Argumentation Accelerated Reinforcement Learning for RoboCup Keepaway-Takeaway

Yang Gao, Francesca Toni

Department of Computing, Imperial College London

Abstract. Multi-Agent Learning (MAL) is a complex problem, especially in real-time systems where both cooperative and competitive learning are involved. We study this problem in the RoboCup Soccer Keepaway-Takeaway game and propose Argumentation Accelerated Reinforcement Learning (AARL) for this game. AARL incorporates heuristics, represented by arguments in Value-Based Argumentation, into Reinforcement Learning (RL) by using Heuristically Accelerated RL techniques. We empirically study for a specific setting of the Keepaway-Takeaway game the suitability of AARL, in comparison with standard RL and hand-coded strategies, to meet the challenges of MAL.

1 Introduction

Multi-agent Learning (MAL) is widely recognised as a complex problem and has attracted much attention. Research on MAL roughly fall into two categories: *cooperative MAL*, where multiple learning agents share the same goal (e.g. [5, 12, 11, 15]), and *competitive MAL*, where different learning agents have different or even opposite goals (e.g. [16, 13]). Argumentation [7], studying the concept of 'good' arguments among conflicting arguments, is widely viewed as a powerful tool in solving conflicts and reaching agreement (see, e.g., [8]), and has been successfully incorporated within learning [18, 10]. We investigate the use of argumentation in MAL where both cooperative and competitive learning are involved, focusing on the RoboCup Soccer Keepaway-Takeaway (*KATA*) game, an integration of two popular testbeds for MAL [21, 14] where there are two competing teams of agents, keepers and takers, collaborating within the teams.

We focus on *Reinforcement Learning* (RL), because it allows agents to learn by interacting with the environment and has been shown to be a generic and robust learning algorithm for MAL [19]. However, when both competitive and cooperative learning are involved in a MAL, the effectiveness of RL could be seriously reduced due to the instability of the environment [22]. To solve this problem, we propose, in the context of KATA games, *Argumentation Accelerated RL* (AARL), which incorporates Value-Based Argumentation [1] into RL by using *Heuristically Accelerated RL* (HARL) techniques [3], so that, when making decisions, agents rely not only on their interacting experiences with the environment, but also domain knowledge in the form of arguments. Further, we test the effectiveness of AARL in the specific setting of 3-keeper-2-taker KATA games.

Concretely, we test AARL for keepers and takers against two different strategies for each type of agent and perform a round-robin style experiment(where each strategy meets all strategies in turn). Our experiments suggest that the AARL-based strategies are competitive in terms of stability, average convergence time and average optimal performance. This work is an extension of our previous work on single-agent Argumentation-Based Reinforcement Learning (ABRL) [10].

The paper is organised as follows: Section 2 gives background. Section 3 describes how to apply AARL to TAKA games and Section 4 presents empirical results. Section 5 describes related works and Section 6 concludes.

2 Background

First we give fundamentals of value-based argumentation. Then we describe *Markov Decision Process*-RL, focusing on the SARSA(λ) algorithm that we use, followed by an introduction of HARL, by means of which we integrate arguments into RL. Finally, we describe the RoboCup Soccer Keepaway-Takeaway games.

2.1 Argumentation Frameworks

An *abstract argumentation framework* (AF) [7] is a pair $(\mathit{Arg}, \mathit{Att})$ where Arg is a set of *arguments* and $\mathit{Att} \subseteq \mathit{Arg} \times \mathit{Arg}$ is a binary relation ($(A, B) \in \mathit{Att}$ is read '*A attacks B*'). $S \subseteq \mathit{Arg}$ *attacks* $B \in \mathit{Arg}$ iff some member of S attacks B. $S \subseteq \mathit{Arg}$ is *conflict-free* iff S attacks none of its members. If $S \subseteq \mathit{Arg}$ attacks all arguments attacking $B \in \mathit{Arg}$, then S *defends* B . Semantics of AFs are defined as sets of "rationally acceptable" arguments (*extensions*), e.g. (given some $\mathsf{F} = (\mathit{Arg}, \mathit{Att})$ and $S \subseteq \mathit{Arg}$):

– S is a *complete extension* for F iff S is conflict-free and $S = \{a | S \text{ defends } a\}$;
– S is the *grounded extension* for F iff S is minimally (wrt $\subseteq$) complete for F.

The grounded extension is guaranteed to be unique, consisting solely of uncontroversial arguments and being thus "sceptical".

In some contexts, the attack relation is not enough to decide what is rationally acceptable, and the "values" promoted by arguments must be considered. *Value-based argumentation frameworks* (VAFs) [1] incorporate values as well as preferences over them into AFs. The key idea is to allow for attacks to succeed or fail, depending on the relative worth of the values promoted by the competing arguments. Given a set V of values, an *audience* *Valpref* is a strict partial order over V (corresponding to the preferences of an agent), and an *audience-specific VAF* is a tuple $(\mathit{Arg}, \mathit{Att}, V, \mathit{val}, \mathit{Valpref})$, where $(\mathit{Arg}, \mathit{Att})$ is an AF and $\mathit{val} : \mathit{Arg} \to V$ gives the values promoted by arguments. *Valpref*, the audience, is a strict partial order over V. We denote $(X, Y) \in \mathit{Valpref}$ by $X >_v Y$.

In VAF, *Valpref* is taken into account in the definition of extensions. The *simplification* of an audience-specific VAF is the AF $(\mathit{Arg}, \mathit{Def})$, where $(A, B) \in \mathit{Def}$ iff $(A, B) \in \mathit{Att}$ and $\mathit{val}(B) \not>_v \mathit{val}(A)$. $(A, B) \in \mathit{Def}$ is read '*A defeats B*'. Then, (acceptable) extensions of a VAF are defined as (acceptable) extensions of its simplification (*Arg*, *Def*). We refer to (*Arg*, *Def*) as the *simplified AF derived from* (*Arg*, *Att*, *V*, *val*, *Valpref*).

2.2 Markov Decision Process

The Markov Decision Process (MDP) is one of the most widely used model for RL and has several variants [22]. An MDP is a tuple (S, A, T, R), where S is the *state space*, A is the *action space*, $T(s, a, s') = Pr(s'|s, a)$ is the *transition probability* of moving from state s to state s' by executing action a, and $R(s, a, s')$ gives the immediate *reward* received when action a is taken in state s, moving to state s'. In many real problems, e.g. RoboCup Keepaway/Takeaway games (see Section 2.4), actions may take variable amount of time. In these cases, Semi-MDP [4] are used to model temporally-extended courses of actions. We use the SMDP version of SARSA(λ) [22] learning algorithm extended, in order to improve the learning speed, with *replacing eligibility traces* [20], outlined as Algorithm 1 below.

Algorithm 1 SARSA(λ) with replacing eligibility traces (adjusted from [22])

```
Initialise $Q(s, a)$ arbitrarily for all states $s$ and actions $a$
Repeat (for each episode):
  Initialise $e(s, a) = 0$ for all $s$ and $a$
  Initialise current state $s_t$
  Choose action $a_t$ from $s_t$ using the $\varepsilon$-greedy policy
  Repeat until $s_t$ is the terminal state:
    Execute action $a_t$, observe reward $r_t$ and new state $s_{t+1}$
    Choose $a_{t+1}$ from $s_{t+1}$ using the $\varepsilon$-greedy policy
    $\delta \leftarrow r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$
    $e(s_t, a_t) \leftarrow 1$
    For all $s$, $a$:
        $Q(s, a) \leftarrow Q(s, a) + \alpha\delta e(s, a)$
        $e(s, a) \leftarrow \gamma\lambda e(s, a)$
    $s_t \leftarrow s_{t+1}$; $a_t \leftarrow a_{t+1}$
```

In this algorithm, $Q(s, a) \in \mathbb{R}$ represents the value of performing action a in state s. α is the *learning rate*, γ is the *discount factor* governing the weight placed on the future rewards, e represents eligibility traces, which store the credit that previous action choices should receive for current rewards, while λ governs how much credit is delivered back to them. *ε-greedy* is a widely used (action-selection) policy, which selects the action with highest $Q(s, a)$ value for a proportion $1 - \varepsilon$ of the trials;for the other ε proportion, actions will be selected randomly. Formally, this policy is defined as:

$$\pi(s_t) = \begin{cases} \arg\max_{a_t} Q(s_t, a_t) & \text{if } q \leq \varepsilon \\ a_{random} & \text{otherwise} \end{cases} \tag{1}$$

where q is a random value uniformly distributed over $[0, 1]$. a_{random} is an action randomly chosen among all those available in state s_t.

2.3 Heuristically Accelerated RL

HARL [3] is a way to solve a MDP problem by explicitly incorporating heuristics within RL. By using HARL, a learning agent's choice of actions is influenced so that more promising actions are more likely to be performed. HARL influences a RL process by overriding the action-selection policy. For example, if we are using the ε-greedy policy (see Equation 1) in the original RL, by using HARL, the policy will be changed as:

$$\pi^H(s_t) = \begin{cases} \arg\max_{a_t}[Q(s_t, a_t) + H_t(s_t, a_t)] & \text{if } q \leq \varepsilon \\ a_{random} & \text{otherwise} \end{cases} \tag{2}$$

where $H_t(s_t, a_t)$ is the *heuristic function* which is defined by the domain expert. For a state-action pair (s_t, a_t), the higher the value of $H_t(s_t, a_t)$, the more promising performing action a_t in state s_t. As for q and ε, they have the same meaning as in Equation 1. Note that HARL only provides the more promising state-action pairs with higher priority to be explored, but does not change the convergence of the original RL algorithm [2]. The heuristic function H_t can be defined a priori or at any moment during learning, and can be updated at any time throughout learning. Later in Section 3.3, we will give the argumentation-based definition of HARL.

2.4 RoboCup Soccer Games

RoboCup Soccer is an international project which aims at providing an experimental framework in which various technologies can be integrated and evaluated[1]. In order to facilitate RL research in this application, two simplified tasks have been developed: the *Keepaway* game [21], and the *Takeaway* game [14]. The basic settings of these games are the same: $N + 1$ ($N \in \mathbb{N}$, $N \geq 1$) *keepers* are competing with N *takers* on a fixed-size court. Keepers are trying to keep possession of the ball within their team for longer time, whereas takers are trying to win possession. The games consist of a series of *episodes*: at the start of each episode, the keeper in the top-left corner holds the ball, while all other keepers are on the right. All takers are initially in the bottom-left corner. An episode ends when the ball goes off the court or any taker gets the ball, and a new episode starts immediately with all the players reset.

In Keepaway, only the keeper holding the ball is learning. All the other keepers and takers are playing in accordance with hand-coded strategies. In Takeaway, however, all takers are learning independently while all keepers are playing in accordance with hand-coded strategies. So Takeaway is a cooperative MAL problem whereas Keepaway is a single-agent learning problem which takes place in a multi-agent scenario. In this paper, we endow both keeper and takers with learning ability, and we call a game with $N + 1$ learning keepers and N learning takers a N-*Keepaway-Takeaway* (N-KATA) game.

[1] See `http://www.robocup.org/` for more information.

In the RoboCup simulation platform, only primitive actions and coordinate positions are available. However, RL cannot *effectively* use this low-level information in Keepaway [21] or Takeaway [14]. So *macro actions* were proposed originally by [21] for Keepaway, and then adjusted by [14] for Takeaway. In particular, there are 2 macro actions for Keepaway:

HoldBall(): stay still while keeping the ball;

PassBall(i): kick the ball towards keeper K_i;

and 2 macro actions for Takeaway:

TackleBall(): move towards the ball to tackle it

MarkKeeper(i): go to mark keeper K_i, $i \neq 1$

where K_i represents the ith closest keeper to the ball - so that K_1 is the keeper in possession of the ball. Takers are indexed in the same way. When a taker marks a keeper, the taker blocks the path between the ball and that keeper. Thus, a taker is not allowed to mark the ball holder, and the action set in N-Takeaway consists of $N + 1$ actions. In addition, *state variables* are proposed by [21] to facilitate the state representation in Keepaway games. In particular, a state is represented by a *state vector* which consists of elements, known as *state variables*, that can be directly used in the agent's decision making. The state variables for the Keepaway games are shown in Table 1. For example, the distances between takers and the ball holder are state variables, because the holder could use this information to decide when to pass the ball and where to pass the ball. As we can see, all state variables are collected in the perspective of the ball holder, because the ball holder is the only learner in Keepaway. We call these state variables *holder-oriented*.

Most existing research on Takeaway uses the holder-oriented state variables (e.g. [14, 17, 6]). However, a taker's *self-oriented* state variables would be more helpful. Also, since multiple takers are learning independently in Takeaway, the state variables should also facilitate coordination between takers. We combine taker's *self-oriented* and some holder-oriented state variables, and use the new state variables in Table 2. Later in Section 5 we will show that compared with the learning takers that use the holder-oriented state variables, the takers using our new state variables have significantly better performance.

3 Argumentation for RoboCup Soccer

In Section 3.1 we give arguments and values for keepers and takers. Then, in Section 3.2, we define the defeat relationship among arguments, by taking the ranking of values into account. As a result, we instantiate VAFs (seen in Section 2.1) for keepers and takers. Acceptable arguments (in the grounded extenstion) for these VAFs recommend actions. Finally, in Section 3.3, we integrate this action recommendation into RL by using HARL techniques.

3.1 Arguments and Values

Arguments are of the form:

State Variable(s)	Description
$dist(K_i, C)$, $i \in [1, N+1]$	Distance between keepers and the centre of the court.
$dist(T_j, C)$, $j \in [1, N]$	Distance between takers and the centre of the court.
$dist(K_1, K_i)$, $i \in [2, N+1]$	Distance between K_1 and the other keepers.
$dist(K_1, T_j)$, $j \in [1, N]$	Distance between K_1 and the takers.
$\min_{j \in [1,N]} dist(K_i, T_j)$, $i \in [2, N+1]$	Distance between K_i and its closest taker.
$\min_{j \in [1,N]} ang(K_i, T_j)$, $i \in [2, N+1]$	The smallest angle between K_i and the takers with vertex at K_1.

Table 1. State variables for learning keeper K_1 in a N-KATA game. ($i, j \in \mathbb{N}$).

$$con(A) \ \text{IF} \ pre(A)$$

where $con(A)$ (the *conclusion* of A) is the recommended action and $pre(A)$ (the *premise* of A) describes under which conditions argument A is applicable.

Arguments and values for keepers. For the learning keeper, we use the same arguments as described in [10], which are designed based on a successful hand-coded strategy for the keeper described in [21]:

- **HD**: **HoldBall()** IF $\min_{1 \le j \le N} dist(K_1, T_j) \ge 7$
- **F**(i): **PassBall**(i) IF $\min_{1 \le j \le N} dist(K_i, T_j) \ge 15$
- **O**(i): **PassBall**(i) IF $\min_{1 \le j \le N} ang(K_i, T_j) \ge 15$

where $i, j \in \mathbb{N}$. We say that these arguments *belong to* the keeper K_1. Note that the thresholds used above, i.e. 7 and 15, are proposed based on empirical results or thresholds used in the hand-coded strategy. Overall, there are $2N + 1$ candidate arguments for K_1 [2]. These arguments can be interpreted as:

- **HD**: hold the ball because all takers are "far": the distance between each taker and K_1 is larger than 7;
- **F**(i): pass the ball to K_i because K_i is "far": the distance between K_i and the K_1 is larger than 15;
- **O**(i): pass the ball to K_i because K_i is "open": the angles between K_i and all the takers, with vertex at K_1, are over 15°.

The arguments are promoting values:

- **RM**: reduce the risk of teammates being marked;
- **RI**: reduce the risk of the ball being intercepted;
- **RT**: reduce the risk of the ball being tackled;

[2] **HD** generates one argument. **F**(i) and **O**(i) generate N arguments each.

State Variable(s)	**Description**
$dist(K_i, Me)$, $i \in [1, N+1]$	Distance between keepers and myself.
$dist(T_j, Me)$, $j \in [2, N]$	Distance between other takers and myself.
$ang(K_i, Me)$, $i \in [2, N+1]$	The angle between the free keepers and myself, with vertex at K_1.
$dist(K_i, K_1)$, $i \in [2, N+1]$	Distance between K_1 and the other keepers.
$dist(T_j, K_1)$, $j \in [2, N]$	Distance between K_1 and the other takers.
$\min\limits_{j \in [1,N]} ang(K_i, T_j)$, $i \in [2, N+1]$	The smallest angle between K_i and the takers with vertex at K_1.

Table 2. State variables for learning taker T_1 in a N-KATA game. State variables of other takers can be obtained similarly. ($i, j \in \mathbb{N}$). The top 3 rows describe self-oriented variables, and the others describe variables about the keepers relative layout.

where *val*(**HD**) = **RM**, *val*($\mathbf{F}(i)$) = **RT** and *val*($\mathbf{O}(i)$) = **RI** with **RM** $>_v$ **RI** $>_v$ **RT**. Note that in standard Keepaway, takers are always trying to tackle the ball. All arguments and values described above are designed based on this assumption. However, in KATA games, takers can not only tackle the ball, but also mark keepers. In other words, these arguments and values inevitably have errors when applied to KATA games. In Section 4, we will make a deeper analysis of the effects of this imperfect domain knowledge on the learning performance.

Arguments and Values for Takers. As for takers, the arguments should not only instruct takers to compete with keepers, but also coordinate takers. We propose the following categories of candidate arguments that *belong to* the taker T_j:

- T_j**TK**: **TackleBall()** IF $j = \arg\min\limits_{1 \le t \le N} dist(K_1, T_t)$
- $T_j\mathbf{O}(i)$: **MarkKeeper(i)** IF $\min\limits_{1 \le t \le N} ang(K_i, T_t) \geq 15$
- $T_j\mathbf{F}(i)$: **MarkKeeper(i)** IF $\min\limits_{1 \le t \le N} dist(K_i, T_t) \geq 15$
- $T_j\mathbf{A}(i)$: **MarkKeeper(i)** IF $j = \arg\min\limits_{1 \le t \le N} ang(K_i, T_t)$
- $T_j\mathbf{C}(i)$: **MarkKeeper(i)** IF $j = \arg\min\limits_{1 \le t \le N} dist(K_i, T_t)$

where $i, j, t \in \mathbb{N}$. For $T_j\mathbf{O}(i), T_j\mathbf{F}(i)$, $T_j\mathbf{A}(i), T_j\mathbf{C}(i)$, $i \in \{2, \cdots, N+1\}$, because K_1 cannot be marked. The intuition behind these arguments is as follows:

- T_j**TK**: T_j should tackle the ball if T_j is the closest to the ball holder (K_1) among all the takers;
- $T_j\mathbf{O}(i)$: T_j should mark keeper K_i if K_i is quite "open": the angles between K_i and all the takers, with vertex at K_1, are over 15°;
- $T_j\mathbf{F}(i)$: T_j should mark keeper K_i if K_i is "far": its distances to all takers are larger than 15;

- $T_j\mathbf{A}(i)$: T_j should mark keeper K_i if the angle between T_j and K_i, with vertex at K_1, is the smallest;
- $T_j\mathbf{C}(i)$: T_j should mark keeper K_i if T_j is closest to K_i.

Overall, in a N-KATA game, there are $4N^2+N$ arguments for takers[3]. These arguments are promoting values:

- **VT**: The ball should be tackled as quickly as possible;
- **VO**: If the ball holder decides to pass, it is very likely to pass the ball to an "open" keeper;
- **VF**: If the ball holder decides to pass, it is very likely to pass to a keeper far from all takers;
- **VA**: The taker with the smallest angle to a keeper is most likely to intercept the ball passed to that keeper;
- **VC**: The taker closest to a keeper can mark it most quickly.

We set $\mathit{val}(T_j\mathbf{TK})=\mathbf{VT}$, $\mathit{val}(T_j\mathbf{O}(i))=\mathbf{VO}$, $\mathit{val}(T_j\mathbf{F}(i))=\mathbf{VF}$, $\mathit{val}(T_j\mathbf{A}(i)) = \mathbf{VA}$, $\mathit{val}(T_j\mathbf{C}(i))=\mathbf{VC}$. Further, we set $\mathbf{VT}>_v \mathbf{VA} =_v \mathbf{VC} >_v \mathbf{VO} >_v \mathbf{VF}$[4]. Note that, for simplicity, we assume the same ranking of values throughout the game, but our technique can be applied with value rankings that change over time.

Applicable arguments. The arguments given so far are candidate arguments that may not be applicable at all times. Indeed, in KATA games, the environment is constantly changing and in each state, an agent has to select the applicable arguments by checking all candidate arguments to see whether their premises hold true in that state. Since takers need to coordinate, we assume that each taker is aware of all other takers' applicable arguments.[5]

For example, consider the scenario in Figure 1. With respect to the learning keeper, since the distances between all takers and K_1 are larger than 7, the argument **HD** is applicable. Also, because the distance between K_3 and K_1 is larger than 15, $\mathbf{F}(3)$ is applicable. The premises of other candidate arguments are not satisfied in this scenario, so they are not applicable. Similarly, we get the applicable arguments for takers: $T_1\mathbf{TK}$, $T_1\mathbf{A}(2)$, $T_1\mathbf{A}(3)$, $T_1\mathbf{C}(2)$, $T_2\mathbf{C}(3)$.

3.2 Defeat Relation and Simplified AFs

For any two arguments P and Q, $\mathit{val}(P) = V_1$ and $\mathit{val}(Q) = V_2$, P *defeats* Q iff $V_1 >_v V_2$ and one of the following two conditions holds:

- P and Q belong to different agents but recommend the same action (i.e. $\mathit{con}(P) = \mathit{con}(Q)$);

[3] Indeed, for taker T_j, $T_j\mathbf{TK}$ gives 1 argument and the other four categories of arguments each give N arguments.

[4] $V_1 =_v V_2$ stands for $(V_1 >_v V_2) \wedge (V_2 >_v V_1)$

[5] This is in line with all existing research on Keepaway/Takeaway games, building upon the assumption that an agent is aware of all agents' locations and the ball's location, i.e. each agent has a *perfect world view*.

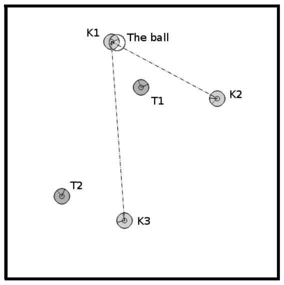


Fig. 1. A scenario of 2-KATA. The size of the court is 40×40.

– P and Q belong to the same agent but recommend different actions (i.e. $con(P) \neq con(Q)$).

Given the applicable arguments and the defeat relation, we obtain simplified AFs (see Section 2.1) for keepers and takers. For example, consider again the scenario in Figure 1. For the keeper, **HD** and **F**(3) belong to the same agent but support different actions, and the value promoted by **HD**: **RM**, is more preferred than the value promoted by **F**(3): **RT**, so **HD** defeats **F**(3). The simplified AF for keeper and takers are shown in Figure 2(a) and Figure 2(b), respectively.

3.3 Argumentation Accelerated RL (AARL)

We use the grounded extension (see Section 2.1) to select the *recommended action* for each agent, because this extension is always unique and, as a result, will not recommend different actions to an agent. For example, consider again the scenario in Figure 1. The grounded extension of the keeper's argumentation framework is {**HD**}, so the recommended action for K_1 is **HoldBall()**. The grounded extension of the takers' argumentation framework is $\{T_1\mathbf{TK}\}$, so T_1 is recommended to **TackleBall()**. Note that the grounded extension of takers does not include arguments belonging to T_2. This means that given the current state and our domain knowledge no action is recommended to T_2. Note that in some scenarios, the grounded extension can be empty, which means that based on the current domain knowledge, there is no convincing enough recommendation can be drawn in this scenario. So additional domain knowledge should be added; otherwise, no actions are recommended and agents will choose actions solely based on the values of each state-action pairs (Q-values, see Section 2.2)

AARL amounts to integrating these recommended actions into RL by using HARL (see Section 2.3) to give these actions higher probabilities to be explored. Because all the arguments and values are designed based on the domain knowledge and are not updated during learning, we define the heuristic function a priori and keep it fixed throughout the learning. In particular, we set the heuristic function of agent A_i as:

$$H(s,a)=\begin{cases} \eta & \text{if } a \text{ is recommended to } A_i \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{tikzcd}
 & & & T_1\mathbf{A}(2) \arrow[d, leftrightarrow] & T_1\mathbf{C}(2) \arrow[dl, leftrightarrow] \\
\mathbf{HD} \arrow[r] & \mathbf{F}(3) & T_2\mathbf{C}(3) \arrow[r, leftrightarrow] & T_1\mathbf{A}(3) & T_1\mathbf{TK} \arrow[l] \arrow[u] \arrow[ul]
\end{tikzcd}$$

(a) (b)

Fig. 2. Simplified AFs for Figure 1: for keeper(2(a)) and takers (2(b)).

where an action a is recommended to A_i iff a is recommended by an argument in the grounded extension of A_i's simplified AF. Because all Q-values are initialised as 0, the heuristic value for the recommended action is the value of η. If an agent does not have any recommended actions, then it uses the standard ε-greedy policy (see Section 2.2). Note that the heuristic function, states and actions have no time index, because they can be applied to any state-action pair.

4 Empirical Results

Our learning algorithm is shown in Algorithm 1) We use the same setting as in [21] for SARSA(λ) and we set $\eta = 2$. For the learning keeper, we use the same rewarding scheme as in [21]:

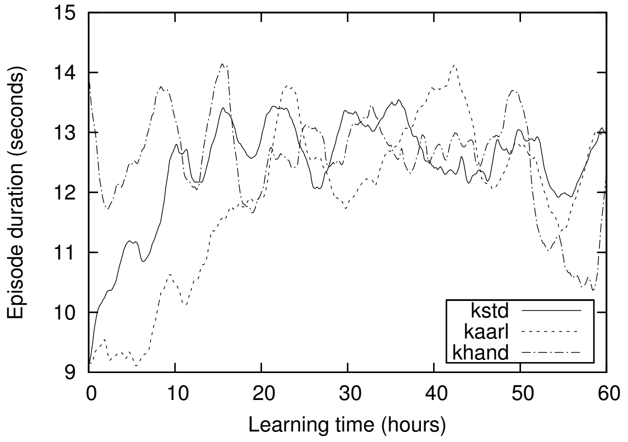
$$r = \mathit{CurrentTime} - \mathit{LastActionTime}$$

where r is the reward, *CurrentTime* is the time when a keeper holds the ball or an episode ends, and *LastActionTime* is the time when a keeper selected the last action. As a result, if the last action was **HoldBall()**, the reward r must be equal to the duration of an episode; if the last action was **PassBall()**, then the farther the target keeper is, the larger the reward will be. So, roughly speaking, this reward system is *distance-oriented*: passing the ball to farther keepers is more encouraged. For takers, the reward is 10 for the last cycle[6] of each episode and -1 for all the other cycles. In order to prevent possible oscillations of the strategy, a taker updates its policy and makes new decisions every 5 cycles (called a *trail*). We conduct one experiment on each combination of strategies. All the experiments are done in RoboCup Soccer Simulator v15.1.0[7]. The hand-coded strategies of keepers are described in [21] (see Section 3.1), and we design a hand-coded strategy for the takers, s.t. takers who have a recommended action would perform it; otherwise, they will tackle the ball.

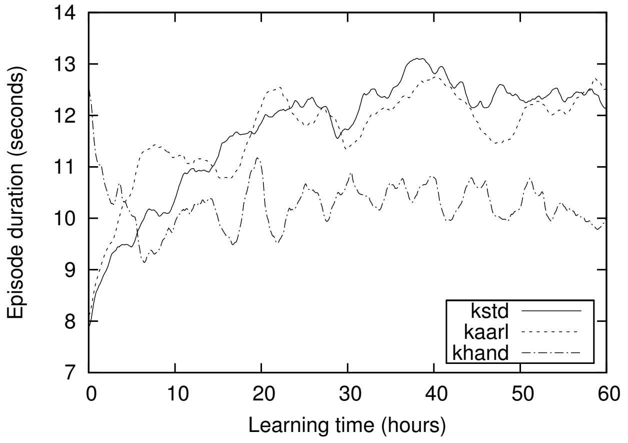
The performances of different combinations of keeper's and takers' strategies are shown in Figure 3. Both keepers and takers have 3 strategies, namely the SARSA(λ)-based strategy, the AARL-based strategy and hand-coded strategy. The SARSA(λ)-based strategy can be viewed as the most "random" strategy, because it uses the standard ε-greedy action selection policy and randomly searches

[6] In the RoboCup Soccer Simulator, each second is divided into 20 equal-length time slots, called *cycles*.

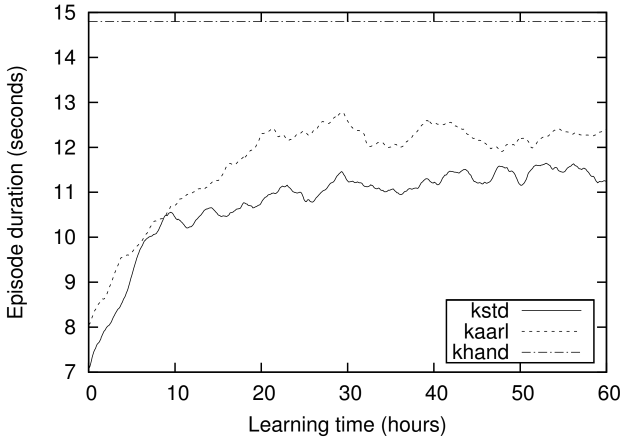
[7] http://sourceforge.net/projects/sserver/



(a) Keepers vs. SARSA(λ) learning takers



(b) Keepers vs. AARL learning takers



(c) Keepers vs. hand-coded takers

Fig. 3. Performances of different learning algorithms for 2-KATA games on a 40×40 court. Each curve represents the performance of a single test. Note that when both keepers and takers are using the hand-coded strategies, the performance is stable and, as a result, we present its average value here (the straight line in 3(c)).

the action space in the early learning stage. Thus, its performance is most unpredictable. On the other hand, the hand-coded strategy can be viewed as a fully argumentation-instructed strategy, because all agents' actions are strictly constrained by the results of the simplified AFs. Hence, the performance of the hand-coded strategy is most predictable and stable. The AARL-based strategy can be viewed as half-random-half-argumentation-instructed, with the arguments used the same as for the hand-coded strategy. Hence, there are 9 combinations overall. We evaluate the performance of each combination in 3 aspects:

initial performance (IP): episodes' average duration in the first hour of learning;
convergence time (CT): how long time does a strategy need to converge;
optimal performance (OP): optimal performance of a strategy.

The performance of each combination, in terms of these 3 properties, is shown in Table 3. Note that for a keeper's strategy, the higher IP and OP, the more successful the strategy. However, for takers, a successful strategy should have lower IP and OP. For both keeper's and takers' strategy, the shorter CT, the better the strategy. We suggest the following conjectures:

1. When a single learner is competing with a group of learning agents, it is better for the single learner to use the random strategy, but for the learning group to use the argumentation-instructed strategy. This is because the group of learners are learning independently, so their emergent behaviours can be hardly predicted and, as a result, any domain knowledge for the single learner would be helpless. Instead, if the single learner is using heuristics, its behaviour is easier to predict. On the other hand, the heuristics would help the group of learning agents to predict the single learning opponent's behaviour more quickly.

2. The AARL-based strategy has the best overall performance, in terms of the stability, average convergence speed and average optimal performance. This is because AARL can be viewed as a tradeoff between the SARSA-based strategy and the hand-coded strategy, so it has the advantages of both.

3. The AARL-based strategy is robust to errors in arguments. We can see that when both sides are using hand-coded strategies, the average episode duration is very high, which means that the keeper's hand-coded strategy is better than the takers'. However, with respect to takers, the performance of the AARL-based strategy is always better than the SARSA-based strategy.

4. When a group of randomly learning agents are competing with a hand-coded opponent, the convergence speed can be very slow, even slower than when the opponent is using a learning strategy. The reason could be that when using the AARL-based or the hand-coded strategy, the learning group have some coordination schemes so their group behaviours can converge faster; when both sides are using learning strategies, they are pushing each other to achieve a Nash Equilibrium[16] [8], so the convergence time can by quicker.

[8] The KATA game can be viewed as a zero-sum game because the goal of the two sides are opposite. However, since our application and algorithm is very different

		Keeper		
		SARSA	AARL	Hand-coded
Takers	SARSA	9.2, 15, 12.7	9.2, 22, 12.7	13.9, > 60, unknown
	AARL	7.9, 20, 12.3	8.1, 22, 12.0	12.3, 10, 10.2
	Hand-coded	7.1, 30, 11.1	8.1, 21, 12.1	14.8, 0, 14.8

Table 3. Summary of performances. Each entry consists of three numbers (in seconds): initial performance, convergence time and optimal performance.

5 Related Works

There is research on improving machine learning by argumentation. Mozina et al. [18] proposed argumentation based machine learning, which combines arguments with the original examples of CN2 algorithm to form argumented examples. The use of arguments significantly improves the performance of CN2. However, the relationships between different arguments are not taken into account in their technique, which restricts the effect argumentation should have. Also, the machine learning technique they considered, CN2, is supervised and fundamentally different from RL. Research has also been devoted to incorporating domain knowledge into RL to improve its performance in Keepaway games. For example, [6] used potential-based reward shaping in Takeaway games and showed that the convergence time can be reduced and group performance can be altered. However, their work does not explicitly consider the domain knowledge representation. Moreover, under the same game settings, their average episode durations are almost twice as long as ours.

With respect to cooperative RL, [5] distinguished and compared two forms of multi-agent RL: *independent learners* (ILs), who only consider its own Q-values when choosing actions, and *joint action learners*, who search the exponential joint action space to maximise the sum of all agents' Q-values. However, the performance of these two learners are almost the same. Our agents can be seen as ILs. [12] used *coordination graph* to restrain the coordination relationships between actions. Actions are selected to maximise the sum of Q-values of only related agents. So in order to know the Q-values of all related teammates, each agent has to compute all these Q-values or get them by communication. This technique is not suitable for real-time applications where computational time is strictly constrained and communication is forbidden, e.g. Takeaway. Some have also explored using Hierarchical RL (HRL) to guide coordination. For example, [11] proposed *Cooperative HRL*, in which coordination is only learnt in predefined *cooperative subtasks* (CSs), defined by domain experts as subtasks where coordination would significantly improve the performance of the whole team. [15] modelled the coordination among agents as *coordination constraints* and used these to limit the joint action space for exploration. In all these HRL approaches, domain knowledge is in the form of hard constraints and the action exploration is strictly constrained by them. Hence, the learning process cannot

from those in [16], we cannot guarantee a Nash Equilibrium can be achieved. The difference between our research and [16] are discussed in Section 5.

correct errors contained in the domain knowledge and the performances of these techniques, as a result, are highly sensitive to the quality of the domain knowledge. Note that there are also research about using argumentation to coordinate cooperative agents [9, 23]. However, their agents do not learn.

For competitive RL, [16] proposed the *minimax-Q-Learning* algorithm for two-player zero-sum Markov game. Based on Littman's work, [13] developed a more general algorithm for n-player general-sum Markov games. Both these approaches are guaranteed to converge to a Nash equilibrium under certain conditions. However, in the Keepaway/Takeaway games, keepers and takers are making decisions asynchronously, i.e. the keeper is making a decision at each time slot whereas takers are making decisions every 5 time slots (see Section 4), and the actions of opponent(s) are difficult or even impossible to identify. For these reasons, the payoff matrices, which are the bases of these approaches, can hardly be built in Keepaway/Takeaway. Another fact worth mentioning is that the application domains of all these cooperative/competitive RL techniques above are simple problems, such as matrix games or 'grid world' where there are finite number of discrete states. However, KATA games are real-time large-scale problems which take place in continuous space, and both cooperative learning and competitive learning are involved. Thus, the application domain we are using is more realistic and complex than most existing research.

6 Conclusions

We presented *Argumentation-Accelerated RL* (AARL) for the 2-KATA game. This is a new approach to RL where domain knowledge is represented and organised as an argumentation framework. We implement AARL using the SARSA(λ) algorithm and conduct experiments in 2-KATA games. The results of our experiments suggest that AARL is competitive with respect to stability, average convergence time and average optimal performance. Further experiments are needed to consolidate our conclusions.

This work is preliminary research on using arguments to solve multi-agent cooperative-competitive learning. Since the arguments we are using (see Section 3) are independent of any specific learning algorithm, we believe that our approach can in principle be integrated within other learning algorithms (not limited to RL) or within RL via other techniques (not limited to HARL). However, as we have mentioned in Section 3.1, the arguments we are using contain obvious faults and have a huge space for improvement. So future work can be done to try out our methodology with other learning methods and more sophisticated arguments. In addition, since the conclusions are based on one specific game and limited experiments, more experiments on more games should be performed so as to test our conclusions more generally.

References

1. Bench-Capon, T.: Persuasion in practical argument using value-based argumentation frameworks. J. Log. Comput. 13(3), 429–448 (2003)

2. Bianchi, R.A.: Using heuristics to accelerate reinforcement learning algorithms (in Portuguese). Ph.D. thesis, University of São Paulo (2004)
3. Bianchi, R.A., Ribeiro, C.H., Costa, A.H.: Accelerated autonumous learning by using heuristic selection of actions. Journal of Heuristics 14, 135–168 (2008)
4. Bradtke, S.J., Duff, M.O.: Reinforcement learning methods for continuous-time markov decision problems. Advances in Neural Information Processing Systems 7, 393–400 (1995)
5. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: The Proc. of AAAI (1998)
6. Devlin, S., Grzes, M., Kudenko, D.: Multi-agent, reward shaping for robocup keepaway (extended abstract). In: Proc. AAMAS (2011)
7. Dung, P.M.: On the acceptability of arugments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial Intelligence 77(2), 321–357 (1995)
8. Fan, X., Toni, F.: Argumentation dialogues for two-agent conflict resolution. In: Proc. of COMMA (2012)
9. Ferretti, E., Errecalde, M., García, A.J., Simari, G.R.: An application of defeasible logic programming to decision making in a robotic environment. In: LPNMR (2007)
10. Gao, Y., Toni, F., Craven, R.: Argumentation-based reinforcement learning for robocup soccer keepaway. In: Proc. of ECAI (2012)
11. Ghavamzadeh, M., Mahadevan, S., Makar, R.: Hierarchical multi-agent reinforcement learning. Autonomous Agents and Multi-Agent Systems 13, 197–229 (2006)
12. Guestrin, C., Lagoudakis, M., Parr, R.: Coordinated reinforcement learning. In: Machine learning international workshop then conference (2002)
13. Hu, J., Wellman, M.P.: Multiagent reinforcement learning: Theoretical framework and an algorithm. In: Proc. of ICML (1998)
14. Iscen, A., Erogul, U.: A new perspective to the keepaway soccer: The takers (short paper). In: Proc. of AAMAS (2008)
15. Lau, Q.P., Lee, M.L., Hsu, W.: Coordination guided reinforcement learning. In: Proc. of AAMAS (2012)
16. Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: Proc. of ICML (1994)
17. Min, H.Q., Zeng, J.A., Chen, J., Zhu, J.H.: A study of reinforcement learning in a new multiagent domain. In: 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (2008)
18. Mozina, M., Zabkar, J., Bratko, I.: Argument based machine learning. Artificial Intelligence 171, 922–937 (2007)
19. Sen, S., Sekaran, M., Hale, J.: Learning to coordinate without sharing information. In: Proc. of AAAI (1994)
20. Singh, S.P., Sutton, R.S.: Reinforcement learning with replacing eligibility traces. Machine Learning 22, 123–158 (1996)
21. Stone, P., Sutton, R., Kuhlmann, G.: Reinforcement learning for robocup soccer keepaway. Adaptive Behavior 13, 165–188 (2005)
22. Sutton, R., Barto, A.: Reinforcement Learning. MIT Press (1998)
23. Tambe, M., Jung, H.: The benefits of arguing in a team. AI Magzine 20(4), 85–92 (1999)