An Argumentative Intentional Model for High Level Reasoning of Mobile Robots

Sebastian Gottifredi, Mariano Tucat
Alejandro J. García, and Guillermo R. Simari

Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur. Av. Alem 1253, (8000) Bahía Blanca, Argentina
{sg,mt,ajg,grs}@cs.uns.edu.ar

Abstract. In this work we present an argumentation based intentional model for a BDI architecture defined formally for high level control of mobile robots. The proposed approach provides a sophisticated way of handling conflicting intentions using the argumentative formalism of Defeasible Logic Programming. To manage the intentional model, we present a special notion of argument disagreement and a new comparison criteria. Finally an implementation for the proposed system is introduced.

1 Introduction

In this work, we extend a BDI architecture presented and defined formally for high level control of a team of mobile robots that participated in the VI Argentine Championship of Robot Soccer (CAFR 2008 [2]). The original BDI architecture represented the higher level of a layered system, in which each layer is associated with a different level of abstraction. The proposed architecture uses a logic-based model for knowledge representation and reasoning. We will extend this architecture by using Defeasible Logic Programming (DeLP) for the representation of intention rules. The intention model will allow the developer to specify the situations where a plan is applicable in a declarative manner. Since plans cannot be executed in parallel, agent intentions will be conflicting. We will provide a formalism based on Defeasible Logic Programming that provides an elegant way of handling these conflicting intentions. We will also introduce a concrete implementation of the extended architecture using argumentative servers.

Mobile robots involved in complex environments require high degree of intelligence or high level capabilities (such as reasoning, knowledge representation, planning, agent communication) integrated with lower level primitives (such as sensor management, basic movements, obstacle avoidance, navigation, etc.). Since 2004 we have been working in mobile robotics, specially in robotic soccer. Our previous researches were focused on those high and lower level areas. In particular, we have developed an obstacle avoiding system [9], researches on sensorial information and basic movements [8], and a multi-agent architecture

Partially supported by CONICET (PIP 5050) and SGCyT UNS

to control the robots [6]. We also have developed a robotic soccer team that participated in the E-League held in Robocup 2004.

We have recently designed a Multi-Agent System Architecture for implementing the control of the team. We developed the system following a three-layer architecture. The main goal of the design and implementation of this architecture was to encapsulate all the low level developments, including the ones mentioned above, in the lower layers of the architecture and allow an easier implementation of high level capabilities in the upper layer. Therefore, AI theories developed on reasoning, knowledge representation, planning, agent communication, among others, can be tested in this real scenario. In particular, we developed a team controlled by a BDI architecture [11,7] to participate in the VI Argentine Championship of Robot Soccer (CAFR 2008 [2]).

The main goal of the leagues we participated in (E-League and CAFR [2]) is to provide an environment where researchers, practitioners and students interact sharing knowledge and expertise while enjoying the games. The leagues provide common basic services to all of the participants, such as vision and communication. Teams can use low cost kits (such as Lego [1]) and concentrate on the development and study of Artificial Intelligence techniques. The most important feature of these leagues is its simple and modular structure. There are only three basic components that must be available to obtain a functional team: a vision module that works as the robot's perception component, a communication module that allows actions to be communicated to the robots, and a control module that is implemented by agents that control the robots on the field of play.

The game of soccer can be seen as a well defined system: the number and type of players, duration of play, allowed behaviors, and punishments (among other aspects of the game) are governed by a well defined set of rules that are known to all participants. However, the interaction among the players cannot be defined beforehand. Each team is composed of players that must cooperate in order to reach their goal of winning the game. They must also take into account the existence of the opposing team, which also has the goal of winning the game.

2 Theoretical Background

In this section we give a brief summary of Defeasible Logic Programming (DeLP) [5], the goal of this summary is to introduce formal concepts used in this work.

DeLP is a formalism that combines results of Logic Programming and Defeasible Argumentation. DeLP provides the possibility of representing information in the form of rules in a declarative manner, and a defeasible argumentation inference mechanism for warranting the entailed conclusions. These rules are the key element for introducing defeasibility and they will be used to represent a relation between pieces of knowledge that could be defeated after all things are considered. Using these rules, common sense reasoning is defeasible in a way that is not explicitly programmed. Defeat should be the result of a global consideration of the corpus of knowledge of the agent performing the inference. Defeasible Argumentation provides the tools for doing this.

In a Defeasible Logic Program (or *de.l.p.* for short) knowledge can be represented using facts and defeasible rules. *Facts* are ground literals representing atomic information or the negation of atomic information using strong negation "$\sim$" (e. g. b, or $\sim a$). *Defeasible Rules* (d-rules) are denoted $L_0 -\!\prec L_1, \ldots, L_n$, where the *head* L_0 is a ground literal and the *body* $\{L_i\}_{i>0}$ is a set of ground literals. (e. g. $\sim a -\!\prec b, c$). A d-rule represents tentative information that may be used if nothing could be posed against it. A d-rule "$Head -\!\prec Body$" expresses that "*reasons to believe in the antecedent Body give reasons to believe in the consequent Head*". D-rules are ground, however, following the usual convention, some examples will use "schematic rules" with variables. When required, the set of facts is denoted Ψ and the set of d-rules Δ.

Strong negation could appear in facts or in the head of d-rules and can be used to represent contradictory knowledge. Observe that from *de.l.p.* contradictory literals could be derived, however, the set Ψ (used to represent non-defeasible information) must be non-contradictory, *i. e.*no pair of contradictory literals can be derived from Ψ. Given a literal L, $\overline{L}$ represents the complement with respect to strong negation. For instance, suppose a *de.l.p.* $\mathcal{P}_1$=(Δ_1,Ψ_1), where $\Delta_1 = \{(a -\!\prec b), (\sim a -\!\prec b, c)\}$ and $\Psi_1 = \{b,c\}$, then both a and $\sim a$ can be defeasibly derived using the first and the second d-rule respectively.

In DeLP when contradictory literals are derived, a dialectical process is used for deciding which literals are warranted. A literal L is *warranted* if there exists a non-defeated argument $\mathcal{A}$ supporting L. To establish if $\langle \mathcal{A}, L\rangle$ is a non-defeated argument, defeaters for $\langle \mathcal{A}, L\rangle$ are considered. A *defeater* is a counter-argument that is preferred to $\langle \mathcal{A}, L\rangle$ by some argument comparison criterion. Counter-arguments of $\langle \mathcal{A}, L\rangle$ are those arguments that *disagree* (are in contradiction) at some point with $\langle \mathcal{A}, L\rangle$.

An *argument* for a literal L, denoted $\langle \mathcal{A}, L\rangle$, is a minimal non-contradictory set of d-rules $\mathcal{A}\subseteq \Delta$, that allows to derive L. A sub-argument of an argument $\langle \mathcal{A}, L\rangle$ is a subset of the d-rules in $\mathcal{A}$. For example, using the *de.l.p.* $\mathcal{P}_1$, the following arguments can be constructed: $\mathcal{A}_1$= $\{a -\!\prec b\}$ for a, and $\mathcal{A}_2$=$\{\sim a -\!\prec b, c\}$ for $\sim a$. Following the example, $\mathcal{A}_2$ is a counter-argument for $\mathcal{A}_1$ (and viceversa) because both support contradictory conclusions.

Given an argument $\langle \mathcal{A}_1, h_1\rangle$ and a counter-argument $\langle \mathcal{A}_2, h_2\rangle$ for $\langle \mathcal{A}_1, h_1\rangle$ these two arguments can be compared in order to decide which one prevails. This is made by the comparison criterion ($\leq$) that defines a partial order among the arguments. Given two arguments $\mathcal{D}$, $\mathcal{A}$ from a *de.l.p.*, $\mathcal{D}$ is a *proper defeater* of $\mathcal{A}$ if $\mathcal{D}$ $\mathcal{D}$ is strictly stronger than $\mathcal{A}$. $\mathcal{D}$ is a *blocking defeater* of $\mathcal{A}$ neither argument is better, nor worse, than the other. In our example, $\mathcal{A}_2$ is a proper defeater for $\mathcal{A}_1$ since $\mathcal{A}_2$ is strictly more specific than $\mathcal{A}_1$. A defeater can attack the conclusion of an argument or an inner point of it. Since defeaters are arguments, there may exist defeaters for them, defeaters for those defeaters, and so on. Thus, a sequence of arguments called *argumentation line* can arise. Note that an argument can not appear more than once in an argumentation line (for more details see [5])

Clearly, for a particular argument $\langle \mathcal{A}_0, h_0\rangle$ there might be more than one defeater. Therefore, many argumentation lines could arise from one argument.

This leads to a tree structure called *dialectical tree* (d-tree) [5], denoted $\mathcal{T}\langle\mathcal{A}_0, h_0\rangle$. In a dialectical tree every node (except the root) is a defeater of its parent, and leaves are non-defeated arguments. In a d-tree every node can be marked as *defeated* (D) or *undefeated* (U): leaves are marked as undefeated; inner nodes are marked as defeated when there is at least a child marked as undefeated, or are marked as undefeated when all its children are marked as defeated. The marked dialectical tree for an argument $\langle\mathcal{A}, h\rangle$ is denoted $\mathcal{T}^*\langle\mathcal{A}, h\rangle$. A literal h will be warranted from a *de.l.p.* $\mathcal{P}$ (noted $\mathcal{P} \mid\sim_w h$), if there exists an argument $\langle\mathcal{A}, h\rangle$ from $\mathcal{P}$, and the the root of $\mathcal{T}^*\langle\mathcal{A}, h\rangle$ is marked as "U". For instance from the *de.l.p.* $\mathcal{P}_1$ the literal $\sim a$ and the elements of Ψ_1 are warranted.

In order to provide an argumentative reasoning service for multi-agent systems, a more flexible implementation of DeLP, called DeLP-server, has been developed [4]. A DeLP-server is a stand-alone program that can interact with multiple client agents. A common (or public) DeLP-program can be stored in a server, and client agents (that can be distributed in remote hosts) may send queries to the server and receive the corresponding answer together with the explanation for that answer. A single DeLP-server can be consulted by several agents, and one particular agent can consult several DeLP-servers simultaneously, each of them providing a different shared knowledge base.

To answer queries, a DeLP-server will use the common knowledge stored in it, together with individual knowledge that clients can send attached to a query, creating a particular *context* for that query. This context is private knowledge that the server will use for answering the query and will not affect other future queries. That is, a client agent cannot make permanent changes to the *de.l.p.* stored in a server . The temporal scope of the context sent in a query [$Context$,Q] is limited and disappears once the query Q has been answered.

3 Design of the agent system

The proposed design considers the construction of the system based on an hybrid architecture combining reaction with deliberation [10]. The most popular hybrid architectures is the three layer architecture, which consists of a *reactive layer*, an *executive layer* and a *deliberative layer*, each of which covers different levels of abstraction of the problem to be solved. The *reactive layer* provides low-level control of the robot. The *executive layer* serves as the glue between the *reactive* and the *deliberative layer*. It accepts directives by the *deliberative layer*, and sequences them for the *reactive layer*. The *deliberative layer* is responsible for controlling the robot behavior by taking high level decisions.

3.1 Reactive layer

We associate with this layer the program running inside the robots and implementing the basic actions they need to be able to act in a dynamic environment such as robotic soccer. This layer also includes the basic hardware and software support that is provided by the league. This involves physical support, such as

infrared transmitters, video camera and communication network. The software provided includes video and command communication servers.

The basic actions represent the minimal unit of change a robot may try to execute in order to modify its environment. The possible basic actions of any robot are directly related to its shape, design and capabilities. In our case, the basic actions implemented inside the robots include three generic movements: moving forward and backward, rotating clockwise or counterclockwise and describing different kinds of arcs.

This layer also includes the basic perceptions. The basic perceptions represent the information any robot may obtain from the environment. This information may correspond to the location of the objects that are part of the environment. It may also represent the orientation and velocities of these objects.

3.2 Executive Layer

This layer serves as the glue between the *reactive* and the *deliberative layer*. It accepts directives by the *deliberative layer*, and sequences them for the *reactive layer*. It also provides perception information about the environment to the *deliberative layer* in order to allow it to reason/decide about the robot behavior. Therefore, this layer is divided in two sub-layers, the *planner manager sub-layer* and the *sensorial sub-layer*.

The *planner manager sub-layer* provides a set of implemented *schematic plans* allowing the *deliberative layer* to control the robot behavior without worrying about low level details. Thus, this sub-layer provides the ability to select and execute actions by planning, and performing such actions as a matter of plan execution. However, this *planner manager* does not decide which *schematic plan* to execute, it only obtains the actual *schematic plan* and selects the best action to perform next in order to accomplish the desired goal of the plan.

A *schematic plan* represents a plan in a highly dynamic environment, in which the sequence of actions needed to achieve the desired goal may vary at the moment of executing the plan. Therefore, these *schematic plans* are divided in several atomic actions, in particular, the basic actions described in the *reactive layer*, and each of these actions depend on the state of the field at the moment immediate before of been executed.

The current *schematic plans* implemented for the domain of robotic soccer are: go to a given object, such as another robot or the ball, pass the ball to a teammate, go to a defensive position, kick the ball, dribble to a given location, and clear the ball out of the defensive zone.

In the *sensorial sub-layer*, the visual information is processed and translated into information that express states of the world. This information is divided in basic perception and processed information. The basic perception, defined in the previous subsection, corresponds to the coordinates, orientation and speeds of the robots and the ball. Examples of processed information are: The robots' and the ball's locations relative to the field, player and/or team that is closest to the ball, distances between different objects on the field, whether the ball is moving and in which estimated direction, or unable to move towards it goal.

These information is provided to the *deliberative layer* as PROLOG predicates, and they are implemented by querying and analyzing the basic perceptions obtained from the video server. Then, the situations modeled through these predicates are used in the upper layer to model the team's game strategy.

3.3 Deliberative Layer

This layer is responsible for the design and implementation of the agents that control the robots behavior. The lower layers allows this layer to obtain high level processed information, use it to decide the behavior of the robots, and finally control the robots. In the deliberative layer, agents will have a deliberative cycle in which they perceive information about the environment and reason/decide about the *schematic plans* to take. The implementation of the *executive layer* allows to specify the elements of the cognitive layer using different knowledge representation and reasoning systems. In [7] a Belief-Desires-Intentions (BDI) model [3] was used to design the agents controlling the robots. In the following section, will present an extension of that approach were DeLP is used as knowledge representation and reasoning mechanism for intentional model.

4 An Argumentative Approach for the Deliberative Layer

In this section we will present an extension of the BDI-model presented in [7]. The proposed architecture uses a logic-based model of the knowledge representation and reasoning. We will extend this architecture by using Defeasible Logic Programming (DeLP) for the representation of the intention rules. The intention model will allow the developer to specify the situations where a plan is applicable in a declarative manner. Since plans cannot be executed in parallel, agent intentions will be conflicting. We will provide a formalism based on DeLP that provides an elegant way of handling these conflicting intentions.

In the BDI model [3] an agent is specified using mentalistic attributes Belief, Desires and Intentions, and uses these components to determine the agent state. Thus the agent will have a belief set containing information about the environment in which it is involved, a desire set containing all the possible goal that the agent have and a set of intentions containing all the possibilities that the agent has to achieve its desires. In the following subsections, we will show the formal aspects of each component corresponding to our approach. For this, we will introduce the following working example:

Example 1.
Consider an agent belonging to the blue team is playing against the yellow team (see Figure 1). Its team is winning the match 2 - 0. Currently, the agent (labelled "me") is near the right part of blue team penalty box and it is carrying the ball. Its teammate "b1" is located in the center of the middle field. One of its opponents "y1" is also located in the center of the middle field and the other opponent"y2" is located in the center of the yellow team penalty area.

Figure 1

4.1 Beliefs

Beliefs of an agent in the BDI Architecture are used to represent a situation of the world that the agent is in. A mobile robot should be able to work in dynamic environments with incomplete knowledge of its environment. That is, robots will need a formalism to reason about this incomplete, uncertain, and changing information. Therefore, we will use Prolog logic formalism to reason about beliefs.

The PROLOG program used to represent agent beliefs will contain: sensed information such as field (X,Y) coordinates of an object, distances between objects, direction that a robot is facing, or if a robot has the ball; strict information such as which robots are teammates, or which robots are opponents; and inference rules to obtain inferred information such as the empty spots in the field or the best places to dribble. Next we will define the belief base of an agent considering these information categories.

Definition 1 (belief base) *The belief base of an agent will be a* PROLOG *program* $\mathcal{BB}=(\phi,\pi,\delta)$*, where* ϕ *is a set of* PROLOG *facts used to represent sensed information obtained from the* executive layer*,* π *is a set of* PROLOG *facts used to represent strict information and* δ *is a set of* PROLOG *rules used to obtain inferred information.*

Example 2. Consider the situation depicted in Figure 1, there the agent from Example 1 will have the following set of sensed information ϕ_1={hasBall(me), pos(middle, center, b1), pos(myPenBox, right, me), pos(middle, center, y1), pos(opPenBox, center, y2), winning}, as strict information will have the following set π_1={teammate(b1), opponent(y1), opponent(y2)}, and suppose that has the following inference rules (sketch) δ_1={ blocked(X) :- pos(Zone,Dir,me), pos(Zone,DirOp,X),opponent(X)}. Thus, its belief base will be $\mathcal{BB} = (\phi_1, \pi_1, \delta_1)$

The facts of the set ϕ are updated every deliberative cycle by the sub-sensorial layer. Since this sensorial information may be used by the inference rules the inferred information can also change every deliberative cycle. Next, we will show how to obtain the elements inferred by $\mathcal{BB}$.

Definition 2 (actual belief) *Let* $\mathcal{BB}=(\phi, \pi,\delta)$ *be a belief base of a agent, The* PROLOG *atom* h *is an* actual belief *from* $\mathcal{BB}$ *(noted* $\mathcal{BB} \vdash h$*) iff* $h \in \phi \cup \pi$ *or* h *can be obtained via a* PROLOG *derivation using the rules of* δ *and the* PROLOG *inference mechanism.If it is the case that an atom* h *is not a actual belief from* $\mathcal{BB}$ *it will be noted* $\mathcal{BB} \nvdash h$

Example 3. The agent with a belief base presented in the example 2, will have the following actual beliefs: blocked(b1) and the elements of ϕ and π.

The belief base will provide the other mental components with the *actual beliefs* to do their computations and decide how to act based in the situation in which the agent is in.

4.2 Desires

Desires in the BDI architecture represent what the agent wants to do. In this work desires will represent high level attitudes that the agent will try to achieve during the match, for instance *attack*, *defend* or *score a goal*. Depending on the situation in which the agent is involved, there are desires can be justified or not. For instance, if the agent has the ball and is loosing the match, then, the desire *defend* is not quite adequate, whereas *attack* is a plausible option. In order to specify this desire behavior, next, we will introduce the desire rules, which are used to determine when a desire is justified.

Definition 3 (desire rule) *A desire rule is a duple* $DR=(d(Imp), Just)$*, where* d *is an atom representing a desire,* Imp *is a number denoting the importance value of the rule desire, and* $Just=\{p_1,\ldots, p_n, not\ c_1,\ldots, not\ c_m\}$ *(*$n \geq 0$ *and* $m \geq 0$*) is formed by a set of atoms* $\{p_1, \ldots, p_n\}$ *representing belief preconditions and a set of extended atoms* $\{not\ c_1, \ldots, not\ c_m\}$ *representing belief restrictions.*

Since desire rules involve belief and desires and both of them are atoms we will assume that beliefs and desires are represented with separate names. Hence, a desire cannot be perceived or derived as a belief. All the specified desire rules determine the agent *desire base* noted DB. Note that the set $D = \{\bigcup d \mid (d(Imp), Just) \in DB\}$ contains all the possible desires.

Example 4. The agent desire base is DB = (defend(5),{hasBall(X),teamMate(X)}), (attack(15),{hasBall(X),teamMate(X), loosing}), {(attack(5),{hasBall(me)}), (defend(15),{hasBall(me),winning}), (defend(20),{hasBall(X),opponent(X)}) }

Using these rules an agent will be able to determine which of its desires are justified in each deliberative cycle. Next, we will define how to obtain them.

Definition 4 (justified desire) *Let* $\mathcal{BB}$ *be a belief base, a desire rule* $DR = (d(Imp), Just)$*, and* $Just=\{p_1,\ldots, p_n, not\ c_1,\ldots, not\ c_m\}$ *with* $n \geq 0$ *and* $m \geq 0$*. Then* d *will be a* justified desire *with an importance value of* Imp *(noted d(Imp)) iff* $\forall i = 1..n\ \mathcal{BB} \vdash p_i$ *and* $\forall j = 1..m\ \mathcal{BB} \nvdash c_j$

Example 5. Suppose that an agent is in the situation described in figure 1, the belief base of example 2 and has the desires rules of example 4 then it will have the following justified desires: attack(5) and defend(15).

The set JD of justified desires will contain all the desires that are justified. Notice that a desire d might be justified by several rules. However, an agent will only use the justified desire with higher importance value. This means that the JD will have one occurrence of each justified desire. The set JD will allow the agent to determine whether intentions it can commit to.

4.3 Intentions

In this work intentions will be used to determine which plan the agent will execute to achieve its desires. Basically, an intention will be used to denote *when* a *schematic plan* is executable. For instance, if the agent believes that it has the ball and desires to attack, it can have intentions to plan "dribble to the opponent area" or plan "shoot to the goal". Thus, with this intention model, the developer will specify different alternatives or intentions to achieve desires. Since these alternatives (the *schematic plans*) are generally conflicting, an agent should use a mechanism to determine which one is the best. Once the best alternative found the agent should commit and execute its actions.

In order to address these issues we will use DeLP as intentional reasoning model. We will use d-rules to model tentative reasons to determine whether an intention is applicable or not. The body of d-rules will be used represent the desires and beliefs involved in the intention. Next, we will define how d-rules are use to specify intention rules.

Definition 5 (intention rule) *Let $D = L_0 -\!\prec L_1, \ldots, L_n$ be an d-rule, we will say that D is a intention rule iff $L_0 = I(X)$, where X is a* schematic plan.

An intention base $\mathcal{IB}$ will be a *de.l.p.* were Δ contains only intention rules. Since an intention involve beliefs and goals, in $\mathcal{IB}$, actual beliefs and justified desires will be considered facts. To refer to these elements in the body of intention rules we will use the modalities $B(\cdot)$ and $D(\cdot)$ for actual beliefs and justifies desires respectively. For instance, if marked(me) is a actual belief and attack(20) is a justified desire, B(marked(me)) and D(attack(20)) will be considered as facts in $\mathcal{IB}$. Note that these facts will change in every deliberative cycle, since actual beliefs and justified desires change.

Example 6. An intention base (sketch) $\mathcal{IB} = (\Delta, \emptyset)$ could be:

$$\begin{aligned}\Delta = \{\ & I(shoot) -\!\prec B(hasBall(me)), notB(blocked(me)), D(attack(X)) \\ & I(dribble) -\!\prec B(hasBall(me)), notB(blocked(me)), D(defend(X)) \\ & I(clear) -\!\prec B(hasBall(me)), B(pos(me,, myPenBox)), D(defend(X)) \\ & I(pass) -\!\prec B(hasBall(me)), B(blocked(me)), D(attack(X))\ \}\end{aligned}$$

The set PI will have all the possible intentions that an agent could have. This set is composed by all the heads of the intention rules in $\mathcal{IB}$.

Remember that an agent will be able to execute one *schematic plan* at a time. Since an intention will determine a plan we should make intentions conflicting. For instance, we want the intentions I(shoot) and I(pass) be conflicting. In DeLP, this can be easily done extending the notion of disagreement:

Definition 6 (Intention Disagreement) *Let $\mathcal{IB}$ be an agent Intention Base, and L_1, L_2 literals form $\mathcal{IB}$. We will say that a and b disagree iff $L_1 = \overline{L_2}$ or if $L_1 = I(X)$, $L_2 = I(Y)$ and $X \neq Y$.*

In order to determine which intention an agent will select, we will use the DeLP inference formalism. That is, from $\mathcal{IB}$ arguments for intentions from PI,

will be constructed. These arguments will represent active possible intentions that an agent has in a deliberative cycle. Based the definition of intention disagreement these arguments will be conflicting. In other words, an argument for I(X) will be a counter argument of an argument for I(Y) if $X \neq Y$. Since we want to commit to one intention, we must determine if the argument for I(X) is a *defeater* of the argument for I(Y). In Section 2, we have shown that the comparison criterion is used on that propose. Next, we will present new comparison criterion for the intention base:

Definition 7 (Intention Comparison Criterion) *Let $\mathcal{IB}$ be an intention base, and $\langle \mathcal{A}, I(X)\rangle, \langle \mathcal{B}, I(Y)\rangle$ from $\mathcal{IB}$. We will say that the argument $\langle \mathcal{A}, I(X)\rangle$ is better $\langle \mathcal{B}, I(Y)\rangle$ (noted $\langle \mathcal{A}, I(X)\rangle < \langle \mathcal{B}, I(Y)\rangle$) if $ValX > ValY$, where $ValX = (\sum vX_i)\ \forall vX_i$, $D(a_i(vX_i) \in \mathcal{A}$ and $ValY = \sum vY_j\ \forall vY_j$, $D(b_j(vX_j) \in \mathcal{B}$ else if $(ValX = ValY)$ and X appears before than Y in the following list: clear, shoot, move, dribble, pass.*

Example 7. Using the actual beliefs and justified desires from examples 3 and 5, and the intention base of example 6, arguments $\mathcal{A}_1$ for $I(shoot)$ and $\mathcal{A}_2$ for $I(dribble)$ can be built. Following the intention comparison criterion, $\mathcal{A}_1 < \mathcal{A}_2$.

The criterion defined above, imposes a total order among the arguments for the possible intentions. Therefore, using this criterion and the DeLP reasoning mechanism, an agent will be able to determine one intention to commit with.

Definition 8 (Intention Commitment) *Let AB be the set of actual beliefs, JD be the set of justified desires and $\mathcal{IB}$=(Ψ, Δ) a intention base, an agent commits to an intention* I(X) *from $\mathcal{IB}$ iff $((\phi \cup \iota \cup \Psi), \Delta) \mid\sim_w$ I(X), where* X *is an* schematic plan*, $\phi = \{B(\beta) | \beta \in AB\}$ and $\iota = \{D(\gamma) | \gamma \in JD\}$*

Remark 1. Since all the elements in PI are conflicting and the comparison criterion imposes a total order among the arguments for elements in PI, an agent will commit with only one intention.

Example 8. Using the actual beliefs and justified desires from Ex. 3 and 5, and the intention base of Ex. 6, the agent will commit with the intention $I(clear)$.
An agent in this proposal may have several intention bases which can be used depending the situation. The motivation behind this idea is to speed up argumentation process, since possible intentions can be divided into several categories. For instance, one base to decide between attacking related intentions and other base to decide between defending related intentions. Other example could be have an intention base to handle reactive intentions and other for complex intentions. However, each deliberative cycle the agent should decide which intention base use. Next, we will present a function used to decide which intention base select based in the actual beliefs and justified desires.

Definition 9 (Intention Base Selection Function) *Let AB be the set of actual beliefs and JD be the set of justified desires, a Intention Base Selection Function is a function IBSF: $AB \times JD \rightarrow \mathcal{IB}$*

For instance, suppose that the agent has an offensive intention base and a defensive intention base, the intention base selection function can determine which base select depending on the importance of the justified desires. Intention rules and commitments complete the agent architecture. These rules allow to developer to specify the situations where a plan is applicable in a declarative manner. DeLP provides an elegant way of handling intention conflicts. Using the special criterion and the argumentative mechanism the agent will always have as much as one intention to commit with.

5 Implementation Issues

This section present the main issues concerning the implementation of the deliberative layer, that is, the BDI architecture explained in the previous section. In particular, we will explain how the introduced components of this layer interact in order to obtain the expected behavior.

Basic perceptions and also processed information are provided as PROLOG predicates, facilitating the obtention of the set of actual beliefs, defined as PROLOG rules and facts. This set is obtained from a PROLOG interpreter with the PROLOG program corresponding to the belief base, by querying it for each possible belief. This can be done by executing `findall(B,b(B),Actual_Beliefs)`.

In order to compute the set of justified desires, a PROLOG program having the program corresponding to the belief base and a PROLOG rule for each desire rule is used. A PROLOG rule for a desire rule has as head the desire with the importance number as parameter, while the preconditions and restrictions form the body of the rule. Therefore, this program is queried for each possible desire, obtaining all the justified desires with their corresponding importance. This can be done by executing `findall(D,d(D),Justified_Desires)`.

As explained in the previous section, our approach allows the agent to have different intention bases. In order to determine the applicable intention, we must define which of the intention bases we will use, depending on the justified desires. Therefore, the Intention Base Selection Function is executed in order to determine which of the intention bases should be used.

Each intention base will be represented as a DeLP program loaded at a DeLP Server. Once we have defined which intention base to use, we will know which of the existent DeLP Servers to query in order to obtain the applicable intention. Therefore, a contextual query will be executed in the corresponding DeLP Server, having as context the set of actual beliefs and the set of justified desires, and as query the atom "i(X)". This contextual query will return the applicable intention corresponding to the intention base, containing a schematic plan to be executed by the executive layer.

The different DeLP Servers are executed as independent processes that may run on the same or different hosts. The agent implementing the BDI architecture will use TCP connections to interact with the DeLP Servers. The overhead imposed for having these servers running as independent processes does not affect the quick reaction of the controlled robot in a dynamic match, since the

use of schematic plans allows the agents to decide the best plan to execute while the robot execute the previous one.

6 Conclusions

In this work we have shown an extension of a BDI architecture for high level control of a team of mobile robots. The original BDI architecture represented the higher level of a layered system, in which each layer is associated with a different level of abstraction. The presented architecture uses a logic-based model of the knowledge representation and reasoning, and we extended it by using DeLP for the representation of the intention rules. The proposed intention model allows the developer to specify the situations where a plan is applicable in a declarative manner. We introduced a special disagreement notion to specify intention conflicts. We also presented a comparison criterion that provides a complete order among the possible intentions. These elements combined with the DeLP formalism provides an elegant way of choosing the applicable intention. The presented formalism allows the use of different intention bases at the same time, speeding up the intention decision process. Finally, we introduced a concrete implementation of the extended architecture, in which DeLP servers represent the different intention bases that an agent will reason with.

References

1. http://www.legomindstorms.com. Lego Mindstorms robots and RCX controllers.
2. http://www.uncoma.edu.ar/cafr2008/. Official webpage of the VI Argentine Championship of Robot Soccer.
3. M. Bratman, D. Israel, and M. Pollack. Plans and resource-bounded practical reasoning. In Robert Cummins and John L. Pollock, editors, *Philosophy and AI: Essays at the Interface*, pages 1–22, Cambridge, Massachusetts, 1991.
4. A. García, N. Rotstein, M. Tucat, and G. Simari. An argumentative reasoning service for deliberative agents. In *KSEM*, 2007.
5. A. Garcia and G. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, 4:95–138, 2004.
6. A. J. García, G. I. Simari, and T. Delladio. Designing an agent system for controlling a robotic soccer team. In *CACIC 2004*.
7. S. Gottifredi, M. Tucat, D. Corbatta, A. J. García, and G. Simari. A bdi architecture for high level robot deliberation. In *CACIC 2008*.
8. F. Martín, M. Tucat, and A. J. García. Soluciones a problemas de percepción y acción en el dominio de un equipo de fútbol de robots. In *CACIC 2004*.
9. N. Rotstein and A. J. García. Evasión de obstáculos con bajo costo computacional para un equipo de fútbol de robots. In *CACIC 2004*.
10. S. Russel and P. Norvig. *Artificial Intelligence: a Modern Approach*. Prentice-Hall, 1995.
11. M. Tucat, S. Gottifredi, F. Vidaurreta, A. J. García, and G. Simari. A layered architecture using schematic plans for controlling mobile robots. In *CACIC 2008*.