FRA-UNIted — Team Description 2024

Thomas Gabel, Berkan Eren, Eicke Godehardt

Faculty of Computer Science and Engineering
Frankfurt University of Applied Sciences
60318 Frankfurt am Main, Germany
{tgabel|godehardt}@fb2.fra-uas.de, {berkan.eren}@stud.fra-uas.de

Abstract. The main focus of FRA-UNIted's effort in the RoboCup soccer simulation 2D domain is to develop and to apply machine learning techniques in complex domains. In particular, we are interested in applying reinforcement learning methods, where the training signal is only given in terms of success or failure. In this paper, we review some of our recent efforts taken during the past year, putting a special focus on tool support and team performance analyses.

1 Introduction

The soccer simulation 2D team FRA-UNIted is the continuation of the former Brainstormers project which has ceased to be active in 2010. The ancestor Brainstormers project was established in 1998 by Martin Riedmiller, starting off with a 2D team which had been led by the first author of this team description paper since 2005. Our efforts in the RoboCup domain have been accompanied by the achievement of several successes such as multiple world champion and world vice champion titles as well as victories at numerous local tournaments. Our team was re-established in 2015 at the first author's new affiliation, Frankfurt University of Applied Sciences, reflecting this relocation with the team's new name FRA-UNIted.

As a continuation of our efforts in the ancestor project, the underlying and encouraging research goal of FRA-UNIted is to exploit artificial intelligence and machine learning techniques wherever possible. Particularly, the successful employment of reinforcement learning (RL, [8]) methods for various elements of FRA-UNIted's decision making modules – and their integration into the competition team – has been our main focus. Moreover, the extended use of the FRA-UNIted framework in the context of university teaching has moved into our special focus.

In this team description paper, we refrain from presenting approaches and ideas we already explained in team description papers of the previous years [1]. Instead, we focus on topics that most recently moved into our focus. We start this team description paper, however, with a short general overview of the FRA-UNIted framework. Note that, to this end, there is some overlap with our older team description papers including those written in the context of our ancestor project (Brainstormers 2D, 2005–2010) which is why the interested reader is also referred to those publications, e.g. to [7, 6].

1.1 Design Principles

FRA-UNIted relies on the following basic principles:

- There are two main modules: the world module and decision making
- Input to the decision module is the approximate, complete world state as provided by the soccer simulation environment.
- The soccer environment is modeled as a Markovian Decision Process (MDP).
- Decision making is organized in complex and less complex behaviors where the more complex ones can easily utilize the less complex ones.
- A large part of the behaviors is learned by reinforcement learning methods.
- Modern AI methods are applied wherever possible and useful (e.g. particle filters are used for improved self localization).

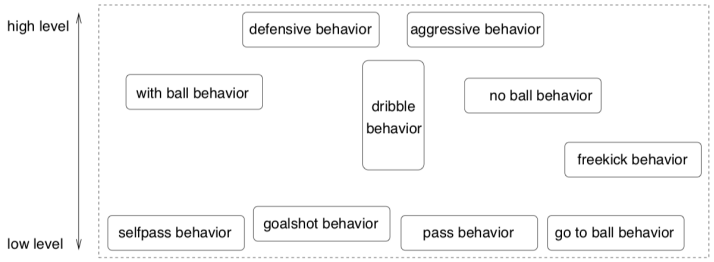

Fig. 1. The Behavior Architecture

1.2 The FRA-UNIted Agent

The decision-making process of the FRA-UNIted agent is inspired by behavior-based robot architectures. A set of more or less complex behaviors realize the agents' decision making as sketched in Fig. 1. To a certain degree this architecture can be characterized as hierarchical, differing from more complex behaviors, such as "no ball behavior", to very basic, skill-like ones, e.g. "pass behavior". There is no strict hierarchical sub-divisioning, which is why a low-level behavior may call a more abstract one. For instance, the behavior responsible for intercepting the ball may, under certain circumstances, decide that it is better to not intercept the ball, but to focus on more defensive tasks and, in doing so, call the "defensive behavior" and delegating responsibility for action choice to it.

2 Replayer: A Tool for Simulating Multiple Futures

The normal course of a simulated soccer match is that a game is played using the Soccer Server [5]. After the match has been completed the server saves a log file which can later be replayed and inspected via the log-playing capabilities that are nowadays integrated into the Rcssmonitor program[1].

[1] https://github.com/rcsoccersim/rcssmonitor

Now, let us dare to make a thought experiment: Imagine how the sequence of events would have changed, e.g., if at timestamp 1352 player 7 of team A would have successfully tackled (which it did not), regardless of a moderate 73% tackle success rate. How would the further course of the game have been influenced in this scenario? What, if we intended to modify our agents' playing and tackling behavior in order to see what kind of events would have emerged in this altered scenario? Questions like these were the motivation for starting to conceptualize and develop a Replayer tool – a soccer software tool with which we are able to play scenarios anew, change our algorithms to see, if we can optimize the outcome of some scenario, or just retry it multiple times.

2.1 Overview of the Project

The Replayer is a tool that allows soccer simulation 2D teams to jump to a timestamp t from which a game is played anew. To achieve this, we build a software that feeds the Soccer Server with the *recorded* commands of each agent and of the coaches and utilizes the offline coach to get an accurate context for times before t. Figure 2 provides an overview of the endeavour.

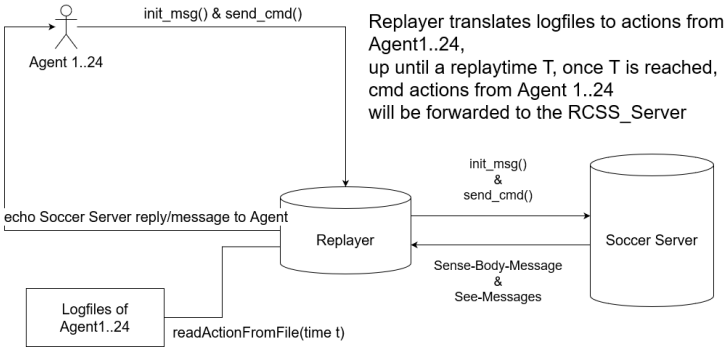

Fig. 2. Architecture of the Replayer Project: Integration with Soccer Server and Connected Player and Coach Agents

The offline coach shall be employed to set positions of all agents and of the ball and to force each connected agent to have accurate (if we assume that the agents are deterministic) or semi-accurate information (if an agent is non-deterministic). After time step t has been reached, the binaries of the connected agents and coaches will take over, i.e. the match proceeds in a normal way, and the commands that they are sending will be simply forwarded to the server. Possible use cases for this project include the opportunity to update some binary (add new or change existing behavior) and to test a specific functionality as well as to replay the situation to see what would happen, e.g., if an action would have succeeded instead of having failed.

We identified two primary milestones to be pursued:

– M1: Connect agents/coaches with Replayer and have a normal game played
– M2: Reading the log (`.rcl`) files, connecting the offline coach, and have a game being replayed through the Replayer and setting the positions via the offline coach

2.2 Connection and Setup Issues

A first important step is to consider how a connection to the Soccer Server is structured and maintained. In Figure 3, we visualize how an initial connection between Replayer and Soccer Server is supposed to be established. First, the Soccer Server is started with port 6003 in order to allow our Replayer to take port 6000, which is the default port for the communication between agents and the Soccer Server in the 2D league. Thus, the player agents will be sending their UDP packages to port 6000, which is taken by the Replayer.

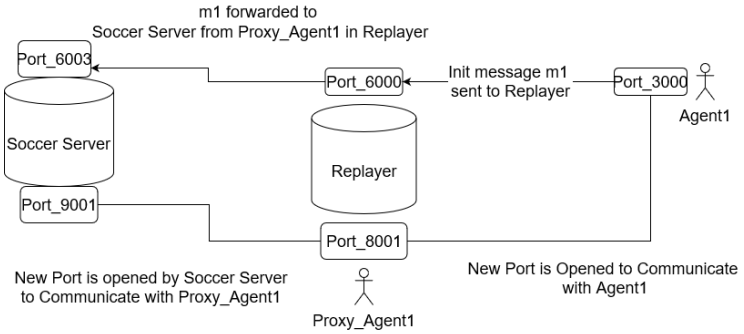

Fig. 3. Message Paths in the Replayer Setup

Upon receiving the first initial message `m1` within the Replayer, a new port corresponding to agent 1 is opened. Here, a Proxy_Agent1 is created in the Replayer which will be responsible for remembering the actions sent by agent 1. We then forward our initial message from port 8001 to the Soccer Server which will open a new port in the Soccer Server that is responsible for Proxy_Agent1. Besides the connection of the player agents we also have to connect the coach agents using a separate port (by default the Soccer Server port for coaches and offline coaches are two separate ones).

2.3 Work In Progress

Current work in the project focuses on implementing the required form of message parsing. Up to now, we are therefore limited in our ability to manage the different types of connection requests. A problem arising from the delineated implementation is that the first eleven agents' plus the first coach's init messages

will be assigned to team A and the subsequent eleven agents' and second coach's init messages will automatically be assigned to team B. As a workaround, we require team A to connect completely before connecting the team B to the Replayer. If a monitor instance tries to connect to the Replayer, it would also be falsely assigned as an agent, as it attempts to connect to port 6000 by default. Hence, another workaround is to start the monitor with parameters commanding it to connect to the correct server port which can easily be done using appropriate command line arguments.

This project is far from finished, and also requires us to add a few extra functionalities implemented in the Soccer Server. For example, we need the possibility to swap from the offline coach to an online coach, while a match is running, or to let the offline coach act as a regular coach after we reached time t.

2.4 Future Work and Open Issues

Among other things, we have to incorporate a parser, which will allow for assigning agents/coaches to their respective teams and manage the monitor's connection without mistaking it as an agent. Our intended parsing functionality will have to include the offline coach's language and ensure that the context of each agent is exactly the same by setting position, velocity, and stamina of each object according to the log files. This will require the Replayer to be able to interpret and translate commands of each agent while it is in the replay portion of the game before time t. Another future idea is to create some "Scenario Management" tool within the Replayer or even outside of it, which might be used to set up more complex scenarios in which machine learning algorithms might be utilized to train portions of the agents' behavior. Once the project has its barebones and a sufficiently functioning program is created, we will share this with the soccer simulation community making it an open-source project.

3 Continuous Integration for Uncovering Subtleties in Team Behavior: The Miracle of Goals at Half Time

Since each match of 2D soccer is tainted to various degrees by randomness, the need to play a large number of matches arises, in order to limit the impact of chance on the overall results. For this reason, our team has developed and established the use of a continuous integration (CI) system which we already presented in an earlier team description paper [2]. While this system is under steady further development [4], we can by now report on successful use cases where its application helped in revealing some of our team's weaknesses.

Figure 4 visualizes a particular observation that was revealed using the mentioned CI system when testing against Helios' 2022 champion binary. That figure provides the accumulated number of goals shot and received from 3050 games plotted as a histogram with windows of 100 time steps width. While our team received on average approximately about 100 goals within 100 time steps throughout the game except for the very beginning (i.e. $100/3050 \approx 0.033$ in each window

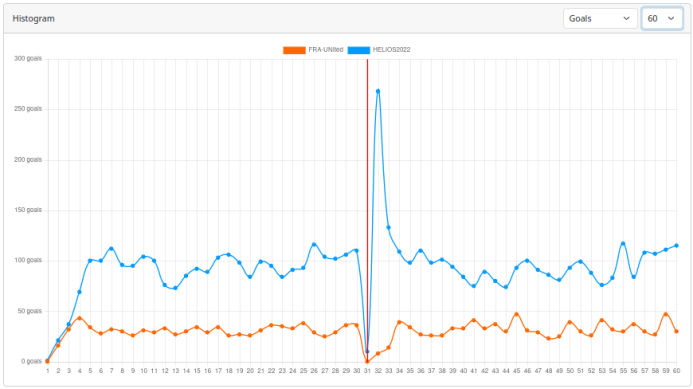

Fig. 4. 3050 Matches in Three Nights: Something peculiar is going at half time breaks.

per match), there is a protruding peak shortly after the half time break, i.e. between time steps 3100 and 3200, where the amount of goals received is about 2.7 times as high as normally during the match.

Since our team, FRA-UNIted, is the only one remaining in the starting field of the soccer simulation 2D competitions which defends using a man-to-man marking approach [3], our first response to the delineated astonishing statistic was to blame the coach, its calculations for determining suitable marking assignments, as well as the way it communicates such re-calculated assignments at the half-time break. Since, however, we found no anomalies through code inspection and by a number of variations introduced to the coach's behavior, we next attempted to flip sides: So, while the series of test matches shown in Figure 4 were performed such that our team played from left to right and Helios from right to left, we now conducted a series with swapped side. Figure 5 shows the resulting statistics which left us even more puzzled.

While the average score in the former series was 0.59 : 1.83, the average score in the second one was 0.59 : 1.77 (from FRA-UNIted's point of view in either case), which represents a non-significant change. Nevertheless, the outstanding peak seemed to have disappeared with Helios scoring approximately three times as many goals as out team in any phase of a match. A more thorough analysis of the depicted observations is currently work in progress and shall be presented in a separate paper.

4 Holes and Clashes: Reunion with a Long-Presumed Dead Adversary

We analyzed the log files of major competitions during past years and found a degradation of the reliability of the hardware setup and its computational power provided to the teams. In particular, we observed that – starting in 2022 – many teams' players did frequently miss to send a command within 100ms.

Fig. 5. 9443 More Matches in Nine More Nights: Changeover with Helios playing from left to right and FRA-UNIted from right to left.

If some agent does not send a command in one time step at all, then usually something is wrong (called a *hole*). If some agent sends in one time step t more than one command, then usually something is very wrong (e.g. two dashes, or a dash and a turn). It usually means that this agent did not manage to send the first of the two commands within the 100ms of the previous time step $t-1$, i.e. it is being sent one step too late. Simultaneously, the agent sends the command of the current time step t. Thus, the server receives the command from the previous *and* the current time step in the same cycle (called a *clash*).

Holes and clashes are bad for any agent. They used to be a big problem in the early days of soccer simulation 2D, when computational resources were scarce as well as when the network connections needed for exchanging messages via UDP were not as reliable as they are today. While we believed that holes and clashes are a problem of another decade and they occur only very rarely or due to programming errors by some team nowadays, we observed their systematic return in competitions of the last and current year.

We analyzed log files from recent world championships tournaments and searched the rcl files for the occurrence of holes and clashed. In RoboCup 2019, there were no problems across all teams (with the exception of Ri-One which we think to be due to a team-internal problem) and a minor issue for YuShan2019 (with, however, less than 1 black hole in the course of two matches). Likewise, there were essentially no issues for RoboCup 2021: Counting the total number of clashes for RoboCup 2021, we found that there was just a single one in the entire tournament (by Helios2021).

Unfortunately, things changed in RoboCup 2022 where, for the first time, a Docker-based environment was used. The summed number of clashes has been increased to 557 in the entire tournament – a number which is certainly far beyond anything acceptable. Apparently, all teams were affected where YuShan2022 was affected most with almost seven clashes per game on average.

Of course, one might argue that the prevention of clashes is in the responsibility of each team itself. However, we strongly believed that the wide re-occurrence of clashes from 2022 onward points to some inherent problem with the way the simulation environment has been set up recently. Our analysis revealed that often the entire simulation seemed to be "frozen" for one time step or for multiple time steps that are close to each another.

We brought our observations to the public, distributing them via the veteran `robocup-sim` mailing list as well as through the Soccer Simulation 2D community channels in Discord. Our ignition spark raised the public awareness of the issue and initiated, as part of the preparation for RoboCup 2023, a very thorough analysis by Omid Amini (team Cyrus) and Alireza Saddraii Rad that has been publicized and is available on: `https://www.rcss.dev/holesclashes`

5 Conclusion

In this team description paper we have outlined the characteristics of the FRA-UNIted team participating in RoboCup's 2D Soccer Simulation League. We have stressed that this team is a continuation of the former Brainstormers project, pursuing similar and extended goals in research and development as well as for teaching purposes. Specifically, we have put emphasis on our most recent research activities and practical implementation of our results.

References

1. Gabel, T., Breuer, S., Sommer, F., Godehardt, E.: FRA-UNIted - Team Description 2019 (2019), Supplementary material to RoboCup 2019: Robot Soccer World Cup XXIII, LNCS, Sydney, Australia
2. Gabel, T., Eren, Y., Sommer, F., Vieth, A., Godehardt, E.: FRA-UNIted - Team Description 2022 (2022), Supplementary material to RoboCup 2022: Robot Soccer World Cup XXVI, LNCS, Bangkok, Thailand
3. Gabel, T., Riedmiller, M.: On Progress in RoboCup: The Simulation League Showcase. In: J. Ruiz-del-Solar, E. Chown, P. Plöger, editors, RoboCup 2010: Robot Soccer World Cup XIV, LNCS. pp. 36–47. Springer, Singapore (2010)
4. Godehardt, E., Allani, M., Vieth, A., Gabel, T.: 1001 Games a Night – Continuous Evaluation of an Intelligent Multi-Agent Based System. In: Proceedings of the 8th International Congress on Information and Communication Technology (ICICT 2023). Springer, London, United Kingdom (2023)
5. Noda, I.: Soccer Server: A Simulator of RoboCup. In: Proceedings of the AI Symposium 1995. pp. 29–34. Japanese Society for Artificial Intelligence (1995)
6. Riedmiller, M., Gabel, T.: On Experiences in a Complex and Competitive Gaming Domain: Reinforcement Learning Meets RoboCup. In: Proceedings of the 3rd IEEE Symposium on Computational Intelligence and Games (CIG 2007). pp. 68–75. IEEE Press, Honolulu, USA (2007)
7. Riedmiller, M., Gabel, T., Hafner, R., Lange, S.: Reinforcement Learning for Robot Soccer. Autonomous Robots 27(1), 55–73 (2009)
8. Sutton, R., Barto, A.: Reinforcement Learning. An Introduction. MIT Press/A Bradford Book, Cambridge, USA (1998)