Reasoning About Intentions: a Defeasible Argumentation Approach

Nicolás D. Rotstein Guillermo R. Simari Alejandro J. García

Artificial Intelligence Research and Development Laboratory
Department of Computer Science and Engineering
Universidad Nacional del Sur
Av. Alem 1253, (8000) Bahía Blanca, Argentina
e-mail: {ndr,grs,ajg}@cs.uns.edu.ar

Abstract

In this work we present an approach to model an action-oriented system controlled by BDI agents using a defeasible argumentation formalism to represent its knowledge. Here, our main concern will be modelling the software agents that drive the physical robots. The chosen agent architecture will be BDI and it will use defeasible argumentation to perform the reasoning part of the system. Provided that our laboratory has the basic setup to implement a robotic soccer team, that is the application domain selected to test this approach.

Keywords: Argumentation - BDI - Logic Programming

1 Introduction

This article addresses the problem of having a robot that must reach a certain goal by means of a given set of actions. In order to achieve this, other problems must be solved first, from the construction of the robots to knowledge representation. Here, our main concern will be modelling the software agents that drive the physical robots. We have chosen the BDI architecture [9, 10], which is "...one of the most promising architectures for the development of intelligent agents, and has become one of the most studied and well known in the literature" [3].

In the BDI model, reasoning about beliefs, desires and intentions must be performed; we will use Defeasible Logic Programming (DeLP) [4] as the reasoning module. DeLP is an argumentative formalism [1, 8] that relies on a defeasible logic program. In our work, this program will contain rules that combine desires and beliefs to provide the agent with the capability of deriving intentions. Then, when the current intention is determined, the robot will use its effectors to perform the physical action that best accomplishes what the software agent intended.

Finally, the application domain chosen is robotic soccer, a system with enough complexity to show the capabilities of our approach. A soccer-playing robot has a variety of actions available

Acknowledgments: This research was funded by Universidad Nacional del Sur, Agencia Nacional de Promoción Científica y Tecnológica (PICT 13096, PICT 15043, PAV 076), and by CONICET (Argentina)

(*e.g.*, make a pass, carry the ball), and it can perform one at a time. Furthermore, the robot has a clear goal, *i.e.*, to score, and a single intention that can fulfill that task: to shoot on goal.

This paper is organized as follows: Section 2 will describe the agents architecture used in our approach; Section 3 will explain the scenario and how knowledge is represented in the form of defeasible rules; Section 4 will show how this approach is applied in a robotic soccer domain.

2 Agents Architecture

In this paper, agents will be implemented using a BDI architecture. An agent built upon a BDI architecture has an internal state that relies on three sets: *Beliefs* ($\mathcal{B}$), *Desires* ($\mathcal{D}$) and *Intentions* ($\mathcal{I}$). At a given moment the agent will have a set of beliefs $\mathsf{B} \subseteq \mathcal{B}$, a set of desires $\mathsf{D} \subseteq \mathcal{D}$, and a set of intentions $\mathsf{I} \subseteq \mathcal{I}$.

The set B of Beliefs will include the current state of the world and some parameters that define the role of the agent. The main idea is that everything that is perceived by the agent (explicitly or implicitly) as well as the parameters that rule its personality are contained in this set.

The set D of Desires will contain the *mental attitudes* that allow the agent to reach its final goal. Every element can be seen as a sub-goal, as the agent will be willing to accomplish it in an appropriate moment to fulfill its task. A particular subset D^c of D will be distinguished: the one containing only the desires that (according to B) can be currently fulfilled; this subset D^c will be the *Current Desires* set (see Definition 3.1).

The set I of Intentions will be a subset of D^c. Observe that there is a link connecting intentions and actions, provided that an action will be an effort to fulfill an intention, given the faulty nature of the robot's domain.

The BDI architecture describes a two-phase loop (Figure 1) that requires to perform reasoning: first, the set D is filtered to build the set D^c and then an intention has to be selected among the elements of D^c. In our approach, we will use DeLP to represent knowledge and derive intentions. Therefore, the argumentation process performed regarding each desire will model the reasoning part of the system. The defeasible knowledge under representation will be based on beliefs and desires encoded as a *Defeasible Logic Program* $\mathcal{P} = (\Pi, \Delta)$, where Π denotes the set of facts and strict rules, while Δ denotes the set of defeasible rules:

- *Facts*, which are ground literals representing atomic information or the negation of atomic information (using the strong negation "$\sim$").
- *Strict Rules* of the form $L_0 \leftarrow L_1, \ldots, L_n$, where L_0 is the *head* and $\{L_i\}_{i>0}$ is the *body*. Each L_i in the body or the head is a literal.
- *Defeasible Rules* of the form $L_0 \prec\!\!- L_1, \ldots, L_n$, where L_0 is the *head* and $\{L_i\}_{i>0}$ is the *body*. Each L_i in the body or the head is a literal.

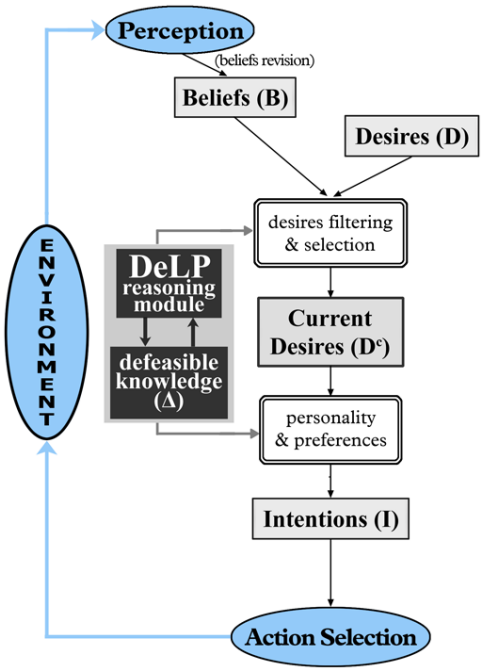

Figure 1: *BDI architecture using DeLP as a reasoning module*

In DeLP a literal L is *warranted* if there exists a non-defeated *argument* $\mathcal{A}$ supporting L. In order to establish if $\langle \mathcal{A}, L \rangle$ is a non-defeated argument, *argument rebuttals* or *counter-arguments* that could be *defeaters* for $\langle \mathcal{A}, L \rangle$ are considered, *i.e.*, counter-arguments that by some criterion are preferred to $\langle \mathcal{A}, L \rangle$. Since counter-arguments are arguments, there may exist defeaters for them, and defeaters for these defeaters, and so on. Thus, a sequence of arguments called an *argumentation line* is constructed, where each argument defeats its predecessor in the line (the interested reader should refer to [4] in order to obtain a more detailed explanation[1]).

In DeLP, given a query Q the possible answers will be: *YES*, if Q is warranted; *NO*, if the complement of Q is warranted; *UNDECIDED*, if neither Q nor its complement are warranted; and *UNKNOWN*, if Q is not in the language of the program.

3 Knowledge, Actions and Desires Representation

In this work we will consider the problem of having an action-oriented system with a single goal. This is a common scenario in the mobile robotics domain, where a robot has certain effectorial capabilities, along with a given task. An interesting issue arises when there is just one action that can be performed at a time, that is, the agent has to select the right one out of an available set of actions. Therefore, the suitability of each action must be weighed. The agent must have the ability of selecting the right action at every moment, *i.e.*, the action that gets it closer to the goal.

[1]The implementation (interpreter) of DeLP that satisfies the semantics described in [4] is currently accessible online at `http://lidia.cs.uns.edu.ar/DeLP`.

3.1 Defeasible Logic Program

As stated in Section 2, in this model knowledge will be encoded as a defeasible logic program and the reasoning will be performed by a formalism of defeasible argumentation. In this kind of logic programs, the programmer must assert reasons for and against certain literals, written in the form of rules. We will consider the following scheme of rules:

$\delta_i \prec \beta_{i_1}, ..., \beta_{i_n}$ (defeasible)
$\delta_j \leftarrow \beta_{j_1}, ..., \beta_{j_m}$ (strict)

Where every δ_i is a literal that stands for a reason for or against a desire. The literals β_{i_j} will represent beliefs.

In addition to this, if a reason for a desire holds, it must be considered as a reason against every other desire. Keep in mind that each desire is directly related to a physical action, and in this work we are assuming that just one action can be performed at a time. Hence, whenever the agent has a reason to perform an action, that must be automatically thought as a reason against the rest of the available actions (*e.g.*, if you have a reason to watch TV, you will have a reason against reading this paper, and *vice versa*). For that reason, the following type of rules must be added to the program for all desires:

$\overline{\delta}_i \prec \delta_j$, such that $i \neq j$

Having this rules implies that a warranted argument for a desire will be a strong opposition against the possibility of warrant of the rest of the desires. Although this seems to mean that the warrant of more than one desire will never happen, the influence of the beliefs within an argument can override this property and allow more than one desire to be warranted.

According to the rules described above, the defeasible program will represent the agent's knowledge by stating reasons for and against every element of the set of Desires based on the set of Beliefs and the set of Desires itself. Let's see an example, where we model the desires and beliefs of a soccer-playing agent that has the ability of performing passes and carrying the ball; then, the sets B and D are:

- $\mathsf{B} = \{$
  $noOneAhead(P)$ (true if player P has no players in front of him),
  $hasBall(P)$ (true if player P has the ball in his possession),
  $betterPosition(P1, P2)$ (true if player P1 is in a better position than P2)$\}$
- $\mathsf{D} = \{$
  $pass(Src, Tgt)$ (represents a reason for player Src to perform a pass to player Tgt),
  $carry(P)$ (represents a reason for player P to carry the ball)$\}$

And the defeasible logic program representing the agent's knowledge will be:

$\mathcal{P} = (\Pi, \Delta) = \{$
$(pass(Src, Tgt) \prec betterPosition(Tgt, Src)),$
$(\sim pass(Src, \_) \leftarrow not\ hasBall(Src)),$
$(carry(P) \prec noOneAhead(P)),$
$(\sim carry(P) \prec not\ hasBall(P))\}$

Rules of the form $\overline{\delta}_i \prec \delta_j$ must be written in order to establish that, when a reason for carrying the ball holds, then there is a reason against performing a pass to a teammate, and *vice versa*. Therefore, according to the set D, this rules are:

$(\sim pass(P, \_) \prec carry(P))$
$(\sim carry(P) \prec pass(P, \_))$

Preference criteria

Here we will address the methodological question of how to choose a preference criterion and we will guide the implementation of a suitable criterion for an agent built following this approach. First, we will introduce the notion of preference criterion, and then, some necessary definitions will be presented.

Given an argument structure $\mathcal{A}_2$ that is a counter-argument for $\mathcal{A}_1$, in order to decide which one prevails, these two arguments must be compared by some criterion. For example, in [4], if the counter-argument $\mathcal{A}_2$ is better than $\mathcal{A}_1$ w.r.t. the comparison criterion used, then $\mathcal{A}_2$ prevails and it will be called a proper defeater for $\mathcal{A}_1$. If neither argument is better nor worse than the other, a blocking situation occurs, and we will say that $\mathcal{A}_2$ is a blocking defeater for $\mathcal{A}_1$. If $\mathcal{A}_1$ is better than $\mathcal{A}_2$, then $\mathcal{A}_2$ will not be considered as a defeater for $\mathcal{A}_1$, and $\mathcal{A}_1$ prevails.

Definition 3.1 (Current desires) *Given the Beliefs* (B) *and Desires* (D) *sets within a BDI architecture, the* Current Desires (D^c) *set will be defined as follows:*

$$\mathsf{D}^c = \{\delta \in \mathsf{D} \mid \textit{there is no warrant}(\overline{\delta})\}$$

Then, the set D^c *will be a subset of the set* D *containing just the desires that, in concordance with the beliefs, can be satisfied in the current situation.*

According to the four possible answers in DeLP, the set D^c is defined:

$$\mathsf{D}^c = \{\delta \in \mathsf{D} \mid answer(\delta) \neq no\}$$

Before stating a necessary property for the comparison criterion, a couple of concepts must be introduced first: *disagreement sub-argument* and *supporting literal*. As explained in [4], if $\langle \mathcal{A}_1, h_1 \rangle$ counter-argues $\langle \mathcal{A}_2, h_2 \rangle$ at literal h, then the sub-argument structure $\langle \mathcal{A}_d, h \rangle$ will be called the *disagreement sub-argument*. In addition to this, we will call *supporting literals* to those that appear on an argument and are not on the head of any defeasible rule included in that argument.

Definition 3.2 (Comparison criterion) *Given two arguments $\mathcal{A}_1$ and $\mathcal{A}_2$, such that $\mathcal{A}_2$ is a counter-argument for $\mathcal{A}_1$, and the* disagreement sub-argument *of $\mathcal{A}_1$ is $\mathcal{A}_d$, then $\mathcal{A}_2$ is better than $\mathcal{A}_1$ if, and only if, $\mathcal{A}_2$'s* supporting literals *contain, at least, one desire, and $\mathcal{A}_d$'s* supporting literals *contain none.*

Any preference criteria used in a system designed by this approach must agree with Definition 3.2. This condition states that desire-based reasons are stronger than those based merely on beliefs. The following example will show how this criterion works. Consider this sets of beliefs, desires and rules:

$$\Delta = \{(a_1 \multimap v), (\sim a_1 \multimap b), (\sim a_1 \multimap a_2), (a_2 \multimap x), (a_2 \multimap y), (\sim a_2 \multimap d), (\sim a_2 \multimap a_1)\}$$

$$\mathsf{D} = \{a_1, a_2, a_3\}$$

$$\mathsf{B} = \Pi = \{v, x, y, b, d\}$$

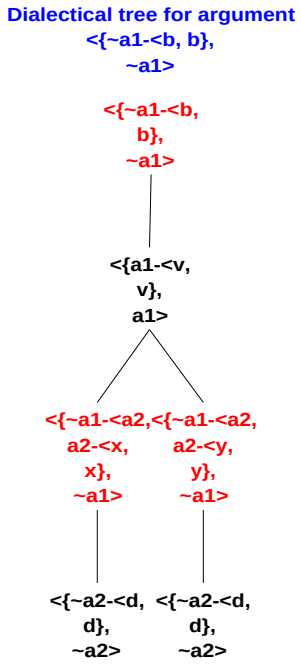

Figure 2: *Dialectical tree for query $\sim a_1$*

In Figure 2 we can see a dialectical tree for query $\sim a_1$ that shows the notion of defeat determined by this comparison criterion works. The root of this tree is blocked by a single defeater, and that argument is properly defeated by two arguments holding $\sim a_1$; this defeat

relationship is justified by the fact that a_1 is supported by the reason v, and the defeaters support their conclusions $\sim a_1$ with the action-related desire a_2. Finally, at the bottom, both proper defeaters are, in turn, blocked by the same argument, ending both argumentation lines.

We will get a variety of dialectical trees representing the reasoning process performed to achieve the answer. The interested reader may refer to `http://lidia.cs.uns.edu.ar/DeLP`.

3.2 Actions and Intentions

In this model, actions will be assumed as a move (or sequence of moves) performed by a robot trying to fulfill an intention previously derived by the software agent. Thus, an intention will be defined in terms of some analysis of the state of the warrant of the current desires, and the actions just will try to mirror the currently selected intention in the best possible way.

Definition 3.3 (Intention) *An intention I is a current desire that is enabled by a set P of preconditions and a set C of constraints of the form not C_i. Policies for obtaining intentions will be denoted in the following way:*

$$I \Leftarrow \{P_1, ..., P_m\},\ not\{C_1, ..., C_n\}$$

Notice that the notation not $\{C_1, ..., C_n\}$ represents $\{not\ C_1, ..., not\ C_n\}$.

Let $K = (\Pi, \Delta)$ be an agent's knowledge base, I will be an enabled intention if every precondition P_i has a warrant built from K and every constraint C_i fails to be warranted.

In the model presented in this paper, intentions are derived from desires. According to the changes in the environment, there could be, eventually, more than one enabled intention, so a preference order must be introduced to make the final choice. Nevertheless, proper agent design should avoid this; if the policies are correctly set, the chances of having a completely undecided situation will be highly diminished.

The selection of the intention will define the agent's personality and may be subordinated by its role in the multi-agent system. The reader must notice that having an intention selected does not mean that the effects of its associated action can be taken as facts.

The effects of the actions cannot have a predictable correlation in the agent's beliefs, because they will be ultimately carried out by a physical robot and its actions can fail or be imprecise. Therefore, the beliefs revision function will be performed by the sensing system of the robot, updating the geometric data, *i.e.*, the coordinates of all the objects in the playfield.

In this section we presented how the faulty nature of the physical environment (where the actions are performed) brings shortcomings which do not exist on a simulation. Because the effects of the actions cannot be determined in advance, there are no planning capabilities and the reasoning must be executed on-the-fly.

4 Application Domain: Robotic Soccer

Robotic soccer has proven to be a complex enough system to test many of the features of any reasoning system. The robots are controlled by software agents, each of which has a set of high-level actions to perform, such as kicking the ball with a given strength or moving in a given direction. Every moment an agent has to choose which action to do next, and that choice can be made by using a reasoning system, in this case, a defeasible argumentation system.

In this section we will consider a software agent that controls the behavior of a physical robot according to the model explained in this article. For the sake of simplicity, we will assume that the agent has the possession of the ball, and analyze the cases under this assumption. This will suffice to get the overall idea of the system's function.

The basic setup to start a robotic soccer match includes: a video camera, infra-red transmitter towers, the robots and computers devoted to run the agents and the video/command servers. This system works as it follows: The agent perceives its environment through the camera (that takes the whole playfield); then, a video server analyzes those images, recognizing the positions of the objects (ball and robots) and sending this information to the software agents. From this data the agents will build their (almost identical, they will differ just on personal features, such as their roles) B set describing the current state of the world. Next, from sets B and D, it will generate the set D^c. Finally, it will select an element from the D^c set as the current intention.

4.1 Agents Architecture

According to Section 2, the BDI sets to describe the internal state of a soccer-playing robot are the following:

- $\mathsf{B} = \{$
  $marked(P)$ (true if player P is marked),
  $oppositeField(P)$ (true if player P is in the opposite side of the field),
  $noOneAhead(P)$ (true if player P has no players in front of him),
  $goalieAway(T)$ (true if the goalkeeper of team T is bad positioned),
  $hasBall(P)$ (true if player P has the ball in his possession),
  $teammate(P)$ (true if player P is a teammate of the player calling the predicate),
  $betterPosition(P1, P2)$ (true if player P1 is in a better position than P2),
  $playerBetween(P1, P2)$ (true if there is a player between P1 and P2),
  $gameResult(R)$ (true if R is the current result of the game),
  $playerRole(R)$ (true if R is the role of the player calling the predicate)$\}$

  This set has predicates built upon the information gathered from the sensorial data and represents the state of the agent's world.

- $\mathsf{D} = \{$
  $shoot(P)$ (represents a reason for player P to shoot on goal),
  $pass(Src, Tgt)$ (represents a reason for player Src to perform a pass to player Tgt),
  $carry(P)$ (represents a reason for player P to carry the ball)$\}$

  This set is hard-wired to the robot's knowledge and contains the available desires that may be elected as an intention at any moment of the game. At every moment during the match, it is filtered and the D^c set is built.

- $\mathsf{I} = [shoot(P) \mid pass(Src, Tgt) \mid carry(P)]$

  The intention currently selected as explained in Section 3.2.

4.2 Defeasible Argumentation

The defeasible logic program has to define reasons for and against every element belonging to the set D and it does so via the following rules:

$\mathcal{P} = (\Pi, \Delta) = \{$

$(shoot(P) \prec oppositeField(P), noOneAhead(P)),$
$(shoot(P) \prec oppositeField(P), not\ marked(P)),$
$(shoot(\_) \prec goalieAway(opposite)),$

$(\sim shoot(P) \leftarrow not\ hasBall(P)),$
$(\sim shoot(P) \prec pass(P, \_)),$
$(\sim shoot(P) \prec carry(P)),$

$(pass(Src, \_) \prec marked(Src)),$
$(pass(Src, Tgt) \prec betterPosition(Tgt, Src)),$

$(\sim pass(Src, \_) \leftarrow not\ hasBall(Src)),$
$(\sim pass(Src, Tgt) \prec playerBetween(Src, Tgt)),$
$(\sim pass(\_, Tgt) \prec marked(Tgt)),$
$(\sim pass(Src, \_) \prec shoot(Src)),$
$(\sim pass(Src, \_) \prec carry(Src)),$

$(carry(P) \prec noOneAhead(P)),$

$(\sim carry(P) \leftarrow not\ hasBall(P)),$
$(\sim carry(P) \prec shoot(P)),$
$(\sim carry(P) \prec pass(P, \_))\}$

In the following examples we will show how a player makes a decision based on this model. Every scenario has a a couple of players belonging to the blue team and two or three yellow team players. We will analyze the reasoning performed by the blue team player labeled 'self'. Regarding knowledge representation, the beliefs predicates will be written according to the Close World Assumption, and everything that cannot be proved will be assumed false.

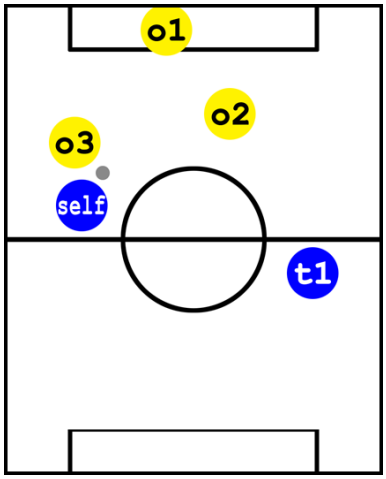

Figure 3: *Player 'self' decides to make a pass*

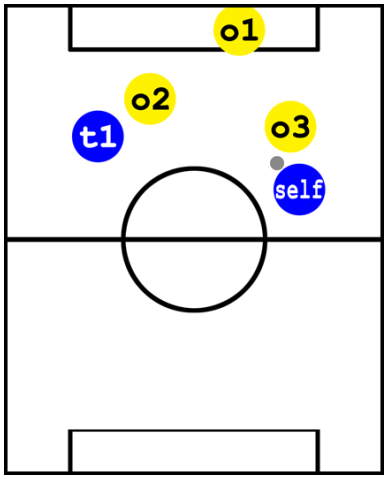

Figure 4: *Completely undecided situation*

4.3 Situation One: a straightforward decision

In this situation, as can be seen in Figure 3, the player has the following Beliefs set:

$$\mathsf{B} = \{marked(self), oppositeField(self), noOneAhead(t1), hasBall(self), teammate(t1), betterPosition(me, t1), playerRole(forward)\}$$

Once built the internal representation of the world, the agent performs DeLP queries over the elements of its set D, gathering their corresponding answers:

- shoot: NO; there are no reasons for shooting on goal, because the opposite goalkeeper is well-positioned, the player 'self' is marked, *etc.* On the other hand, there is one undefeated reason for not shooting on goal: a pass can be performed to teammate 't1', who is free.
- pass: YES; there is an undefeated reason to perform a pass to teammate 't1': player 'self' is marked by opponent 'o3'.
- carry: NO; there are no reasons for carrying the ball, because the path between 'self' and the goal is obstructed by opponent 'o3'. But there is a reason for not carrying the ball: a pass can be performed to teammate 't1'.

In this situation, the player has a clear choice: the selected intention must be *perform a pass to 't1'*, because it is the only current desire and it is warranted.

4.4 Situation Two: reasoning upon indecision

In this case, shown by Figure 4, player 'self' builds the following Beliefs set:

$$\mathsf{B} = \{marked(self), marked(t1), oppositeField(self), oppositeField(t1), hasBall(self), teammate(t1), betterPosition(t1, me), playerRole(forward)\}$$

The querying process over each desire throws the following answers:

- shoot: UNDECIDED; there are no reasons supporting this argument; on the other hand, there are two arguments holding not to shoot on goal, based on different reasons to perform a pass to 't1', but both are, in turn, defeated by an argument saying that 't1' is marked by opponent 'o2'.
- pass: UNDECIDED; the fact that both players are marked by an opponent, despite 't1' is better positioned than player 'self', results in the reasoner module being incapable of deriving a definite answer.
- carry: UNDECIDED; as can be seen in the Figure, 'self' cannot advance with the ball, so there are no arguments for it. Two reasons support not to carry the ball, and they are defeated in the same fashion as the reasons for not to shoot on goal were.

Now, in this situation, what should the agent do?, it couldn't make a decision for one of its three possible actions, but it has to choose one among them. Taking into consideration that the 'UNDECIDED' answer points that no warrant could be built for the queried literal nor its complement, the agent should determine its final intention with a policy like:

$$shoot(self) \Leftarrow not\{\sim shoot(self)\}$$

Therefore, when the *not to shoot on goal* (negated form of) desire is not warranted, the agent chooses to shoot on goal as its intention. This kind of policy describes the personality of an agent that shoots whenever it has an opportunity.

5 Conclusions

This article presents an approach to solve the problem of having an action-oriented system such as a robot with effectorial capabilities and a goal to complete. The robots are controlled by software agents based on a BDI architecture that reason via a defeasible argumentation module (DeLP). This module uses a defeasible logic program in the form of rules that combines desires and beliefs. Therefore, the agent will query about its desires and will derive a single intention, depending on the answers obtained. Finally, the robot will perform an action trying to satisfy the selected intention.

The application domain selected is interesting in the sense that its usefulness is two-folded: it give us the possibility of grounding the theoretical ideas developed in our laboratory, as well as the necessary feedback to fine-tune them. We also have the basic setup to verify the result of the physical actions made by real robots. Now that we have a concrete approach, we can start the development of a robotic soccer team that uses argumentation to perform reasoning.

References

[1] Chesñevar, C. I., Maguitman, A. G., and Loui, R. P. Logical Models of Argument. *ACM Computing Surveys 32*, 4 (Dec. 2000), 337–383.

[2] Dung, P. M. On the acceptability of arguments and its fundamental role in nonmonotonic reason, logic programming, and N-person games. *Artificial Intelligence 77* (1995), 321–357.

[3] Falappa, M. A., García, A. J., and Simari, G. R. Belief dynamics and defeasible argumentation in rational agents. In *Proceedings of the 10th International Workshop on Non-Monotonic Reasoning (NMR2004)* (June 2004).

[4] García, A. J., and Simari, G. R. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming 4*, 1 (2004), 95–138.

[5] Governatori, G., Maher, M., Antoniou, G., and Billington, D. Argumentation semantics for defeasible logic. *Journal of Logic and Computation 14* (2004), 675–702.

[6] Prakken, H., and Sartor, G. Argument-based extended logic programming with defeasible priorities. In *Working Notes of 3rd ModelAge Workshop: Formal Models of Agents* (Sesimbra, Portugal, 1996), P.-Y. Schobbens, Ed.

[7] Prakken, H., and Sartor, G. A dialectical model of assessing conflicting arguments in legal reasoning. *Artificial Intelligence and Law 4*, 3-4 (1996), 331–368.

[8] Prakken, H., and Vreeswijk, G. Logical systems for defeasible argumentation. In *Handbook of Philosophical Logic, 2nd ed.*, D.Gabbay, Ed. Kluwer Academic Pub., 2000.

[9] Rao, A. S. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World* (Eindhoven, The Netherlands, 1996), R. van Hoe, Ed.

[10] Rao, A. S., and Georgeff, M. P. BDI-agents: from theory to practice. In *Proceedings of the First International Conference on Multiagent Systems* (San Francisco, 1995).

[11] Simari, G. R., and Loui, R. P. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence 53*, 2–3 (1992), 125–157.