

A thesis submitted in partial satisfaction of the requirements
for the degree of Master of Computer Science and Engineering
in the Graduate School of the University of Aizu

Learning Soccer Player Behavior On Real-World Data



by

Victor Khaustov

March 2018

© Copyright by Victor Khaustov, March 2018

All Rights Reserved.

The thesis titled
Learning Soccer Player Behavior On Real-World Data

by
Victor Khaustov

is reviewed and approved by:

Main referee

Associate Professor

Maxim Mozgovoy

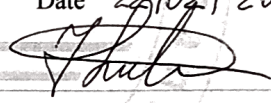
Date 22/02/2018



Professor

Ihor Lubashevsky

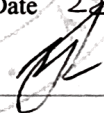
Date 22/02/2018



Senior Associate Professor

Evgeny Pyshkin

Date 22/02/2018



THE UNIVERSITY OF AIZU

March 2018

Contents

Chapter 1 Introduction	1
Chapter 2 Related work.....	2
Chapter 3 Processing the data.....	5
3.1 Movements detection.....	5
3.2 Passes recognition	6
3.3 Shots recognition.....	7
3.4 Substitutions, injuries, red cards and incorrect data	7
Chapter 4 Description of implementation	9
Chapter 5 Experiments	17
Chapter 6 Conclusion.....	20
References	21

List of Figures

Figure 2.1 State-action graph.....	2
Figure 2.2 Generalization levels.....	3
Figure 3.1 Real-world data visualization.....	6
Figure 3.2 Manual annotating of passes and shots using ANVIL Software.....	7
Figure 4.1 Interaction between subsystems.....	9
Figure 4.2 A hierarchy of main subsystems.....	10
Figure 4.3 Game situation adapters.....	11
Figure 4.4 Game events analysis.....	12
Figure 4.5 Action converter.....	13
Figure 4.6 Action extractor.....	14
Figure 4.7 Action executing process.....	15
Figure 4.8 Knowledge subsystem.....	16
Figure 5.1 A part of knowledge graph.....	17
Figure 5.2 Midfield Player decision-making.....	18
Figure 5.3 Decision-making of another team's midfielder.....	18
Figure 5.4 Defender's decision making on midfielder's knowledge.....	19

Abstract

The analysis of digitalized real soccer game data is an important task for a variety of problems, ranging from sports analytics to data-driven game AI systems. The purpose of the thesis is to present a way to learn AI on real-world soccer data. Since spatial data (player and ball movements) is typically available as a series of game snapshots containing player coordinates, a method to extract real game actions, such as movements, passes, and shots is proposed. The resulting data is used for learning AI. The architecture of an active learning system for representing, storing and retrieving knowledge is discussed in the thesis. The main principle behind the proposed system is asynchronous decision-making about each AI agent's actions, using the agent's up-to-date knowledge about a game. An AI agent's knowledge about what is happening in the game is represented by semantic directed graphs - acting graphs and generalization trees both growing and being analyzed in real time. The experiments with real-world data show that the proposed system is able to learn the behavior of human soccer player of a particular role and reproduce actions in similar game situations.

Chapter 1 Introduction

Growing amount of real-world human behavior data brings new opportunities to learn Artificial Intelligence (AI) on it. For example, most popular games, such as soccer, basketball, ice-hockey collect data for calculating game statistics (possession, pass accuracy, goal strikes count, distance run, speed etc.). A widely used solution on the market is TRACAB system by ChyronHego Corporation [1]. TRACAB system uses sophisticated image-processing technology to trace the position of all moving objects on the field of play with a frequency of 25 times per second. The product is a live data feed that contains all the X, Y and Z positional coordinates for each identified object, including soccer players, referee and the ball.

The problem of learning AI on real-world data is important, and it not only can lead to improvements in the sport industry by helping teams to recognize their strength and weaknesses against particular opponent, but also can improve computer game industry by delivering human-like AI agents into soccer simulator games.

The thesis presents an attempt to learn a human-like AI on the real-world data of five soccer matches of J1 League (Japanese top division soccer league). Believability (human-likeness) is stated as an important characteristic of AI-controlled agents. In its turn, believability can be achieved with methods that can learn and reproduce human behavior [2, 3].

A method for developing AI system for learning soccer player behavior on real-world data proposed in the thesis is based on decision trees and it employs not only players' coordinates, but their actions, such as movements, passes and shots. For that reason, proposed approach requires the initial data to be preprocessed and actions to be extracted. Running of the experiments in automated mode on implemented system was described in a journal article [4].

The thesis is organized as follows. The second chapter looks into related work, describes several approaches that can be used for the task of building believable AI systems and explains the method used in the thesis. The third chapter explains how to handle the real-world data: how to convert coordinates of the objects on the field into movement actions and how to extract passes and shot actions. The fourth chapter dives into details on how to implement and use the proposed AI system. The fifth chapter presents the results of experiments performed on implemented system and last chapter gives a brief conclusion.

Chapter 2 Related work

When agent's believability is stated as a goal, the approach of building AI systems usually refers to one of the following three categories:

1. Rely on expert knowledge. Decision-making logic can be directly hand-coded according to an expert view of the given domain. In certain cases, this solution can be adequate, and result in a solid AI system [5].
2. Mimic human decision-making process. A system can be designed on the basis of contemporary psychological theories of human behavior [6].
3. Rely on actual logs of human behavior. In the most straightforward form, this approach can be reduced to replaying human decision-making logs verbatim [2], but usually it is implemented via machine learning: agents are trained on human data to provide the same actions as human players in similar situations [3].

The approach used in the thesis falls into the third category. It is a behavior-capture (learning by observation) system based on case-based reasoning and case-based planning [7]. Its behavior-capture ability provides feasible solution for development of AI agents by transferring knowledge from the experts such as professional soccer players directly to AI agents and capturing different behaviors from different players within one system.

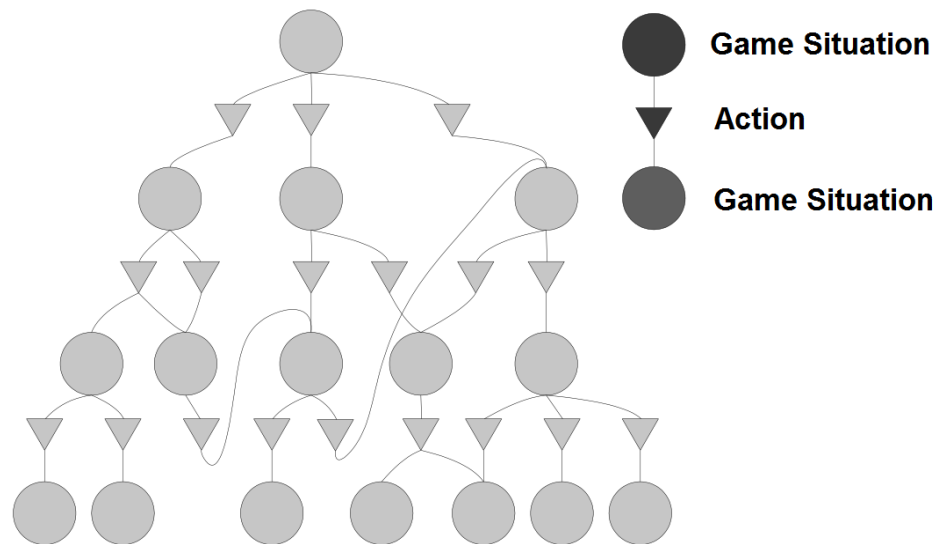


Figure 2.1 State-action graph

Learning by observation is an efficient form of knowledge acquisition which requires the expert demonstrating the task being learned. An AI agent "observes" the human game play and derives a knowledgebase based on the player actions given each game situation. By analyzing a soccer match frame-by-frame, the agent processes the raw data based on the decision logic/factors that are provided to it and develops action graphs and situation generalization structures that are the basis of its behavior. The behavior can be evaluated by having the agent play the game in the place of the person. At any point, the action graph can be edited to change behavior through additional game training. The

researcher also has the option to update the decision logic or the factors the agent is considering when making decisions.

Knowledge is represented as state and action (decision) pairs organized in a directed graph (action graph) (as shown in Figure 2.1) coupled with a state generalization directed acyclic graph (generalization graph). Generalization graph groups states into several levels of abstraction.

The Figure 2.2 shows an example of an action graph coupled with a generalization forest. A plane with black nodes illustrates a game situation space viewed on a lowest level of abstraction. Two trees with red nodes illustrate a sample generalization graph built ‘on top’ of an action graph.

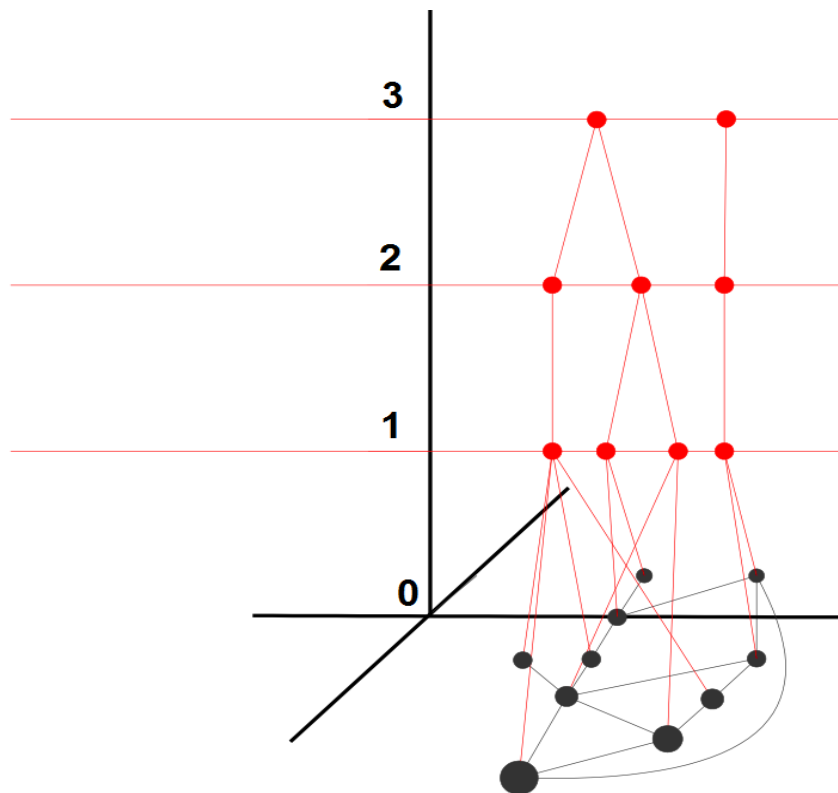


Figure 2.2 Generalization levels

During learning, AI agents automatically add new nodes and arcs (i.e. states and transition) to their action graphs, and also add related nodes and arcs to their generalization graphs. AI agent’s decision-making process consists of a situation matching step and an action matching and planning step.

During situation matching step AI agents search for similar states in their action graphs utilizing the generalization graphs and extract a set of relevant actions. The situation matching step can be described as a case-based k-nearest neighbors reasoning. By extracting relevant cases from the knowledge, AI determines a set of actions, applicable in the current situation.

Action matching prioritizes possible actions according to their applicability to the current state. Actions that are applicable to the current state are further analyzed. Such

analysis creates sequences of actions and examines their applicability based on the current situation, current action and the learnt situation and action sequences in the knowledge.

The action matching step utilizes an auxiliary application-specific rule-based system. The rule-based system has two purposes: introduce domain-dependent common sense base logic (which is not a subject to automatic learning) into the system, and to be able to use different factors depending on the type of the action.

Proposed system utilizes knowledge of sequences in the <situation, action> space. It allows to preserve learned action sequences when necessary. To support optional efficient self-learning to achieve set goals AI system can use active online reinforcement learning with delayed rewards.

Learning of team behavior is structured as concurrent learning of collaborative behaviors of individual team members. Each agent learns how to act in a team by observation of a human soccer player's actions with actions of other teammates during training sessions. This process allows capturing of players' collaborative behavior to a team of agents. Every agent learns to act in a team due to anticipation of teammates' actions learned during training sessions.

Chapter 3 Processing the data

For the game of soccer, digitalized recordings of human behavior can be obtained with TRACAB technology [1] that relies on video streams obtained from six still video cameras installed in a stadium. These streams are processed with tracking software that identifies player and ball coordinates at any game moment. Resulting data files contain X, Y coordinates of each player on the field and X, Y, Z coordinates and speed of the ball with a frame rate of 25 frames per second.

Such tracking data can be used in a variety of tasks, mostly in the field of sport analytics. However, TRACAB format does not contain high-level information describing the actions that actually take place on the game field. In particular, it lacks vital information analytical about player movements, passes and shots on goal. While the task of extracting this information can be automated, the process is not straightforward, since numerous difficulties have to be addressed. In the subsequent sections such difficulties will be discussed and possible solutions will be provided.

Since the primary goal of the thesis is designing soccer game AI system proposed in a previous chapter, the primary interest of this chapter is on extracting high-level actions of players, such as movements, shots, and passes. For other purposes (e.g., soccer analytics) one might also need to detect various game events, such as red cards, goals, throw-ins, and substitutions.

To ensure that the data is correct real-world data visualization tool was developed. The example of visualization in action is shown in Figure 3.1. Video of matches was compared to the visualized data by watching them side by side. It was found that in many cases the data does not correctly reflect the situation on the field.

3.1 Movements detection

For the proposed system player movements should be represented as “move in direction D for N frames” instead of more obvious “move to the point (X, Y).” To decrease overall complexity of the system movement actions are chosen to be of eight possible directions: forward, backward, left, right, left-forward, right-forward, left-backward, and right-backward. To determine movement direction, an angle between X-axis and the direction vector $\{X_1 - X_0, Y_1 - Y_0\}$ is calculated (where X_1, Y_1 are player coordinates in the current frame, and X_0, Y_0 are player coordinates in the previous frame), and the value is rounded to the closest direction. The consecutive frames with the same movement directions are combined into a single movement action with a duration equal to the number of consecutive frames.

While there are advantages of this approach like dimensionality reduction, reducing the number of directions may decrease system’s accuracy. To compensate accumulating errors, one of possible solutions is to keep track of player movements performed with

simplified 8-direction game control, and if the difference between the resulting player coordinates and the original (precise) coordinates reaches a certain threshold value, it is corrected by introducing additional movement actions.

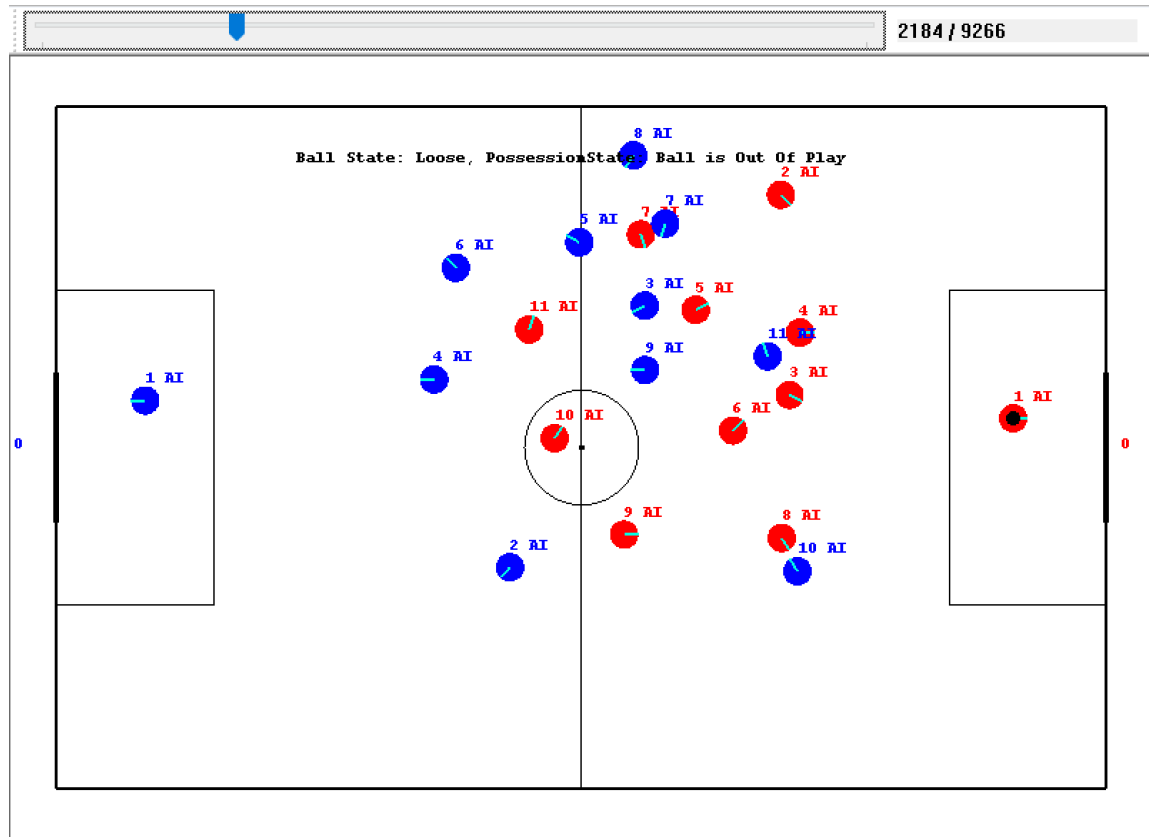


Figure 3.1 Real-world data visualization

3.2 Passes recognition

When the distance between the ball and some player is less than certain predefined radius, such situation is recognized as ball possession. Once a player loses the ball, and the next player to possess the ball happen to belong to the same team, the situation is recognized as a pass action, and the other player is set as a pass target.

The most difficult situations to recognize are inaccurate and intercepted passes, since the crucial goal is to guess player's intention and identify the possible pass target. In addition, inaccurate passes should not be confused with tackles. The tackles can be identified by examining whether any of the opponents was present in the immediate neighborhood of the player and on the way of the ball. The trajectory lines of all the teammates and of the ball can be confirmed for their intersections. The teammate with the closest point of intersection of its trajectory and the ball trajectory is considered to be the pass receiver.

3.3 Shots recognition

To extract shot actions, the following situation is more likely to be detected: a player loses control of the ball, and later the ball crosses the goal line or appears to be controlled by the goalkeeper. For the shots on goal the intersection of trajectory of the ball and a goal line is set as an intended shot point. For the shots off target the closest point inside the goal is set as an intended shot point. To distinguish a shot from an inaccurate pass the speed of the ball and teammates' positions can be taken into account. All ball movements with ball speed above a certain threshold are considered as shots. If the speed of the ball is low, it is recognized as pass only if there are potential pass receivers close to the ball trajectory (see the previous section).

3.4 Substitutions, injuries, red cards and incorrect data

Raw TRACAB data sets do not contain information of player substitutions. Each player has an associated “jersey number” that can get a value of -1 on any frame, which means that the given player is not active (substituted, inured, or got a red card). Consequently, the number of players of the field is not constant. In the beginning of the game, each side has 11 players, but during the match certain players might be removed.

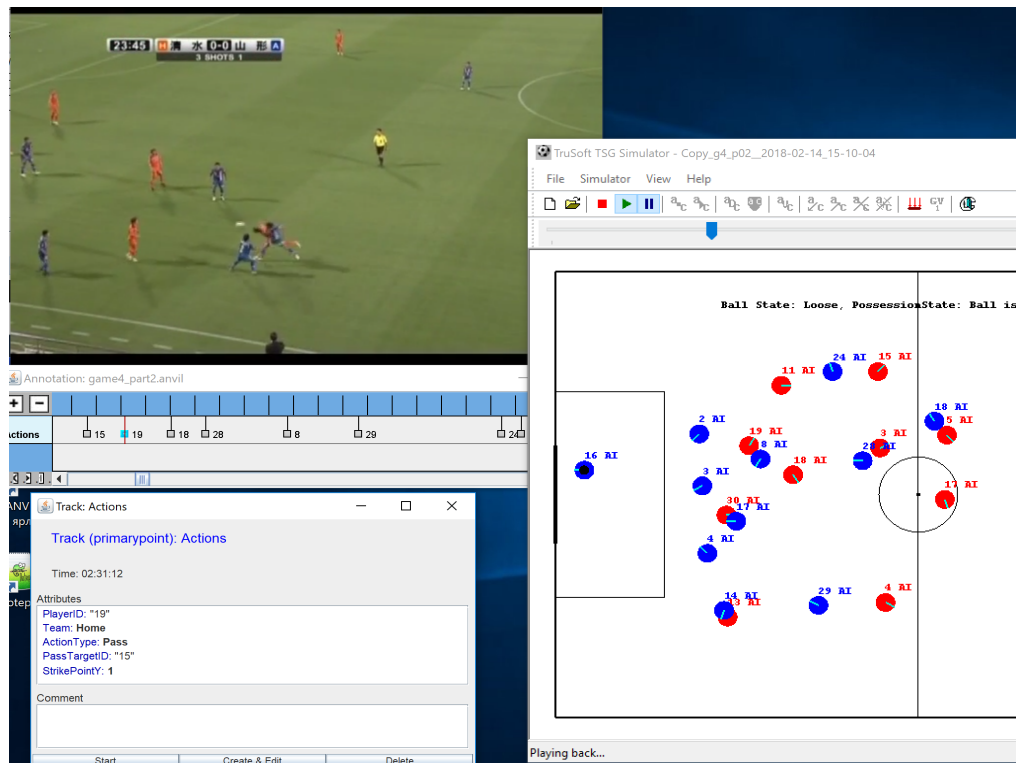


Figure 3.2 Manual annotating of passes and shots using ANVIL Software

The player data provided by the system is not always correct. Sometimes a ball or a player appear on the wrong side of the field (most probably, due to tracking error). Therefore, during the preprocessing stage the previous frame information should be stored to compare it with the current frame player coordinates and check whether the system-provided data can be considered possible, given the maximum speed values of players and the ball. If the number of matches in the dataset is small the actions can be manually annotated by ANVIL Software (as shown in Figure 3.2).

Chapter 4 Description of implementation

Proposed AI system deals with numerous subsystems, both generic and game-specific. The structure of subsystems and their interaction is a topic of this chapter.

The interaction between the separate subsystems of the proposed AI system is shown in Figure 4.1

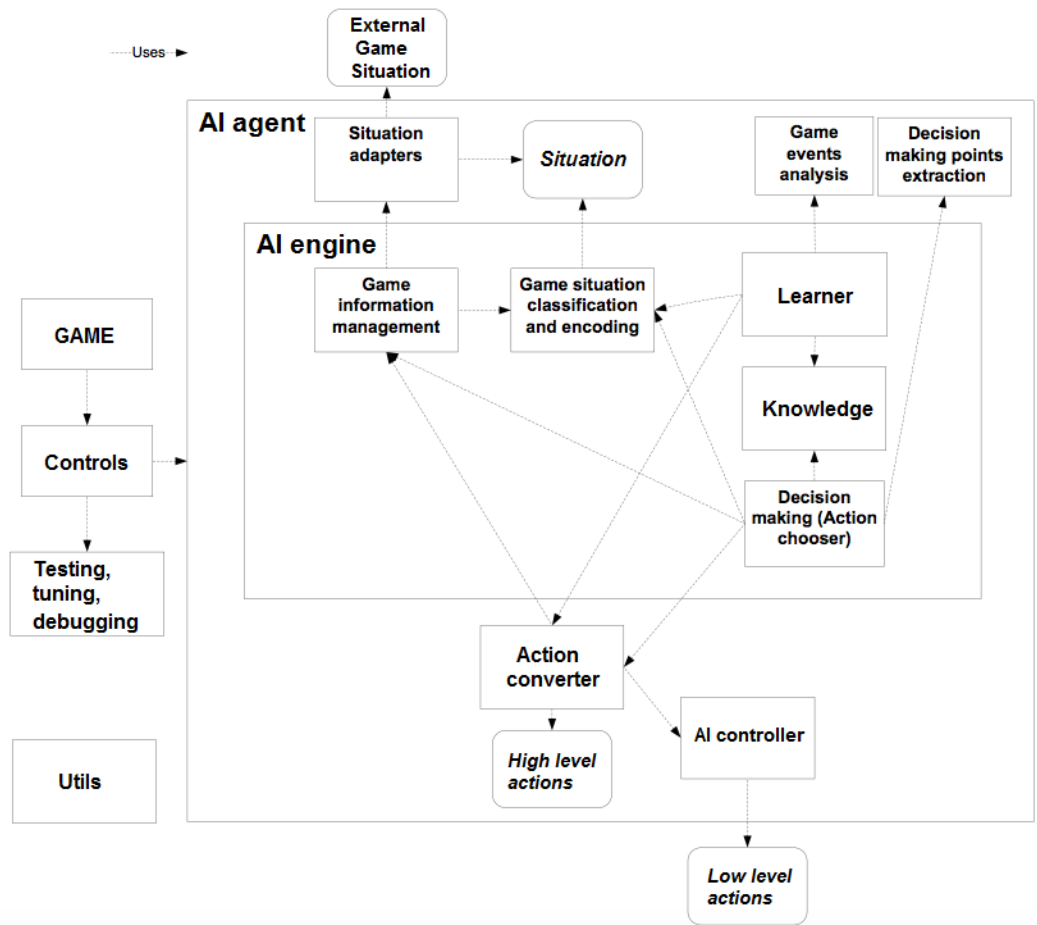


Figure 4.1. Interaction between subsystems

Separate subsystems form a hierarchical relationship, shown in Figure 4.2. As it can be seen, most elements are direct successors of either AI agent or AI engine.

AI Agent subsystems, introduced in this chapter, serve as an interface between general AI technology and soccer-specific modules. AI agents have to be able to analyze situations and actions that occur in the game, react to game events, and pass the decisions to the game engine.

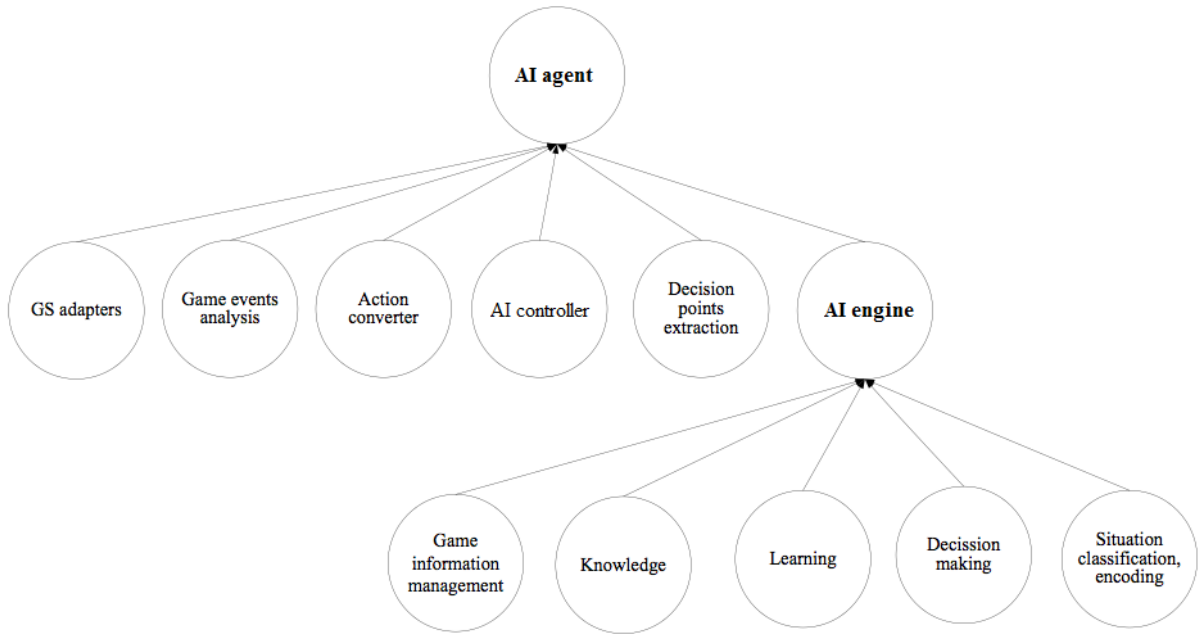


Figure 4.2. A hierarchy of main subsystems

Game Situation (GS)

Game situation (or game state) is a complete game situation description for a soccer game. It is considered as “raw” description of a game state on a certain frame.

Game Situation Recording (GS Recording)

Real-world data is processed and converted into game situation recording. This is a complete recording of the internal game states needed for AI agent’s knowledge of the game environment. Recordings are classified as either training recordings (in which an AI agent learns) or acting recordings in which the agent acts using the knowledge that it acquired during learning. Stored recording files allow to review how any given game was played and how the AI agent acted or learned during gameplay. AI tools allow viewing the recordings and analyzing every decision made by AI agent step by step. Furthermore, it is possible to review AI’s “grounds” for every decision – referring to specific instances where AI agent has learned a tactic from a human player in previous training sessions. Both “learning” and “acting” recordings can be replayed. Recordings are stored and they can be viewed using a dedicated tool.

AI can learn from the recordings. Therefore, it is possible to retrain an AI agent if something is changed in the agent’s learning algorithms. This important feature allows to play training games very early in the development of an AI agent. It is possible to learn from any part of a recording — and therefore to edit any knowledge.

GS Recording Files

GS recording files play an important role in the development and training of AI agents:

- Every part of the match is permanently stored as a GS recording file or immediately discarded (if during that part the ball was out of play and if the part is not going to be used in training or testing).
- Each GS recording file may contain all the attributes of a soccer game.
- Each GS recording file might have a number of tags.
- Structured notes (allowing qualitative interpretation of a GS recording file). GS recording files can be organized in a number of subsets to create training or testing sessions.
- Each subset of GS recording files can contain any combination of plays.

GS recording file tags can be organized in a way allowing a semi-automatic system to create subsets of GS recordings. The format for GS recording should allow the support for versions.

Each GS recording (a recorded play) can be used for the purposes of:

- AI agents' training;
- AI agents' testing.

AI Game Situation (ACGS)

AI game situation is a “game situation as it seen by the agent”. It can differ from “raw” description, e.g. with the presence of heuristic values and with explicit knowledge of an agent-controlled game player.

Game Situation Adapters

GS Adapters (shown in Figure 4.3) convert (adapt) a game situation object to the corresponding AI game situation.

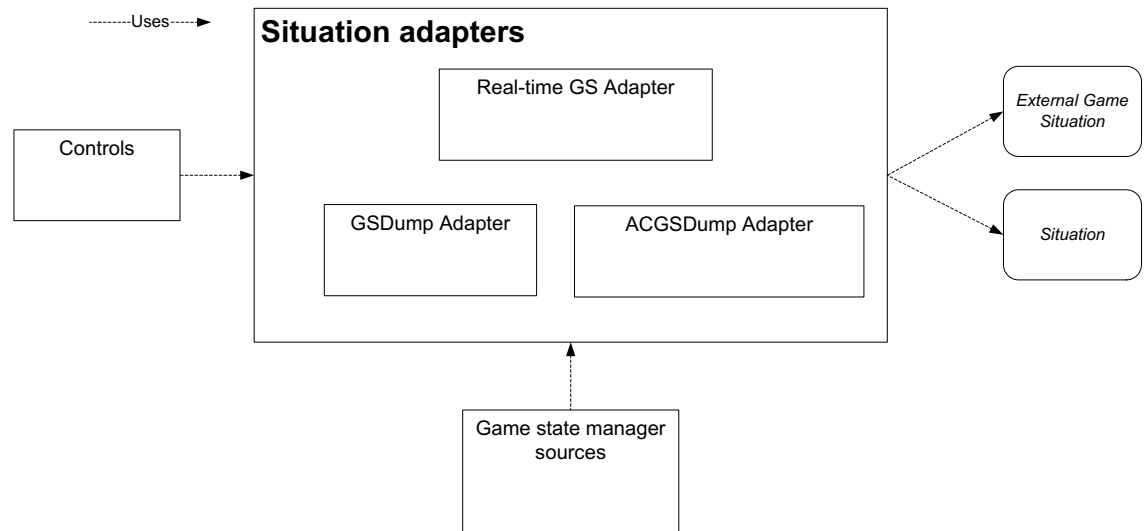


Figure 4.3. Game situation adapters

Real-time GS Adapter

Real-time GS Adapter is designed to convert raw game situation data that comes directly from a game engine.

GS Recording Adapter

GS Recording Adapter converts an element of GS recording into the corresponding AI game situation object.

Game Action

Game action is an elementary action of the game AI engine works with. Logically, game actions performed by all entities in the game are part of a game situation. Technically, AI technology does not enforce to store actions in GS objects.

AI Action

AI actions are high-level actions stored in the agent's knowledgebase. AI actions are related to game actions in the same way as AI game situation is related to "raw" game situation. AI actions are directly executable by the agent.

Game Events Analysis

Game events analysis (shown in Figure 4.4) is a logical module that examines the events in the game that are important for AI engine's learning and acting.

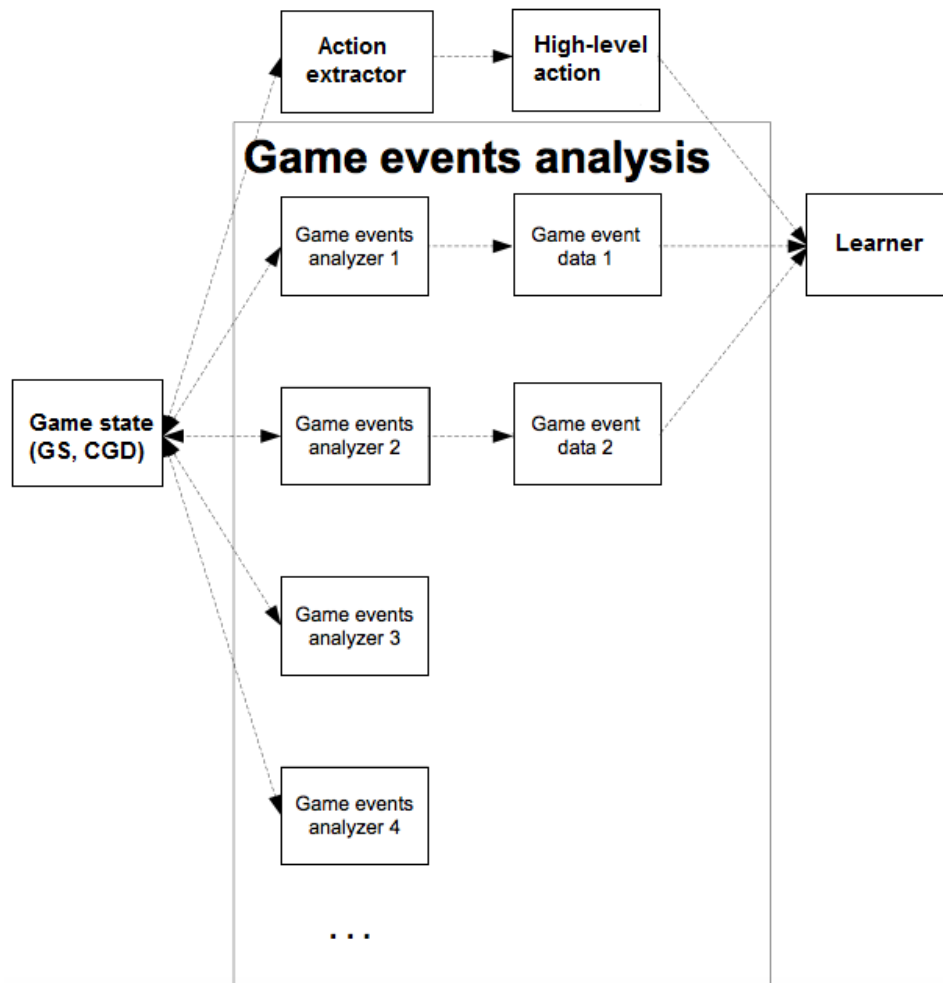


Figure 4.4. Game events analysis

For example, a soccer AI agent could perform the following tasks:

- Analysis for special game situations (scoring a goal etc.)
- Analysis for game phases (switching from attack to defense, etc.) and other events.
- Analysis for the possible actions, i.e. which types of actions are possible to execute in the current situation. Having this task completed, the game is divided into game stages corresponding to a particular set of actions that are allowed to be executed.

Decision Making Points Finding and Extraction

Decision making points are time moments when the agent either acts or learns new actions. The actual work of this module is determined by the mode of the system (learning or acting).

During acting this mechanism is used to determine when the system should perform the next AI action. During learning this module assigns decision making points for a teacher.

Action Converter

The purpose of Action converter subsystem is twofold. On a learning phase, it extracts high level game actions from a sequence of game situations and low level actions (i.e. converts low level actions to high level actions). On an acting phase, it executes a given AI action (i.e. converts high level actions to low level actions). Generally, this subsystem works as a converter between low level game actions and high level AI actions. The structure of Action converter is shown in Figure 4.5

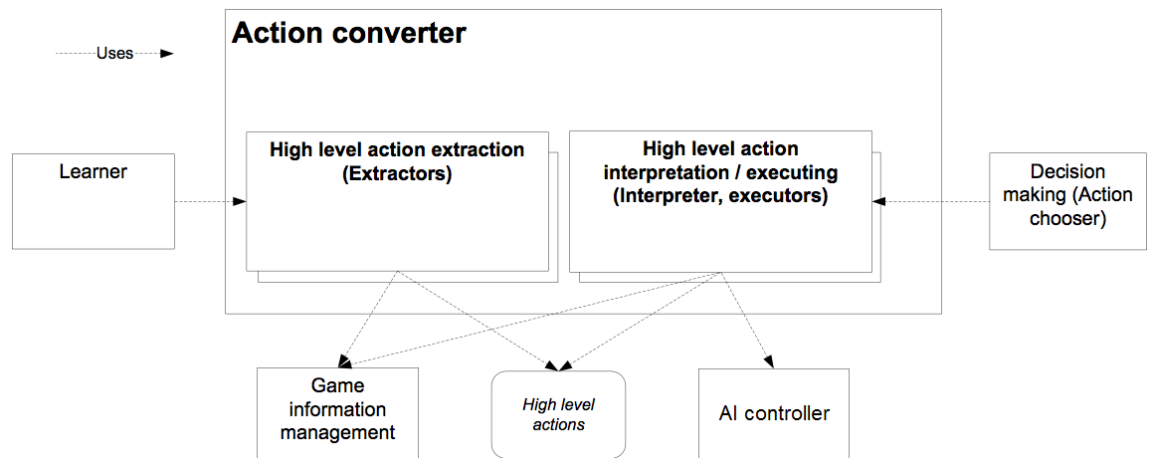


Figure 4.5. Action converter

AI Action Extractor

This subsystem provides means for extraction of high level AI actions from a sequence of game situations and low-level game actions. A soccer game action extractor is shown in Figure 4.6

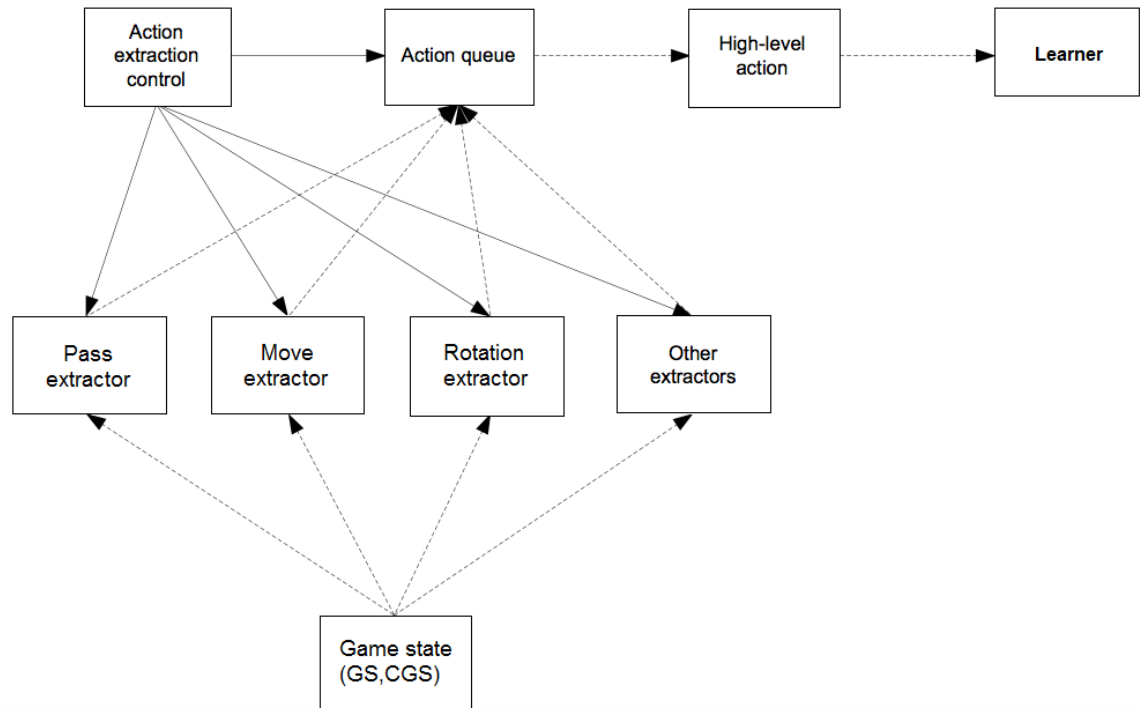


Figure 4.6. Action extractor

AI Action Executor

This subsystem's goal is to execute high level AI actions as a series of low level game actions while taking the following into account:

- the subsystem should handle differences between the current situation and the situation during learning;
- the subsystem should be able to adjust the sequence of game actions in case of game situation changes during acting.

The diagram of the subsystem is shown in Figure 4.7

Local Classified Game Situation (LCGS)

In most cases, game situation includes lots of information that AI will not use. Some aspects of game situation may be important only for the particular decision-making mode or for a particular AI's subsystem.

LCGS is a representation for a single aspect of game situation, important for AI's knowledge and decision-making process. This representation can be a simple value from the situation tracked with a certain level of details or it may be a value of some kind of heuristic.

Local Classifier (LC)

LCs may be very simple, or they may be non-trivial subsystems that evaluate some heuristic values.

Local classifiers may be static or dynamic. Static local classifiers can create LCGS using situation data on one frame. Dynamic classifiers need some "history", though it is

better to get the necessary information from one frame (i.e. to avoid creating dynamic classifiers), if possible.

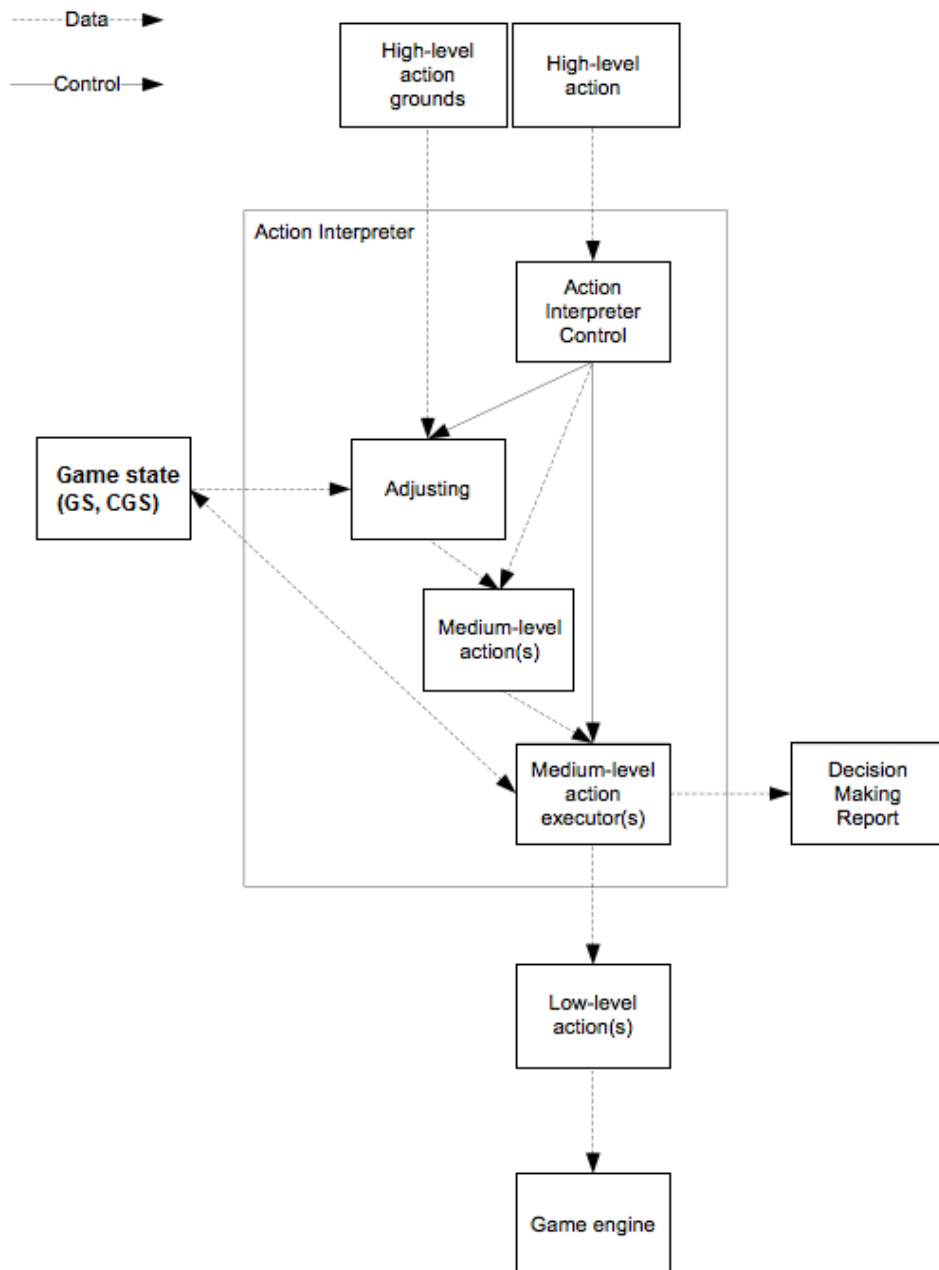


Figure 4.7. Action executing process

Compound Objects and Zoom Levels

A game situation can be analyzed on different levels of abstraction (called zoom levels, Figure 2.2). This functionality is essential for searching similar game situations. For example, two game situations can be treated as equal if they match at the lowest (the most detailed) zoom level, and as “similar enough” if they match on the higher levels of abstraction only. The most detailed zoom level is referenced as “zoom level zero” or “Z0”.

Acting Graph and Knowledge

The behavior of any AI agent is determined by the contents of its knowledgebase, represented in the form of acting graph (Figure 2.1). The nodes of this graph are individual game states, and the arcs are game actions. If states S_1 and S_2 are connected with the arc labeled A, it means that the game is switched from the state S_1 to the S_2 by executing action A.

Acting graph is connected to the system of zoom levels, so that AI engine can analyze the relations between the game situations on different levels of abstraction.

The scheme of Knowledge subsystem is shown in Figure 4.8.

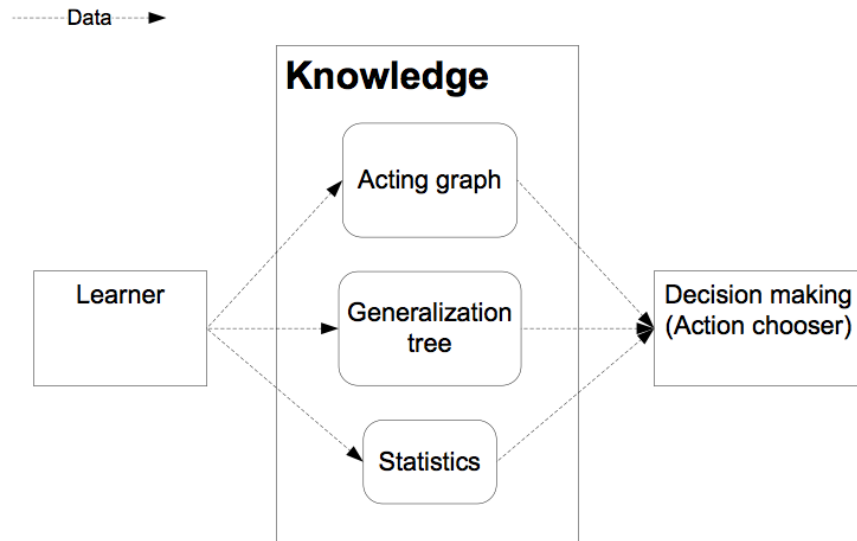


Figure 4.8. Knowledge subsystem

Chapter 5 Experiments

To evaluate the implemented system that observes logs of real-world soccer matches several experiments have been conducted. The system can learn human player behavior from a single match or numerous matches. The experiments aimed to test the ability of the system to learn the behavior of a player of a particular role on the field.

The data for experiments consists of 5 real-world soccer matches of a top Japanese soccer league (J1 league). To exercise the proposed system the dataset was divided into learning and acting by a proportion of 80:20. Learning dataset of 500 000 frames contained about 54 000 actions. Acting dataset of 132 000 frames contained 13 500 actions.

A role of the midfielder was chosen for learning. Figure 5.1 shows a part of a knowledge graph generated during learning phase.

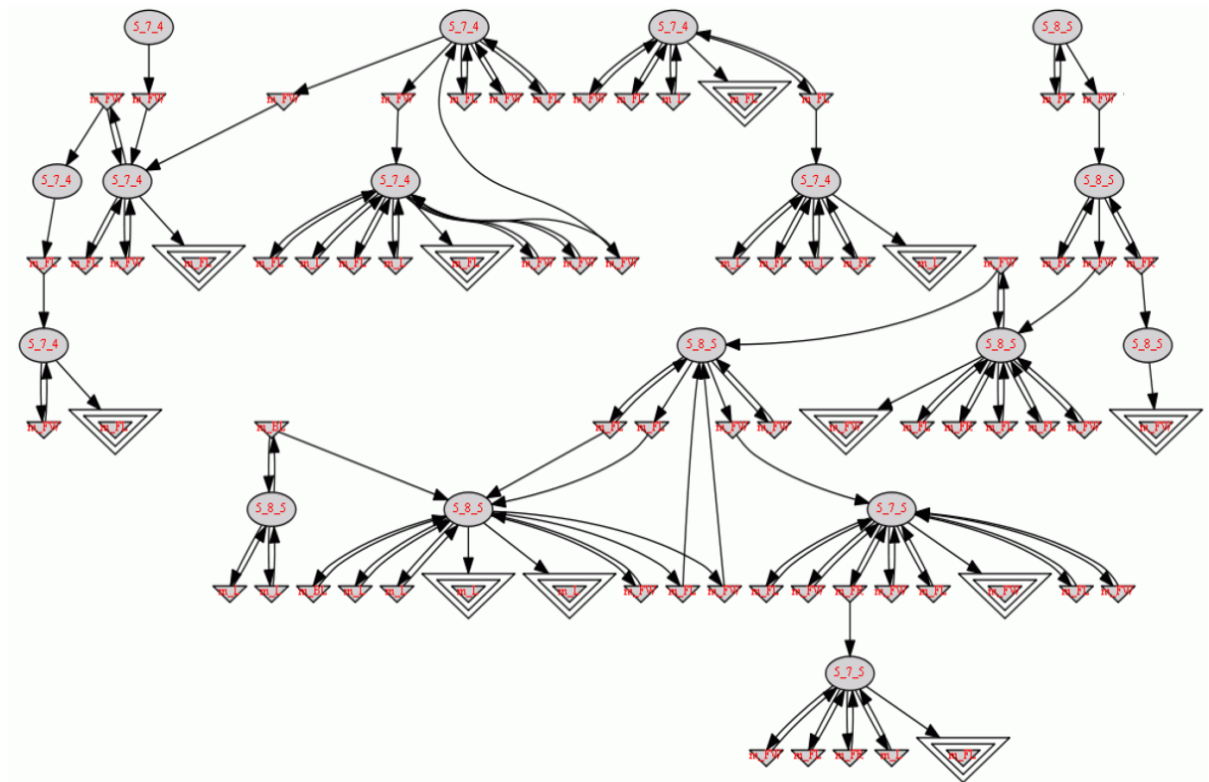


Figure 5.1 A part of knowledge graph

First, the resulting AI agent's knowledge was tested on acting dataset of the player itself. Figure 5.2 shows a chart of decisions retrieved from a knowledge graph. In 14% of cases AI agent was able to find a corresponding action on the lowest generalization level (Level 0) where the highest number of game attributes is taken into account. And in 65% cases the agent was able to retrieve knowledge from less precise generalization Level 1. Only in 1 % of cases agent could not find a match in the knowledgebase.

Midfield Player decision-making

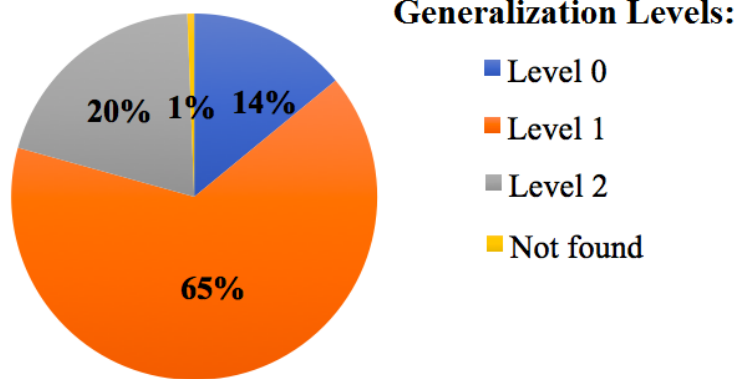


Figure 5.2 Midfield player decision-making during training

Next, the AI agent's knowledge was tested on acting dataset of the player of the same role (midfielder) but from a different team. As shown in Figure 5.3, there were no corresponding knowledge found on the generalization Level 0, but in 45% of cases it was discovered on the generalization Level 1 and in 53% on the generalization Level 2.

Midfield Player (Other Team)

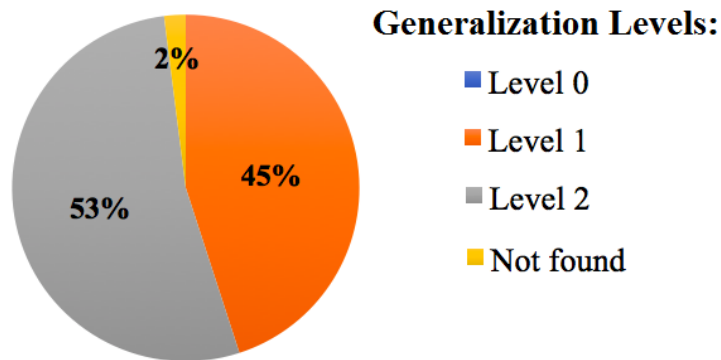


Figure 5.3 Decision-making of another team's midfielder

Finally, when the decision-making process was tested on acting dataset of the player of the different role (defender) of the same team (Figure 5.4), the AI agent could not find a corresponding action in the knowledgebase in 63% of cases, and the rest of the actions were discovered on the generalization Level 2, which is the least precise level containing only a fraction of game attributes.

Defender's decision-making

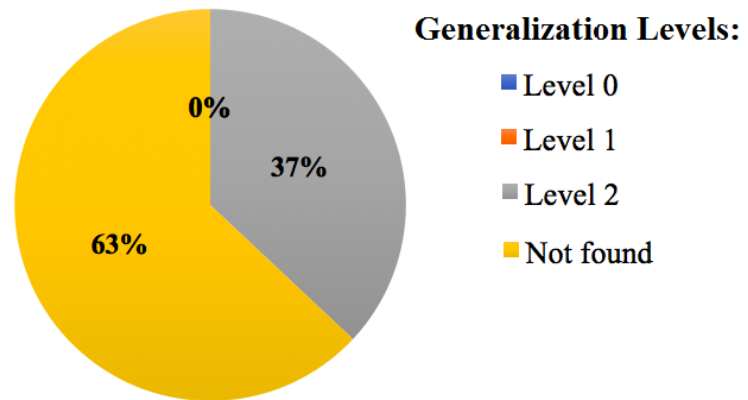


Figure 5.4 Defender's decision-making on midfielder's knowledge

Chapter 6 Conclusion

An active learning system, including methods for representing, storing and retrieving knowledge, was presented in the thesis. The way to use it with real-world soccer data was described. It was shown that using the system with real-world soccer data and learning player behavior on it requires additional data processing steps to recognize players actions. Typical challenges of recognizing actions have to be addressed to use tracking of real soccer matches for the purposes of learning player behavior. Furthermore, while preprocessing the data one should take care of possible substitutions, red cards and frames containing incorrect information.

The main principle behind the proposed system is asynchronous decision-making about each AI agent's actions, using the agent's up-to-date knowledge about a game. Knowledge is organized hierarchically in a flexible number of levels of abstraction – zoom levels. Each zoom level corresponds to a set of specific game situation representations. Game situation representations are game dependent but flexible.

An AI agent's knowledge about what is happening in the game is represented by semantic directed graphs — acting graphs and generalization trees both growing and being analyzed in real time. Acting graphs store “game situation -> action -> game situation” – type data. Generalization trees provide means for structured hierarchical analysis of game data — the physical organization of “zoom levels.”

The proposed system allows development of AI agents, providing a standardized way to test and analyze AI agents' behaviors. It is possible to see grounds for every decision that an AI agent makes. It is possible to look into AI knowledge and analyze its structure. AI decisions are always transparent compared to Neural Networks approach. The experiments with real-world data showed the ability of proposed system to learn player behavior of a particular role.

References

- [1] TRACAB Technology. Project homepage: <http://tracab.com>
- [2] I. V. Karpov, J. Schrum, and R. Miikkulainen, “Believable Bot Navigation via Playback of Human Traces,” in *Believable Bots: Can Computers Play Like People?*, P. Hingston, Ed, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 151–170.
- [3] J. Ortega, N. Shaker, J. Togelius, and G. N. Yannakakis, “Imitating Human Playing Styles in Super Mario Bros,” *Entertainment Computing*, vol. 4, no. 2, 2013, pp. 93–104.
- [4] V. Khaustov, M. Mozgovoy. “Teaching Automated Software Testing with Appium and Soccer Simulator,” *NOSU Bulletin*, 2017, pp. 124-127.
- [5] A. M. Mora, F. Aisa, P. García-Sánchez, P. Á. Castillo, and J. J. Merelo, “Modelling a Human-Like Bot in a First Person Shooter Game,” *International Journal of Creative Interfaces and Computer Graphics (IJCICG)*, vol. 6, no. 1, pp. 21–37, 2015.
- [6] Asensio, Joan Marc Llargues et al, “Artificial Intelligence approaches for the generation and assessment of believable human-like behaviour in virtual characters,” *Expert Systems with Applications*, vol. 41, no. 16, pp. 7281–7290, 2014.
- [7] M. Mozgovoy and I. Umarov, “Behavior Capture with Acting Graph: A Knowledgebase for a Game AI System,” in *Databases in Networked Information Systems (DNIS): 7th International Workshop*, S. Kikuchi, A. Madaan, S. Sachdeva, and S. Bhalla, Eds, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 68–77, 2011.
- [8] Kipp, M. (2014) “ANVIL: A Universal Video Research Tool,” in: J. Durand, U. Gut, G. Kristofferson (Eds.) *Handbook of Corpus Phonology*, Oxford University Press, Chapter 21, pp. 420-436