Preliminary version of article published in AI Magazine 32 (2).

An Application of Transfer to American Football: From Observation of Raw Video to Control in a Simulated Environment

David J. Stracuzzi, Alan Fern, Kamal Ali, Robin Hess,

Jervis Pinto, Nan Li, Tolga Könik, and Dan Shapiro

Abstract

Automatic transfer of learned knowledge from one task or domain to another offers great potential to simplify and expedite the construction and deployment of intelligent systems. In practice however, there are many barriers to achieving this goal. In this article, we present a prototype system for the real-world context of transferring knowledge of American football from video observation to control in a game simulator. We trace an example play from the raw video through execution and adaptation in the simulator, highlighting the system's component algorithms along with issues of complexity, generality, and scale. We then conclude with a discussion of the implications of this work for other applications, along with several possible improvements.

Introduction

Automatic transfer of learned knowledge from one task or domain to another is a relatively new area of research in the artificial intelligence community. Broadly speaking, the goal is to apply knowledge acquired in the context of one task to a second task in the hopes of reducing the overhead associated with training and knowledge engineering in the second task. The underlying idea is that the human and computational costs of revising and adapting the previously learned knowledge should be less than that of building a new knowledge base from scratch.

This article provides an overview of recent work on a system for transferring knowledge between two tasks in American football. In the initial task, known as the *source*, the system must learn to recognize plays, including the patterns run by individual players, from videos of college football games. The second task, known as the *target*, requires the system to execute and improve upon the plays observed in the source in a simulated football environment, called Rush 2008.[1]

This case study into transfer from recognition into control knowledge illustrates an ability to load performance knowledge into an agent by demonstration, as opposed to manual encoding or search and exploration. Our work uniquely spans the full range of problems associated with this knowledge conversion process includes learning to process the raw video into a symbolic format, learning procedures that capture the play observed in video, and adapting those procedures to the simulated environment. Central to this process is the problem of how to translate from the declarative knowledge used in parsing the videos into the procedural knowledge used to control the simulator. The football domain provides an ideal environment to pursue this work for two reasons. First, the domain exhibits substantial structure, which is an important aspect of transfer. Second, football is a complex and knowledge intense domain, so that both interpretation and physical participation in the game are challenging for inexperienced humans, much less computer agents. Ideally, the system should apply the knowledge structures acquired in the source to support high levels of performance in the target, similar to those observed in the original video.

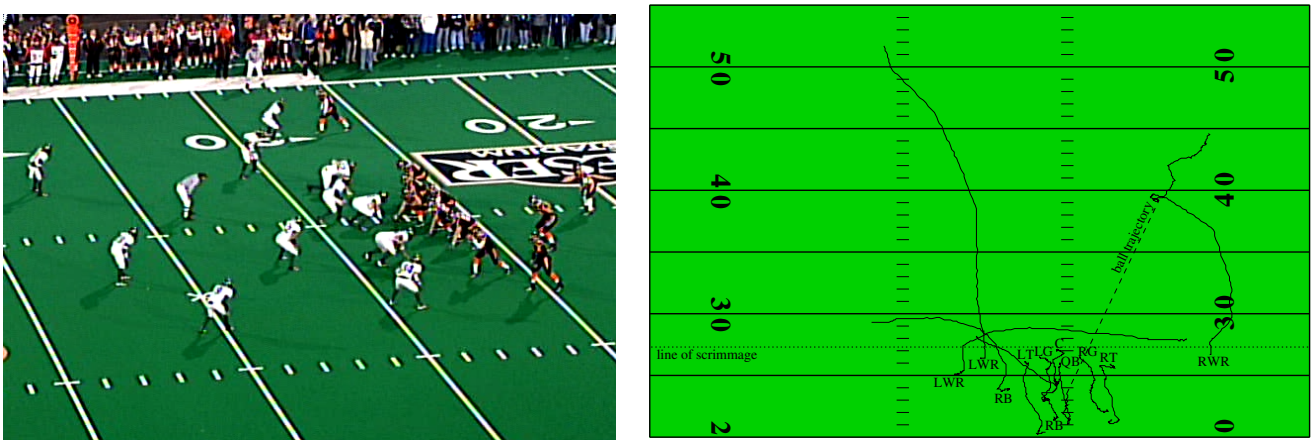
The goal of this presentation is to describe an end-to-end transfer system that connects to real-world data, including the technology that we developed to address the associated issues of compexity, generality, and scale. We begin with a brief review of American football, including a summary of the video footage and simulated environment. Next, we visit each learning and performance task in order, tracing a single play from raw video through execution and adaptation. After this, we discuss some of the implications of this work, including strengths and weaknesses of the approach along with possible next steps and other potential applications. Finally, we conclude with several remarks on the future of transfer.

Transfer in American Football

American football is a competitive team sport that combines strategy with physical play. Two eleven-player teams play the game on a 100 by $53\frac{1}{3}$ yard field (slightly smaller than a typical soccer pitch). The objective is for the offense (team with possession of the ball) to score points by advancing the ball down the field until reaching the opponent's end, while the defense works to stop forward progress. The rules governing ball control and motion are complex, but for this article, only the notion of a single play is important.

A play begins with the ball placed at the location of the end of the previous play, known as the line of scrimmage. The teams line up horizontally on opposite sides of the ball. Play begins when the center (C) snaps, or hands-off, the

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



(a) (b)

Figure 1: A sample video frame (a) showing initial player positions, along with a trace of offensive player movement throughout the play (b).

ball to the quarterback (QB), who then attempts to advance the ball using a variety of techniques. This article presents a passing play, in which the quarterback attempts to pass (throw) the ball to a receiver who has moved downfield. Ideally, the receiver catches the ball and continues to run downfield until tackled (stopped) by the opposing team. Again, the rules are complex, but a typical passing play includes five players that can legally receive the ball.

Transfer Task

Broadly speaking, the objective of our transfer system is to gain sufficient knowledge by observation of football videos to support control and adaptation in a simulated environment. More specifically, the source task requires the system to recognize plays executed by the offense in the video. This includes recognizing the patterns run by the individual players, along with the actions required to achieve those patterns. Toward this end, we provide the system with a set of twenty videos, each of which depicts one successful passing play. The plays were selected to illustrate a variety of different behaviors and patterns.

The videos used in the source task each depict a single passing play as executed by the Oregon State University football team in a real game. The videos are shot from a panning and zooming camera located near the top of the stadium, and correspond to those used by coaches for reviewing and preparing for games. The camera operator attempts to keep as much of the action in view as possible. This differs from typical television footage, which tends to focus primarily on the ball.

Figure 1 (a) shows a sample video frame, taken immediately prior to the start of play, from the play that we trace throughout this article. Figure 1 (b) shows a trace of each offensive player's motion throughout the play. Notice how the quarterback (QB) drops backward from the line of scrimmage to buy time and space from oncoming defenders (not shown), while the receivers (LWRs, RWR, and RBs) run a variety of patterns down and across the field in an effort to get clear of defenders and receive a pass from the quarterback. The role of the five players (LT, LG, C, RG, RT) near the center of the field is to block oncoming defenders from reaching the quarterback before he throws the ball to an open receiver. In this case, the quarterback throws the ball to the right wide-receiver, who catches it and runs an additional five yards down field before the defense tackles him.

In the target task, the system must first apply the knowledge gained in the source to control the offensive players in the simulator, and then adapt transferred play to improve performance in the target domain. The transfer system receives low-level information, such as player location, direction, and current action, from the simulator on each clock tic. In response, it must control all eight offensive players on a tic-by-tic basis using low-level commands such as *stride*, *block*, and *throwTo*. For the purposes of this article, the rules and objectives of the simulated game are the same as in standard football.

In spite of these similarities, the transfer system must still contend with a variety of other differences that make play adaptation in the target important. For example, the Rush 2008 football environment used for the target task simulates an eight player variant of standard American football. The simulator also uses a wider field than regulation college football, which produces more open play when combined with the reduced number of players. Importantly, our transfer system controls only the offensive players, while the simulator controls the defense based on one of several strategies. This means that the defensive tactics employed by the simulator may look nothing like the tactics employed in the video. As a result, the transfer system may need to adapt its offensive tactics accordingly.

Overview of Approach

Our transfer system consists of three distinct parts, as shown in Figure 2. The first part, which corresponds to the source learning task, takes the raw video along with labeled examples as input and applies statistical machine learning tech-

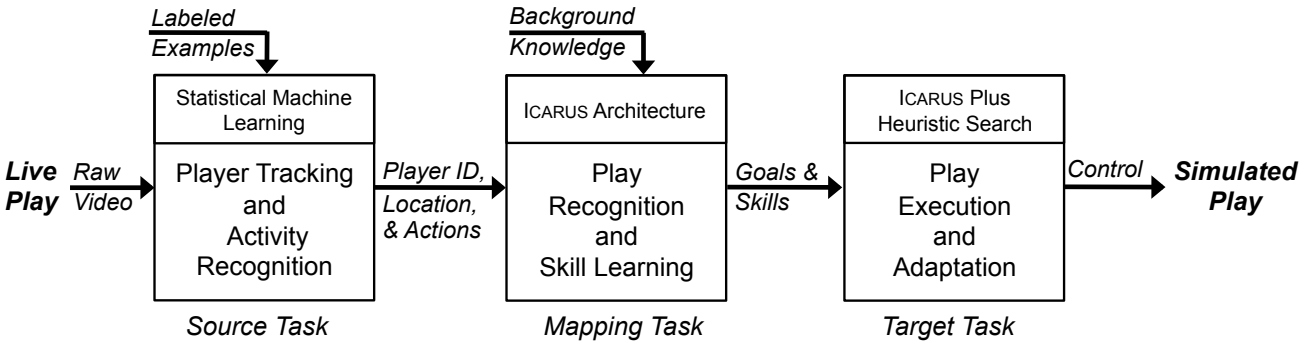


Figure 2: Transfer system overview.

niques to distinguish among the individual players on the field and to recognize the activities undertaken by each player. In the second part, the system must map recognition knowledge acquired in the source into the procedural knowledge required by the target. For this, the system uses the ICARUS cognitive architecture (Langley and Choi 2006), which provides a framework for defining intelligent agents. ICARUS first interprets the play-specific knoweldge acquired in the source in the context of general football background knowledge, and then constructs a set of procedures for executing the observed play. In the third part, the system uses ICARUS to control players during simulation, and adds a heuristic search mechanism to support adaptation of the learned plays to the simulated environment. The following three sections provide a more detailed look at each learning and performance task addressed by the transfer system.

The Source Task: Learning to Annotate Raw Video with Actors and Actions

The performance goals in the source domain are to recognize the activities occuring on the field, and to annotate the video appropriately. In particular, the result of learning should be an annotation system that receives raw video of a football play and outputs a low-level symbolic description of the actors and actions in the play. These low-level descriptions will then serve as the primitive vocabulary that ICARUS uses for extracting higher-level knowledge about football plays.

The problem of computing symbolic annotations from raw video of American football is complicated by several factors. These include the large number of players on the field (22 offensive and defensive players), the large degree of occlusion produced when many players converge on one location, erratic player movement, and the uniformity of the players' appearances. Prior work has attempted to compute symbolic descriptions of football plays (Intille and Bobick 1999), but has relied heavily on human assistance for certain parts of the vision pipeline, such as player tracking and video registration. We are not aware of any prior work that allows for full automation of the annotation process. Here we describe our recent efforts in developing such a system by leveraging modern ideas from machine learning, probabilistic inference, and computer vision.

Our football play annotation system consists of four main components as diagrammed in Figure 3. First, we map the input video to a model of the field in a process known as registration. Second, we identify the initial locations and roles of each player on the field, here called formation recognition. Next, we track each player's location throughout the play. Finally, we segment and label each player track with durative actions. All of these tasks fall within the leftmost box of Figure 2. In the remainder of this section, we give a brief overview of each of these components.

Video Registration

As noted earlier, the video used in this work comes from a camera located in a fixed position at the top of the stadium. During each play, the operator pans and zooms significantly to capture all of the action on the field. This causes the precise video coordinates to become meaningless, since there is not a fixed relationship between video coordinates and locations on the football field. The purpose of our video registration subsystem is therefore to compute a mapping between the raw input video and an overhead scale model of the football field.

The second box of Figure 3 shows the result of projecting a video frame onto the football field according to the computed mapping. This is an important first step in the annotation process because it allows us to model the action of the football game in terms of the static coordinate frame defined by the football field (i.e. in terms of yard and hash lines) rather than in terms of video coordinates. This component of our system does not involve learning but is rather a fixed, stand-alone pre-processor of the video.

Standard registration approaches use image features, such as those computed using SIFT (Lowe 2004), to find point correspondences between the images or video and the model with which they are being registered. These correspondences then get used to analytically compute registration transforms. Unfortunately the image feature matching techniques used by these approaches are typically able to match only those image features which are *distinctive*, that is, image features whose visual appearance is unique from the others. In football video, however, many of the image features that are most useful for registration, such as hash marks and numbers, have similar or identical appearances. For this reason, standard registration approaches generally perform very

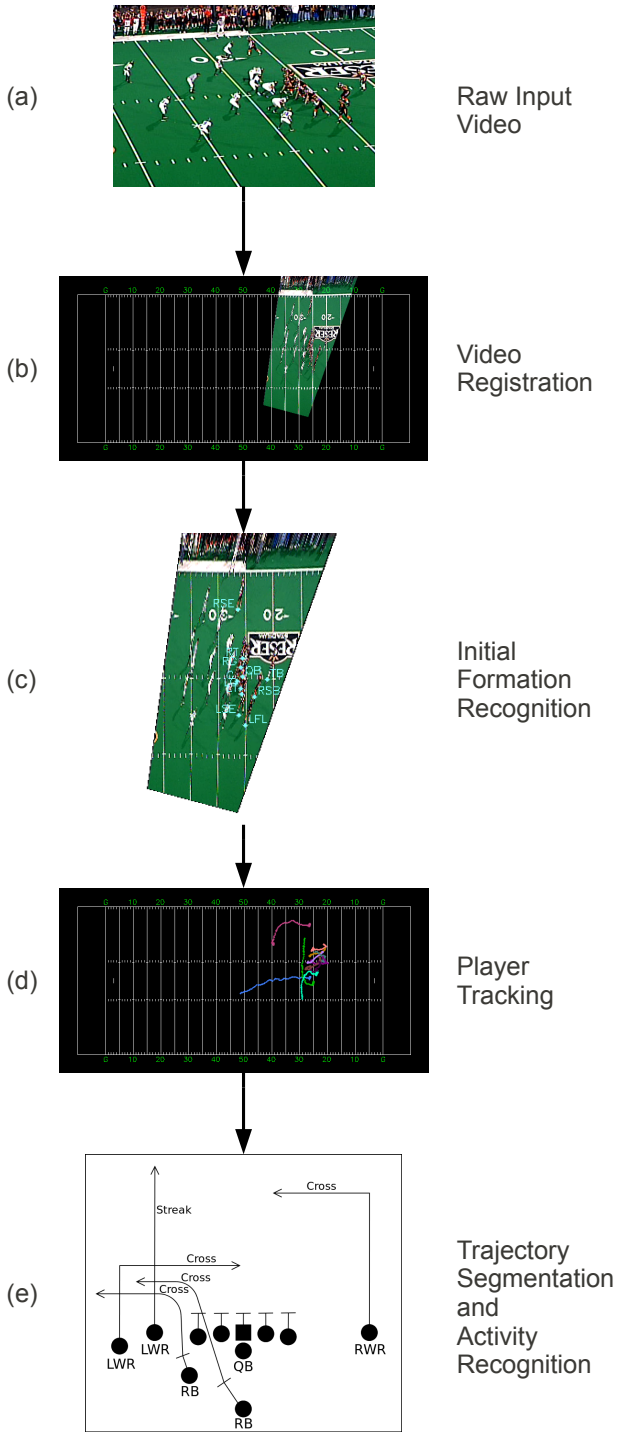


Figure 3: An outline of the main components of the source task system, which annotates the raw input video of a football play with the players' roles and actions.

poorly on football video.

To overcome this difficulty, we developed a registration algorithm that uses a measure of *local* distinctiveness, as opposed to the *global* distinctiveness measures used by standard approaches, to find video-to-model correspondences even for image features with near-identical appearances (Hess and Fern 2007). For football video, this algorithm yields results that are typically accurate to within half a yard on the football field. We use the transformations computed using this algorithm in all later stages of video annotation.

Initial Formation Recognition

Recognizing initial player formations entails computing the locations and determining the roles (e.g. quarterback, wide receiver, etc.) of all the players on the field for a particular play. ICARUS makes extensive use of this information when constructing and executing play procedures, so it represents a key output of the source learning task. We also use the location information to initialize the player trackers we describe below. An example output of our formation recognizer is shown in the third box of Figure 3, where each dot indicates the position of a player and the label next to each dot (e.g. QB for quarterback) assigns a role to the player at that location.

In practice, each player's role and his location relative to other players on the field are closely related. This is useful piece of information since players in our input video appear nearly identical, making it impossible to determine a player's identity based on his appearance alone. We therefore rely on the relative spatial configuration of the players to determine their roles.

Our approach to formation recognition treats the entire formation as a single object composed of parts (the players). We model each player based on his appearance and his ideal location relative to other players, as determined by his role. The key step of this component is to learn appearance models from the videos using standard maximum-likelihood techniques. Given the model we determine the optimal set of players and their locations by minimizing an energy function that takes players' appearances into account along with deformations from their ideal spatial configuration. We perform this minimization using a combination of branch-and-bound search and an efficient form of belief propagation. This model, called the mixture-of-parts pictorial structure (MoPPS) (Hess, Fern, and Mortenson 2007), achieves over 98% accuracy in classifying players' roles and is accurate to within just over half a yard in determining players' locations.

Player Tracking

Every football play consists of a set of carefully planned motion trajectories followed by the players, and reasoning about the action of a football play requires knowing these trajectories. However, tracking the motion of football players is one of the most difficult instances of the object tracking problem. It exhibits nearly all of the most challenging aspects of multi-object tracking, including a significant number of players to be tracked, players with near-identical

appearances, occlusion between players, complex interactions sometimes involving upwards of five or ten players, erratic motion of the players and camera, and changing motion characteristics that depend on both a player's role and the time stage of the football play.

To solve this tracking problem requires a system with a great deal of flexibility. To this end, we designed a particle-filter based tracking framework that employs pseudo-independent trackers and a particle weighting function. Here, pseudo-independent means that the individual trackers perform inference independently, but have the previous states of other trackers available as observations. The particle weighting functiom takes the form of a log-linear combination of weighted arbitrary feature functions (similar to the model used in conditional random fields). Importantly, this model allows us to use different parameters/weights for each player role, which is beneficial since players of different roles exhibit very different typical behaviors. The information about which role each player in the initial video frame has is obtained by the MoPPS model described above and is treated as part of the input to our tracker.

Tuning the parameters by hand is far too difficult for this task. Thus, we designed an efficient, discriminative, supervised training procedure for this tracking framework that learns to improve performance based on errors observed during tracking. This approach is described in detail by Hess and Fern (2009), where it is shown to significantly outperform other state-of-the-art tracking systems on the football tracking problem. An example output of our tracker is shown in the fourth box of Figure 3, where the trajectory of each of the eleven offensive players is projected onto the football field model.

Labeling Player Tracks with Durative Actions

Given the availability of player tracks, we now must annotate those tracks with the durative (non-instantaneous) actions that occur along those tracks. For this, we first defined a set of role-dependent, parameterized actions that correspond to the basic player activities used by coaches to describe football plays. For example, wide receivers will typically get associated with actions that describe passing patterns such as the *cross pattern* shown in Figure 3, in which the player runs forward and then turns across the field. While not shown in the figure, each such action is parameterized by distance and directional information. For example, the cross pattern is parameterized by the number of yards that the player runs forward before turning, along with the direction of the cross (in or out).

Given this action vocabulary, the system must segment the tracks of each player according to distinct actions and label each segment by the action. Most players typically perform a single action per play, but it is not unusual for certain players (e.g. quarterbacks and running backs) to perform a sequence of two or three actions. For this, our system uses an instance-based learning algorithm to simultaneously segment and label the player tracks. The algorithm receives a representative training set of labeled trajectories, which serve as prototypes. Given this data set and a new track to label the system performs the following steps: (1) find the "most similar" track in the labeled dataset, (2) find the segmentation that best corresponds to the segmentation of the most similar track, and (3) return the query track with the segments computed in the previous step labeled by the activities in the most similar track.

The key challenge is to define a notion of similarity of player tracks and the related problem of finding the most similar segmentation between two tracks. For this purpose, we note that a track is simply a time-series of two dimensional points, or a vector-valued signal. By treating our tracks as signals we can draw on prior work on measuring the similarity between signals. Specifically, we apply Dynamic Time Warping (DTW) (Sakoe and Chiba 1978), a well-known algorithm for matching temporally-warped signals. Given a distance measure between any two points in a signal, DTW produces a correspondence between points in the two signals that minimizes the cumulative distances between corresponding points. Such a correspondence can also be viewed as giving a segmentation of one signal relative to the other.

It remains to describe our distance measure used between two points on player tracks. We first defined a set of features that describe each two dimensional point in a trajectory in terms of the direction, speed, and distance covered by the player in a small window of time around a point. We then define the distance between two points to be a weighted Euclidean distance between the feature vectors of those points, where the weights were hand-selected parameters.

Given a query track, we can now use DTW to find the track in the dataset that returns the lowest cost. Since this nearest track in the labeled data set has its durative actions labeled, we simply use the DTW alignment to give each point on the query trajectory the same label as the corresponding point on the nearest trajectory. Thus DTW can be used to perform both labeling and segmentation efficiently. Figure 4 shows the output of the DTW algorithm for a query track (in red) and its nearest match (in green). Each track corresponds to a receiver running a cross pass pattern. The grid represents the field and the points shown are the actual field co-ordinates contained in each track. Notice how DTW correctly aligns the trajectories at the point where the player suddenly changes direction. This is important since it allows us to correctly compute the parameters of the query pattern.

The method performs perfectly for linemen due to the uniformity of their activity, but performance varies for receivers and running backs. The segmentation boundaries are generally accurate though they rarely match the training data exactly. However, certain pairs of segment labels are frequently confused due to qualitative similarity. For example, passing patterns such as "slant" and "cross" can often look quite similar and can be difficult even for human labelers to agree upon. 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 resulting system constitutes the first end-to-end system for labeling football plays from video and was sufficiently accurate to serve as good source data for the overall transfer task.

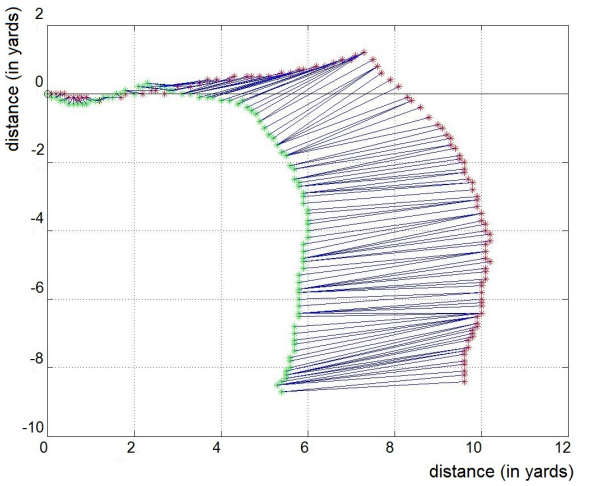


Figure 4: Cost and alignment returned by the DTW algorithm for two "cross" patterns with different parameters.

The Mapping Task: Learning Executable Procedures from the Annotated Video

After completing the source learning task, we now have available a low-level, symbolic description of the videos that includes information such as the identity, location, and action of each player on the field for each video frame, along with some more durative action labels, such as receiver running patterns. The next step is to convert this play recognition knowledge into a form suitable for control. One possible approach is to convert the action labels directly into a script, based on the low-level actions available in the simulator. This would certainly provide the control needed to execute the observed plays in Rush, but in the absense of the recognition information acquired in the source such scripts could not adapt to any changes in the environment.

Toward this end, our approach uses the ICARUS cognitive architecture (Langley and Choi 2006) to represent and apply both forms of knowledge in the simulator. In the following, we review cognitive architectures in general, and ICARUS in particular. The remainder of this section then describes how ICARUS uses and converts the recognition knowledge acquired in the source task, along with other background knowledge, into the procedures or skills needed for play execution in Rush.

Background on the ICARUS Architecture

A cognitive architecture (Newell 1990) serves as a template for intelligent agents by specifying the processes, memories, and knowledge representation languages that remain constant across domains. The specific mechanisms used by a given architecture reflect its objectives. ICARUS aims to capture many facets of human cognitive ability by modeling qualitatively results from psychology and cognitive science. More importantly, ICARUS takes a unified view of cognition (Newell 1990), which means that the architecture focuses on modeling the many important interactions among cognitive processes such as inference, execution, problem solving, and learning. We provide a brief introduction to ICARUS here; see Langley and Choi (2006) for a more detailed overview of the architecture, its assumptions, and its mechanisms.

ICARUS bears a close relationship to other well-known cognitive architectures, such as Soar (Laird, Newell, and Rosenbloom 1987) ACT-R (Anderson 1993). Like these, ICARUS assumes a distinction between long-term and short-term memories, and that both contain modular elements that can be composed dynamically during performance and learning. These architectures also share the notion that cognitive behaviors occur in cycles that retrieve and instantiate long-term structures and then use selected elements to drive action. One key distinction that is relevant to our work on transfer is that ICARUS places conceptual and skill knowledge into separate stores, and index skills by the goals that they achieve. These long-term memories are organized hierarchically, such that complex structures are defined in terms of simpler ones. As we will see, this structure plays a key role in creating skills for the annotated video clips.

Figure 5 shows the basic ICARUS performance system. A cycle begins when perceptual information that describes objects in the environment gets deposited into a short-term *perceptual buffer*. This then triggers the inference process, which matches the contents of the perceptual buffer against long-term conceptual memory to produce beliefs. Each concept describes a class of environmental situations in a relational language similar to PROLOG. Beliefs are therefore specific instances of relations that hold among objects in the environment. The inference process computes the deductive closure of the perceptual buffer with the conceptual memory, and deposits the resulting beliefs into a short-term *belief memory*. Next, the execution process attempts to achieve the first unsatisfied goal stored in *goal memory* by selecting an applicable skill from long-term skill memory based on the contents of belief memory. If a skill that can achieve the current goal applies in the current environment, the architecture selects it and executes the first applicable action. This in turn alters the environment, which produces a new set of percepts, and initiates the next cycle.

In the remainder of this section, we discuss the steps taken to apply ICARUS to the problem of transferring knowledge acquired by observation of football plays into knowledge suitable for executing plays in a simulator. We provide examples and demonstrations of the processes outlined above as needed, and highlight the learning method that underpins that mapping from observed to executable knowledge. We then focus on execution and refinement of the learned skills in the simulator in the following section.

A Symbolic Representation of the Video

The first step in using ICARUS to convert source recognition knowledge to target control knowledge is to define the set of percepts that the system will observe from the preprocessed video and the simulator. Strictly speaking, the percepts from the video stream and the simulator need not be identical. One could reasonably expect the transfer system to map between the source and target representations, as demonstrated by Hinrichs and Forbus (2007) and Könik et

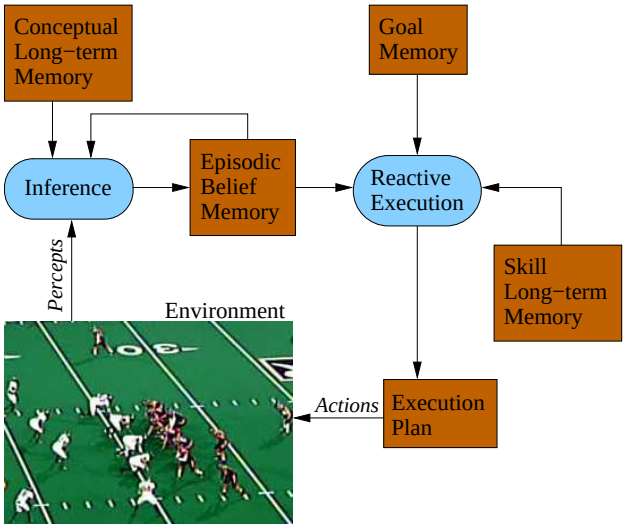


Figure 5: The ICARUS performance system.

(time 143)			
(ball BALL1	carriedby OFF05)		
(agent OFF00	role RWR	x 18.7	y -23.3
	direction N	team OFFENSE	action MOVE)
(agent OFF01	role RB	x 6	y -28.6
	direction N	team OFFENSE	action MOVE)
(agent OFF05	role QB	x 6.6	y -24.1
	direction S	team OFFENSE	action MOVE)
(agent OFF08	role C	x 6.3	y -22.9
	direction S	team OFFENSE	action BLOCK)
(agent DEF11	role DB	x 18.2	y -21.1
	direction SE	team DEFENSE	action DEFEND)
(agent DEF12	role DB	x 13.1	y -11.9
	direction NE	team DEFENSE	action DEFEND)

Table 1: A selection of percepts describing a single frame in the video.

al (2009) for example. In this case, the extra mapping step adds unnecessary complication, as the source output and target input representations align well.

Recall that the source preprocessing step produced four outputs for each of the offensive players on the field: (1) an identifier and role, (2) a location, (3) a direction of motion, and (4) a long-term action, such as *block* or *5-yard cross pattern*. The first three of these correspond exactly to the percepts produced by Rush 2008, and the fourth shows only minor differences in terminology and form. Although a simple, direct mapping would be sufficient to let the system control the simulator based on experience gained by observing the videos, we take a slightly different approach here to provide the system with more control over performance in the target.

The high-level labels used in the source task play an important role in keeping the source learning task tractable and accurately describe the action that takes place on the field. They could also be used to provide high-level control in Rush as done in earlier versions of this work. However, they do not lend themselves to the sort of reactive control model of which ICARUS is capable and on which successful execution in a simulator that follows different defensive strategies depends. To improve this, we mapped the high-level action labels into a sequence of low-level, short-duration, labels. For example, a *5-yard cross pattern* would get mapped into a sequence of directional *move* labels, which when combined with the directional labels, is sufficient to reproduce the original pattern in an incremental manner.

Table 1 shows a sample subset of percepts that ICARUS receives from a single frame in the video. Along with the identity, location, direction and actions of offensive players, ICARUS perceives the identity of the ball carrier and is aware of the passage of time (denoted by the *time* percept. Notice that the system perceives the defenders as well as the offensive players, but that the action associated with the defense is always *defend*. This is because ICARUS controls the offense only, and cannot know the plans or actions taken by the defense except by observation of position and direction. Time steps in the simulator are encoded similarly.

Play Interpretation

The second step in learning executable skills from the video-based perceptual stream is to interpret the observed players and actions into higher-level beliefs about events on the field. The resulting beliefs describe events at various levels of abstraction and over varying periods of time. Reasoning such as this is one of the principle tasks that any intelligent agent must perform as a part of interacting with its environment. ICARUS does this via an inference process that matches perceptual information against concept definitions stored in long-term memory. In the context of learning skills from observed behavior, reasoning serves as an important part of explaining which actions were taken in a given context and why. More generally, reasoning helps to determine which actions to take in order to achieve a given goal.

ICARUS beliefs are instances of generalized concepts, and represent specific relations among entities in the environment. Each belief has two timestamps indicating the first and last times during which the belief held continuously. ICARUS retains all beliefs inferred during the current episode (here, one football play). This allows the agent to reason about events over time. As noted earlier, the inference process in ICARUS simply computes the deductive closure of the perceptual buffer with the concepts stored in long-term memory. Although the use of deductive closure cannot scale to very complex domains, it is sufficient to handle interpretation of the events related to the eleven offensive players in a regulation football play. Stracuzzi et al (2009) provide additional details on temporal inference in ICARUS.

The concepts used in interpreting the plays were manually constructed, as ICARUS does not yet include a mechanism for inducing concept hierarchies from observed sequences. The hierarchy used here included 67 concepts over six layers of abstraction and interprets the complete suite of 20 dif-

```
; Who moved in which direction and when
((moved ?agent ?dir)
  :percepts    ((agent ?agent direction ?dir team OFFENSE action MOVE)))

; Who has possession of the ball and when
((possession ?agent ?ball)
  :percepts    ((ball ?ball carriedby ?agent)))

; Who caught the ball and when
((caught-ball ?agent ?ball)
  :relations   (((possession AIR ?ball) ?air-start ?air-end)
                ((possession ?agent ?ball) ?pos-start ?pos-end))
  :constraints ((= ?air-end (- ?pos-end 1))))

; Indicates that ?agent moved in ?dir consistently until someone caught the ball
((moved-until-ball-caught ?agent ?gen-dir)
  :relations   (((eligible-receiver ?agent ?role) ?elig-start NOW)
                ((moved ?agent ?dir) ?move-start NOW)
                ((caught-ball ?receiver ?ball) ?catch-start NOW))
  :constraints ((< ?move-start ?catch-start)))

; Indicates that ?agent completed a slant pattern: ran downfield ?dist
; yards, then turned toward ?dir and ran until someone caught the ball
((slant-pattern-completed ?agent ?dist ?gen-dir)
  :relations   (((moved-distance-in-direction ?agent ?dist N) ?north-start ?north-end)
                ((moved-until-ball-caught ?agent ?gen-dir) ?dir-start NOW)
                (slant-pattern-direction ?gen-dir))
  :constraints (≤ ?north-end ?dir-start))
```

Table 2: A selection of concepts from the football domain. For the purposes of illustration, these concept definitions have been simplified slightly.

(CAUGHT-BALL RWR BALL1)	259	259
(DROPPED-BACK QB 5)	187	191
...		
(MOVED-DISTANCE QB 5 S)	187	191
(MOVED-DISTANCE RWR 10 N)	208	212
(MOVED-DISTANCE RWR 14 NW)	237	253
...		
(PASS-COMPLETED QB RWR)	259	259
(POSSESSION QB BALL1)	140	236
(POSSESSION AIR BALL1)	237	258
(POSSESSION RWR BALL1)	259	NOW
...		
(SLANT-PATTERN RWR 10 NW)	259	259
(SLANT-RECEPTION RWR 10 NW)	340	NOW
...		
(SNAP QB BALL1)	140	140
(THREW-BALL QB BALL1)	237	237

Table 3: Sample beliefs inferred for the football domain.

ferent passing plays used in training and testing the source learning mechanisms. Importantly, the hierarchy includes a significant amount of shared substructure among the various concepts. As we discuss below, the shared substructure provides generality and scalability to our transfer system.

Table 2 shows sample football concepts. Each concept has a head, which consists of a predicate with arguments, and a body, which defines the situations under which the concept holds true. The relations field specifies the subconcepts upon which the concept depends along with their associated time stamps, which correspond to the time stamps on beliefs. The constraints field then describes the temporal relations among the subconcepts by referring to these time stamps.

The hierarchical relationship among concepts is a key property of the sample concepts shown in Table 2. For example, the first two explicitly test the agent's percepts, checking for ball possession and player movement respectively. `Caught-ball` tests for the event that some player caught a thrown ball by checking whether the ball was in the `AIR` in one time step, and then in possession of a player in the next. `Moved-until-ball-caught` tests for the event that an eligible receiver ran continuously in a specific direction (not necessarily starting from the beginning of the play) until the some player caught the thrown ball. Note the use of `NOW`, which only matches if the belief holds in the current time step. Finally, `slant-pattern-completed` identifies situations in which a receiver runs downfield for a specified number of yards, then turns 45 degrees, and continues running until the ball is caught (possibly by another player). Strictly speaking, the dependence on catching the ball implies that ICARUS cannot recognize receiver patterns if the ball is not caught (a failed pass attempt). While true, this has no impact on performance, since a failed pass typically ends the play and obviates the need for continued recognition and control.

ICARUS updates belief memory after receiving updated percepts at the start of each cycle by matching the concept definitions to percepts and existing beliefs in a bottom-up manner. Table 3 shows some of the beliefs derived from the play shown in Figure 1. Note the cumulative nature of some of the beliefs. For example, the three `possession` beliefs indicate the history of ball possession for the entire play (340 cognitive cycles). Other beliefs, such as `slant-pattern-completed` or `slant-reception-completed` indicate the event that a durative procedure has completed (the latter indicates that the player has run his slant pattern, then caught the ball and ran with it until being tackled by the defense). This temporal understanding of events plays a critical role in letting ICARUS interpret and control complex procedures such as those used in football.

Notice also that the high-level beliefs inferred by ICARUS may not correspond exactly to the high-level labels originally assigned by the source labeling system. For example, a slant pattern (receiver runs straight downfield, then turns and continues downfield at a 45 degree angle) may look similar to a streak (receiver runs straight downfield only) if the slant angle is substantially less than 45, or if the streak angles off to one side. These differences are a matter of interpretation, and have negligible impact on performance in the target. Figure 6 shows the play diagram as recognized by ICARUS for the play shown in Figure 1, including pattern labels for the receivers, the running backs, and the quarterback as recognized by ICARUS. Note the minor differences in pattern labels with respect to the bottom panel of Figure 3.

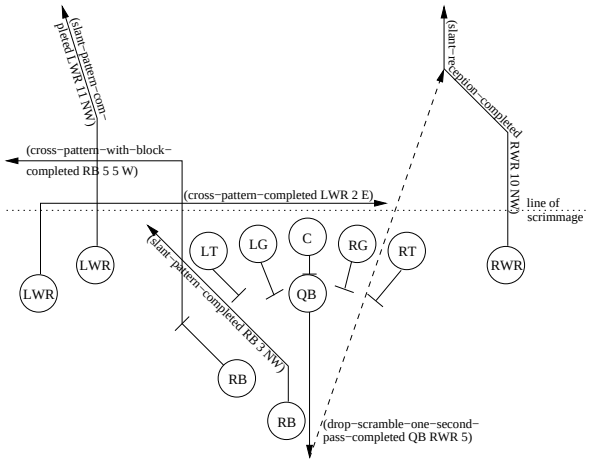


Figure 6: A graphical depiction of ICARUS' symbolic play interpretation.

Explaining What Happened and Learning New Skills

Having interpreted the low-level perceptual stream into higher-level beliefs about events on the field, the next step is now to construct explanations for the goals achieved by each player. These explanations, along with the structure of the concepts, determine the structure of the skills learned by ICARUS. The input to the explanation process consists of a specific goal achieved by a player, the concepts and beliefs computed in the previous step, the observed perceptual sequence, and the low-level actions available to the players. Previously acquired skills, such as those derived by processing other offensive players on the field, may also be available to the system.

The explanation process in ICARUS is based on the architecture's means-ends analysis problem-solver (Li et al. 2009). ICARUS typically problem-solves in the physical world by interleaving problem-solving steps with execution steps. However, in the context of observing video, the agent has no control over the physical world, so the architecture resorts to mental planning. The objective here is to explain, in terms of actions and skills, how various events come to pass. In particular, the architecture attempts to explain the events that have been identified as the top-level goals for each offensive player on the field.

For the work reported here, the system trivially identifies the player goals as the highest level (most abstract) beliefs inferred for each player. In general, this is not a robust approach to the problem, and goal identification is an open area of research that we have not studied extensively with ICARUS. Consider for example the play shown in Figure 1. ICARUS receives inputs such as those shown in Table 1 and applies concepts such as those in Table 2 to produce beliefs like the ones in Table 3. The goals correspond to the beliefs inferred from top-level concepts (those upon which no other concept depends) for each player. Examples include `(DROP-SCRAMBLE-ONE-SECOND-PASS-COMPLETED QB RWR 5)`, which states that the quarterback (QB) should drop back five yards, wait for one second, and then complete a pass to the right wide-receiver (RWR). For the right wide-receiver, the system assigns the goal `(SLANT-RECEPTION-COMPLETED RWR 10 NW)`, which states that he should run a ten yard, northwest slant pattern (run ten yards forward, then run diagonally to the northwest) before catching and running downfield with the ball until tackled.

ICARUS explains these goals using a combination of conceptual and skill knowledge. Ideally, the system would find existing skills that explain how the goals were achieved from the initial state. This would mean that the architecture already had the skills necessary to control the player stored in long-term memory. When this is not the case, ICARUS falls back on using the conceptual hierarchy to explain how the goal belief was derived. This process continues recursively down through the hierarchy, with subconcepts becoming subgoals to explain, until the system reaches a point at which some known skill or action explains the achievement of the current subgoal. The resulting explanation hierarchy then provides the framework for a hierarchy of new skills that achieve the decomposed goals and subgoals.

To create new skills from the explanation, ICARUS performs additional analysis on the explanation to determine important characteristics such as which achieved events should serve as subgoals and start conditions in skills, and in some cases which events can be ignored. The details for the learning and explanation processes appear in Li et al (2009).

Table 4 shows examples of the primitive skills given to ICARUS before skill learning (top) and of the new skills acquired by observation of the play video (bottom). Primitive skills in ICARUS depend exclusively on actions executable in the environment. Once activated, primitive skills continue to execute the listed actions until the goal specified by the head is satisfied or the architecture decides to execute a different skill. In this case, `moved-distance-in-direction` executes until `?agent` has moved `?dist` yards in `?dir`. `moved-until-ball-caught` is similar, except that `?agent` runs until some player catches the ball as indicated by an instance (belief) of the concept in Table 2.

The learned skills on the bottom of Table 4 build upon the primitives in the same way that the concept definitions build upon each other. The main difference is that ICARUS must determine additional control information not present in the concept hierarchy, such as which subconcepts serve as start conditions for the new skills, and which serve as subgoals. For example, when unpacked, the start condition for `slant-pattern-completed` tests whether `?agent` is an eligible receiver and whether `?dir` is a legal direction for a slant pattern. The latter precondition is mentioned by the concept definition, while the former is mentioned in the concept definition of the `moved-until-ball-caught` subgoal.

The Target Task: Executing and Adapting the Learned Procedures in the Simulator

The final task in transferring knowledge gained by observation of football videos into control knowledge is to ap-

```
; Causes ?agent to run in ?dir for a distance of ?dist
((moved-distance-in-direction ?agent ?dist ?dir)
  :start      ((agent-team ?agent OFFENSE))
  :actions    ((*run ?agent ?dir)))

; Causes ?agent to move in direction ?dir until he catches the ball
((moved-until-ball-caught ?agent ?dir)
  :start      ((eligible-receiver ?agent ?role))
  :actions    ((*run ?agent ?dir)))
```

```
((slant-pattern-completed ?agent ?dist ?dir)
  :start      ((SCslant-pattern-completed ?agent ?dir))
  :subgoals ((moved-distance-in-direction ?agent ?dist N)
             (moved-until-ball-caught ?agent ?dir)))

((slant-reception-completed ?agent ?dist ?dir)
  :subgoals ((slant-pattern-completed ?agent ?dist ?dir)
             (ran-with-ball-until-tackled ?agent ?ball)))
```

Table 4: Sample skills for the football domain. The upper panel shows two low-level (primitive) skills provided to the system, while the lower panel shows two high-level skills learned by ICARUS.

ply and adapt the learned skills to the new environment. In the mapping task, the system converted declarative knowledge suitable for state and event recognition into procedural knowledge suitable for controlling players. However, the identified goals and learned skills remain biased by the conditions observed in the source domain. For performance in the target, the system must appropriately adapt the goals and skills into the simulated environment.

Like most research on transfer of learning, this work is concerned not only with applying the transferred knowledge in the target domain, but also with leveraging that knowledge into improved target performance. Ideally we would evaluate this improvement by showing that our system gains more yards per play with the transferred knowledge than without. However, the knowledge acquired in the source substantially constrains the problem of improving performance in the target. With a search space size of approximately 10^{30}, learning to control a simulated football team without any prior knowledge is intractable, making a direct comparison uninformative. In response, we compare the performance of our system to a naive baseline system along with several other points of reference, in an effort to establish a relative understanding of performance. Our results show that ICARUS can use the skills learned from the source videos to bootstrap its way to good target performance on an otherwise intractable task.

Initial Performance of the Transferred Procedures

To execute the observed plays in Rush, we first assign each offensive player a goal from those identified during play interpretation. Although ICARUS can use the individual player goals to control play in the simulator, there are many possible play combinations given that the observed video includes eleven players while the simulator uses only eight. The first step in applying the learned skills in the simulator is therefore to remove three of the player goals observed in the video. In general the system could do this by experimenting with various player configurations to determine which performs best in the simulator. In practice, a simple heuristic works quite well in this situation.

In regulation college football, five players (the offensive linemen) start on the line of scrimmage adjacent to the ball and typically block oncoming defenders from reaching the quarterback. A typical Rush play uses only three linemen, so the system simply drops the two outer players, LT and RT in Figure 1 (b). In selecting the third player to drop, the system first rules out any player that touches the ball during play. This typically includes the quarterback and either a receiver (LWRs and RWR) or a running back (RBs). In plays such as the one illustrated throughout this article, one of the running backs often remains behind the line of scrimmage throughout the entire play to act as an extra blocker for the quarterback. If such a running back is present in a play, the system drops that player, otherwise, the system drops the player that ends up farthest from the ball. Finally, the system extracts the initial positions of the selected players from the source video and scales them to fit the wider Rush field ($53\frac{1}{3}$ versus 60 yards). Again, the system could determine these initial positions automatically, but this simple scaling works well in practice.

To establish an initial performance level for the transferred skills, we used ICARUS to execute the test play five times in Rush with no additional learning. We then compared the resulting performance to that of a naive play, to a manually constructed play, and to the outcome of the original video play. The naive play simply sent all five receivers and running backs straight down the field to receive a pass while the quarterback threw the ball immediately upon receiving the ball. The manually constructed play was created by a Rush expert to recreate the source play as closely as possible using a play design language built into the simulator. The language includes high-level constructs for controlling the offensive players, similar to the instructions that a coach might give, in addition to the low-level commands that ICARUS uses. Our performance metric was simply the mean yardage gained by the play over five attempts (though we have only one instance of the play in the source video for comparison).

Figure 7 shows a typical instance of the test play as executed by ICARUS in Rush. Notice the structural similarity to the player tracks from the video, shown in Figure 1 (b). The simulated play is essentially an idealized version of the video play, as the simulated players run in straight lines and make crisp 45 and 90 degree turns instead of noisy lines and rough arcs. This follows from the simplified physics used by the Rush simulator. The other noteworthy difference between the simulated and real plays relates to performance after the catch. In the video, the receiver gains an additional five yards after catching the ball by avoiding potential tacklers. By contrast, the simulated player runs straight downfield, and is tackled almost immediately. This is partly a limitation of the simulator, but also follows from our use of a simplified player model. An important point of future work in this project is to expand the conceptual hierarchy to support a more detailed interpretation of player action. This

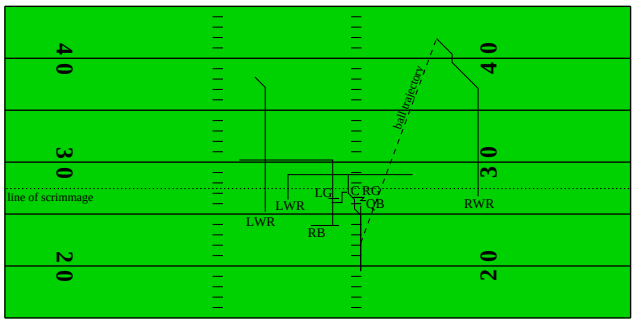


Figure 7: A trace of the test play executed by ICARUS in the simulator prior to adaptation.

in turn would support learning of a more reactive skill set by providing additional recognition and therefore additional response capacity.

Initial results for executing the play without additional adaptation were generally good. ICARUS averaged 14.2 yards per attempt, which was better than both the naive play at 1.3 yards, and the hand-constructed at 10.4 yards per attempt. This demonstrates a strong transfer effect and that the transfer system was clearly able to recognize the expert source play, then transfer, and exploit that knowledge for play improvement. However, the system did not perform as well as the 17.1 yards gained by the OSU football team (a single play). That ICARUS should underperform the video before adaptation is no surprise. The offensive players in Rush do not possess the same attributes, such as speed or agility, as the OSU players. This necessarily causes the timing and synchronization of the play to be off. Likewise, the Rush defense need not make the same choices as those of their video counterparts. Moreover, the extra field width and smaller number of players changes the dynamics of the game substantially, much like the difference between arena and collegiate football. As a result, we expect the adaptation of the learned skills to the new environment to have substantial impact on the outcome of the play.

Adapting Player Goals for Rush

The final step in our transfer system is then to make the goals given to individual players more suitable to the new environment. More specifically, we adapt the goal parameters, such as the distance run by a receiver before turning, or the direction of the turn. We do not adapt the structure of the induced skills in this work, for example by switching the order of the down-field and cross-field components of a receiver's pattern. On the contrary, we assume that the observed structure is correct, and that only the parameters require adaptation to the new environment.

Our approach to goal adaptation operates outside of the ICARUS framework, which does not yet support goal evaluation or revision. The adaptation process operates only over the distance and direction parameters associated with the top-level goals assigned to each offensive player. For example, in `(slant-reception-completed RWR 10 NW)` only the second and third parameters get adjusted. The

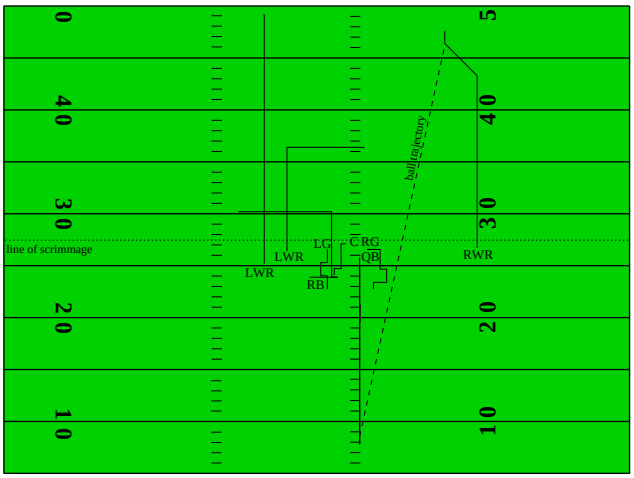


Figure 8: A trace of the test play executed by ICARUS in the simulator after adaptation.

system does not adjust the player named in the first parameter. The current concept hierarchy does not capture the many rules that govern player roles and locations, so the system cannot determine whether a revised formation is legal. Thus, we view the player designations as part of the play structure (who does what) as opposed to part of the parameterization (how far, or in which direction).

The system optimizes the selected parameters using a form of hill-climbing search. Initially, all of the parameters take on their observed values from the video. The system then randomly chooses one parameter to change, and evaluates the performance of the resulting play as above. The best parameter set then gets selected for the final play. Könik et al (2010) provide additional detail on the search algorithm.

Figure 8 shows the play as executed by ICARUS after training. The most salient feature of the adapted play is the substantial increase in distances covered by the quarterback (QB) and the wide receivers (LWRs and RWR). The leftmost wide receiver gets assigned a 34 yard slant pattern (which he does not complete, as evidenced by the lack of a 45 degree turn to the left), while his nearest teammate gets assigned an 11 yard cross pattern. These are much longer than the initial 11 and 2 yard patterns respectively assigned based on the observed source video. The right wide receiver (RWR) similarly gets assigned a 20 yard slant pattern, as opposed to the initial 10 yard pattern. These deep receiver patterns make reasonable sense, given the relatively sparse distribution of defenders in Rush.

Also noteworthy is the 18 yard drop (backward motion) assigned to the quarterback. Though unusual in collegiate or professional play, this makes sense given that ICARUS does not support temporal parameters to skills. The system uses the quarterback's drop distance as a timing mechanism for passing. The behaviors of the running back and three linesmen remain unchanged qualitatively. This particular execution produced a gain of 20.1 yards, though other attempts with the same parameter set produce more or fewer for an average of 18.2.

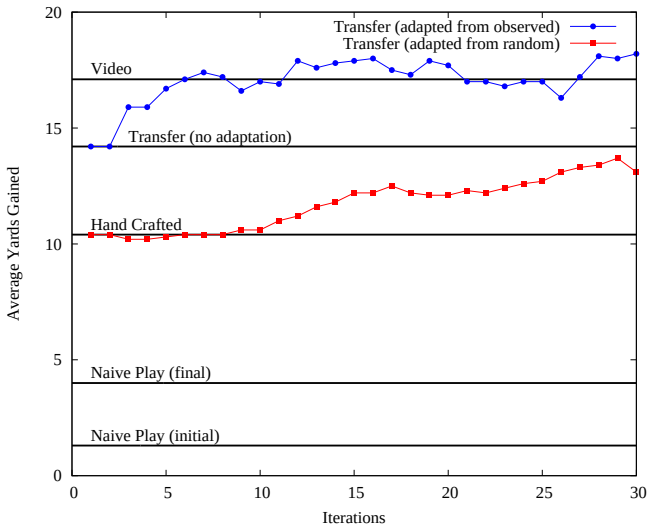


Figure 9: Learning curves for adjusting the parameters associated with player goals.

Figure 9 shows the results of target learning. The horizontal lines show, from top to bottom, the performance of the video play as a point of reference, our system using the initial transferred parameters, the manually encoded version of the play, and the naive play. The performance of the final version of the naive play followed from 1620 iterations of a hillclimbing procedure similar to the one described above and averaging 4.0 yards per attempt. The upper curve on the graph shows the change in performance of the transfer system over thirty rounds of search, while the lower curve shows the same except that the initial parameter values were set randomly instead of based on the observed video values.

These results suggest two main conclusions. First, the limited performance of the naive play in comparison to the transfer system illustrates the impact of the source knowledge on the target task. Not only did the transfer system gain many more yards per attempt than the naive play, but the process of adapting the play to the simulated environment required many fewer search iterations. To a large exent the observed transfer effect was determined by our choice of a naive play. A play composed of random legal motions would have exhibited less potential, while a play composed by a football fan would probably have generated a smaller transfer effect. Nevertheless, the naive play provides a point of comparison that, in conjunction with the other reference points illustrates the power of transfer in this domain.

The results also suggest that both the structure and the observed parameters play an important role in achieving good performance. The transferred structure substantially constrains the space of action combinations that the system must consider by aggregating sequences of low-level actions into a small number of long-term, higher-level patterns. The observed parameters then provide a reasonable starting point for the search. Thus, an otherwise intractable problem of learning to control an entire football team becomes manageable.

Discussion

Like any system aimed at performing a complex task, understanding the capabilities of a transfer system is critical to determining the appropriateness of that system to a given problem. In the artificial intelligence community, the effort to characterize key properties of transfer systems has only recently begun. However, psychological researchers have studied transfer in humans for over a century, producing a variety of dimensions for comparison. Three of these apply particularly well to the problem of characterizing artificially intelligent transfer systems. We do not argue that these dimensions provide a complete characterization of transfer capabilities; they merely serve as a starting point for an important and open problem.

First, Robertson (2001) describes the *reproductive* versus *productive* dimension, in which reproductive refers to the direct application of existing knowledge to a new task while productive refers to knowledge that gets adapted for use in the target. Our system is therefore an instance of a productive transfer system, as it modifies the parameters of the plays to better suit the target environment. The initial performance of the play in Rush, prior to executing the parameter search, provides an example of the reproductive performance of the system. As demonstrated in our example play, adaptation of the source knowledge may have a substantial impact on performance.

A second dimension concerns *knowledge* transfer versus *problem-solving* transfer (Mayer and Wittrock 1996). Here, knowledge transfer refers to the case in which knowledge (declarative, procedural, or otherwise) relevant to one task supports performance of a second task. In contrast, problem solving transfer implies that the experience of solving one problem improves the system's ability to solve a subsequent problem. From an artificial intelligence perspective, the two tasks may share little or no domain-specific knowledge, but the search strategies or biases are similar. Our system currently supports only knowledge transfer. However, one can imagine a variant of our system that also performs problem solving transfer. For example, the system could extract heuristic properties of "good" plays from the source videos, and then used those heiristics to guide the search for play improvements in the target.

A third dimension relates to the amount of processing done to knowledge (or problem-solving biases) in order to convert the information to a suitable representation for the target task (Salomon and Perkins 1989; Mayer and Wittrock 1996). *High-road* transfer requires the system to actively retrieve, map, or otherwise infer relationships and similarities among knowledge components in the two tasks. *Low-road* transfer is automatic in the sense that the system need not apply any computational effort to apply the transferred knowledge in the target. Our system has elements of both. In the context of converting the observed actions into structured skills, our system takes a high-road approach. However, with respect to the assumption that the action and entity names are the same in both the source and target, our system is decidedly low-road.

Future Work

As noted throughout the article, the full set of source domain video covers 20 different passing plays. Although our presentation focuses on the details of single play, we have, to varying degrees, tested the components of transfer system on larger subsets of the videos. For example, we have tested successfully both the source recognition subsystem and the ICARUS play interpretation (recognition) module on the full set of 20 plays. Conversely, we have applied the explanation, learning, and execution components of ICARUS to three plays, while the adaptation algorithm has been tested on only two. The results of these tests were analogous to the results shown above, suggesting that our approach has the potential to generalize and scale across the broader domain of football. The following discussion of future work highlights some of the steps needed to generalize and scale the system to the full football domain and beyond.

One of the major challenges in transfer comes in trying to transfer knowledge among weakly related domains. Consider the extent of mapping and analogy required to transfer knowledge gained by observation and playing of rugby or soccer into football. Though many concepts and skills of teamwork and motion may apply in the abstract, the specific rules of play share almost nothing. This requires a much deeper form of transfer than we have managed here, and of which the community at large has only started to consider. The next big step for the work presented in this article is therefore to incorporate methods for recognizing and applying more abstract relationships among domains. Initial efforts based on analogy (see Hinrichs and Forbus, this issue) and Markov Logic (Kok and Domingos 2007) appear promising and are the most relevant to this work. Nevertheless, research along these lines has only just begun, and opportunities for advancing the field abound.

A second challenge relates to the extent to which transferred knowledge gets reused. Our system presently attempts to recreate the observed source play in the simulator, making minor adjustments to tailor performance to the new environment. However, with improvements to the system's ability to consider alterations to the plays, the system could easily mix and match the contents of several observed plays into a broad variety of original plays. This is important because it implies the ability to leverage a small amount of carefully selected training into a wide array of skills and behaviors.

On a smaller scale, two other improvements will play an important role in making our transfer system more robust. The first concerns symbol mapping. Currently our system assumes that the source and target description languages are equivalent, which is not typically true. Könik et al (2009) describe an approach to this problem that would produce a substantially more robust transfer system. The second line deals with handling coordination among players explicitly. Currently each player receives an individual goal at the beginning of each play with successful achievement ensuring the needed synchronization among players. A better solution would be for the system to encode all of the individual player goals and player coordination information into a single, top-level team goal. This would allow players who initially fail to achieve their goals (for example due to a collision with a defenseman) to recover and try to complete the play anyway.

Related Work

Research on procedural knowledge transfer is relatively rare as compared to work on transfer in classification settings. One popular framework for work on procedural transfer is reinforcement learning, where the object of transfer is often a value function (Taylor and Stone 2005). This often requires auxiliary mappings between domain states and actions, and the technology for acquiring those mappings can become quite involved as the gap between source and target environments grows. For example, Liu and Stone (2006) employ a structure mapping engine to find analogies among qualitative dynamic Bayes nets expressing soccer keep-away tasks, while Torrey et al (2006) transfer situation-action rules obtained by inductive logic techniques, but employ them to provide advice for constructing a value function. See Taylor and Stone (this issue) for a more detailed introduction to transfer in reinforcement learning.

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

Our work on reusing recognition knowledge for play design continues a trend towards bridging increasingly distant source and target tasks. However, in contrast with some of the examples described the above, we bridge distinct purposes versus representations. This shifts emphasis from the mapping technologies that enable transfer onto the power of the skill representation itself. Choi et al. (Choi et al. 2007) argue that the generalization inherent in ICARUS skills naturally supports performance in similar tasks.

One important and practical benefit of our transfer system is its ability to load complex, domain-specific knowledge into agents. Such agents may be used to populate auxiliary characters in games or training simulators, or may serve as adversaries to human players or trainees. At present, constructing game agents is an important, but expensive task. Most recent efforts at building game agents focus on finite state machines designed prior to play. However, the complexity of the automaton, hierarchical or otherwise, increases dramatically with the complexity of the game (Orkin 2006). Likewise, scripts for controlling agent behavior are both popular and difficult to compose for complex games and situations.

Other artificial intelligence-based approaches to loading knowledge into agens include that of Kelley et al (2008), who use off-line planning with hierarchical task networks to generate scripts automatically. This approach requires

expert knowledge to build the hierarchical task networks, while our system acquires similar knowledge automatically. An alternate approach uses reinforcement learning algorithms (Bradley and Hayes 2005; Nason and Laird 2005) to allow an agent to learn through experience. While our approach shares a number of ideas with these, it differs in its use of analytical learning, which tends to require many fewer observations or experiences than reinforcement learning.

Concluding Remarks

Research in transfer of learning has great potential to impact the ways in which we build and train autonomous agents. In this article, we focused on training agents to perform complex tasks in complex environments based on observation of human performance, which is one application of transfer to agent construction. More specifically, our work provides a clear proof-of-concept demonstration for three key capabilities: action recognition in a complex environment, transfer of action recognition into procedural knowledge, and adaptation of the constructed procedures to a new environment.

Our work also suggests that cognitive architectures provide an appropriate vehicle for performing such tasks. Although we augmented the ICARUS architecture substantially to perform the tasks discussed in this article, we can imagine an extended version of the architecture that integrates many of the same capabilities. Expanding the presented transfer system into one that performs robustly across a variety of tasks and domains will require substantial innovation along several fronts. Nevertheless, our work suggests the potential for the approach to generalize and scale to a broad class of problems.

Acknowledgements

This material is based on research sponsored by DARPA under agreement FA8750-05-2-0283, and NSF under grant IIS-0546867. The U. S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of DARPA, NSF, or the U. S. Government.

References

Anderson, J. R. 1993. *Rules of the Mind*. Hillsdale, NJ: Lawrence Erlbaum.

Bradley, J., and Hayes, G. 2005. Group utility gunctions: Learning equilibria between groups of agents in computer games by modifying the reinforcement signal. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 1914–1921. Edinburgh, UK: IEEE Press.

Choi, D.; Könik, T.; Nejati, N.; Park, C.; and Langley, P. 2007. Structural transfer of cognitive skills. In *Proceedings of the Eighth International Conference on Cognitive Modeling*. Ann Arbor, MI: Taylor and Francis / Psychology Press.

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

Hess, R., and Fern, A. 2007. Improved video registration using non-distinctive local image features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recogntion*. Minneapolis, MN: IEEE Press.

Hess, R., and Fern, A. 2009. Discriminatively trained particle filters for complex multi-object tracking. In *Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Miami, FL: IEEE Press.

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

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

Intille, S., and Bobick, A. 1999. A framework for recognizing multi-agent action from visual evidence. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 518–525. Orlando, FL: AAAI Press.

Kelley, J. P.; Botea, A.; and Koenig, S. 2008. Offline planning with hierarchical task networks in video games. In Darken, C., and Mateas, M., eds., *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*. Stanford, CA: AAAI Press.

Kok, S., and Domingos, P. 2007. Statistical predicate invention. In *Proceedings of the 24th International Conference on Machine Learning*, 433 – 440. Corvallis, OR: ACM Press.

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

Könik, T.; Ali, K.; Shapiro, D.; Li, N.; and Stracuzzi, D. J. 2010. Improving structural knowledge transfer with parameter adaptation. In *Proceedings of the 23rd Florida Artificial Intelligence Research Society (FLAIRS) Conference*. Daytona Beach, FL: AAAI Press.

Laird, J. E.; Newell, A.; and Rosenbloom, P. S. 1987. Soar: An architecture for general intelligence. *Artificial Intelligence* 33(1):1–64.

Langley, P., and Choi, D. 2006. A unified cognitive architecture for physical agents. In *Preceedings of the Twenty-First National Conference on Artificial Intelligence*, 1469–1474. Boston: AAAI Press.

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*, 1858–1863. Amsterdam, Netherlands: Cognitive Science Society, Inc.

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

Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. *Intl. Journal of Computer Vision* 60(2):91–110.

Mayer, R., and Wittrock, M. 1996. Problem-solving transfer. In Berliner, D., and Calfee, R., eds., *Handbook of Educational Psychology*. New York: MacMillan Library Reference USA. 47–62.

Nason, S., and Laird, J. E. 2005. Soar-rl: Integrating reinforcement learning with soar. *Cognitive Systems Research* 6(1):51–59.

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

Orkin, J. 2006. Three states and a plan: The AI of F.E.A.R. In *Game Developers Conference*.

Robertson, S. 2001. *Problem Solving*. Philadelphia, PA: Psychology Press / Taylor and Francis.

Sakoe, H., and Chiba, S. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26(1):43–49.

Salomon, G., and Perkins, D. N. 1989. Rocky roads to transfer: Rethinking mechanisms of a neglected phenomenon. *Educational Psychologist* 24:113–142.

Stracuzzi, D. J.; Li, N.; Cleveland, G.; and Langley, P. 2009. Representing and reasoning over time in a symbolic cognitive architecture. In *Proceedings of the 31st Annual Meeting of the Cognitive Science Society*, 2986–2991. Amsterdam, Netherlands: Cognitive Science Society, Inc.

Taylor, M. E., and Stone, P. 2005. Behavior transfer for value-function-based reinfocement learning. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, 53–59. Utrecht, The Netherlands: ACM.

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

David Stacuzzi is a senior member of the technical staff at Sandia National Laboratories. He received his doctorate from the University of Massachusetts at Amherst in 2006, followed by a postdoc at Stanford and three years as research faculty at Arizona State. His research interests include machine learning, cognitive architectures, and cognitive modeling.

Alan Fern is an associate professor of computer science at Oregon State University. He received a PhD in computer engineering at Purdue University in 2004. His research interests are in the areas of machine learning, automated planning, and high-level computer vision.

Kamal Ali has been doing research in machine learning and data mining since 1987. He has worked in both academia and industry, and is currently co-founder at Metric Avenue and serving as editor for a book on machine learning and data mining for astronomy. At the time of this work, he was a senior research scientist at the Computational Learning Lab at CSLI at Stanford University.

Rob Hess is currently a PhD candidate in the Computer Science Department at Oregon State University. He earned his BS at the University of Delaware. His current research interests are in Computer Vision and Artificial Intelligence and include object tracking and learning in structured spaces. He has also done work in information visualization and open-source education.

Jervis Pinto is a PhD candidate in the Machine Learning group at Oregon State University in beautiful Corvallis, Oregon where he works in the research area of Reinforcement Learning. He has a bachelor's degree in computer engineering from Mumbai University, India.

Nan Li is a PhD student in Computer Science at Carnegie Mellon University. She received her Masters degree from Arizona State University and her Bachelors degree from Peking University, both in Computer Science, in 2009 and 2006 respectively. Her research interests include artificial intelligence, machine learning, and information retrieval.

Tolga Könik is a postdoctoral research scientist with the Logic Group at Stanford University. His research combines ideas from artificial intelligence, machine learning, and cognitive science with the goals of reducing the difficulty and cost of building AI systems, enabling users who have no programming background to create or customize them, and providing AI systems with the ability to improve themselves by interacting with humans.

Daniel Shapiro is the Executive Director of the Institute for the Study of Learning and Expertise (ISLE), where he conducts research on cognitive architectures, and cognitive systems that learn. He directed ISLE's prime contract on the DARPA Transfer Learning Program. Dr. Shapiro is the chair for the Innovative Applications of AI conference in 2011, and a member of the advisory board for the Value Driven Design Institute (VDDI). He received his PhD in Management Science and Engineering from Stanford University in 2001.