# EduBAI: An Educational Platform for Logic-Based Reasoning

Dimitrios Arampatzis[1,2], Maria Doulgeraki[1,2], Michail Giannoulis[2], Evropi Stefanidi[1,2], and Theodore Patkos[1,2](✉)

[1] Institute of Computer Science, FO.R.T.H., Heraklion, Greece
{arabatzis,mdoulger,evropi,patkos}@ics.forth.gr
[2] Computer Science Department, University of Crete, Heraklion, Greece
giannoulis@csd.uoc.gr

**Abstract.** The field of logic-based Knowledge Representation and Reasoning has produced powerful formalisms for modeling commonsense knowledge in Artificial Intelligence. In this paper, we present EduBAI, an educational platform that helps users familiarize themselves with the main tenets of commonsense reasoning in dynamic, causal domains by means of an interactive entertaining environment. We present the design and implementation of the platform, along with the rationale of sample game tactics of diverse modeling complexity.

**Keywords:** Answer set programming · Event Calculus · Logic-based reasoning

## 1 Introduction

Formal theories for reasoning about action and change are well-established logical theories for reasoning about the dynamics of changing worlds, contributing solutions to domains as diverse as high-level robot cognition, Ambient Intelligence, complex event detection, and others. The heterogeneous application areas of these formalisms is attracting the interest of researchers from a variety of disciplines and with diverse backgrounds. Given the quite advanced technical training needed to master logic-based reasoning, it becomes evident that the educational process of understanding both the theoretical and the practical aspects of commonsense reasoning is a challenging task.

This paper aims to facilitate the learning process of logic-based reasoning, by introducing EduBAI (Educational Basketball playing using AI), a game platform and accompanying material that can assist students and researchers having basic knowledge of the underlying formalisms in writing, executing and evaluating logic-based axiomatizations. The main contributions of this paper can be summarized as follows:

This project has received funding from the Hellenic Foundation for Research and Innovation (HFRI) and the General Secretariat for Research and Technology (GSRT), under grant agreement No 188.

J. MacIntyre et al. (Eds.): AIAI 2019, IFIP AICT 559, pp. 464–472, 2019.
https://doi.org/10.1007/978-3-030-19823-7_39

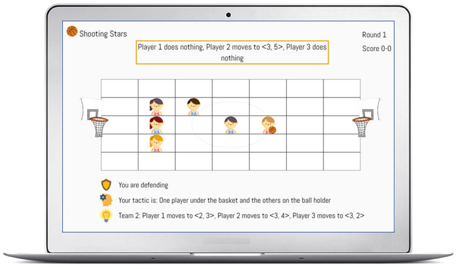
- We present a fully operational, interactive educational platform, that enables users to focus on improving their logic programming skills in an entertaining way, hiding technical details regarding in-game communication, visualization of outputs etc.
- We offer a testbet for experimenting with a rich range of reasoning features. The game scenario involves automating the decision making of 3-player teams competing against each other in a basketball match. The gameplay gives the ability to explore aspects related to executing causality-based reasoning, geospatial reasoning, multi-player coordination, temporal reasoning, probabilistic reasoning, function optimization, among others.
- We offer a set of example tactics of diverse level of modeling complexity that can be used as educational material for users interacting with the platform.

The gamification nature of our approach aims to offer a more entertaining way of introducing logic-based programming, motivated by the observation that toy problems in the form of grid settings and logic puzzles have proven to be an effective way of understanding insights of AI programming.

The rest of the paper is structured as follows. Section 2 describes in detail the game dynamics, the methodology used, as well as the possibilities for learning reasoning aspects through diverse team tactics. Section 3 presents implementation details, while Sect. 4 discusses related work and future directions.

## 2 Platform Design and Methodology

EduBAI is an interactive educational platform implementing a basketball game between two 3-player teams. The overall goal of the platform is to enable users model the intelligence of their team using formal languages, such as the Event Calculus and ASP, in the context of a dynamic, non-deterministic domain.


Shooting Stars

Round 1

Score 0-0

Player 1 does nothing, Player 2 moves to <3, 5>, Player 3 does nothing

You are defending

Your tactic is: One player under the basket and the others on the ball holder

Team 2: Player 1 moves to <2, 3>, Player 2 moves to <3, 4>, Player 3 moves to <3, 2>

**Fig. 1.** Snapshot of the EduBAI UI

The Event Calculus [4,6] is a narrative-based many-sorted first-order language for reasoning about action and change, which explicitly represents temporal knowledge, enabling reasoning about the effects of a narrative of events along a time line. It also relies on a non-monotonic treatment of events, in the sense that by default there are no unexpected effects or event occurrences. The game dynamics of EduBAI, such as the players' actions, their effects etc, are described as Event Calculus axioms.

ASP [3,5] is a declarative problem solving paradigm oriented towards complex combinatorial search problems. A domain is represented as a set of logical rules, whose models, called answer sets, correspond to solutions to a reasoning task, such as progression or planning. Sets of such rules, or answer set programs, come with an intuitive, well-defined semantics, having its roots in research in knowledge representation, in particular nonmonotonic reasoning. EduBAI implements a translation of Event Calculus theories into ASP rules, which are then executed by the state-of-the-art Clingo ASP reasoner[1].

Figure 1 shows the main user interface of EduBAI, displaying a snapshot of a game, including the position and state of each player in the court, as well as informative messages describing the progress of the game. The platform is free-to-use and available online for testing and play.[2]

## 2.1 Main Concepts, Rules and Game Mechanics

Each of the six players can move within the premises of the EduBAI basketball court, represented as a 5 × 7 grid. The players can perform only a limited set of actions, namely *shoot, move* (up, down right, left) and *pass.* At most one action can be performed at each timepoint by each player, except from the player holding the ball, who cannot stay inactive. Player actions are contingent on appropriate preconditions; for instance, a player can perform a pass only if she is in possession of the ball. The players cannot move beyond the boundaries of the court, but any number of players can be located in the same cell at any moment.

At each timepoint, first the attacking team decides its actions for all its players and, then, the defending team takes turn, having observed the opponent's moves. All player actions are sent to the Sports Caster Server, a component that decides non-deterministically the outcome of all actions, based on the following probabilities:

- a *shoot* action has 95% probability of being successful if it occurs in the same cell as the basket. Distance from the basket reduces the prior probability by 17% per cell. Each opponent in the same cell reduces the probability by 25%.
- a *pass* action has 99% probability of being successful if the teammate receiving the ball is in the same cell. Distance reduces this probability by 2% per cell. Each opponent in the same cell as the person making a pass reduces the probability by 25%.
- a *move* action always succeeds.

A successful shot within 1-cell distance from the basket or less counts for 2 points, all other shots count for 3 points. Indicatively, a player shooting from medium range distance having 1 opponent has 53% success probability; 3-pointers without defense succeed 61% of the times, at best; even the longest pass without defense is rather easy (79%).

The game proceeds in rounds. At the beginning of each round, the human users decide the starting position of their players, as well as the defending and attacking tactic that their team will follow during the round; these tactics cannot change in the course of the given round, therefore once this initialization step is completed, the game proceeds without any human intervention. A round

[1] https://potassco.org/.
[2] The EduBAI platform: http://139.91.183.97:8181/edubai/.

concludes if a successful shot is made or if the ball goes out of play due to an unsuccessful pass. Finally, 10 consecutive moves by the attacking team without a shot cause the attack to end and the possession of the ball to change hands.

## 2.2 Axiomatizing the Game Dynamics

One of the objectives of the EduBAI platform is to help users understand the main principles of causality-based reasoning. This is achieved by separating the axiomatization of the game dynamics, such as the rules and constraints, from the team tactics described in the next subsection. The platform includes a sample encoding of the game dynamics to be used as reference, which the users can choose to study, use or ignore, if wishing to define their own game environment.

The game dynamics are axiomatized using the Event Calculus formalism and cover a range of features that are commonly met in a typical dynamic domain. The formal definition of the main domain objects is needed first, such as the players, the ball, the cells etc. The Event Calculus relies on the notion of fluents to characterize the state of any time-varying property these objects may possess. Such properties, for instance, may concern the position of each player, the player holding the ball etc. Since the state of fluents changes as a result of event occurrences, users need to also define appropriate actions, as specified in the game description (Sect. 2.1), along with any other event considered relevant.

The domain axiomatization then follows, covering various causality features, such as the effects of events, preconditions on effects, as well as preconditions on event occurrences, triggered events caused by the occurrence of other events, concurrent event execution, causal constraints etc. The commonsense law of inertia, a key concept in causal domains, becomes also part of the axiomatization, enabling users to specify the state of fluents that remain unaffected by events. Finally, the initial state of all fluents needs to be specified, in order to complete the axiomatization of a game instance.

Table 1 shows example rules comprising part of the game environment axiomatization, expressed as Event Calculus axioms and written in the ASP syntax

**Table 1.** Sample of the game environment axiomatization.

| |
|---|
| Example of fluent and event definitions |
| $fluent(player\_pos(TM,(X,Y))) :- player(TM), pos(X,Y).$ |
| $event(pass(TM1,TM2)) :- player(TM1), player(TM2), TM1 != TM2.$ |
| |
| Example of a context-dependent effect axiom: the *move left* action |
| $initiates(move(TM,left), player\_pos(TM,(X,Y-1)),T) :-$ |
| $\quad holdsAt(player\_pos(TM,(X,Y)),T), time(T).$ |
| $terminates(move(TM,left), player\_pos(TM,(X,Y)),T) :-$ |
| $\quad holdsAt(player\_pos(TM,(X,Y)),T), time(T).$ |
| |
| Example of event occurrence constraint: a player cannot shot unless holding the ball |
| $:- happens(shoot(TM),T), not\ holdsAt(ball\_holder(TM),T), time(T), player(TM).$ |

that is used by the Clingo reasoner.[3] For example, the "move left" action is axiomatized as two rules specifying which the new set of position coordinates should be (*initiates* axiom), invalidating at the same time the old coordinates (*terminates* axiom). Similarly, an example of a constraint is shown at the bottom of the table. A constraint axiom is written as a rule with a body, but without a head, meaning that the body should not be satisfied. In this case, the *shoot* event cannot happen by a player not possessing the ball. After some familiarization with the syntax, the declarative nature of such rules often makes it easier for the human user to understand the intended meaning and the rule correlation.

## 2.3 Axiomatizing the Team Tactic Intelligence

The modeling of the dynamics of the EduBAI domain, described previously, covers certain elementary, yet crucial aspects in the process of representing causal domains. More advanced features of logic-based reasoning can be introduced as an attempt to implement intelligent team tactics to outsmart the opponent.

**Table 2.** Sample of team tactics axiomatization.

| |
|---|
| Example of a choice rule generating all combinations of specific actions (exactly one) |
| $1\{happens(doNothing(D), T); happens(move(D, left), T); happens(shoot(D), T)\}1 :- player(D), time(T).$ |
| |
| Example of optimization: assign player markings minimizing overall distance |
| $holdsAt(marking(Def, Att), T) :- defender(Def), attacker(Att), time(T).$ |
| $:- holdsAt(marking(Def1, Att), T), holdsAt(marking(Def2, Att), T), Def1 != Def2.$ |
| $holdsAt(manhattanDist(Def, Att, \|X1 - X2\| + \|Y1 - Y2\|), T) :- holdsAt(marking(Def, Att), T), holdsAt(player\_pos(Def, (X1, Y1)), T), holdsAt(player\_pos(Att, (X1, Y1)), T).$ |
| $\#minimize\{Dist : holdsAt(manhattanDist(Def, Att, Dist), T)\}.$ |

Next, we discuss aspects that can be taken into consideration when implementing the reasoning part of a basketball team in EduBAI, specifying the features that can be learned by the user.[4] Table 2 give examples of encodings.

*Simple, Pre-defined Tactics:* Users can implement a rich set of simple tactics by only moving their own players along the lines of a predefined plan, ignoring any information about the opponents, the probability of action success etc. Examples include moving directly towards the basket before shooting without risking any pass, moving the teammates away from the player holding the ball to avoid crowding defenders in that cell, etc.

Such tactics can easily be implemented by only relying on simple Event Calculus constructs that model the effects of actions, in addition to a powerful construct provided by ASP called *choice rule*. Choice rules are typically used in

[3] Arguments starting with a capital letter denote universally quantified variables.

[4] Fully implemented tactics can be found at https://github.com/DimitrisArr/EduBAI.

ASP programs to specify which combinations of predicates should be included in an answer set. In our case, they are used to produce the set of all possible plans of actions to be taken by the players. For instance, the first rule shown in Table 2 generates all combinations of valid *shoot* or *move toward the basket* actions to be considered as a potential next step. Appropriate state constraints can also be added, to eliminate for instance answer sets suggesting a *shoot* action when the distance is above a given threshold.

*Considering the Opponent Players' Positions:* Apparently, taking into account the position of opponent players is the next reasonable step, especially if defensive tactics are to be implemented. This can be as simple as mirroring the opponents' moves, but it can also involve the axiomatization of topological relations, such as maintaining sufficient distance to accomplish double-player blocking when a shoot is attempted. Defining proper predicates to capture such topological relations offers a very preliminary introduction to geospatial reasoning.

*Taking into Account Probabilities:* Taking advantage of the percentage of success of actions can significantly improve the intelligence of a team tactic. The modeling of uncertainty can directly be introduced in domain axioms or it can be implemented with more advanced probabilistic extensions of the Event Calculus.

*Optimizing a Cost Function:* The coupling of modeling probabilities with ASP optimization statements can produce powerful tactics. In particular, teams can easily implement a rich variety of cost functions to minimize relevant aspects, in order to help their teams decide whether to move, shoot or pass, given the state of their opponents and the success percentages. The ruleset shown in Table 2 is part of a tactic that decides which attacker each defender should cover, attempting to minimize the Manhattan distance of all defending players from the attackers.

*Considering Past/Future Actions:* The incorporation of temporal reasoning can also improve significantly the effectiveness of team plans. The easiest step is to consider future timepoints, in order to estimate whether following a given course of actions can reach a possible state that achieves high probability of success for a shoot or pass, given the current state of players. Even more challenging can be to also take into consideration the history of opponent's moves, in order to figure out their tactics and react accordingly. The explicit modeling of time, as supported by the Event Calculus, facilitates temporal reasoning in practical domains, where the time clock needs to also be taken into account.

## 3 Implementation

EduBAI consists of multiple web services that expose their functionality either via a Rest API or a web-socket connection. The core components of the platform, shown in Fig. 2, are as follows:

User Service: this is a nodeJS service that stores, manages, and serves data about users. It provides a Restful API which allows users to register, login and query the system about other users. A noSQL database is used to store data.

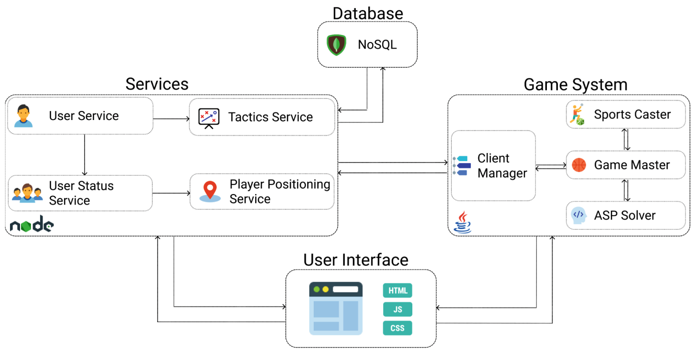


**Fig. 2.** EduBAI system architecture

User Status Service: This service is responsible for monitoring if a user is offline, online or in a game session (i.e., in a basketball match). This service also provides an invitation mechanism, where a user can start a game session by inviting another online user. Users can access the service via a Restful API alongside with a websocket for real time communication.

Tactics Service: It manages custom tactics created by users. It provides a Restful API developed using nodeJS that allows the uploading of ASP code.

Players Positioning Service: This service provides a real time connection between users, using nodeJS and webSockets, when they are in the same game session. This connection is used in order to send data about the positioning of the players on the court and for setting up the initial position of the players and ball in each new round.

Game System: This is a web service responsible for running a game session. The Game System is developed using Java and provides a websocket server where multiple users can connect. ASP code is executed using Clingo, a command line tool that is executed as a child process. After the execution of the logic program defining the tactics, the results are forwarded to the Sports Caster Server. This component implements a simple multi-thread java server that receives XML messages arriving from the Game Master and decides the outcome of the player actions based on the aforementioned probabilities (Sect. 2.1).

Graphical User Interface (GUI): The GUI is Web based and implemented using HTML, CSS and JavaScript. It is designed with respect to Human-Computer Interaction (HCI) principles. Learnability, i.e., the ease of effective interaction between a new user and the system, is achieved by following specific HCI guidelines, such as adopting a common Look 'n' Feel throughout the whole system, in order to achieve consistency among all pages. In addition, Web design guidelines are followed for predictability when positioning Web elements; e.g., the logout button is on top right of the page, which is where a user expects to find it.

## 4 Related Work and Conclusions

Probably the closest to ours platform in terms of scope and objectives is the React! tool [2]. React! is an educational environment designed to teach AI students the main formalisms and principles of cognitive robotics. It is a domain-agnostic tool that provides guidelines and explanations for modeling dynamic causal domains to solve planning problems. The user can choose among different SAT and ASP solvers to execute the reasoning tasks.

In a broader sense, a similar purpose is being served by platforms that support users in the complex task of producing syntactically correct and valid logic programs. For instance, the SeaLion system [1] is an IDE for facilitating the task of ASP encoding.

Such tools though do not employ some scoring mechanism or adopt a competitive style of interaction to motivate the participation of users in the learning process. Such a practice is typically found in systems following the Games With A Purpose (GWAP) paradigm, which has become a rather popular approach. There are many relevant systems, as for instance the one presented in [7], which relies on the Event Calculus, similarly to EduBAI. It implements a game where players take the role of a teacher to train a robot to answer simple questions.

Overall, EduBAI is a platform that intends to help users improve their logic programming skills through an entertaining interactive environment. Relying on the set of predefined tactics as guidelines, the users can apply their own creative thinking, in order to devise complex and effective tactics to win human or computer opponents. Part of our future plans involve extending the platform with friendly interactive features that will enhance the educational character, as for example an embedded ASP editor to enable users to encode their tactics or a tutorial-style assistant to walk new user through the main functionalities.

## References

1. Busoniu, P.A., Oetsch, J., Puhrer, J., Skocovsky, P., Tompits, H.: SeaLion: an eclipse-based IDE for answer-set programming with advanced debugging support. Theor. Pract. Logic Programm. **13**(4–5), 657–673 (2013)
2. Dogmus, Z., Erdem, E., Patoglu, V.: ReAct!: an interactive educational tool for AI planning for robotics. IEEE Trans. Educ. **58**(1), 15–24 (2015)
3. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings 5th International Joint Conference and Symposium on Logic Programming, IJCSLP 1988, pp. 1070–1080. MIT Press (1988)
4. Kowalski, R., Sergot, M.: A logic-based calculus of events. New Gener. Comput. **4**, 67–95 (1986)
5. Marek, V.W., Truszczynski, M.: Stable models and an alternative logic programming paradigm. In: Apt, K.R., Marek, V.W., Truszczynski, M., Warren, D.S. (eds.) The Logic Programming Paradigm: A 25-Year Perspective, pp. 375–398. Springer, Heidelberg (1999). https://doi.org/10.1007/978-3-642-60085-2_17

6. Miller, R., Shanahan, M.: Some alternative formulations of the event calculus. In: Kakas, A.C., Sadri, F. (eds.) Computational Logic: Logic Programming and Beyond. LNCS (LNAI), vol. 2408, pp. 452–490. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45632-5_17
7. Rodosthenous, C., Michael, L.: A hybrid approach to commonsense knowledge acquisition. In: Proceedings of the 8th European Starting AI Researcher Symposium (STAIRS 2016), pp. 111–122. IOS Press (2016)