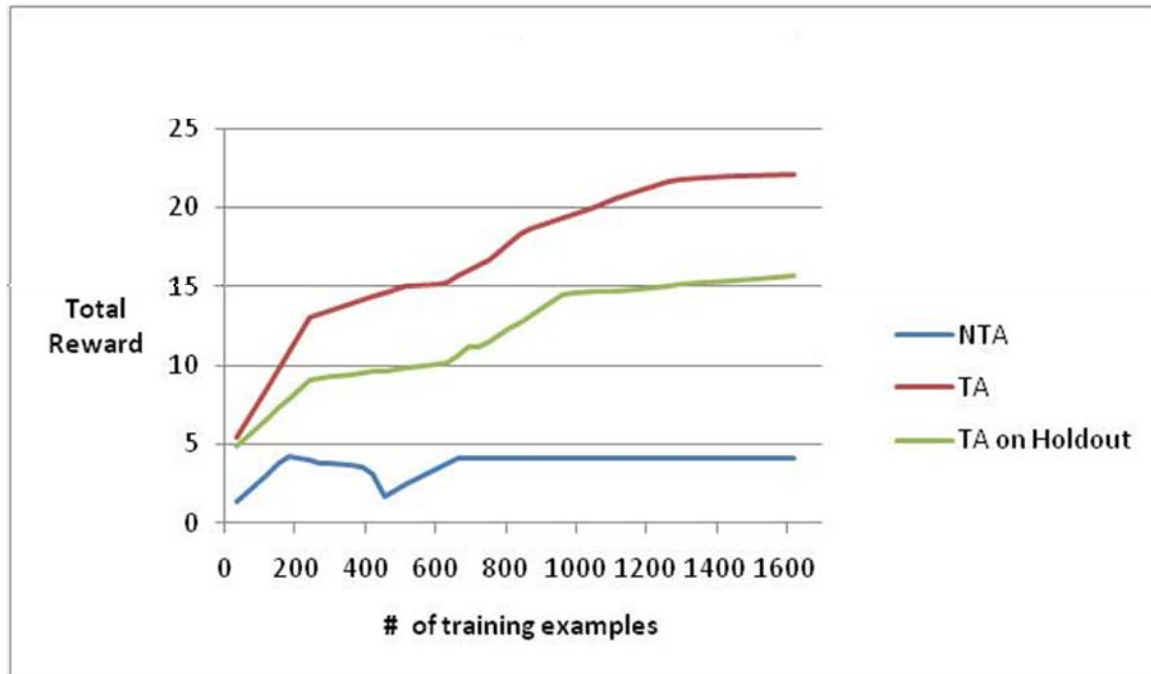


Figure 10. Average performance results when learning to improve American football plays, with and without knowledge transfer from play recognition.

3. Conclusions

The Transfer Learning project was a basic research effort whose goal was to enable machines to acquire knowledge in one domain and use it to improve performance in another. In the course of this effort, ISLE developed technology for both *near* and *far* transfer, which can be distinguished by the conceptual and representational distance between the source and target tasks. Our research focused on methods that could utilize a small number of experiences to acquire and transfer significant structural knowledge among tasks set in a logic-oriented framework. We developed techniques that acquired hierarchical relational knowledge using explanation based learning techniques, and that transferred that knowledge using analogical representation mapping.

More broadly, our research produced several key conclusions:

- Explanation based learning techniques support relational and hierarchical knowledge acquisition across multiple logic-oriented domains.
- Acquired knowledge can be automatically generalized and expressed in a form that natively supports *near* transfer, i.e., use within different tasks set in the same environment/problem domain.
- Analogical representation mapping supports *far* transfer, i.e., reuse in disparate domains, providing the objects, descriptions, and/or solutions are structurally similar.
- The capacity for, and benefit of transfer is roughly proportional to the similarity between the domains.

As the state of the art advances, we can expect transfer learning technology to address larger distances between source and target tasks, support transfer between new families of tasks and among solution methods, bridge greater representational divides, and discover and exploit deeper similarities that link source and target domains (among other developments). While the field is in its early stages, the transfer learning program has played a major role in establishing and popularizing this area of research, while laying the groundwork that will support a wide variety of future advances.

4. References

1. Choi, D., Könik, T., Nejati, N., Park, C., & Langley, P., Structural Transfer of Cognitive Skills. (2007). In Proceedings of the Eighth International Conference on Cognitive Modeling, pp. 115 – 120. Oxford, UK: Taylor & Fransis/Psychology Press.
2. Falkenhainer, B., Forbus, K. D., and Gentner, D. (1989). The Structure-mapping Engine: Algorithm and Examples. *Artificial Intelligence* 41(1): 1-63.
3. Genesereth, M. R., Love, N., and Pell, B. (2005). General Game Playing: Overview of the AAAI Competition. *AI Magazine* 26(2): 62-72.
4. Hess, R., Fern, A., and Mortenson, E. (2007). *Mixture-of-parts pictorial structures for objects with variable partsets*. In Proceedings of the Eleventh IEEE International Conference on Computer Vision. Rio de Janeiro, Brazil: IEEE Press.
5. Holyoak, K. J., and Thagard, P. (1989). Analogical Mapping by Constraint Satisfaction. *Cognitive Science* 13: 295-355.
6. Könik, T., Ali, K., Stracuzzi, D.J., Li, N., Shapiro, D. (2010). Improving structural knowledge transfer with parametric adaptation. The 23rd Florida Artificial Intelligence Research Society (FLAIRS) Conference. Daytona Beach, FL: AAAI Press.
7. 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.
8. Li, N., Stracuzzi, D. J., Langley, P., and Nejati, N. (2009). *Learning hierarchical skills from problem solutions using means-ends analysis*. In Proceedings of the 31st Annual Meeting of the Cognitive Science Society, Amsterdam, Netherlands.
9. Nau, D., Cao, Y., Lotem, A., and Munoz-Avila, H. (1999). *SHOP: Simple hierarchical ordered planner*. Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (pp. 968–973). Stockholm: Morgan Kaufmann.
10. Nejati, N., Langley, P., and Könik, T. (2006). *Learning hierarchical task networks by observation*. Proceedings of the Twenty-third International Conference on Machine Learning (pp. 665–672). New York: ACM Press.
11. Shapiro D., Könik T., & O'Rorke P. (2008). Achieving far transfer in an integrated cognitive architecture. Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (pp. 1325 – 1330). Menlo Park, California: AAAI Press.
12. Stracuzzi, D. J., Cleveland, G., Könik, T., Shapiro, D., Molineaux, M., Aha, D., and Ali, K. (2009). *Constructing Game Agents from Video of Human Behavior*. In Proceedings of the Fifth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), Stanford, CA.

5. List of Acronyms

AI	Artificial Intelligence
AP	Advanced Placement
GGP	General Game Playing
HTN	Hierarchical Task Network
NTC	non-transfer case
TC	transfer case
UCT	Urban Combat Testbed

Appendix A: Learning to Annotate Video of American Football Plays

Alan Fern

Oregon State University

A.1. Introduction

Oregon State University joined the ISLE team in year 3 of the transfer learning program and was responsible for the source learning task. This task involved learning to form semantic descriptions of football plays from raw video. The work involved two primary phases. The focus of the first phase was on data collection, data format specification, and the development of tools for manual annotation of video. The focus of the second phase was to develop learning algorithms for learning to automate the annotation process from raw video. Below we summarize the work in these phases and our key scientific advances.

A.2. Data Collection

In collaboration with the Oregon State University football team game video was obtained for three home football games. Each game included videos of about 50 offensive plays of the OSU team. From these plays the Oregon State team selected videos of twenty plays to serve as the source data for the transfer learning program. The plays were constrained to be successful passing plays and were selected to depict a variety of different formations, passing patterns, and results. Figure A-1 shows a typical frame from one of the videos. Note that this video is the same video used by coaches and is representative of what most college and professional teams use for analysis.



Figure A-1. Frame from the initial moments from one of the source football videos.

A.3. Data Format

In collaboration with the ISLE team, Oregon State defined file formats for describing football plays. Each play was associated with 5 files.

1. **Raw Video File:** the original video of the play.
2. **Track File:** contained data about the x-y field locations of each of the 22 players (11 offensive and 11 defensive) in each frame of the video.
3. **Player Types File:** contained a mapping from numeric player id's, which are used in all data files, to the player position type of the player. The types considered in the source data includes: quarterback, center, right/left tackles, right/left guards, right/left tight ends, right/left wide receivers, right/left wingbacks, fullbacks, tailbacks.
4. **Ball File:** this file provided information about the position of the ball in each frame of the video. The ball could be tagged as one of the following: being in the position of a particular player, being in the air, neither (generally on the ground).
5. **Activity Label File:** this file provided a description of the activities that each player conducted during the video. For each of the offensive players the file specifies a segmentation of the video frames with a semantic activity tag assigned to each segment. A detailed description of the possible activity labels is available with the full source data distribution.

A.4. Manual Data Set Construction

The above data files were manually constructed for the 20 selected plays by Oregon State University. The purpose of providing the manual data was to provide the ISLE team with initial data to work with and to provide training data for the Oregon State University learning work. The most difficult part of the data construction was to provide tracking data, which requires specifying the field location of each player in each video frame, where a typical play can contain 200 to 300 frames. To make this process possible Oregon State developed a key-point tracking tool that allowed for semi-automatic creation of the trajectory data. The tool used video registration techniques developed in prior work by OSU [1] to maintain a mapping between screen coordinates and football field coordinates. A human is then able to scroll through the video and place key points for the location of each player. The system then interpolates between the key points using polynomial regression and displays the track to the user, who can enter more key points if the interpolation is not accurate enough. Using this system it was possible to track the 22 players in approximation 1.5 to 2 hours per play. A screen capture of the tracking tool is shown in Figure A-2.



Figure A-2. Screen shot of the key-point player tracking tool.

A.5. Automated Annotation of Raw Football Video

The second phase of the project involved developing a system that could learn to automatically create the annotation data for football plays from raw video. This included producing all of the data files above automatically, with the exception of the ball file. It was determined before the beginning of Y3 that requiring automated tracking of the ball would have a high risk of failure and thus it was decided to provide the ball information manually.

Figure A-3 shows a block diagram of the automated video annotation system. Raw video is fed into the video registration algorithm [1] and then the registered video is used by the mixture-of-parts model to infer the initial types and locations of each player. The player type and initial position information is then used to initialize the player tracker, which then attempts to track all 22 players throughout the video. The result is a set of player tracks, giving the x-y field locations of the players at each point in the play. These tracks are fed to the activity labeler, which assigns activity labels to each segment of the player tracks. The source learning task included learning to track the players and label the activities as indicated in the figure. Below we overview the components of the system that follow video registration.

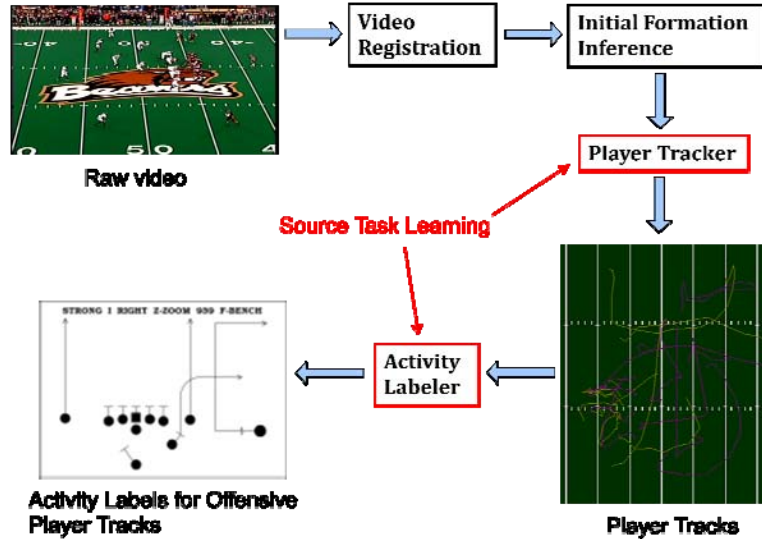


Figure A-3. Diagram of the main components and steps in the automated video annotation system.

A.6. Initial Formation Inference

In order to produce the player type files and to provide an initialization seed for automated player tracking we used an algorithm developed in work prior to the program. In particular, Oregon State used their model called the mixture-of-parts pictorial structures and associated inference algorithms [2]. The complicated aspect of this problem is that player type and initial position information are highly dependent on one another, which requires solving a difficult joint modeling and inference problem. The mixture-of-parts model provides an efficient mechanism for representing these dependencies and for performing the inference. The model is very accurate and performed nearly perfectly for the 20 plays that were selected for the transfer learning project.

A.7. Learning to Track Players

Tracking football players was the most challenging aspect of the Y3 work. This problem encompasses some of the most challenging aspects of multi-object visual tracking. The players being tracked in the football domain move very erratically and the characteristics of a player's motion changes substantially depending on the player's type and the time stage of the video. In the first several seconds of each video, for example, nearly all of the players stand virtually still, while perhaps one or two move only gradually. However, after the ball is snapped (a one-time event that occurs in every football play) all of the players begin to move very quickly, and the tracker must be able to adapt accordingly. In addition, football players interact in complex ways, frequently in very large groups. For example, in every play, a group of five linemen on the offense stand shoulder to shoulder in a line and attempt to block a group of about three to five defensive players who, in turn, attempt to break through the offensive line. Many other complicated interactions take place between smaller groups of players throughout the course of a play. Our initial attempts at implementing and applying state-of-the-art multi-object trackers to this problem resulted in dismal performance.

To solve the tracking problem Oregon State took a supervised learning approach, where a tracker was trained based on training data depicting ground truth tracks for a number of videos. The tracker architecture was based on pseudo-independent particle filters, where each player was associated with a particle filter that tracked the player's location. The trackers were independent with the exception that they were allowed to use information about the state estimates of nearby players in the previous frame when making inferences about the current frame. In this sense the trackers were only pseudo-independent, rather than fully independent.

The key idea behind the approach was to parameterize the particle filters by log-linear probability models that were used to rank particles proposed by a fixed proposal distribution. This framework represents a strict generalization of the standard Bayesian filter paradigm, but offers the advantage of allowing for discriminative training of the model and for incorporating complex features into the model. The key problem is to learn the weighting coefficients of the log-linear filter models with the primary goal to successfully track all the players. For this purpose, Oregon State introduced a novel error-driven training approach which was tightly integrated with the filtering process. In particular, learning proceeds by actually running the filters until a filter is determined to be in error (too far away from the ground truth). When a filter is in error its parameters are updated in a way that attempts to prevent a similar error in the future. The filter is then reset to the ground truth and tracking continues.

The full model and learning algorithm are described in a recent paper [3], which is included in the appendix. The empirical results were a huge improvement over the prior state-of-the-art systems, reducing the tracking failure rate by a factor of approximately 0.5 and reducing the localization error by approximately a factor of 2/3.

A.8. Learning to Label Activities

Given player tracks Oregon State developed an instance-based learning algorithm for segmenting the track of each offensive player and labeling the segments by the appropriate label. The outline of the approach for labeling a given query track is quite simple: 1) Find the “most similar” track in labeled data set to the query track, 2) Find the best segmentation of the query track that best matches the segmentation of the most similar track, 3) Return the query track with the segments computed in step 2 labeled by the activities in the most similar track.

The key aspect of the approach that must be specified is how to judge the similarity of tracks and how to perform the best segmentation. For this purpose, Oregon State used the well known dynamic time warping (DTW) algorithm as both a segmentation and similarity computation. The DTW algorithm is intended to find correspondences between numeric time-series where the time scales may be warped relative to one another, perhaps not uniformly. Such warpings are often observed in football trajectories since players move at different speeds and for different amounts of time during each activity, however, the general characteristics of the trajectories are often similar for similar activities. DTW between two football trajectories can be computed via dynamic programming and is very efficient. The key step in getting good results was to design the similarity measure for individual time points, which is used by DTW to judge the goodness of a warping between two signals. For this purpose we defined a number of feature functions that operated on pairs of points and weighted those features through a trial an error process on a validation set to maximize performance. Given this function we then applied DTW between a query and all trajectories in our data base to find the best match and then used the segmentation computed by DTW in order to assign labels to the segments of the query. Figure A-4 shows some examples of queries and the corresponding best matches found by DTW for trajectories corresponding to passing patterns.

The system performed perfectly for lineman due to the uniformity of their activity. For receivers and backs the performance was more varied. Generally, the segmentation boundaries were quite accurate, though rarely matching the training data exactly. However, there were pairs of segment labels that would frequently be confused due to the qualitative similarity between them. For example passing patterns such as slant and cross can often look quite similar, and are even difficult for human labelers to agree upon in many cases. Most errors were of this near miss variety, with fewer errors being more serious, e.g. classifying a passing pattern as a pass blocking activity. The final results, however, represent the first end-to-end system for labeling football plays at any level of accuracy and were sufficiently accurate to serve as good source data for the ISLE team.

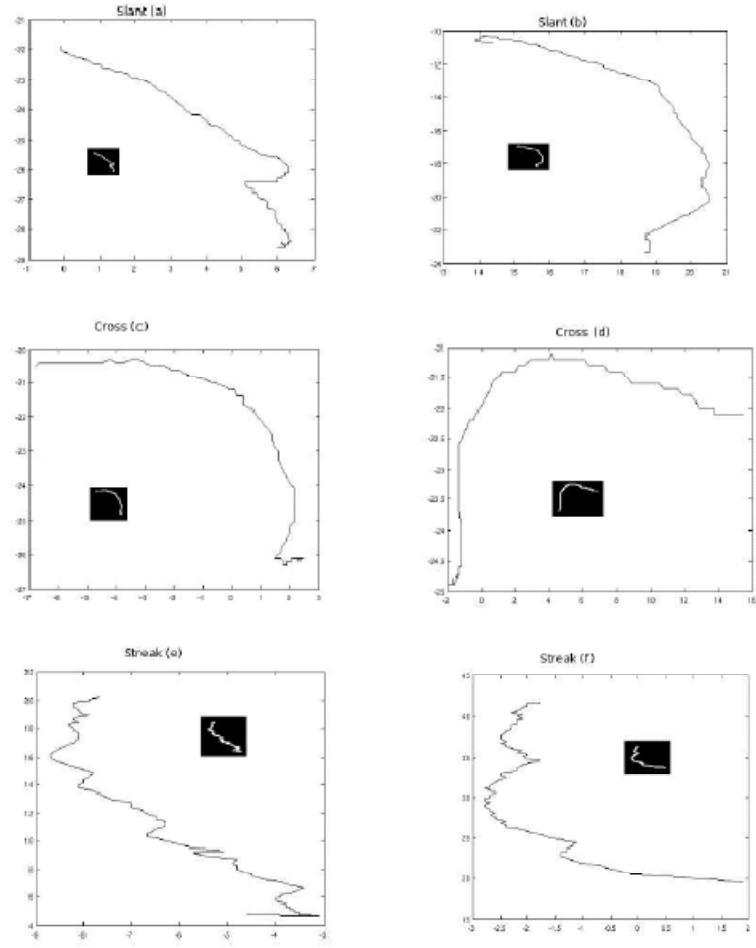


Figure A-4. Some example query trajectories (left column) corresponding to individual passing patterns and the best matches returned by the DTW algorithm (right column).

References

- [1] Robin Hess and Alan Fern. (2007). Improved Video Registration using Non-Distinctive Local Image Features. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2007)*.
- [2] Robin Hess, Alan Fern, and Eric Mortenson. (2007). Mixture-of-Parts Pictorial Structures for Objects with Variable Part Sets. *IEEE International Conference on Computer Vision (ICCV-2007)*.
- [3] Robin Hess and Alan Fern. (2009). Discriminatively Trained Particle Filters for Complex Multi-Object Tracking. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2009)*.

Appendix B: Skill Learning and Activity Recognition for Transfer

David J. Stracuzzi

Arizona State University

B.1. Background and Motivation

The primary role of Arizona State University in the transfer learning project has been to support ISLE in expanding the ICARUS architecture to meet performance goals. More specifically, ASU has focused on expanding ICARUS' representational and inference capabilities to support temporal reasoning, and on improving the architecture's existing skill learning mechanisms to support the new representational capabilities. In the context of the American football domain, this means that ASU's work focused almost exclusively on the dual source tasks of interpreting the plays depicted in the video footage, and acquiring skills to execute the plays based on those interpretations.

The following report provides an overview of the play recognition and skill learning tasks, along with a summary of the key ideas employed by ASU's approach to them. The related tasks of mapping the learned plays into the Rush simulator (target domain) and adapting those plays to improve performance were performed by ISLE and are described elsewhere. We begin with a brief review of the source and target environments. We then discuss the play recognition task in detail, including temporal extensions to ICARUS, provided background knowledge, and our approach to parsing the play videos. Next, we describe our approach to skill learning, which relies heavily on the results from the play recognition methods. We also illustrate both the recognition and learning tasks with a specific example drawn from one of the videos. Finally, we discuss the implications of our approach for transfer of learning along with relationships to other work, and conclude remarks on possible next steps.

B.2. American Football

The specific transfer task considered by the ISLE team was to first observe video of twenty college football plays, learning to recognize and execute the plays, and then to execute and adapt those plays in a simulated football environment. ASU's role in the transfer task fit between those of Oregon State University and ISLE. OSU focused on low-level aspects of play recognition, such as determining the role, location and direction of each player in each frame. ASU used these results to generate a symbolic description of each play on a frame-by-frame basis. This symbolic description then served as the perceptual input to ICARUS, which in turn parsed the observed frames into high-level beliefs about the specific actions and goals pursued by each player on the field, and skills for executing the observed plays in the simulator.

B.3. Observed Videos

As noted, ASU's first step was to construct a symbolic description of each video, since ICARUS is not well suited to handling raw video data. Each frame includes the current time (frame number), the identity of the player in possession of the ball, and information on each player, including his identity, role, location coordinates, direction of motion, team, and current action. The last of these is derived from the label assigned to the player by OSU. ASU parses each play with a more detailed language than that used by OSU, so the original labels assigned by OSU are simply mapped into a set of low-level actions, such as *blocking*, *runningto*, and *waiting*. Note that these actions corresponded only approximately to the executable actions available in Rush.

B.4. The Rush Simulator

In Rush, ICARUS controls only the offensive players, each of which has a unique role in the simulator. Rush provides percepts for all of the players and objects on the field. The offensive players and the agents that control them, all share the same set of percepts. Like the percepts generated from the videos, these include the location, direction and low-level action of each player on the field (including the defense) on each tick. Also included is the location of the ball, which is indicated either by player currently in possession or AIR if the ball has been thrown, but not caught. The simulated players are controlled by assigning them a one or more instructions either before or during play. Rush provides a set of low-level commands, such as *stride north* or *throwto RWR*, that ICARUS uses to control the simulated players on a tick-by-tick basis. The simulator controls the defensive players by selecting one of several defensive strategies at random.

B.5. Performance Goals

ASU had two performance objectives in learning skills from the observed videos. The learned skills needed to demonstrate high fidelity with respect to original video, and needed to exhibit utility consistent with the observed videos when executed in the simulator. Together, these imply that the architecture has captured the key features of the observed play. More importantly, high fidelity skills that exhibit reasonable utility provide a manageable starting point for learning in the simulated environment. Skills that exhibit poor utility, or do not behave reasonably, require far more adaptation in the simulator than skills that perform well. This is undesirable from both a computational point of view, because adaptation in the target is expensive, and from a transfer point of view, because more demand for adaptation in the target implies that less information was transferred from source. This is an important insight that allows qualitative evaluation of transfer.

B.6. Play Recognition

Play recognition is a critical component of learning in the source domain for ICARUS. The goal is to highlight specific events and relationships among the players depicted in the video. This is a generalization of ICARUS' previous recognition abilities, which focused on detecting non-temporal relationships among the objects in a domain. Temporal events and relationship are of particular importance in football, as many of the behaviors executed by players depend on specific temporal orderings of events and relationships.

For example, consider a five-yard cross-left receiver's pattern, in which the receiver runs downfield for five yards, turns 90 degrees left, and then continues to run until the ball is caught. Notice that the receiver could eventually reach the same point by first running across the field, and then running downfield. These two approaches result in various running patterns, however, each can have important consequences. Running across the field first might send a receiver through the quarterback's pocket, which may disrupt both the offensive linemen, and the receiver. Likewise, the fact that the receiver finishes his pattern in the same location is only of consequence if the quarterback throws the ball to him at that point. If the quarterback throws sooner, then the receiver will be out of position with respect to the ball. In the remainder of this section, we provide an overview of the mechanisms that ICARUS uses to represent and reason about temporal relationships [3].

B. 7. Temporal Extensions to Icarus

Several architectural modifications are required for encoding and reasoning time in ICARUS. The first focuses on encoding temporal information with beliefs by including start and end time stamps, which indicate the first and last times at which a belief held. ICARUS already maintains an internal notion of time, based on cognitive cycles, that is now used to set these values. This augmented representation lets ICARUS distinguish beliefs about past events from ones about the present.

A second extension is to expand the temporal scope of belief memory by retaining all of the beliefs held throughout an episode. This is equivalent to providing the architecture with an episodic belief memory, whereas previously belief memory included only those beliefs that held on the current cycle. All beliefs contained in the episodic memory are generated through inference, which is based on the agent's percepts, so belief memory maintains a record of experiences in the environment *from the agent's perspective*.

The importance of episodic memory is well established, but the memory alone provides little improvement to an architecture's capabilities. To exploit this memory, ICARUS' concept language also requires modification. First, the `:relations` field, which lists the lower-level concepts that support a higher-level definition, expands to reference the time stamps assigned to beliefs. Second, we add a new `:constraints` field that represents simple arithmetic tests over time values referenced in the `:relations` field. Thus, this field lets ICARUS apply temporal constraints to the relationships established among the relations.

The architecture's inference process also expands to support the changes in belief and conceptual memories. The fundamental mechanism, which computes in a bottom-up manner the deductive closure of conceptual memory with the belief and perceptual memories, remains unchanged. The only difference is that the time stamps and temporal constraints must be matched in addition to the percepts and relations fields. No new specialized control is required.

B.3.2. Background Knowledge

Given the ability to recognize temporal relationships among objects and events, the next step toward parsing football plays is to add background knowledge about football to ICARUS. The knowledge is encoded using ICARUS' concept language, such that each concept describes a class of environmental situations and potentially includes temporal relationships among the referenced objects and events. Importantly, the background conceptual knowledge captures state-like descriptions of plays, and does not contain any information about how such states should be achieved. Learning procedures for achieving these states and events therefore remains a challenge as discussed below.

B.3.3. A Play Parsing Example

Consider now an example drawn from one of the source domain videos. Figure B-1 (a) shows the paths followed by all eleven offensive players, while Figure B-1(b) shows the corresponding play diagram with labels for each player's high-level goal. The recognition task is then to observe the symbolic encoding of the play and construct a set of beliefs that capture the information shown in the diagram. In particular, ICARUS relies on highly structured knowledge representations, so the resulting beliefs about the play will contain information at a variety of levels of abstraction.

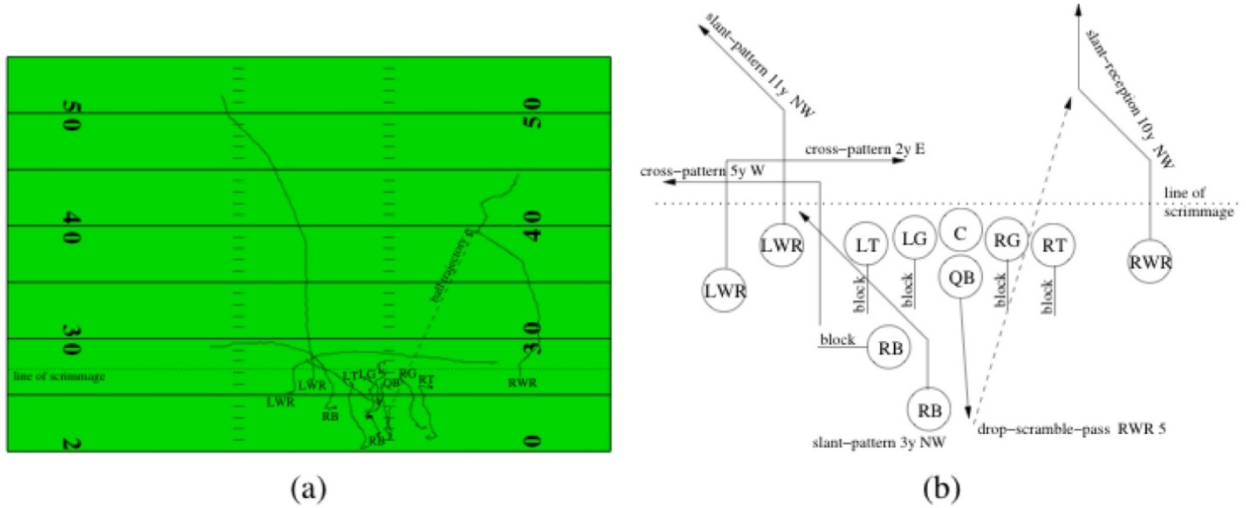


Figure B-1. Paths followed by offensive players (a) and corresponding play diagram with high-level labels for each player's goals (b) for one of the videos.

Table B-1 shows a subset of beliefs related to the quarterback inferred by ICARUS over the course of the play. Some of the beliefs indicate simple events, such as **SNAP-COMPLETED**, which holds at all times after the center snaps the ball to the quarterback. Other beliefs indicate specific properties of a state or sequence of states, such as **POSSESSION**, which indicates the range of times over which the quarterback had possession of the ball. Note that ICARUS performs play recognition in an online manner, meaning that the architecture attempts to determine the salient properties of the sequence as it unfolds, rather than through a post-hoc analysis. As a result, ICARUS sometimes generates beliefs about events and processes that have not yet completed. For example, there are **MOVED-DISTANCE** and **DROPPED-BACK** beliefs for each yard of progress in the quarterback's drop. This follows from ICARUS' lack of a priori knowledge about the intended drop distance. Also notice the structural relationships between

beliefs that hold earlier in the play, and those that hold later. For example the sequence of events DROPPED-BACK, SCRAMBLED, and PASS-COMPLETED combine to form the higher-level event DROP-SCRAMBLE-ONE-SECOND-PASS-COMPLETED, which represents the top-level goal for the quarterback in this play.

Table B-1. A selection of inferred beliefs about the quarterback for the play shown above.

(SNAP-COMPLETED QB BALL1)	140	NOW
(POSSESSION QB BALL1)	140	236
(MOVED QB S)	140	191
(MOVED-FROM QB 6.6 -24 S)	140	191
...		
(MOVED-DISTANCE-IN-GENERAL-DIRECTION QB 4 S)	177	186
(MOVED-DISTANCE-IN-GENERAL-DIRECTION QB 5 S)	187	191
...		
(DROPPED-BACK QB 4)	177	186
(DROPPED-BACK QB 5)	187	191
(SCRAMBLED-FOR-ONE-SECOND QB)	215	229
(THREW-BALL QB BALL1)	237	237
(PASS-COMPLETED QB RWR)	259	259
(DROP-SCRAMBLE-ONE-SECOND-PASS-COMPLETED QB RWR 5)	259	NOW

B.4 Learning Skills from Observed Behavior

The ICARUS architecture has traditionally learned new skills in the context of problem solving. The architecture executes in a given domain until it reaches an impasse, or point at which no known skills are capable of achieving the agent's goals. In response, the architecture resorts to backward-chaining search from the goal. Though effective, this approach can become computationally expensive when there are many possible paths to consider. The football domain offers an almost unlimited set of possibilities, since ICARUS controls all eight offensive players in the Rush simulator on a tick-by-tick basis. With this in mind, ASU converted the source task of video play recognition into an opportunity to learn skills for target based on the observed behavior of the human players.

Our approach to skill learning from the observed video runs in three steps. First, the system observes the video of the game and infers beliefs as described above. Next, the architecture attempts to explain how the goal was achieved using both existing conceptual and procedural knowledge. Finally, the architecture uses the explained events to construct new skills, which can then be executed in the simulator. We summarize briefly the skill learning methods below, leaving a more detailed discussion for Li et al. [2].

B.4.1. Generating Explanations

The objective of the explanation process in ICARUS is to show how the states in the observed sequence relate to each other, and how the progression relates to the goals and sub goals of a given agent. The explanation therefore provides important insight into the procedures that an observed agent must have applied in order to achieve its goals. In ICARUS, the observing agent tries to explain the observed trace using a combination of known skills and concepts.

ICARUS first tries to explain its observations using existing skills by retrieving any skills that can achieve the top-level goal that was identified during play recognition. Skills whose start conditions are not consistent with the observed trace are ignored. The agent then selects a candidate skill and parses the observation sequence into subsequences based on the times when the start conditions and sub goals were achieved (using the temporal beliefs). The explanation process then recurses on each subsequence (corresponding to a start condition or sub goal) until the entire observation sequence is explained by a sequence of primitive skills.

If ICARUS fails to explain the observation sequence using skills, it attempts to explain it using concepts. Using the belief corresponding to the observed goal and the lower-level beliefs that support it, the agent divides the observed sequence into temporal subsequences. As with skill-based explanations, the agent recursively attempts to explain these sub goals. The observation sequence is considered successfully explained if either (1) there exists a skill for the goal that is applicable at the beginning of the trace, in which case there is no need to learn, or (2) all the sub goals of the current trace have already been successfully explained. Figure B-2 shows a partial explanation for the quarterback based on the beliefs shown in Table B-1.

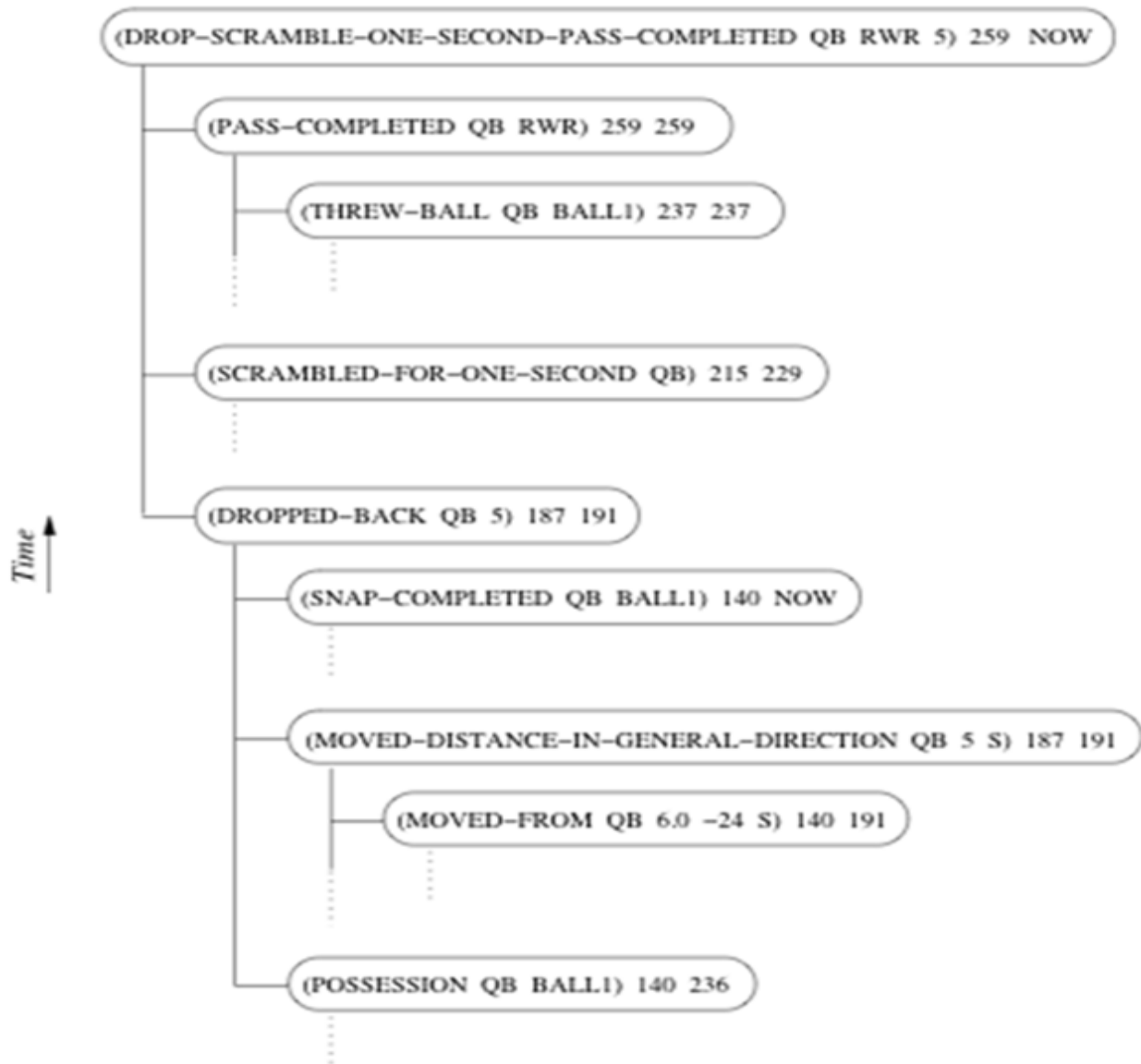


Figure B-2. A partial explanation for the quarterback's observed behavior.

B.4.2. Constructing Skills

After explaining an observation sequence, the agent can learn skills for the achieved goals. These new skills then let the agent achieve similar goals in the future under similar conditions without additional problem solving. Note that new methods are constructed on top of existing methods based on the derived explanations. In the context of football, this means that the number of new skills acquired by observing a sequence of plays diminishes rapidly to zero, since only previously unobserved behaviors require additional skill learning. We consider this important point further in Section 0.

New skills are generated in two ways, depending on how the explanation was constructed. If the explanation is based on skill knowledge, then the sub goals of the new skill are the chronologically ordered start conditions and sub goals from the skills that provided the explanation that were achieved during the observed sequence. The start condition for a new skill includes the start conditions and sub goals that were true before or during the first cycle of the observed subsequence.

If the explanation is based on conceptual knowledge, then the learner retrieves the concept definition corresponding to the goal. The sub goals of the new skill correspond to the sub concepts that were achieved during the observed sequence in chronological order. The start condition of the new skill includes the generalized lower-level beliefs (sub goals from the explanation) that were true before or during first cycle of the observed subsequence.

Note that the methods for acquiring skills described above are not equivalent to compiling information already contained in the provided concept hierarchy into an operational form. The learning mechanism must recognize which parts of a concept definition are start conditions, and which are sub goals. If we simply consider all sub concepts in a definition to be sub goals, then ICARUS will conclude that the learned behaviors apply in many situations that they do not. The result would be that agents take many fruitless actions in the environment. Similarly, the temporal constraints in a concept definition only provide partial ordering information about the sub goals in a new method. Although the concepts in football tend to provide a total ordering over the sub goals, this is not generally true, and our approach does not rely on this property.

B.4.3. Learning Skill Preconditions

The final step of skill construction is to determine the skill precondition, which specifies the environmental conditions that must hold in order for the skill to apply, and which may depend on temporal relations. In particular, many concept definitions include information about the temporal relationships among its sub concepts, and therefore the skill's sub goals. The learned skill should therefore retain this information in its start condition.

ICARUS does not currently support temporal conditions in skill definitions. Instead, the agent constructs a new concept to represent the start condition by extracting information from the original concept definition. Note that the learned start condition concepts describe both what must be true in the current state and conditions that must have held in the past to ensure that the skill applies. Our learning mechanism is incremental in the sense that constructed concepts and skills can serve as domain knowledge for future trace explanation. This enables our mechanism to acquire increasingly complex domain knowledge over time.

B.5. Discussion and Conclusions

The ICARUS-based approach for representing and acquiring transferable football knowledge is based on three principles. The first is abstraction. The temporal representation described above allows the architecture to abstract away many unimportant details of the source and target domains, such as specific attributes of the individual players, so that both background and learned knowledge transfer easily from source to target. Although experiments such as those conducted by Li et al. [1] show that some adjustment in the target is necessary, the initial, untuned performance of the transferred skills is reasonable with respect to the domain.

A second principle relates to high-fidelity learning. The skills learned by ICARUS from the source video reproduce the observed behavior quite faithfully in the Rush simulated. In some cases, the learned skills reproduced the observed plays with greater fidelity than hand-crafted agents based on Rush's built-in play-control commands. When combined with abstract knowledge representations, this should result in learned skills that perform robustly across a variety of offensive and defensive team configurations. Pilot experiments suggest that this is true, though we have not yet confirmed this with detailed, formal experiments.

The third principle is that the learned skills should compose to form new, unobserved behaviors to the extent possible. Ideally, ICARUS should be able to observe a variety of different behaviors in a variety of situations, and then combine the skills learned from these observations in new ways. This allows the architecture to increase the breadth of its execution capabilities much faster than if it needed to view the combinations before executing them. Experiments from the football domain suggest that ICARUS can compose behaviors without prior observation. For example, the number of new skills learned from a given video observation drops rapidly toward zero as the number of observed videos increases. This implies that ICARUS is able to explain (and by extension execute) previously unseen play combinations based primarily on existing skills. This ability has important consequences for transfer, since new domains will often require agents to apply known skills in ways that have not been previously considered.

Next steps for this work include expanding the architecture to acquire the concept definitions currently provided as background knowledge automatically from the observed video traces. This represents a significant expansion of the ICARUS architecture, which has not traditionally focused on inductive learning, and would operate in a complementary fashion to the analytical concept learning methods employed as part of the skill learning techniques described above. Expansion into concept learning will also play an important role in adding detail to the conceptual knowledge currently available to architecture. For example, the football agents currently have limited knowledge of interaction with the defensive players. As a result, they typically do not make significant efforts to avoid being tackled. Increasing the resolution of the agent's conceptual knowledge would in turn allow the agent to increase the resolution of its skill knowledge, such as by learning to out-manuever oncoming tacklers after receiving a pass. Such expansions would allow agents to take advantage of existing architectural properties, such as reactivity to the physical world, which then further improve ICARUS' ability to transfer learned knowledge among domains.

In summary, ASU's work on the transfer project focused on representation and learning in the source domain. The representation of temporal relationship plays a key role in the architecture's ability to recognize and capture information about football from the source video. Likewise, the expanded skill and precondition learning methods are important for both capturing temporally constrained procedural knowledge, and for allowing the architecture to avoid backward-chaining problem solving in the target domain, which would have been intractable. Finally, we concluded that the most important step toward making ICARUS' ability to transfer learned knowledge more robust is expanding the architecture's ability to identify and learn new concepts based on either experience in the source domain, or observed behaviors. These would allow agents to expand their understanding of the source problem, thereby giving the agents additional material on which draw in the target.

References

1. Li, N., Stracuzzi, D. J., Cleveland, G., Könik, T., Shapiro, D., Molineaux, M., & Aha, D. W. (2009a). Constructing game agents from video of human behavior. *Proceedings of the Fifth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Stanford, CA: AAAI Press.
2. Li, N., Stracuzzi, D. J., Langley, P., & Nejati, N. (2009b). Learning hierarchical skills from problem solutions using means-ends analysis. *Proceedings of the 31st Annual Meeting of the Cognitive Science Society*. Amsterdam, Netherlands: Cognitive Science Society, Inc.
3. Stracuzzi, D. J., Li, N., Cleveland, G., & Langley, P. (2009). Representing and reasoning over time in a symbolic cognitive architecture. *Proceedings of the 31st Annual Meeting of the Cognitive Science Society*. Amsterdam, Netherlands: Cognitive Science Society, Inc.

Appendix C: Hierarchical Task Networks in Transfer Learning

Dana Nau, Ugur Kuter
University of Maryland College Park

C.1 Introduction

This report describes the research conducted at University of Maryland under DARPA's Transfer Learning contract. We have worked on two main research topics: *learning Hierarchical Task Networks (HTNs)* and *combining heuristic search techniques in HTN planning*. We have developed new algorithms in these topics and published our results in major conferences such as IJCAI, AAAI, and ILP. We have implemented and integrated our planning and learning algorithms in Transfer Learning systems developed by ISLE under this contract. We describe the impact and the results of our work in this program.

C.2. Summary of Research Conducted

One of the objectives of transfer learning is transferring knowledge and skills learned from a variety of previous situations, called *source* problems, to the current, previously unencountered problems(s), called the *target(s)* (where significant differences may exist between these two problem types). For this, a transfer learner needs to have some knowledge about the underlying characteristics of both source and target problems. Transfer can be especially effective when such knowledge can be represented suitably structured, e.g., in a relational fashion as in reinforcement learning and/or in a hierarchical fashion as in Hierarchical Task Networks (HTNs).

In our work, we have been using HTNs as a means for describing higher level problem-solving knowledge and studying how to generate plans with HTNs as well as how to learn such HTN-based planning knowledge for the purposes of transfer. The subsequent sections summarize our main research accomplishments under the Transfer Learning contract.

C.3. HTN Planning for Guiding Transfer Learning

In transfer learning scenarios, planning with HTNs typically take place both in the source planning domain and in the target planning domain. The former aims to support the learning process by producing plans as learning examples. The latter provides an evaluation framework (i.e., a simulation of execution) for the transferred knowledge. In either case, the HTNs are learned by machine-learning algorithms that depend on the observations in the world. In many planning domains, however, observations to learn HTN knowledge may not be completely available and thus, the learned HTNs do not typically have an expert-level quality.

Our objective in this direction was to develop ways to use heuristic guidance to replace some of the HTN based knowledge that would otherwise be needed for an HTN planner to generate solution plans. In automated planning, a planner that can use domain-independent heuristic information (e.g., FF [6], AltAlt [9], SGPlan [2], HSP [1], FastDownward [5], and LPG [4]) usually does not need expert-provided domain knowledge, since the planner itself computes a heuristic for each domain. This makes the domain formalization simpler and the planner easier to use; but the planner may often perform much worse than a planner that exploits specific domain knowledge.

We have developed three approaches that combine the two schools of thought above in different ways.

Integrating Heuristic Selection in SHOP2. In this work, we extended the definition of an HTN method (i.e., an operational procedure that describe how nonprimitive tasks are decomposed into their subtasks) to include a *goal expression*, i.e., a single logical atom that will be true in the state of the world when/if all of the subtasks are successfully accomplished in the current state.² A method's goal expression describes the possible goal states reachable by the SHOP2 planner from the current state by accomplishing the current task. This enabled us to use domain-independent heuristics during task decomposition.

The planner takes as input a domain-independent heuristic function and uses it to compute how close the goal of the method is to the state in which the decomposition is performed. Given the current state and the goal expression of a method, the heuristic function computes a score value of using that method for the current task in the current state. The planner can compute a score using any heuristic function originally developed for existing state-space search planners, such as *Manhattan Distance* heuristic and the distance-based heuristics as in FASTFORWARD [6].

This approach is also useful in planning with learned HTNs, i.e., HTN knowledge produced by a machine learning algorithm as opposed to an expert. We have used this approach successfully in the Transfer Learning Go/Nogo tests in order to plan with HTNs learned by the learning algorithms such as Icarus and LIGHT in the program.

Duet = SHOP2 x LPG. In this work, we have developed a formalism that enabled us to combine SHOP2 and LPG, a well-known domain-independent classical planner, in a unified planning architecture called Duet. We extended the SHOP2 and LPG formalisms to allow the planners to communicate in Duet by generating subgoals of a planning problem for each other. Duet organizes the planning process by passing these subgoals to the individual planners until no subgoals are left to achieve.

² The use of goal expressions associated with HTN methods is originated from the previous work on ICARUS, a machine-learning system capable of producing hierarchical knowledge similar to HTNs [10]

In our experiments, we varied the amount of HTN-based domain-specific knowledge available to Duet and compared its performance with LPG's and SHOP2's performance as stand-alone planners. Even with just a small amount of domain-specific knowledge (e.g., ``choose the least-fragile object and move it to the target museum"), Duet usually generated solutions faster than LPG. With more domain-specific problem-solving knowledge (e.g., how to properly stack art objects on top of each other), Duet ran faster and solved more problems than both LPG and SHOP2. Although SHOP2's performance could have been improved, this would have required much more time for hand-crafting its knowledge base.

This work is published in the 2008 proceedings of European Conference on Artificial Intelligence (ECAI-08).

Translating HTNs into PDDL. We have developed a translation methodology that takes an HTN knowledge base and an HTN planning domain description and an HTN planning problem and translates them into a classical planning domain description and classical planning problem. Any classical planner can use the produced domain description in order to solve the translated planning problem -- hence the HTN-based knowledge in its operations.

Our experiments show that by translating partial HTN models into PDDL, we can substantially improve a classical planner's performance. In experiments with the well-known Fast-Forward (FF) planner [6] on more than 3500 planning problems, the translated knowledge improved FF's running time by several orders of magnitude, and enabled it to solve much larger planning problems than it could otherwise solve.

This work is published in the 2009 proceedings of International Joint Conference on Artificial Intelligence (IJCAI-09).

C.3.1 Learning HTN knowledge from Plan Traces

It is crucial to develop learning techniques in order to produce planning knowledge when human contributions are limited or unavailable, either in the form of writing HTNs or devising heuristic functions. The subsequent sections summarize our research:

Learn2SHOP. Our objective was to develop HTN learning formalisms and algorithms that use computer simulations in a planning domain in order to acquire data about the unknown outcomes of the actions. We have developed the Learn2SHOP architecture, which departs significantly from the previous works on AI planning and learning in that its modular architecture integrates HTN planning, concept learning, and computer simulations.

The overall algorithm employed by Learn2SHOP takes as input example solutions to given problems. It acquires data on the performance of these examples through simulation in the game's actual environment. Using simulations during the planning and learning process enables the system to get information about the outcomes of the actions. This data is then used in a concept learning algorithm to determine the applicability of the various HTN methods to the given game.

As learning progresses, Learn2SHOP becomes more and more certain of which HTN methods are best in which situations and performs better at the provided game. If the game is switched, Learn2SHOP will not be confused but will proceed as expected: applying lessons already learned that work, and re-learning those lessons which worked before but do not apply to the new environment.

In Phase 1, the experiments with Learn2SHOP have demonstrated the advantages of integrating planning, learning, and simulation in the benchmark MadRTS real-time strategy game engine. We have tested Learn2SHOP in a transfer-learning task in this game, where the objective was to take knowledge that was acquired under one model and harness it in the learning within another model (e.g., taking lessons that were learned in one game scenario and using them in other game scenarios). The experiments performed by an objective third-party, namely Naval Research Laboratories, demonstrated the effectiveness of our integrated system in a suite of performance measures of knowledge transfer.

HTN-Maker. Furthermore, we have developed a way to formalize semantics of the tasks (i.e., activities) and to use that formalism in order to learn methods for decomposing tasks into smaller tasks. Intuitively, our formalism associates an activity with the conditions that must be held in the world in order to start that activity and the effects that will be realized when the activity ends. Existing HTN formalisms, such as the ones described in [3, 8, 11], that associate semantics to tasks typically does so through the methods that decompose those tasks -- i.e., traditionally, the meaning of a task is given through what subtasks must be accomplished in order to accomplish the task and through the conditions under which such accomplishments must be made.

We describe HTN-Maker (short for *Hierarchical Task Networks with Minimal Additional Knowledge Engineering Required*), an offline and incremental algorithm for learning HTN methods (i.e., both the structural relationships between tasks and their subtasks as well as the conditions under which a method must be applied to a task) using semantic information provided for the tasks in a planning domain. During learning, HTN-Maker uses *hierarchical goal regression*, a form of goal regression in order to generate the preconditions of the learned HTN methods and to identify the subtask relationships in a particular method. Unlike previous work on goal regression [7], HTN-Maker's goal regression propagates goals over HTNs, i.e., both over the actions and over the task hierarchy through previously learned methods.

Our work was published in the 2008 proceedings of AAAI conference (AAAI-08).

C.4. Conclusions

We have described our research on planning and learning with HTNs under DARPA's Transfer Learning contract. We have developed several planning and learning systems. Some of which were integrated and helped the ISLE's transfer learning effort in DARPA's Go/Nogo evaluations, and others produced pure research results that were published in highly prestigious AI conferences.

References

1. B. Bonet and H. Geffner. Planning as heuristic search: New results. In *European Conference on Planning*, pages 360--372, Durham, UK, 1999. Springer-Verlag.
2. Y. Chen, C. Hsu, and B. Wah. Temporal planning using subgoal partitioning and resolution in {SGPlan}. *JAIR*, 26:323--369, 2006.
3. Kutluhan Erol, James Hendler, and Dana S. Nau. Semantics for hierarchical task-network planning. Technical Report CS TR-3239, UMIACS TR-94-31, ISR-TR-95-9, University of Maryland, March 1994.
4. Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. Planning through stochastic local search and temporal action graphs. *JAIR*, 20:239--290, 2003.
5. M. Helmert. The Fast Downward planning system. *JAIR*, 26:191--246, 2006.
6. J. Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *JAIR*, 14:253--302, 2001.
7. T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1, 1986.
8. Dana Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, William Murdock, Dan Wu, and Fusun Yaman. SHOP2: An HTN planning system. *JAIR*, 20:379--404, December 2003.
9. N. Nguyen, Subbarao Kambhampati, and R. Nigenda. Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. *Artificial Intelligence*, 2002.
10. N. Nejati, P. Langley, and T. Könik. Learning hierarchical task networks by observation. In *Proc. of ICML '06*, pages 665--672, New York, NY, USA, 2006. ACM.
11. Austin Tate. Project planning using a hierarchic non-linear planner. Technical Report 25, Department of Artificial Intelligence, University of Edinburgh, 1976.

Appendix D: Advances in Reinforcement Learning for Transfer

Peter Stone

The University of Texas at Austin

D.1. Introduction

The Transfer Learning (TL) program directly enabled and supported a large body of successful and influential research within the UT Austin Learning Agent Research Group (LARG). The program directly led to a completed Ph.D. dissertation which was published as a book (Matthew Taylor), another Ph.D. dissertation is near completion (Gregory Kuhlmann), two journal articles in one of the top machine learning journals (JMLR) [12, 14], and fourteen conference papers in top conferences such as ICML [11], ECML [7, 3], IJCAI [1, 2], AAAI [15, 4, 5, 13], and AAMAS [8, 16, 6, 10], and others [9]. All of these articles and papers are indexed together at http://www.cs.utexas.edu/~pstone/Papers/bib2html/class_rescat.html#Machine%20Learning:%20Transfer%20Learning. In addition, many more articles and papers in areas related to TL, such as abstraction in reinforcement learning and multi-agent learning, were partially supported.

This document briefly summarizes the different threads of research carried out as a part of the TL program within LARG at UT Austin.

D.2. Transfer Learning in Reinforcement Learning

The main line of research supported by TL was Matthew Taylor's Ph.D. research on Transfer Learning in Reinforcement Learning. The thesis was published as a book by Springer Verlag, and was runner up for the IFAAMAS-08 Victor Lesser Distinguished Dissertation Award.

Reinforcement learning (RL) methods have become popular in recent years because of their ability to solve complex tasks with minimal feedback. While these methods have had experimental successes and have been shown to exhibit some desirable properties in theory, the basic learning algorithms have often been found slow in practice. Therefore, much of the current RL research focuses on speeding up learning by taking advantage of domain knowledge, or by better utilizing agents' experience. The ambitious goal of transfer learning, when applied to RL tasks, is to accelerate learning on some target task after training on a different, but related, source task. Matt's dissertation demonstrates that transfer learning methods can successfully improve learning in RL tasks via experience from previously learned tasks. Transfer learning can increase RL's applicability to difficult tasks by allowing agents to generalize their experience across learning problems.

The dissertation presents inter-task mappings, the first transfer mechanism in this area to successfully enable transfer between tasks with different state variables and actions. Inter-task mappings have subsequently been used by a number of transfer researchers. A set of six transfer learning algorithms are then introduced. While these transfer methods differ in terms of what base RL algorithms they are compatible with, what type of knowledge they transfer, and what

their strengths are, all utilize the same inter-task mapping mechanism. These transfer methods can all successfully use mappings constructed by a human from domain knowledge, but there may be situations in which domain knowledge is unavailable, or insufficient, to describe how two given tasks are related. We therefore also study how inter-task mappings can be learned autonomously by leveraging existing machine learning algorithms. Our methods use classification and regression techniques to successfully discover similarities between data gathered in pairs of tasks, culminating in what is currently one of the most robust mapping-learning algorithms for RL transfer.

Combining transfer methods with these similarity-learning algorithms allows us to empirically demonstrate the plausibility of autonomous transfer. We fully implemented these methods in four domains (each with different salient characteristics), showed that transfer can significantly improve an agent's ability to learn in each domain, and explored the limits of transfer's applicability.

The central question examined in the dissertation is the following:

Given a pair of related RL tasks that have different state spaces, different available actions, and/or different representative state variables,

1. how and to what extent can agents transfer knowledge from the source task to learn faster or otherwise better in the target task, and
2. what, if any, domain knowledge must be provided to the agent to enable successful transfer?

The primary contribution of the dissertation is to answer the first part of the above question by demonstrating that TL is feasible. For this purpose, we introduce *inter-task mappings*, a construct that relates pairs of tasks that have different actions and state variables. Inter-task mappings are the field's first construct to enable such transfer techniques.

The first TL method to use inter-task mappings is *Value Function Transfer*, which can transfer between agents³ in tasks with different state variables and actions, assuming that both agents use *temporal difference* (TD) learning algorithms and represent the learned value function in the same manner. Experiments demonstrate that this method can significantly improve learning, but there may be situations where it is inapplicable because an agent does not use TD learning, or because agents use different representations. However, the inter-task mapping construct is robust enough to work in a variety of settings, and this dissertation fully explores the application of inter-task mappings as the core of multiple algorithms. Matt's dissertation introduces the following methods, all of which utilize inter-task mappings:

³ It is reasonable to frame TL as transferring from an agent in a source task to a *different* agent in a target task, or to consider training an agent in a source task and then having it *move* into the target task. The dissertation assumes transfer between different agents, but the two views are equivalent.

- Value Function Transfer is described above.
- Q-Value Reuse on Value Function Transfer by directly reusing a learned source task action-value function, allowing for transfer between TD agents with different function approximators.
- Policy Transfer modifies the structure and weights of neural network action selectors to transfer between direct policy search learning methods.
- TIMBREL directly transfers experience data between tasks in order to improve learning on a model-based learning method in the target task, without placing any requirements on the type of source task learning method.
- Rule Transfer learns production rules that summarize a source task policy learned with any RL method, and provides the rules as advice to a TD learner in a target task.
- Representation Transfer allows experience from an agent trained in a task to be reused in the same task by an agent with a different representation (as defined by the learning algorithm, the function approximator, and the function approximator's parameterization), or in a different task.

As a whole, these methods show that inter-task mappings can be used as a core component in multiple algorithms, allowing for transfer between many different types of learners and learning representations. Additionally, these methods show that different types of knowledge can be successfully transferred, emphasizing that inter-task mappings are a very general construct that allow for significant flexibility in specific transfer algorithms.

A second contribution of Matt's dissertation is to answer the latter part of the above question by demonstrating that inter-task mappings can be learned autonomously. While all of the TL methods in the dissertation function well with mappings provided by a human, a human may sometimes be unable to generate such a mapping, either because she does not have the requisite domain knowledge, or because the agent is fully autonomous. A pair of mapping-learning methods are therefore introduced to address this potential shortcoming. The first uses classification, in conjunction with some limited domain knowledge, to learn a mapping between two tasks. The second gathers data in both tasks, uses regression to learn a simple model, and then selects an inter-task mapping by testing different possible mappings against the model offline. This second method is significantly more robust than existing methods that learn such relationships between tasks, and is capable of enabling autonomous transfer.

1. There are many ways to formulate and address the transfer learning problem. Matt's research differs from prior approaches in three ways: The TL methods enumerated above use inter-task mappings to transfer between tasks with differences in the action space and state variables, which increases their applicability (relative to many existing transfer methods). Our algorithms are also applicable when the transition function, reward function, and/or initial state differ between pairs of tasks⁴.
2. Our methods are competitive with, or are able to outperform, other transfer methods with similar goals.

⁴ The dissertation uses the *Markov Decision Process* (MDP) framework.

3. We introduce two methods that are able to *learn* inter-task mappings in order to define relationships between pairs of tasks without relying on a human to provide them. Such methods are necessary for achieving autonomous transfer but remain a relatively unexplored area in the literature.

D.3. Transfer in General Game Playing

The second Ph.D. dissertation supported and directly inspired by the TL program will be that of Gregory Kuhlmann. Greg is planning on defending his thesis in the Spring of 2010.

Greg's focus is on the problem of General Game Playing. In this paradigm, the challenge is to design an agent that can receive descriptions of previously unseen games and play them without human input. This arrangement precludes us from doing the game analysis ourselves, and instead motivates research on automated techniques. Following the general game playing paradigm, Greg's thesis will present a system for performing automated domain analysis based on the description of the game. The complete agent leverages this knowledge to improve its initial competency as well as its ability to learn and transfer knowledge between similar games.

Much of the intellectually interesting part of game playing is found in game analysis. One approach is to analyze a game's rules and develop heuristics directly from static analysis. But static analysis of a game's formal description can be complemented by experience playing the game. Through internal simulation, the player may refine some of the knowledge generated during domain analysis. Examples include testing state invariants, discovering cooperative roles, and learning the weights of the heuristic evaluation function.

To make the most of this learned experience, the player should be able to reuse its knowledge in the future when faced with a new, but similar game. Such transfer learning can be evaluated in several ways, including its improvement to the player's initial competency in the new task or its increased learning rate in the new task.

The simplest form of transfer is just direct reuse of learned knowledge when faced with a game that was played previously. The problem becomes more difficult as the source task and the target task begin to share less overlap. For any transfer to be possible, the agent must have a means to recognize similarity between games. This thesis will present methods for automatically identifying similar games and deciding what knowledge to transfer.

In addition to using transfer for increased generalization in the agent's lifelong learning, transfer learning can also be used to speed up learning on a completely new task by automatically generating the source task for itself. There are several examples from the literature in which learning has been shown to improve by first learning a simpler source task before transferring that knowledge to the target task. Typically, the simpler task is chosen by a human. However, Greg's thesis will contribute a method for automatically generating source tasks for some games. Greg demonstrates how the agent can speed up learning in this way. This thesis builds directly upon the previous dissertation work of Matthew Taylor (described above), specifically by using reinforcement learning and value function transfer to automate the domain mapping through analysis of the game's formal description.

D.4. Transfer Using Structure Mapping

In a separate line of research led by postdoc Yaxin Liu, we built upon Matt Taylor's Value Function Transfer as well as Ken Forbus' Structure Mapping Engine to demonstrate the ability to build automatic transfer maps.

Feasible transfer often benefits from knowledge about the structures of the tasks. Such knowledge helps identifying similarities among tasks and suggests where to transfer from and what to transfer. In this work, we show how such knowledge helps transfer in reinforcement learning (RL) by using structure mapping to find similarities. Structure mapping is a psychological theory about analogy and similarities and the structure mapping engine (SME) is the algorithmic implementation of the theory. SME takes as input a source and a target represented symbolically and outputs a similarity score and a mapping between source entities and target entities. To apply structure mapping to transfer in RL, we need a symbolic representation of the RL tasks, namely, the state space, the action space, and the dynamics (how actions change states). To this end, we adopt a qualitative version of dynamic Bayes networks (DBNs). Dynamic Bayes networks are shown to be an effective representation for MDP-based probabilistic planning and reinforcement learning. Although the probabilities in DBNs are too problem-specific to be relevant for transfer, the dependencies represented as links are useful information. The qualitative DBN (QDBN) representation thus ignores probabilities but uses different types of links for different types of dependencies. In this line of research, we specialized and optimized SME to work with QDBNs efficiently using heuristic search to find the best maximal mapping, since QDBNs typically involve at least an order of magnitude more entities than previous SME applications.

As an application of the optimized SME for QDBNs, we generate the mapping of states and actions and thus the transfer functionality automatically, using domain knowledge represented as QDBNs. The main contribution of this research is to use structure mapping to find similarities between the source and target tasks based on domain knowledge about these tasks, in the form of QDBNs in particular, and to automatically construct mappings of state variables and actions for transfer.

D.5. Other Research

In addition to the above lines of research directly related to the transfer learning program, the project supported and inspired several other lines of related lines of research. Most notably, Nicholas Jong's upcoming dissertation on "Automatic Induction of Generalization Hierarchies for Reinforcement Learning" and David Pardoe's thesis on "Adaptive Trading Agent Strategies Using Market Experience" will both include some elements of transfer learning. Both are expected to defend in 2010.

Meanwhile, I fully expect that the exciting transfer learning accomplishments that were achieved during the program will continue to shape the research of my more junior students for many years to come.

References

1. Bikramjit Banerjee and Peter Stone. General game learning using knowledge transfer. In *The 20th International Joint Conference on Artificial Intelligence*, pages 672--677, January 2007.
2. Nicholas K. Jong and Peter Stone. State abstraction discovery from irrelevant state variables. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 752--757, August 2005.
3. Gregory Kuhlmann and Peter Stone. Graph-based domain mapping for transfer learning in general games. In *Proceedings of The Eighteenth European Conference on Machine Learning*, September 2007.
4. Yaxin Liu and Peter Stone. Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pages 415--20, July 2006.
5. Alexander A. Sherstov and Peter Stone. Improving action selection in MDP's via knowledge transfer. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, July 2005.
6. Matthew E. Taylor, , and Peter Stone. Towards reinforcement learning representation transfer. In *The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, May 2007.
7. Matthew E. Taylor, Nicholas K. Jong, and Peter Stone. Transferring instances for model-based reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases*, volume 5212 of *Lecture Notes in Artificial Intelligence*, pages 488--505, September 2008.
8. Matthew E. Taylor, Gregory Kuhlmann, and Peter Stone. Autonomous transfer for reinforcement learning. In *The Seventh International Joint Conference on Autonomous Agents and Multiagent Systems*, May 2008.
9. Matthew E. Taylor, Gregory Kuhlmann, and Peter Stone. Transfer learning and intelligence: an argument and approach. In *Proceedings of the First Conference on Artificial General Intelligence*, March 2008.
10. Matthew E. Taylor and Peter Stone. Behavior transfer for value-function-based reinforcement learning. In Frank Dignum, Virginia Dignum, Sven Koenig, Sarit Kraus, Munindar P. Singh, and Michael Wooldridge, editors, *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 53--59, New York, NY, July 2005. ACM Press.
11. Matthew E. Taylor and Peter Stone. Cross-domain transfer for reinforcement learning. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, June 2007.
12. Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(1):1633--1685, 2009.
13. Matthew E. Taylor, Peter Stone, and Yaxin Liu. Value functions for RL-based behavior transfer: A comparative study. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, July 2005.

14. Matthew E. Taylor, Peter Stone, and Yaxin Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(1):2125--2167, 2007.
15. Matthew E. Taylor, Shimon Whiteson, and Peter Stone. Temporal difference and policy search methods for reinforcement learning: An empirical comparison. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence*, pages 1675--1678, July 2007. Nectar Track.
16. Matthew E. Taylor, Shimon Whiteson, and Peter Stone. Transfer via inter-task mappings in policy search reinforcement learning. In *The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, May 2007.

Appendix E: Transfer Learning with Markov Logic Networks

Ray Mooney

University of Texas at Austin

E.1. Introduction

Ray Mooney's group at the University of Texas developed transfer learning methods for *Markov logic networks* (MLNs). These methods allow an MLN learned for a source domain to be used to aid the learning of an MLN for a new related target domain. This work was conducted in coordination with the research on MLNs lead by Pedro Domingos at the University of Washington (UW). All of the software developed was integrated with UW's Alchemy system for MLNs. By exploiting MLNs' ability to combine the expressiveness of first-order logic with the robustness of probabilistic graphical models (Richardson and Domingos, 2006), our methods allow transfer learning in rich, structural domains where even the representation of the target and source domains may be different. Almost all other transfer-learning methods assume identical feature-vector representations in the source and target and cannot be applied in such cases.

There are two aspects to learning an MLN: 1) acquiring the structure, or first-order logic clauses, and 2) setting the weights. While weight learning is relatively quick, structure learning is very computationally intensive. Therefore, we focused on MLN structure learning because it could particularly benefit from transfer. We view transferring an MLN to a new domain as consisting of two subtasks: *predicate mapping* and *theory refinement*. In general, the set of predicates used to describe data in the source and target domains may be partially or completely distinct. Therefore, the first transfer task is to establish a mapping from predicates in the source domain to predicates in the target domain. For example, consider transferring an MLN learned to model data about individuals and their relationships in an academic department to modeling data from the International Movie Database (IMDB). A predicate mapping between the two domains might establish that directors are like professors, actors are like students, and movies are like research papers. Once a mapping is established, clauses from the source domain can be translated to the target domain. However, these clauses may not be completely accurate and may need to be revised, augmented, and re-weighted in order to properly model the target data. This step is similar to previous work in theory refinement (Richards and Mooney, 1995; Wrobel, 1996), except the theory to be revised is *learned* in a previous domain rather than manually constructed by a human expert. Below we elaborate on the mapping and revision steps.

E.2. Predicate Mapping

The goal of this step is to find the best way to map a source MLN into a target MLN. The quality of a mapping is measured by the performance of the mapped MLN on the target data, as estimated by the likelihood of the data given the model. To allow each clause to potentially use a different mapping, we map each source clause separately. To find the best predicate mapping for a clause, we search the space of all legal mappings. A mapping is legal if each source predicate in a given clause is mapped either to a *compatible* target predicate or to the "empty" predicate, which erases all literals of that source predicate from the clause. Two predicates are *compatible* if they have the same arity and the types of their arguments are consistent with the current type constraints. For any legal mapping, a type in the source domain is mapped to at most one corresponding type in the target domain, and the type constraints are formed by requiring that these type mappings are consistent across all predicates in the clause.

For example, if the current type constraints are empty, then the source predicate `Publication(title, person)` is considered to be compatible with the target predicate `Gender(person, gend)`, and the type mappings `title → person` and `person → gend` are added to the current type constraints. All subsequent predicate mappings within the current clause must conform to these type constraints. For example, with these constraints, the source predicate `SamePerson(person, person)` is compatible with the target predicate `SameGender(gend, gend)` but not compatible with the target predicate `SamePerson(person, person)`. The best predicate mapping for a clause is the one whose translated target clause gives the highest likelihood to the target training data. This process is repeated to find the best predicate mapping for each source clause. Shown below is the best predicate mapping found by the algorithm for a given source clause and the resulting translated target clause:

Source clause:

`Publication(T, A)   Publication(T, B)   Professor(A)   Student(B)`
`¬SamePerson(A, B)   AdvisedBy(B, A)`

Best predicate mapping:

`Publication(title, person) → MovieCredit(movie, person)`
`Professor(person) → Director(person)`
`Student(person) → Actor(person)`
`SamePerson(person, person) → SamePerson(person, person)`
`AdvisedBy(person, person) → WorkedFor(person, person)`

Mapped target clause:

`MovieCredit(T, A)   MovieCredit(T, B)   Director(A)   Actor(B)`
`¬SamePerson(A, B)   WorkedFor(B, A)`

E.3. Revising the Mapped Structure

Next we describe how the mapped structure is revised to improve its fit to the target data. The revision algorithm has three steps and is similar to that of FORTE (Richards and Mooney, 1995), which revises theories in first-order logic.

1. Self-Diagnosis:

The purpose of this step is to focus the search for revisions only on the inaccurate parts of the transferred MLN. The algorithm inspects the MLN and determines for each clause whether it should be shortened, lengthened, or left as is. For each clause, C , this is done by considering every possible way of viewing C as an implication in which one of the literals is placed on the right-hand side of the implication and is treated as the conclusion and the remaining literals serve as the antecedents. The conclusion of a clause is drawn only if the antecedents are satisfied and the clause “fires.” Thus, if a clause makes a wrong conclusion in the target data, it is considered for lengthening because the addition of more literals, or conditions, to the antecedents will make them harder to satisfy, thus preventing the clause from firing. On the other hand, there may be clauses that fail to draw the correct conclusion in the target data because there are too many conditions in the antecedents that prevent them from firing. In this case, we consider shortening the clause.

2. Structure Update:

Clause updates are performed using beam search similar to that used by Kok and Domingos (2005); however, we do not consider all possible additions and deletions of a literal to each clause. Rather, we only try removing literals from the clauses marked for shortening and adding literals to clauses marked for lengthening. Thus, the search space is constrained first by limiting the number of clauses considered for updates, and second, by restricting the kind of update performed on each clause.

3. New Clause Discovery:

To learn additional new clauses for the target domain that do not have analogs in the source domain, we use *relational pathfinding* (RPF) (Richards and Mooney, 1992). RPF is a data-driven approach to clause discovery designed to overcome plateaus and local maxima in the search space. RPF views the relational domain as a graph in which the constants are the vertices, and two constants are connected by an edge if they are related by some predicate. Clauses are formed based on the path of relations that connect the arguments of a known predicate instance. For example, a clause defining the Cousin relationship can be learned from the Parent-Parent-Child-Child path connecting two known cousins.

E.4. Experimental Evaluation of MLN Transfer Learning

We compared the performance of several MLN structure-learning methods. We refer to the structure-learning method developed by Kok and Domingos (2005) as *Alchemy*. We ran *Alchemy* “from scratch” without transfer learning (*Alchemy Scratch*); as well as using it to revise the MLN learned in the source domain and then automatically mapped to the target domain using our predicate mapping procedure (*Alchemy Transfer*). We compared these methods to our complete transfer system (*Revision + Pathfinding*), using both our automatic predicate mapping and theory revision procedures.

We used three real-world relational domains: IMDB, UW-CSE, and WebKB. Each dataset is broken down into *mega-examples*, where each mega-example contains a connected group of facts. Individual mega-examples are disconnected and independent of each other. The IMDB database is organized as five mega-examples, each containing information about four movies, their directors, and the first-billed actors. The UW-CSE database was first used by Richardson and Domingos (2006). The database is divided into mega-examples based on five areas of computer science. It lists facts about people in an academic department (i.e. Student, Professor) and their relationships (i.e. AdvisedBy). The WebKB database contains information about entities from the “University Computer Science Department” data set, compiled by Craven *et al.* (1998).

We used the two metrics employed by Kok and Domingos (2005), the area under the precision-recall curve (AUC) and the conditional log-likelihood (CLL). The AUC is useful because it demonstrates how well the algorithm predicts the few positives instances in the data. The CLL, on the other hand, determines the quality of the probability predictions output by the algorithm. To calculate the AUC and CLL of a given MLN, one uses it to perform inference, providing some of the facts in the test mega-example as evidence and testing the predictions for the remaining ones. Like Kok and Domingos (2005), we tested the instances of each of the predicates of the domain in turn, providing the rest as evidence, and averaging the results across all predicates.

Learning curves for each performance measure were generated using a leave-1-mega-example-out approach, averaging over k different runs, where k is the number of mega-examples in the dataset. In each run, we reserved a different mega-example for testing and trained on the remaining $k-1$ mega-examples, which were provided one by one. All systems observed the same sequence of mega-examples.

A sample learning curve is shown below for transferring from UW-CSE as the source to IMDB as the target. Here we additionally tested the performance of systems that do not use the automatic mapping but are provided with an intuitive hand-constructed global mapping that maps: Student $\rightarrow$ Actor, Professor $\rightarrow$ Director, AdvisedBy/TempAdvisedBy $\rightarrow$ WorkedFor, Publication $\rightarrow$ MovieCredit, Phase $\rightarrow$ Gender, and Position $\rightarrow$ Genre. The last two mappings are motivated by the observation that Phase in UW-CSE applies only to Student and Gender in IMDB applies only to Actor, and similarly Position and Genre apply only to Professor and Director respectively. The systems using the automatic mapping perform much better because it discovers a mapping for each clause that allows the source knowledge to adapt better to the

target domain. In addition, our *revision* algorithm outperforms the Alchemy approach. Comprehensive results from our evaluation are presented by Mihalkova, Huynh, and Mooney (2007).

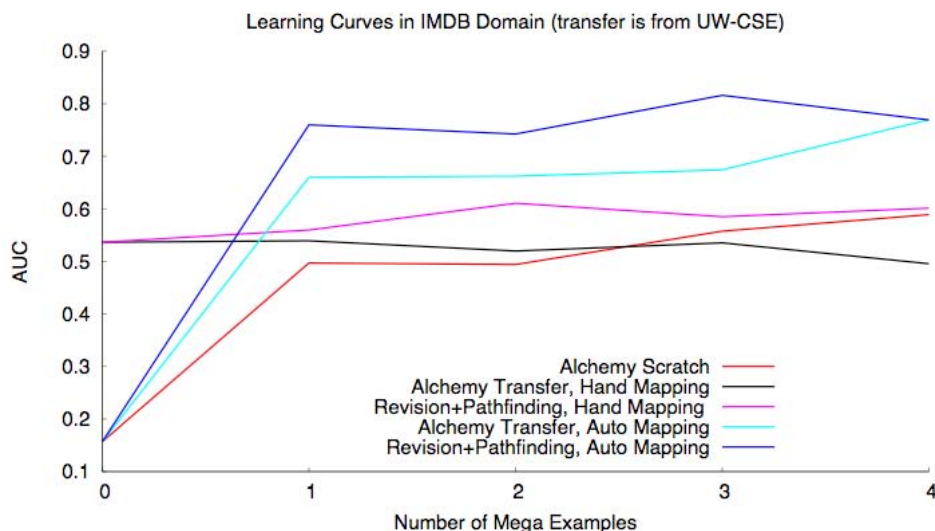


Figure E-1. Learning Curves in the IMDB Domain

E.5. Transfer Learning for MLNs with Minimal Target Data

We also developed an alternative predicate-mapping algorithm that is still effective when there is very minimal training data in the target domain. In such cases, the target data is insufficient to accurately judge the accuracy of the complete translated MLN resulting from a proposed predicate mapping. Therefore, we developed a predicate-mapping algorithm for target data that consists of just a handful of entities. In the extreme case, only a single entity is known. This setting may arise in a variety of real-world situations. For instance, when a new social networking site is launched, data is available on only a few initial registrants. The popularity of the site depends on its ability to make meaningful predictions that would allow it to suggest promising friendships to users. However, the sparsity of available data makes learning an effective model from scratch infeasible.

We developed an alternative predicate-mapping algorithm for this situation, SR2LR (which stands for Short-Range To Long-Range) that is based on the observation that a good model for the source domain contains two types of clauses: short-range ones that concern the properties of a single entity, and long-range ones that relate the properties of several entities. Because possible mappings of the short-range clauses to the target domain can be directly evaluated on the limited available target data, the key is to use the short-range clauses in order to find mappings between the relations in the two domains, which are then used to translate the long-range clauses.

In one experiment evaluating SR2LR on transfer learning from UW-CSE to IMDB, we measured the accuracy of predicting `workedUnder` as representative of an interesting relation that is reasonably predictable. To do this, we considered 5 distinct orderings of the constants of type `person` in each IMDB mega-example, and provided the first n to the systems, with n ranging from 2 to 40, where the smallest mega-example has 44 constants of type `person`. Each point on the learning curves is the average over all training instances with that many known entities. The AUC results are shown below. As can be seen, SR2LR outperforms our basic predicate mapping algorithm (mTAMAR) in this situation and maintains its effectiveness even as more data becomes available. Further evaluation of SR2LR is provided by Mihalkova and Mooney (2009).

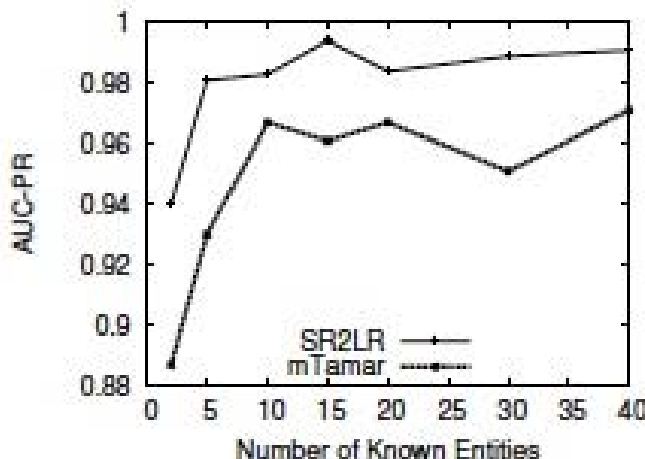


Figure E-2. Comparison of SR2LR and mTamar.

E.6. Non-Transfer Learning for MLNs

In addition to developing transfer learning algorithms for MLNs, we also developed improved algorithms for learning their structure and parameters even when knowledge from a prior domain is not available. Previous structure learning methods for MLNs take a *top-down* approach, heuristically searching the space of increasingly complex models, guiding the search by scoring models using a statistical measure of their fit to the training data. Such top-down approaches follow a “blind” generate-and-test strategy in which many potential changes to an existing model are systematically generated independent of the training data, and then tested for empirical adequacy. For complex models such as MLNs, the space of potential revisions is combinatorially explosive and such a search can become difficult to control, resulting in convergence to suboptimal local maxima. *Bottom-up* learning methods attempt to use the training data to directly construct promising structural changes or additions to the model and thereby avoid such local maxima. Therefore, we developed a bottom-up structure-learning method, BUSL, for MLNs and demonstrated its superior performance on several benchmark problems (Mihalkova and Mooney, 2007).

Most MLN learning algorithms, including BUSL, are non-discriminative and attempt to learn a set of clauses that is equally capable of predicting the truth value of any predicate given an arbitrary set of evidence. However, in many learning problems, there is a specific *target predicate* that must be inferred given evidence data about other *background predicates* used to describe the input data. Most traditional *Inductive Logic Programming* (ILP) methods focus on discriminative relational learning (Dzeroski, 2007); however, they do not address the issue of uncertainty. Discriminative methods have been developed for parameter learning in MLNs (Singla and Domingos, 2005; Lowd and Domingos, 2007); however, they do not address structure learning. We found that existing MLN structure learning methods perform very poorly when tested on several benchmark ILP problems on relating the activity of chemical compounds to their structure. This led us to develop new discriminative structure and parameter learning algorithms for MLNs whose performance on these problems surpasses that of traditional ILP methods. The overall approach uses traditional ILP methods to construct a large number of potentially useful clauses, and then uses a new discriminative MLN parameter learning method to properly weight them, preferring to assign zero weights to clauses that do not contribute significantly to overall predictive accuracy, thereby eliminating them. On several benchmark ILP problems, we demonstrated improved performance of our system over existing MLN and ILP methods (Huynh and Mooney, 2008).

Like other discriminative training algorithms for learning MLN weights, our initial method attempts to maximize the conditional log likelihood (CLL) of the target predicates given the evidence from the background predicates. If the goal is to predict accurate target-predicate probabilities, this approach is well motivated. However, in many applications, the actual goal is to maximize an alternative performance metric such as classification accuracy or F-measure. Max-margin methods are a competing approach to discriminative training that are well-founded in computational learning theory and have demonstrated empirical success in many applications (Cristianini and Shawe-Taylor, 2000). They also have the advantage that they can be adapted to maximize a variety of performance metrics in addition to classification accuracy (Joachims, 2005). Therefore, we developed Max-Margin MLNs (M3LNs) by instantiating an existing general framework for max-margin training of structured models (Joachims, Finley and Yu, 2009). Extensive experiments in two real-world MLN applications demonstrated that M3LNs generally produce improved results when the goal involves maximizing predictive accuracy metrics other than CLL (Huynh and Mooney, 2009).

Bibliography

1. Cristianini, N., Shawe-Taylor, J. (2000). An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press.
2. Dzeroski, S. (2007). Inductive logic programming in a nutshell. In L. Getoor and B. Taskar (Eds.), Introduction to statistical relational learning, 57–92. Cambridge, MA: MIT Press.
3. Huynh, T.N. and Mooney, R.J. (2008). Discriminative Structure and Parameter Learning for Markov Logic Networks. ICML-08, Helsinki, Finland.
4. Huynh, T.N. and Mooney, R.J. (2009). Max-Margin Weight Learning for Markov Logic Networks. ECML/PKDD-09, Bled, Slovenia, Part 1, pp. 564-579.
5. Joachims, T. (2005) A support vector method for multivariate performance measures. ICML-05.
6. Joachims, T., Finley, T., and Yu, C.N. (2009) Cutting-plane training of structural SVMs. Machine Learning 77: 27-59.
7. Kok, S., and Domingos, P. (2005). Learning the structure of Markov logic networks. ICML-2005.
8. Kok, S., Singla, P., Richardson, M., and Domingos, P. (2005). The Alchemy system for statistical relational AI. Technical report, Department of Computer Science and Engineering, University of Washington. <http://www.cs.washington.edu/ai/alchemy>.
9. Lowd, D., and Domingos, P. (2007). Efficient weight learning for Markov logic networks. PKDD-07, pp. 200–211.
10. Mihalkova, L., Huynh, T., and Mooney, R.J. (2007). Mapping and revising Markov logic networks for transfer learning. AAAI-07, Vancouver, BC.
11. Mihalkova, L. and Mooney, R.J. (2007). Bottom-Up Learning of Markov Logic Network Structure. ICML-07, Corvallis, OR.
12. Mihalkova, L. and Mooney, R.J. (2009). Transfer Learning from Minimal Target Data by Mapping across Relational Domains. IJCAI-09, Pasadena, CA, pp. 1163-1168.
13. Richards, B. L., and Mooney, R. J. (1992). Learning relations by pathfinding. AAAI-92.
14. Richards, B. L., and Mooney, R. J. (1995). Automated refinement of first-order Horn-clause domain theories. Machine Learning 19(2):95–131.
15. Richardson, M., and Domingos, P. (2006). Markov logic networks. Machine Learning 62:107-136.
16. Singla, P., and Domingos, P. (2005). Discriminative training of Markov logic networks. AAAI-2005, pp. 868–873.
17. Wrobel, S. (1996). First order theory refinement. In De Raedt, L., ed., Advances in Inductive Logic Programming. Amsterdam: IOS Press.

Appendix F: Deep Transfer through Markov Logic Networks

Pedro Domingos
University of Washington

F.1. Introduction

Inductive learning has traditionally been defined as generalizing from training instances to test instances from the same distribution. Unfortunately, in real applications, training and test data often come from different distributions. Humans are able to cope with this much better than machines. In fact, humans are even able to take knowledge learned in one domain and apply it to an entirely different one. For example, Wall Street firms often hire physicists to solve finance problems. Even though these two domains have superficially nothing in common, training as a physicist provides knowledge and skills that are highly applicable in finance (e.g., solving differential equations and performing Monte Carlo simulations). In contrast, a model learned on physics data would simply not be applicable to finance data, because the variables in the two domains are different. Transfer learning addresses this issue by explicitly assuming that the source and target problems are different. In shallow transfer, test instances are from the same domain, but have a different distribution. In deep transfer, test instances are from a different domain entirely (i.e., described by different predicates). Work to date falls mainly into what may be termed shallow transfer while deep transfer remains largely unaddressed. The inability to discover structural regularities that apply to many different domains, irrespective of their superficial descriptions is arguably the biggest gap between current learning systems and humans.

We have successfully developed an approach to deep transfer as described in a series of papers [1,2,3]. Markov logic forms the basis of our approach [4]. The key innovation behind our approach is a second-order extension of Markov logic that introduces predicate variables into the representation language. Using predicate variables in place of predicate names allow us to represent knowledge in a domain-independent fashion. The central problem is deciding what is the most appropriate knowledge to transfer. Often, the data representation does not explicitly encode the high-level regularities best suited for transfer and it is necessary to discover them. To address this problem, we defined the problem of statistical predicate invention, which is the discovery of new concepts, properties and relations from data, expressed in terms of the observable ones, using statistical techniques to guide the process and explicitly represent the uncertainty in the discovered predicates. We automatically invented predicates by clustering objects, attributes and relations which capture arbitrary regularities over all relations. Each cluster represents a high-level relation or concept. We developed both top-down (i.e., divisive) and bottom-up (agglomerative) clustering approaches. We studied the benefit of constructing multiple different clusterings versus searching for the single best clustering. This was all done in an unsupervised and domain-independent manner. We also developed a system for automatically using the source data to discern high-level similarities between logical formulas by lifting a set of first-order formulas into second-order logic by replacing all predicate names with predicate variables. Those second-order formulas that capture the strongest regularities are transferred to

the target domain where they provide a declarative bias for structure learning. Using this approach, we have successfully transferred learned knowledge among social network, molecular biology and web domains. In addition to improved empirical performance, our approach discovered patterns that include broadly useful properties of predicates, like symmetry and transitivity, and relations among predicates, such as homophily.

F.2. Background

Markov logic networks (MLNs) combine logic and probability by attaching weights to *first-order logic* rules [5], and viewing these as templates for features of *Markov networks* [6].

In first-order logic, formulas are constructed using four types of symbols: constants, variables, functions, and predicates. Constants represent objects in the domain of discourse (e.g., people: *Anna*, *Bob*, etc.). Variables (e.g., x , y) range over the objects in the domain. Predicates represent relations among objects (e.g., *Friends*), or attributes of objects (e.g., *Student*). Variables and constants may be typed. An *atom* is a predicate symbol applied to a list of arguments, which may be variables or constants (e.g., *Friends*(*Anna*, x)). (In this report, we use *predicate* and *relation* interchangeably.) A *ground atom* is an atom all of whose arguments are constants (e.g., *Friends*(*Anna*,*Bob*)). A *world* is an assignment of truth values to all possible ground atoms. A database is a partial specification of a world; each atom in it is true, false or (implicitly) unknown. A clause is a disjunction of non-negated/negated atoms.

A Markov network or Markov random field is a model for the joint distribution of a set of variables $\mathbf{X} = (X_1, \dots, X_n) \in \mathcal{X}$. It is composed of an undirected graph $\mathbf{G}$ and a set of potential functions ϕ_k . The graph has a node for each variable, and the model has a potential function for each clique in the graph. A potential function is a non-negative, real-valued function of the state of the corresponding clique. The joint distribution represented by a Markov network is given by $P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \prod_k \phi_k(\mathbf{x}_{\mathbf{C}_k})$ where $\phi_k(\mathbf{x}_{\mathbf{C}_k})$ is the state of the k th clique (i.e., the state of the variables that appear in that clique). Z , known as the *partition function*, is given by $Z = \sum_{\mathbf{x} \in \mathcal{X}} \prod_k \phi_k(\mathbf{x}_{\mathbf{C}_k})$. Markov networks are often conveniently represented as *log-linear models*, with each clique potential replaced by an exponentiated weighted sum of features of the state, leading to $P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \exp(\sum_j w_j f_j(\mathbf{x}))$. A feature may be any real-valued function of the state. This report will focus on binary features $f_j(\mathbf{x}) \in \{0,1\}$. In the most direct translation from the potential-function form, there is one feature corresponding to each possible state $\mathbf{x}_{\mathbf{C}_k}$ of each clique, with its weight being $\log \phi_k(\mathbf{x}_{\mathbf{C}_k})$. This representation is exponential in the size of the cliques. However, we are free to specify a much smaller number of features (e.g., logical functions of the state of the clique), allowing for a more compact representation than the potential-function form, particularly when large cliques are present. Markov logic takes advantage of this.

A Markov logic network (MLN) is a set of weighted first-order formulas. Together with a set of constants representing objects in the domain, it defines a Markov network with one node per ground atom and one feature per ground formula. The weight of a feature is the weight of the first-order formula that originated it. The probability distribution over possible worlds $\mathbf{x}$ specified by the ground Markov network is given by $P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \exp(\sum_{i \in \mathcal{F}} \sum_{g \in \mathcal{G}_i} w_i g_i(\mathbf{x}))$, where Z is the partition function, $\mathcal{F}$ is the set of all first-order formulas in the MLN, $\mathcal{G}_i$ is the set of groundings of the i th first-order formula, and $g_i(\mathbf{x}) = 1$ if the i th ground formula is true and $g_i(\mathbf{x}) = 0$ otherwise. Markov logic enables us to compactly represent complex models in non-i.i.d. domains.

Algorithms have been proposed for learning the weights for each formula (e.g., [7]), as well as for learning the formulas themselves (e.g., [8]). We will focus on the structure learning algorithm of Kok and Domingos [8], which we will call MSL. Structure learning consists of two components: constructing clauses and evaluating clauses. Clause construction follows an inductive logic programming style search [9]. When learning a network from scratch, MSL starts with an empty clause and specializes it by successively adding literals to the clause. Additionally, the algorithm can refine an existing network to correct errors in the clauses. Here, it considers both removing and adding literals to a clause as well as flipping the sign of a literal in the clause. The algorithm uses a beam search to find the current best clause and add it to the network. The search ends when no clause improves the score of the MLN. To evaluate the merit of each clause, it uses weighted pseudo-log-likelihood (WPLL). To avoid overfitting, each clause receives a penalty term proportional to the number of literals that differ between the current clause and the initial clause. MSL is implemented in the publicly available Alchemy package [10].

F.3. Statistical Predicate Invention

F.3.1. System Description

In statistical predicate invention (SPI), we seek to discover new concepts, properties and relations from data, expressed in terms of the observable ones, using statistical techniques to guide the process and explicitly represent the uncertainty in the discovered predicates. These can in turn be used as a basis for discovering new predicates, which is potentially much more powerful than learning based on a fixed set of simple primitives.

We create the Multiple Relational Clusterings (MRC) system [1] as a first step towards a general framework for SPI. MRC automatically invents predicates by clustering objects, attributes and relations in an unsupervised manner, without requiring the number of clusters to be specified in advance. The invented predicates capture arbitrary regularities over all relations, and are not just used to predict a designated target relation. MRC learns multiple clusterings, rather than just one, to represent the complexities in relational data. MRC is based on the observation that, in relational domains, multiple clusterings are necessary to fully capture the interactions between objects. Consider the following simple example. People have coworkers, friends, technical skills, and hobbies. A person's technical skills are best predicted by her coworkers' skills, and her hobbies by her friends' hobbies. If we form a single clustering of people, coworkers and friends will be mixed, and our ability to predict both skills and hobbies will be hurt. Instead, we should cluster together people who work together, and simultaneously cluster people who are friends with each other. Each person thus belongs to both a “work cluster” and a “friendship cluster.” (See Figure F-1.) Membership in a work cluster is highly predictive of technical skills, and membership in a friendship cluster is highly predictive of hobbies.

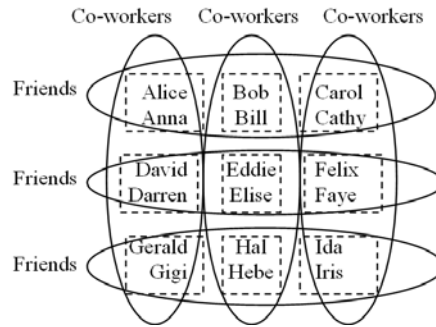


Figure F-1. Example of Multiple Clusterings.

F.3.1.1. Model

We define our model using finite second-order Markov logic, in which variables can range over relations (predicates) as well as objects (constants). We use the variable r to range over predicate symbols, x_i for the i th argument of a predicate, $\mathbf{x}_i$ for a cluster of i th arguments of a predicate (i.e., a set of symbols), and $\mathbf{r}_i$ for a clustering (i.e., a set of clusters or, equivalently, a partitioning of a set of symbols). For simplicity, we present our rules in generic form for predicates of all arities and argument types, with n representing the arity of relation r ; in reality, if a rule involves quantification over predicate variables, a separate version of the rule is required for each arity and argument type.

The first rule in our MLN for SPI states that each symbol belongs to at least one cluster. This rule is hard, i.e., it has infinite weight and cannot be violated.

$$\forall x \exists \gamma \ x \in \gamma$$

The second rule states that a symbol cannot belong to more than one cluster in the same clustering. This rule is also hard.

$$\forall x, \gamma, \gamma', \Gamma \ x \in \gamma \wedge \gamma \in \Gamma \wedge \gamma' \in \Gamma \wedge x \in \gamma' \Rightarrow \gamma = \gamma'$$

If r is in cluster γ_r and x_i is in cluster i , we say that $r(x_1, \dots, x_n)$ is in the combination of clusters

$(\gamma_r, \gamma_1, \dots, \gamma_n)$. The next rule says that each atom appears in exactly one combination of clusters, and is also hard. $\mathcal{C}(\Gamma_r, \Gamma_1, \dots, \Gamma_n)$ is defined as the set of tuples constructible by taking the cross product of clusters in $\Gamma_r, \Gamma_1, \dots, \Gamma_n$.

$$\forall r, x_1, \dots, x_n \exists! c \in \mathcal{C}(\Gamma_r, \Gamma_1, \dots, \Gamma_n) \ r(x_1, \dots, x_n) \in c$$

The next rule is the key rule in the model, and states that the truth value of an atom is determined by the cluster combination it belongs to. This rule is soft. The “+” notation is syntactic sugar that signifies that there is an instance of this rule with a separate weight for each tuple of clusters $(\gamma_r, \gamma_1, \dots, \gamma_n)$.

$$\forall r, x_1, \dots, x_n, +\gamma_r, +\gamma_1, \dots, +\gamma_n \ r \in \gamma_r \wedge x_1 \in \gamma_1 \wedge \dots \wedge x_n \in \gamma_n \Rightarrow r(x_1, \dots, x_n)$$

To combat the proliferation of clusters and consequent overfitting, we impose an exponential prior on the number of clusters, represented by the formula

$$\forall \gamma \exists x \ x \in \gamma$$

with negative weight $-\lambda$. The parameter λ is fixed during learning, and is the penalty in log-posterior incurred by adding a cluster to the model. Thus larger λ 's lead to fewer clusterings being formed.

A cluster assignment $\{\mathbf{F}\}$ is an assignment of truth values to all $r \in \mathcal{R}$ and $x_i \in \mathcal{X}_i$ atoms. Learning consists of finding the cluster assignment that maximizes $P(\{\mathbf{F}\}|R) \propto P(\{\mathbf{F}\}, R) = P(\{\mathbf{F}\})P(R|\{\mathbf{F}\})$, where R is the truth assignments to observable ground atoms. MRC simplifies the learning problem by performing hard assignments of symbols to clusters (i.e., instead of computing probabilities of cluster membership, a symbol is simply assigned to its most likely cluster). This allows the *maximum a posteriori* (MAP) weights of the atom prediction rules, and the MAP log-posterior to be computed in closed form. We use top-down search to greedily find the cluster assignment that maximizes the log-posterior.

The basic idea is the following: when clustering sets of symbols related by atoms, each refinement of one set of symbols potentially forms a basis for the further refinement of the related clusters. MRC is thus composed of two levels of search: the top level finds clusterings, and the bottom level finds clusters. At the top level, MRC is a recursive procedure whose inputs are a cluster of predicates $\mathcal{P}_r$ per arity and argument type, and a cluster of symbols $\mathcal{X}_i$ per type. In the initial call to MRC, each $\mathcal{P}_r$ is the set of all predicate symbols with the same number and type of arguments, and $\mathcal{X}_i$ is the set of all constant symbols of the i th type. At each step, MRC

creates a cluster symbol for each cluster of predicate and constant symbols it receives as input. Next it clusters the predicate and constant symbols, creating and deleting cluster symbols as it creates and destroys clusters. It then calls itself recursively with each possible combination of the clusters it formed. For example, suppose the data consists of binary predicates $r(x_1, x_2)$, where x_1 and x_2 are of different type. If r is clustered into r_1^1 and r_2^1 , x_1 into x_1^1 and x_2^1 , and x_2 into x_2^1 and x_3^1 , MRC calls itself recursively with the cluster combinations (r_1^1, x_1^1, x_2^1) , (r_1^1, x_1^1, x_3^1) , (r_1^1, x_2^1, x_3^1) , (r_2^1, x_1^1, x_2^1) , (r_2^1, x_1^1, x_3^1) , etc.

Within each recursive call, MRC uses greedy search with restarts to find the MAP clustering of the subset of predicate and constant symbols it received. It begins by assigning all constant symbols of the same type to a single cluster, and similarly for predicate symbols of the same arity and argument type. The search operators used are: move a symbol between clusters, merge two clusters, and split a cluster. (If clusters are large, only a random subset of the splits is tried at each step.) A greedy search ends when no operator increases posterior probability. Restarts are performed, and they give different results because of the random split operator used. MRC terminates when no further refinement increases posterior probability, and returns the finest clusterings produced.

F.3.2. Empirical Evaluation

In our experiments, we compare MRC with the Infinite Relational Model (IRM) [11] and MLN structure learning (MSL) [8]. We compared the systems on four datasets: Animals (which relates a small set of animals and their features); Unified Medical Language System (UMLS; a biomedical ontology); Kinship (which describes kinship relationships of an Australian aboriginal tribe); and Nations (which describes features and relationships of nations). For each dataset, we performed ten-fold cross-validation by randomly dividing the atoms into ten folds, training on nine folds at a time, and testing on the remaining one. We measured the average conditional log-likelihood of the test atoms given the observed training ones (CLL), and the area under the precision-recall curve (AUC). The experimental results are shown in Figure F-2. From the figure, we see that MRC does as well as the comparison systems on the smaller two datasets (Animals and Nations) and outperforms them by a large margin on the larger two datasets (UMLS and Kinship). In the figure, Init is the initial clustering formed by MRC.

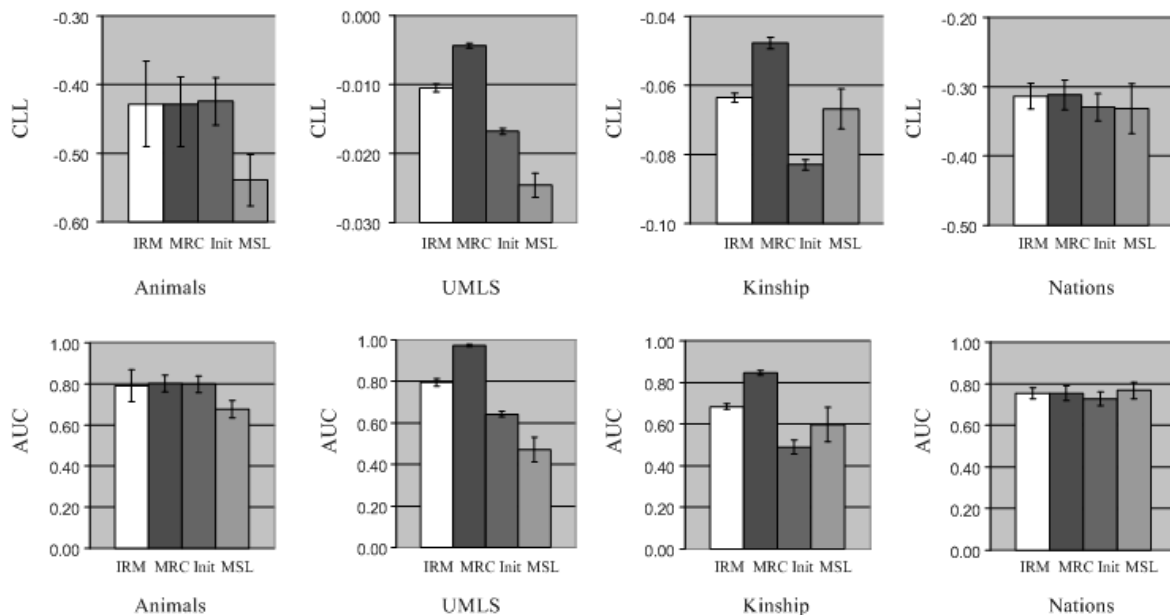


Figure Error! No text of specified style in document.-2. Experimental Results.

F.4. Semantic Network Extractor

We create the Semantic Network Extractor (SNE) system as a first step in addressing the long-standing goal in AI of building an autonomous agent that can read and understand text. SNE uses the TextRunner system [12] to extract $r(x,y)$ triples (e.g., *orbits(space_shuttle,moon)*) from a large corpus of Web pages. Using the triples as input, SNE creates a simple semantic network by jointly clustering objects and relations. In its algorithm, SNE allows information from object clusters it has created at each step to be used in forming relation clusters, and vice versa. Each object cluster corresponds to a concept, and each relation cluster corresponds to a high-level relation.

F.4.1. System Description

To create SNE, we modify the MRC system so that it scales to the Web. SNE uses a bottom-up agglomerative clustering approach rather than MRC's top-down generate-and-test one. While MRC's 'blind' generate-and-test approach may work well for small datasets, it will not be feasible for large Web-scale datasets like the one used in our experiments. For such large datasets, the search space will be so enormous that the top-down algorithm will generate too many candidate moves to be tractable.

SNE's bottom-up agglomerative clustering algorithm begins by assigning each symbol to its own unit cluster. Next it tries to merge pairs of clusters of each type. It creates candidate pairs of clusters, and for each of them, it evaluates the change in posterior probability if the pair is merged. If the candidate pair improves posterior probability, it stores it in a sorted list. It then iterates through the list, performing the best merges first, and ignoring those containing clusters that have already been merged. In this manner, it incrementally merges clusters until no merges

can be performed to improve posterior probability. To avoid creating all possible candidate pairs of clusters of each type (which is quadratic in the number of clusters), it makes use of canopies [13]. A canopy for relation symbols is a set of clusters such that there exist object clusters $\mathcal{C}_x$ and $\mathcal{C}_y$, and for all clusters $\mathcal{C}_r$ in the canopy, the cluster combination $(\mathcal{C}_r, \mathcal{C}_x, \mathcal{C}_y)$ contains at least one true ground atom $r(x, y)$. We say that the clusters in the canopy share the *property* $(\mathcal{C}_x, \mathcal{C}_y)$. Canopies for object symbols x and y are similarly defined. SNE only tries to merge clusters in a canopy that is no larger than a parameter **CanopyMax**. This parameter limits the number of candidate cluster pairs it considers for merges, making its algorithm more tractable. Furthermore, by using canopies, it only tries “good” merges, because symbols in clusters that share a property are more likely to belong to the same cluster than those in clusters with no property in common.

SNE also differs from MRC in having an exponential prior on the number of cluster combinations with true ground atoms rather than on the number of clusters. Unlike MRC which finds multiple clustering, SNE finds a single clustering.

F.4.2. Empirical Evaluation

We conducted experiments to investigate the efficacy of jointly clustering relations and objects vis-à-vis clustering them separately (i.e., clustering relations but not objects, and vice versa). We also investigated the effectiveness of SNE against three other relational clustering systems: Multiple Relational Clustering (MRC) [1], Information-Theoretic Co-clustering (ITC) [14], and Infinite Relational Model (IRM) [11].

All experiments were conducted on a large Web dataset consisting of 2.1 million $r(x, y)$ triples (publicly available at http://knight.cis.temple.edu/~yates/data/resolver_data.tar.gz) extracted in a Web crawl by the information extraction system TextRunner [12]. Each triple takes the form $r(x, y)$ where r is a relation symbol, and x and y are object symbols. Some example triples are: *named_after(Jupiter, Roman_god)* and *upheld(Court, ruling)*. There are 15,872 distinct r symbols, 700,781 distinct x symbols, and 665,378 distinct y symbols. Two characteristics of TextRunner's extractions are that they are sparse and noisy. To reduce the noise in the dataset, we only considered symbols that appeared at least 25 times. This leaves 10,214 r symbols, 8942 x symbols, and 7995 y symbols. There are 2,065,045 triples that contain at least one symbol that appears at least 25 times. In all experiments, we set the **CanopyMax** parameter to 50. We also made the closed-world assumption for all systems (i.e., all triples not in the dataset are assumed false). Because the other relational clustering systems do not scale to the Web dataset, we had to modify them to use SNE's search algorithm. We also limited MRC to find a single clustering (it is able to find multiple) for an apple-to-apple comparison with SNE.

We evaluated the clusterings learned by each model against a gold standard that we manually created. The gold standard assigns 2688 r symbols, 2568 x symbols, and 3058 y symbols to 874, 511, and 700 non-unit clusters respectively. We measured the pairwise precision, recall and F1 of each model against the gold standard. Pairwise precision is the fraction of symbol pairs in learned clusters that appear in the same gold clusters. Pairwise recall is the fraction of symbol pairs in gold clusters that appear in the same learned clusters. F1 is the harmonic mean of precision and recall.

Figure F-3 shows a snippet of the semantic network learned by SNE. Table F-1 shows the performance of SNE when it clusters relations and objects jointly and when it clusters them separately. From that figure, we can see that SNE has the better overall F1 when it clusters relations and objects jointly (SNE) instead of separately (SNE-Sep). We show the best F1s in bold. Table F-2 compares performance of SNE to those of three other relational clustering systems, and shows that SNE has the best overall F1 score. From Table F-3 which shows the runtimes of the various systems, we see that SNE scales well relative to the other systems. We also evaluated the systems in terms of the semantic statements that they learned where a semantic statement is a cluster combination with one true ground atom. We found that SNE outperforms the other systems in terms of the fraction of correct semantic statements discovered (see [2] for details). We also found the clusters discovered by SNE agrees well with those in a publicly available ontology WordNet [15].

Table F-1. Performance when SNE Clusters Relations and Objects Jointly and Separately (SNE-Sep).

Systems	Relation			Object		
	Precision	Recall	F1	Precision	Recall	F1
SNE	0.452	0.187	0.265	0.509	0.062	0.110
SNE-Sep	0.597	0.116	0.194	0.535	0.046	0.085

Table F-2. Performance of SNE and Three Other Relational Clustering Systems.

Systems	Relation			Object		
	Precision	Recall	F1	Precision	Recall	F1
SNE	0.452	0.187	0.265	0.509	0.062	0.110
IRM	0.201	0.089	0.124	0.280	0.042	0.073
ITC	0.773	0.003	0.006	0.617	0.025	0.048
MRC	0.054	0.044	0.049	0.045	0.009	0.015

Table F-3. Runtimes of SNE and Three Other Relational Clustering Systems.

Systems	Runtimes (hrs)
SNE	5.5
IRM	9.5
ITC	72.0
MRC	1.1

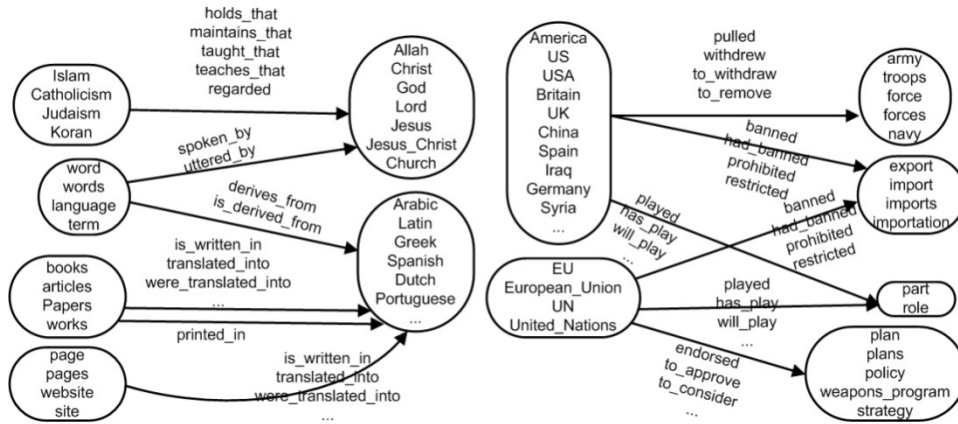


Figure F-3. Snippet of Semantic Network Learned by SNE.

F.5. Deep Transfer Via Second-Order Markov Logic

F.5.1. System Description

The formulas in an MLN capture regularities that hold in the data for a given domain. However, the knowledge that the formulas encode is specific to the types of objects and predicates present in that domain. Our system, called Deep Transfer via Markov Logic (DTM), uses second-order Markov logic, where formulas contain predicate variables [1] to model common structures among first-order formulas. To illustrate the intuition behind DTM, consider the formulas:

$$\text{Complex}(z, y) \quad \text{Interacts}(x, z) \Rightarrow \text{Complex}(x, y)$$

$$\text{Location}(z, y) \quad \text{Interacts}(x, z) \Rightarrow \text{Location}(x, y).$$

Both are instantiations of:

$$r(z, y) \quad s(x, z) \Rightarrow r(x, y),$$

where r and s are predicate variables. Using predicate variables allows DTM to represent high-level structural regularities in a domain-independent fashion. This knowledge can be transferred to another problem, where the formulas are instantiated with the appropriate predicate names.

Given a set of first-order formulas, DTM converts each formula into second-order logic by replacing all predicate names with predicate variables. It then groups the second-order formulas into cliques. Two second-order formulas are assigned to the same clique if and only if they are over the same set of literals. DTM evaluates which second-order cliques represent regularities whose probability deviates significantly from independence among their subcliques. It selects the top k highest-scoring second-order cliques to transfer to the target domain. The transferred knowledge provides a declarative bias for structure learning in the target domain.

The four key elements of DTM, introduced in the next subsections, are: (i) how to define cliques, (ii) how to search for cliques, (iii) how to score the cliques and (iv) how to apply cliques to a new problem.

F.5.1.1. Second-Order Cliques

DTM uses second-order cliques to model structure. This representation is preferable to arbitrary second-order formulas because multiple different formulas over the same predicates can capture the same regularity. A clique groups those formulas with similar effects into one structure. A set of literals with predicate variables, such as $\{r(x, y), r(y, x)\}$, defines each second-order clique and the states, or features, are conjunctions over the literals in the clique. It is more convenient to look at conjunctions than clauses as they do not overlap, and can be evaluated separately. DTM imposes the following restrictions on cliques: the literals in the clique are connected and no cliques are the same modulo variable renaming. The states of a clique are all possible ways of negating the literals in the clique subject to the following constraints: no two features are the same modulo variable renaming and two distinct variables are not allowed to unify.

F.5.1.2. Search

DTM works with any learner than induces formulas in first-order logic. We evaluated two strategies for inducing formulas in the source domain: exhaustive search and beam search.

Exhaustive search. Given a source domain, the learner generates all first-order clauses up to a maximum clause length and a maximum number of object variables. The entire set of clauses is passed to DTM for evaluation.

Beam search. Exhaustive search does not scale well, as the number of clauses it produces is exponential in the clause length and it is computationally infeasible to score long clauses. Beam search, a common strategy for scaling structure learning, is used in MSL (the standard MLN structure learning algorithm). However, transfer learning and structure learning have different objectives. In transfer learning, the goal is to derive a large, diverse set of clauses to evaluate for potential transfer to the target domain. Structure learning simply needs to induce a compact theory that accurately models the predicates in the source domain. The theories induced by MSL tend to contain very few clauses and thus are not ideal for transfer. An alternative approach is to induce a separate theory to predict each predicate in the domain. However, the resulting theory may not be very diverse, since clauses will contain only the target predicate and predicates in its Markov blanket (i.e., its neighbors in the network). A better approach is to construct models that predict sets of predicates. Given the final set of learned models, DTM groups the clauses into second-order cliques and evaluates each clique that appears more than twice.

F.5.1.3. Second-Order Clique Evaluation

Each clique can be decomposed into pairs of subcliques, and it should capture a regularity beyond what its subcliques represent. For example, the second-order clique $\{r(x, y), r(z, y), s(x, z)\}$ can be decomposed into the following three pairs of subcliques: (i) $\{r(x, y), r(z, y)\}$ and $\{s(x, z)\}$, (ii) $\{r(x, y), s(x, z)\}$ and $\{r(z, y)\}$, and (iii) $\{r(z, y), s(x, z)\}$ and $\{r(x, y)\}$. To score a second-order clique, each of its first-order instantiations is evaluated. To score a first-order clique, DTM checks if its probability distribution is significantly different from the product of the probabilities of each possible pair of subcliques that it can be decomposed into.

The natural way to compare these two distributions is to use the K-L divergence: $D(p||q) = \sum p(x) \log (p(x)/q(x))$, where p is the clique's probability distribution, and q is the distribution it would have if the two subcliques were independent. We use Bayesian estimates of the probabilities with Dirichlet priors with all $\alpha_i = 1$. For each first-order instantiation of a second-order clique, DTM computes its K-L divergence versus all its decompositions. Each instantiation receives the minimum K-L score over the set of its decompositions, because any single one could explain the clique's distribution. Each second-order clique receives the average score of its top m first-order instantiations, in order to favor second-order cliques that have multiple useful instantiations.

F.5.1.4. Transfer Mechanism

The next question is how to make use of the transferred knowledge in the target domain. A key component of an inductive logic programming (ILP) [9] system is the declarative bias. Due to the large search space of possible first-order clauses, devising a good declarative bias is crucial for achieving good results with an ILP system. In ILP, two primary methods exist for expressing a declarative bias, and both forms of bias are often used in the same system. The first method

restricts the search space. Common strategies include having type constraints, forcing certain predicate arguments to contain bound variables, and setting a maximum clause length. The second method involves incorporating background knowledge in the form of hand-crafted clauses that define additional predicates which can be added to a clause under construction. Effectively, background knowledge allows the learner to add multiple literals to a clause at once and overcome the myopia of greedy search. It is important to note that these common strategies can easily be expressed in second-order logic. DTM can be viewed as a way to learn the declarative bias in one domain and apply it in another, as opposed to having a user hand-craft the bias for each domain.

To apply a second-order clique in a target domain, DTM decomposes the clique into a set of clauses, and transfers each clause. It uses clauses instead of conjunctions since most structure learning approaches, both in Markov logic and ILP, construct clauses. In Markov logic, a conjunction can be converted into an equivalent clause by negating each literal in it and flipping the sign of its weight. We investigate three different ways to reapply the knowledge in the target domain:

Transfer by Seeding the Beam. In the first approach, the second-order cliques provide a declarative bias for the standard MLN structure search. DTM selects the top k cliques that have at least one true grounding in the target domain. At the beginning of each iteration of the beam search, DTM seeds the beam with the clauses obtained from each legal instantiation of a clique in the target domain. This strategy forces certain clauses to be evaluated in the search process that would not be scored otherwise and helps overcome some of the limitations of greedy search.

Greedy Transfer without Refinement. The second approach again picks the top k second-order cliques that have at least one true grounding in the target domain and creates all legal instantiations in the target domain. This algorithm imposes a very stringent bias by performing a structure search where it only considers including the transferred clauses. The algorithm evaluates all clauses and greedily picks the one that leads to the biggest improvement in WPLL. The search terminates when no clause addition improves the WPLL.

Greedy Transfer with Refinement. The final approach adds a refinement step to the previous algorithm. The MLN generated by the greedy procedure serves as a seed network during standard structure learning. The MLN search can both refine the clauses picked by the greedy procedure to better match the target domain and induce new clauses to add to the theory.

F.5.2 Empirical Evaluation

We evaluated our approach on three real world data sets. The Yeast Protein data come from the MIPS (Munich Information Center for Protein Sequence) Comprehensive Yeast Genome Database as of February 2005 [16,17]. The data set includes information on protein location, function, phenotype, class, and enzymes as well as protein-protein interaction and protein complex data. For this data set, we attempt to predict the truth value of all groundings of two predicates: Function and Interaction. The WebKB data consists of labeled Web pages from the computer science departments of four universities [18]. The data also contains information about which words appear on each Web page and links between pages. For this data set, we again try two tasks: predicting the class label for each Web page and predicting whether a hyperlink exists between each pair of Web pages. The Social Network data consists of pages collected from the Facebook social networking Web site. The data consist of information about friendships among individuals and properties of individuals, such as hobbies, interests and network memberships.

F.5.2.1. High-Scoring Second-order Cliques

An important evaluation measure for transfer learning is whether it discovers and transfers relevant knowledge. In fact, DTM discovers multiple broadly useful properties. Among cliques of length three with up to three object variables, $\{r(x, y), r(z, y), s(x, z)\}$ is ranked first in all three domains. This represents the concept of homophily, which is present when related objects (x and z) tend to have similar properties (y). The clique $\{r(x, y), r(y, z), r(z, x)\}$ is ranked second in Yeast and fourth in WebKB and Facebook and represents the concept of transitivity. Both homophily and transitivity are important concepts that appear in a variety of domains. Therefore it is quite significant that DTM discovers and assigns a high ranking to these concepts. In all three domains, the top three cliques of length two are: $\{r(x, y), r(z, y)\}$, $\{r(x, y), r(x, z)\}$, and $\{r(x, y), r(y, x)\}$. The first clique captures the fact that particular feature values are more common across objects in a domain than others. The second clique captures the fact that pairs of values for the same feature, such as words in the WebKB domain, commonly co-occur in the same object. The final clique captures symmetry, another important general property of relations.

F.5.2.2. Methodology and Results

We compared the DTM algorithm to the Markov logic structure learning (MSL) [8] and to TAMAR [19], another transfer learning algorithm. We tried four source-target pairs: Facebook to Yeast Protein, WebKB to Yeast Protein, Facebook to WebKB and Yeast Protein to WebKB. Each target domain was divided into four disjoint subsets, which we called mega-examples. We selected a subset of the mega-examples to train the learner on and then tested it on the remaining mega-examples. We repeated this train-test cycle for all possible subsets of the mega-examples. Within each domain, all algorithms had the same parameter settings. Each algorithm was allotted 100 hours per database to run. In each domain, we optimized the WPLL for the two predicates we were interested in predicting. For DTM we tried two settings, $k = 5$ and $k = 10$ for the number k of second-order cliques transferred to the target domain. We tried two settings for exhaustive search. The first evaluated all clauses containing at most three literals and three object variables and the second evaluated all clauses containing up to four literals and four object variables. To evaluate each system, we measured the test set conditional log-likelihood (CLL) and area under the precision-recall curve (AUC) for each predicate.

Figure F-4 shows representative learning curves for predicting protein function when transferring cliques of up to length four from the WebKB domain into the Yeast domain. DTM consistently outperforms MSL. Figure F-5 shows representative learning curves for predicting whether two Web pages are linked when transferring cliques of up to length four from Facebook into WebKB. As in Figure F-4, DTM consistently outperforms MSL. TAMAR does significantly worse than all other approaches. TAMAR was unable to map theories from WebKB into the Yeast domain in the allotted time. Overall, we found that DTM using greedy transfer with refinement works better than the other approaches. Furthermore, exhaustive search works better than beam search for finding second-order cliques.

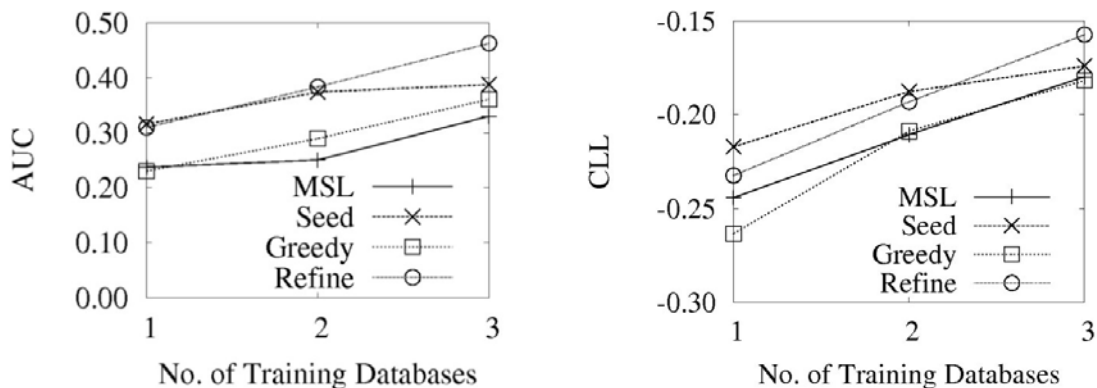


Figure F-4. Learning curves for the Function predicate when transferring second-order cliques of length four from the WebKB domain.

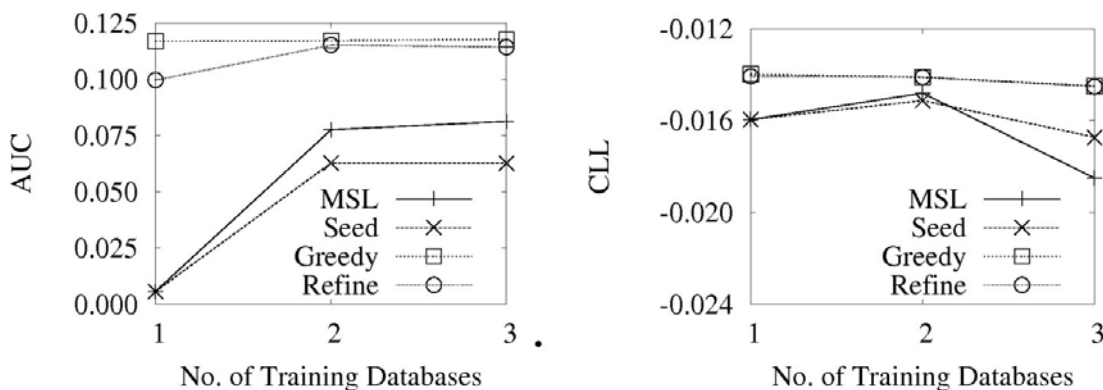


Figure F-5. Learning curves for the Linked predicate when transferring second-order cliques of length four from the Facebook domain.

F.6. Conclusions

We have successfully developed our approach as planned. As a first step towards achieving deep transfer, we explored techniques for statistical predicate invention. Our MRC and SNE systems automatically invent predicates in a domain-independent, unsupervised manner by clustering objects and relations. MRC used a top-down, divisive clustering strategy to construct predicate definitions. SNE extended MRC to web scale by using a bottom-up, agglomerative approach. This is an important first step towards achieving deep transfer because the available data often does not explicitly encode the high-level regularities best suited for transfer and it is necessary to automatically invent them. Our DTM system performed deep transfer by identifying domain-independent regularities in the form of second-order Markov logic clauses, and using them to guide discovery of new structure in the target domain. Initial experiments in bioinformatics, Web and social network domains showed that DTM outperformed standard structure learning, and discovered significant regularities like homophily and transitivity.

References

- [1] Kok, S. & Domingos, P., “Statistical Predicate Invention”, *Proceedings of the Twenty Fourth International Conference on Machine Learning*, Corvallis, Oregon, 2007.
- [2] Kok, S. & Domingos, P., “Extracting Semantic Networks from Text via Relational Clustering”, *Proceedings of the Nineteenth European Conference on Machine Learning*, Antwerp, Belgium, 2008.
- [3] Davis, J. & Domingos, P., “Deep Transfer via Second-Order Markov Logic”, *Proceedings of the Twenty Sixth International Conference on Machine Learning*, Montreal, Canada, 2009.
- [4] Richardson, M. & Domingos, P., “Markov Logic Networks”, *Machine Learning*, **62**, 2006 pp. 107-136.
- [5] Genesereth, M. R. & Nilsson, N. J., **Logical Foundations of Artificial Intelligence**, Morgan Kaufmann, San Mateo, CA, 1987.
- [6] Pearl, J., **Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference**, Morgan Kaufmann, San Francisco, CA, 1988.
- [7] Lowd, D., & Domingos, P., "Efficient Weight Learning for Markov Logic Networks", *Proceedings of the Eleventh Conference on the Practice of Knowledge Discovery in Databases*, Warsaw, Poland, 2007.
- [8] Kok, S. & Domingos, P., “Learning the Structure of Markov Logic Networks”, *Proceedings of the Twenty Second International Conference on Machine Learning*, Bonn, Germany, 2005.
- [9] Dzeroski, S. & Lavrac, N. (Eds.). **Relational Data Mining**. Springer, Berlin, Germany, 2001.
- [10] Kok, S.; Sumner, M.; Richardson, M.; Singla, P.; Poon, H.; Lowd, D.; Wang, J. & Domingos, P., “The Alchemy System for Statistical Relational AI (Technical Report)”, Department of Computer Science and Engineering, University of Washington, 2009.
- [11] Kemp, C.; Tenenbaum, J. B.; Griffiths T. L.; Yamada T. & Ueda, N., “Learning Systems of Concepts with an Infinite Relational Model”, *Proceedings of the Twenty-First Conference on Artificial Intelligence*, Boston, Massachusetts, 2006.
- [12] Banko, M.; Cafarella, M. J.; Soderland, S.; Broadhead, M. & Etzioni, O., “Open Information Extraction from the Web”, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, Hyderabad, India, 2007.
- [13] McCallum, A.; Nigam, K. & Ungar, L., “Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching”, *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.

- [14] Dhillon, I. S.; Mallela, S. & Modha S., "Information-Theoretic Co-Clustering", *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, 2003.
- [15] Gelbaum, C., editor, **WordNet: An Electronic Lexical Database**, MIT Press, Cambridge, MA, 1998.
- [16] Mewes, H. W.; Frishman, D.; Gruber, C.; Geier, B.; Haase, D.; Kaps, A.; Lemcke, K.; Mannhaupt, G.; Pfeiffer, F.; Schuller, C.; Stocker, S. & Weil, B., "MIPS: A Database for Genomes and Protein Sequences", *Nucleic Acids Research*, **28**, 2000, pp. 37–40.
- [17] Davis, J.; Burnside, E.; Dutra, I.; Page, D.; & Costa, V. S., "An Integrated Approach to Learning Bayesian Networks of Rules", *Proceedings of the Sixteenth European Conference on Machine Learning*, Porto, Portugal, 2005.
- [18] Craven, M. & Slattery, S., "Relational Learning with Statistical Predicate Invention: Better Models for Hypertext", *Machine Learning*, **43**, 2001, pp. 97–119.
- [19] Mihalkova, L.; Huynh, T. & Mooney, R. J. "Mapping and Revising Markov Logic Networks for Transfer Learning", *Proceedings of the Twenty Second Conference on Artificial Intelligence*, Vancouver, Canada, 2007.