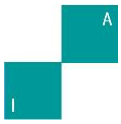
Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial
ISSN: 1137-3601
revista@aepia.org
Asociación Española para la Inteligencia Artificial
España

Gottifredi, Sebastian; Tucat, Mariano; Corbatta, Daniel; García, Alejandro J.; Simari, Guillermo R.
A BDI Architecture for High Level Robot Deliberation
Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, vol. 14, núm. 46, 2010, pp. 74-83
Asociación Española para la Inteligencia Artificial
Valencia, España

Inteligencia Artificial 46(2010), 74-83
doi: 10.4114/ia.v14i46.1540



INTELIGENCIA ARTIFICIAL

http://erevista.aepia.org/

A BDI Architecture for High Level Robot Deliberation

Sebastian Gottifredi†, Mariano Tucat†, Daniel Corbatta, Alejandro J. García†, 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,dc,ajg,grs}@cs.uns.edu.ar

Abstract In this work we present a BDI agent architecture to provide mobile soccer robots a high level reasoning mechanism. This architecture is build on top of layered system, where each layer is associated with a different level of abstraction in terms of robotic specification. The proposed approach allows the declarative specification of goal driven robot behavior. However in order to cope with the dynamics of mobile robotics the reasoning mechanism allows reactivity when needed. Syntax, semantics and interactions of the proposed architecture mental components are defined and studied.

Keywords: Mobile Robotics, Multi-Agent Systems, Agent Architectures.

1 Introduction

The development and implementation of any type of computational system requires an adequate analysis of requirements that allows the definition of the system to model, restrictions imposed (be them on requirements or on the specification), and performance parameters by which the behavior of the system can be evaluated. Robotic soccer is a way of putting different developments in intelligent agents into practice. This includes developments in autonomous, cooperative, competitive, reasoning, learning, and revision systems [15, 9]. Furthermore, it is useful in identifying problems related to aspects concerned with vision and communication. This type of problems cannot be all taken into account beforehand, and therefore demand that the system be designed to be robust enough to recover from eventualities of this type. Robotic soccer is a complex domain, and it is necessary to take into account several aspects related to the robots. Each robot has sensors and effectors which are prone to failure. The environment is dynamic so there is no chance of knowing in advance the situations that can arise in a game. Therefore, it is necessary to be able to recover from adverse situations like sensorial or effectorial failure, and the decisions needed to carry out the recovery process have to be taken quickly.

As mentioned above, mobile robots involved in complex environments, such as robotic soccer, require high degree of intelligence integrated with lower level capabilities. Robots should be able to react quickly in a highly dynamic environment and also to reason about strategies and robot behavior at a high level. Therefore, a robot may need to decide which action (move forward, rotate, etc.) execute next, in order to progress in the desired direction. For example, it may decide to move forward if it is facing the desired location, or it may decide to rotate in order to end looking the goal (desired location). However, it may

ISSN: 1988-3064(on-line)

also need to decide among different strategies or possible high level behaviors. As an example, a robot may need to decide whether to try to go after the ball or to stay in a defensive position, among other possibilities.

The system designed and implemented for controlling the robots need to be aware of the environment characteristics, reaction capabilities and robot behavior requirements. Therefore, the system controlling the robots need to be able to combine reactive response with high level behavior.

In this work, we present the design used as the basis for a Multi-Agent System implemented for the control of a team of robots for the VI Argentine Championship of Robot Soccer (CAFR 2008 [1]). The implementation of the system was carried out following a layered design; the objective of this design is to have a set of Service Layers, each of which is associated with a different level of abstraction. Each layer solves a different set of problems by means of the services that it offers; services of a layer can then be used by the upper layers.

In particular, we will present and formally define a BDI architecture for high level control of mobile robots. The proposed formalism will be used to model the upper layer of the layered system. This architecture uses a logic-based model for knowledge representation and reasoning. The model provides a useful tool to declaratively design and implement agent prototypes and reasoning systems. As a BDI specification, each mental component will provide benefit to the whole reasoning process. Beliefs will provide a way to reason with perceptual information (obtained from lower layers) using a logic programming mechanism. Desires will allows a declarative way to express high level strategies, and to select between them in an elegant way. Intentions will allow the developer to specify the situations where a plan is applicable in a declarative manner, and use internal state as part of the reasoning process to handle history. We will also provide the architecture with a set of reactive rules that allows the agent to work correctly in critical situations were fast computation and reflexive actions are needed.

2 Background

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 research was focused on those high and lower level areas. In particular, we have developed an obstacle avoiding system [13], research on sensorial information and basic movements [11], and a multi-agent architecture to control the robots [8]. We also have developed a robotic soccer team that participated in the E-League held in Robocup 2004.

We designed a new Multi-Agent System 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 to participate in the VI Argentine Championship of Robot Soccer (CAFR 2008 [1]). The main goal of the leagues we participated in 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 and concentrate on the development and study of Artificial Intelligence techniques, as the ones mentioned above. 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.

Each team has one or more auxiliary computers in which the agents are executed. These agents communicate with the vision component in order to obtain information about what happens on the field, and send messages to the robots by means of a communication module. Even though the league does not define a standard platform for the construction of the robots, it does impose restrictions over the

processing and memory capacity. This allows the use of low cost robotic kits, many of which fall under these restrictions. The system we developed was implemented using Lego Mindstorms kits, which are within the rules of the league. Each team is composed of three or four robots, one of the robots acting as a goalkeeper. There are restrictions over the size of the robots, their shape, and the components used in their construction. Even though the robots do not communicate amongst themselves, the processes that control them can do so.

In the following section we will describe a multi-agent system designed to cope with these requirements, also trying to facilitate the testing of new theories developed on reasoning, knowledge representation, planning, agent communication, among others AI techniques.

3 Design of the agent system

The proposed design considers the construction of the system based on an hybrid architecture combining reaction with deliberation [14]. In particular our design is based in a very popular approach, the three layer hybrid architecture (see Figure 1), 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.

As it will be explain next, each layer provides services to the upper layer. These services are implemented using services provided by the lower layer. Therefore, the design or implementation of each service can be modified without provoking considerable changes in the client layers.

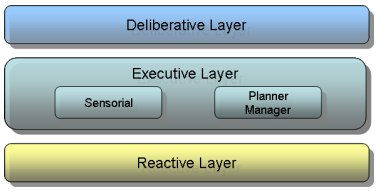


Figure 1: architecture used for the implementation of the Matebots team.

3.1 Reactive layer

The reactive layer implements the basic actions that the robot needs 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, communication network, and common software. The software that is provided by the league includes video and command communication servers.

This layer provides an array of basic actions to the executive layer in order to allow the construction of specific sequences of actions with a higher level goal, such as going to the ball. Basic actions of this layer include three generic movements: moving forward and backward, rotating clockwise or counterclockwise and describing different kinds of arcs. These actions may also vary in the velocity of the wheels, allowing the robots to execute movements with different speeds and precision.

The video server, called *Doraemon* [3], is part of the software support provided by the league. A video camera covers the play field and this server processes the images that it obtains, generating information packets that are then made available to be used by the agents that control both teams' robots. The packets that are generated by the video server provide information about the objects that are on the play field. Information about the position, orientation, and speed of these objects (the robots and the ball) is transmitted. This information will be captured and processed by the executive layer, as we will explain in the following subsection.

The league also provides a command communication software called *Command Server* (abbreviated CS from now on), which allows the agents controlling the robots to send messages to the robots in the field. As we have mentioned, the processing and memory capabilities of the robots is limited, and the control software must therefore reside and execute on auxiliary computers. In this way, the decision processes are carried out by these agents in the upper layers, and the decisions are then communicated to the robots through the CS. The decisions communicated to the robots are the actions to execute, which are implemented in the program running inside the robots. The frequency by which an agent can send actions to the robots through the CS is limited by the physical characteristics of the transmission method used by the robots, in our case Lego Mindstorms kits using IR.

3.2 Executive Layer

This layer is divided in two sub-layers, the *sensorial sub-layer* and the *planner manager sub-layer*. In the former sub-layer, the visual information is processed and translated into information that express states of the world. This information is divided in basic information and inferred information. The latter sub-layer provides a set of implemented schematic plans allowing the *deliberate layer* to control the robot behavior without worrying about low level details.

The entire layer is implemented as an interface between the *reactive layer* and the *deliberate layer*, and has been developed in the C programming language providing PROLOG predicates for the *deliberate layer*. This decision has several advantages. As we mentioned before, the environment is highly dynamic, which causes the states of the world to change quickly. Therefore, it is necessary for the robots to be able to react accordingly to this dynamism. Moreover, the information obtained by the robots can be wrong due to sensorial failure; after recovering from such a failure, the current situation could be completely different from the one previously perceived. In this type of cases, the system has to be able to analyze new situations, and quickly decide which actions should be taken by the robots.

The *sensorial sub-layer* provides basic information (such as the coordinates and speeds of the robots and the ball), and also inferred information (elements obtained using inference rules that contain the basic information), such as: the robots' and the ball's locations relative to the field, player or team closest to the ball, distances between different objects on the field (between players, rival players, etc). This information is provided to the *deliberate layer* as PROLOG predicates, which are implemented by querying and analyzing the information from the video server. Then, the situations modeled through these predicates are used in the *deliberate layer* to model and implement the team's game strategy.

The *planner manager sub-layer* is responsible for the execution of schematic plans. 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, such as rotation and forward and backward movements, and each of these actions depend on the state of the field at the moment immediate before of been executed. The schematic plans implemented are: go to a given object (such as another robot or the ball), pass the ball to a teammate, go to a zone in the field, kick the ball, dribble to a given location, and clear the ball out of the defensive zone.

The existence of this layer allow us to disregard the physical structure of the environment in which the team of robots is embedded. For example, it is possible to implement the services of this layer based on a simulated environment like the *FIRA SimuroSot*. If the interface of the services offered by this layer remain unchanged, then the rest of the *deliberate layer* can also remain unchanged.

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, as in any other agent system, agents will have a deliberative cycle where they perceive information about the environment and reason/decide which the *schematic plans* execute. The implementation of the *executive layer* allows to specify the elements of the cognitive layer using different knowledge representation and reasoning systems. One alternative is to use the Belief-Desires-Intentions (BDI) model [6] to implement agents controlling the robots. In particular, we developed a team using this architecture in order to participate in the VI

Argentine Championship of Robot Soccer (CAFR 2008 [1]). In the following section we will present a detailed explanation of how the BDI architecture is used in our approach.

4 Architecture description for high level reasoning

One of the most promising agent architecture for developing intelligent agents is BDI. The BDI model is recognized and studied in the literature [7], and involves two aspects in the area of intelligent agents: a solid agent theory and a concrete agent architecture. The former is a well known agent logic, based on the theory of practical reasoning presented in [5]. The latter is an architecture [6],[12], based on the agent logics of [5], for building agents that are able to reason about their environment and deliberate about their actions. A BDI agent is specified using the mentalistic attributes Belief, Desires and Intentions, and uses these components to determine the agent state. Thus, an agent will have a belief set containing information about the environment in which it is involved, a desire set containing all the possible goals the agent have and a set of intentions containing all the possibilities the agent has to achieve its desires.

In this section we will show how to integrate the concepts of the BDI architecture with the lower level layers. Next, we will show the role of each mental components in our approach and introduce their syntax, semantics, and interactions. In order to do so, we will introduce the following working example, which describes a particular situation in the match.

Example 1 *Consider an agent belonging to the blue team is playing against the yellow team (see figure 2). Its team is loosing the match 0 - 2. Currently the agent (labelled "me") is moving towards the opponent area and it is located in the left part of middle field. Its teammate "b1" is located in the right part of the middle field and carries the ball. One of its opponents "y1" is located in the right part of the middle field and the other opponent"y2" is located in the center of the yellow team penalty area.*

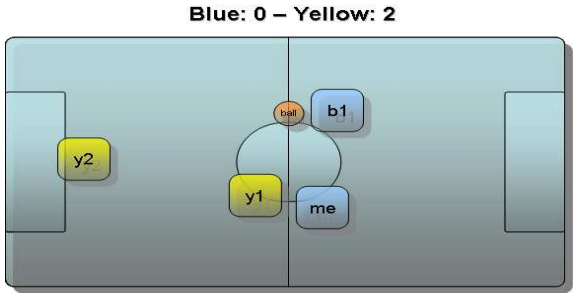


Figure 2: Situation of example 1

4.1 Beliefs

The beliefs of an agent in the BDI model are used to represent the situation of the world that the agent is in. In this approach, beliefs represent all the information the agent obtains from the field in each deliberative cycle. This information is divided in basic information (elements directly perceived from the field like X,Y coordinates) and inferred information (elements obtained using inference rules that contain the basic information, like the best place to attack).

A mobile robot should be able to work in dynamic environments with incomplete knowledge of them. Our particular domain involves incomplete information about the objects in the field and a degree of uncertainty of the effects of their actions. Therefore, the robots will need a formalism to reason about this incomplete, uncertain, and changing information. In order to address those issues, we will use a logic programming formalism to obtain, in each deliberative cycle, the set of elements the agent believes in. In particular will be used PROLOG as the knowledge representation language and as belief inference

mechanism. This introduces interesting advantages for the agent developer such as declarative belief specifications and non-monotonic reasoning.

In our approach, agent's beliefs will be represented by a PROLOG program called belief base. This base can contain sensed information, such as field (X,Y) coordinates of an object, distances between objects, direction that a robot is facing, or whether 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* $\mathcal{P}_b$=(ϕ, π, δ), *where* ϕ *is a set of* PROLOG *facts used to represent sensed information,* 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 2, there the agent from Example 1 will have the following set of sensed information* ϕ_1={*hasBall(b1), pos(middle, right, b1), pos(middle, left, me), pos(middle, left, y1),pos(opPenBox, center, y2), loosing*}, *as strict information will will have the following set* π_1={ *teammate(b1), opponent(y1), opponent(y2)* }, *and suppose that has the following inference rules (sketch):*

$$\delta_1 = \left\{ \begin{array}{l} offensiveZone(opPenBox, Opdir) : -teamMate(X), pos(Zone, Dir, X), oposite Dir(Dir, ODir) \\ defensiveZone(Zone, Opdir) : -teamMate(X), pos(Zone, Dir, X), oposite Dir(Dir, ODir) \\ blocked(X) : -pos(Zone, Dir, me), pos(Zone, Dir, X), opponent(X) \end{array} \right\}$$

Finally, its belief base will be $\mathcal{P}_b = (\phi_1, \pi_1, \delta_1)$

The fact set ϕ is updated every deliberative cycle by the sub-sensorial layer. Since this sensorial information will be surely used by the inference rules, the inferred information will also change every deliberative cycle. Therefore, in order to allow other mental components to use them, we will next define which elements are obtained from the $\mathcal{P}_b$ in a deliberative cycle.

Definition 2 (actual belief) *Let* $\mathcal{P}_b$=(ϕ, π, δ) *be a belief base of an agent, The* PROLOG *atom h is an* actual belief *of* $\mathcal{P}_b$ *(noted* $\mathcal{P}_b \vdash h$*) 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* $\mathcal{P}_b$ *it will be noted* $\mathcal{P}_b \nvdash h$

Example 3 *The belief base presented in Example 2 will have the following actual beliefs: offensiveZone(opPenBox,right), defensiveZone(middle,right), blocked(me) 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. The computation of an actual belief will be made by demand of the requesting component. Thus, the agent will be able to reason using a non-monotonic approach and act reactively in a efficient way since the belief elements will only be calculated when needed without adding overhead to the overall deliberative cycles.

4.2 Desires

The 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. The agent will have a number of desires that it will be continuously trying to achieve. However, depending on the situation in which it is involved, there are desires that 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. Therefore, agents should reason about their desires to select the ones that could be actually justified or selectable. 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 triplet* DR=($d, Just, Imp$), *where d is an atom representing a desire, and* $Just$={p_1,..., p_n, *not* c_1,..., *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, and* Imp *is a number denoting the importance value of the rule desire.*

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 $\mathcal{P}_d$. Note that the set $D = \{\bigcup d \mid (d, Just, Imp) \in \mathcal{P}_d\}$ contains all the possible desires that the agent have.

Example 4 *The agent desire base(sketch) is*

$$\mathcal{P}_d = \left\{ \begin{array}{l} (attack,\{hasBall(me),\ loosing\},20) \\ (attack,\{hasBall(X),teamMate(X),\ loosing\},15) \\ (defend,\{hasBall(X),teamMate(X)\},5) \\ (defend,\{hasBall(X),opponent(X)\},20) \end{array} \right\}$$

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

Definition 4 (justified desire) *Let $\mathcal{P}_b$ be a belief base, a desire rule $DR = (d, Just, Imp)$, where $Just$=$\{p_1,\ldots,\ p_n,\ not\ c_1,\ldots,\ not\ c_m\}$ 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 $\forall i = 1..n\ \mathcal{P}_b \vdash p_i$ and $\forall j = 1..m\ \mathcal{P}_b \nvdash c_j$*

Example 5 *Suppose that an agent is in the situation described in Figure 2, the belief base of Example 2 and has the desires rules of Example 4 then it will have the following justified desires: [attack,15] and [defend,5].*

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

This desire oriented approach allows a declarative way to express high level strategies, and to select among them in an elegant way. Moreover, it allows developing agents that behave depending the match attitudes, which are based on the high level information provided by the belief base.

4.3 Intentions

The intentions in the BDI architecture specify how the agent can achieve its justified desires. In this work, intentions will represent the plans that the agent can execute to reach its desires. For instance if the agent desires to attack, it can have intentions to plan a dribble to the opponent area, plan a shoot to make a goal or just plan a move to the opponent area. Thus, the developer will be able to specify different alternatives or intentions to achieve a desire. However, depending on the situation in which the agent is involved, there are intentions that can be applicable or not. For instance, when the agent does not has the ball its not possible to dribble or shoot. Once that the agent determines which intentions are applicable, it should commit to one of them and execute the plan. The intentional model of the agent should contemplate that there will be an executing intention. Thus, intention applicability reasoning process will involve actual beliefs, the justified desires and the running plan. In order to specify this intentional behavior we will introduce the intention rules.

Definition 5 (intention rule) *An intention rule IR is an ordered tuple IR=(κ, β, cp, π), where: κ (header goal) is an atom representing the desire to be achieved, cp is an atom representing the running plan precondition, π is a plan to be executed, and β=$\{p_1,\ldots,\ p_n,\ not\ c_1,\ldots,\ not\ c_m\}$ ($n \geq 0$ and $m \geq 0$) is a guard 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 restrictions for IR. An intention-rule (κ, β, cp, π) is also denoted $\kappa \leftarrow \beta\ [cp] \mid \{\pi\}$. All the specified intention rules determine the agent* intention base *noted $\mathcal{P}_i$.*

Example 6 *The agent intention base(sketch) is*

$$\mathcal{P}_i = \left\{ \begin{array}{l} attack \leftarrow hasBall(me), position(opPenBox, Dir, me)\ [] \mid \{shoot\} \\ attack \leftarrow hasBall(me)\ [] \mid \{dribbleTo(opPenBox, Dir)\} \\ attack \leftarrow offensiveZone(Zone, Dir)\ [] \mid \{goto(Zone, Dir)\} \\ defend \leftarrow defensiveZone(Zone, Dir)\ [] \mid \{goto(Zone, Dir)\} \\ \leftarrow pos(myPenArea, X, me), hasBall(me)\ [] \mid \{clear\}\ \ \} \end{array} \right\}$$

For instance, the first rule specifies that if the agent wants to attack, it has the ball, and it is near the opponent penalty area, it will be adequate to shoot to the goal.

Note that the plans of the IR of the previous example are those described in Section 3.2. Using these rules, an agent will be able to determine which intentions are applicable each deliberative cycle. Next, we will how to determine which intentions are applicable.

Definition 6 (applicable intention) *Let $\mathcal{P}_b$ be a belief base, JD the set of justified desires, and IR is an intention rule $\kappa \leftarrow \beta\ [cp] \mid \{\pi\}$ where β=$\{p_1,\ldots,\ p_n,\ not\ c_1,\ldots,\ not\ c_m\}$ with $n \geq 0$ and $m \geq 0$. Then IR will be* applicable *iff $\forall i = 1..n\ \mathcal{P}_b \vdash p_i$ and $\forall j = 1..m\ \mathcal{P}_b \nvdash c_j$, $\exists[\kappa, Imp] \in JD$, and cp is the running plan. From an applicable intention rule, the applicable intention will be a pair (π, Imp)*

Example 7 *Suppose that an agent is in the situation described in Figure 2, it has the belief base of Example 2 and it has the justified desires of Example 5. Then, it will have the following applicable intentions: (goto(opPenArea,right),15) and (goto(middle,left),5) since the third and the fourth intention rule are applicable.*

The test if *cp* is the plan the agent is currently executing is made by an interaction with the executive layer. Note that a rule can have an empty *cp*, this means that the rule does not has a running plan precondition, and in this case the condition over the running plan is always met. Also an intention rule can have an empty κ (a reactive rule), this means that the rule does not have a desire to achieve, and will be used to represent reactive reasoning. In this case the justified desire condition is always met. The intention reactive rules will be treated differently from the other intention rules in the deliberative cycle in order to benefit agent reactivity. The set of all applicable intentions obtained from non reactive rules will be called AI.

Since only one intention from AI should be selected to execute, we will introduce one possible criterion to select one: Let $[IR1, ..., IRn]$ be the sequence of the applicable intention rules of the agent, the order of this sequence is determined by the rule order in the agent specification. First, select those elements of AI with higher value of Imp. If there are more than one element that meet the first condition, select then the element obtained from the first intention rule appearing in $[IR1, ..., IRn]$ between those elements that meet the first condition. For instance, In Example 7, the selected intention will be *(goto(opPenArea,right),15)*

Once an applicable intention is selected, its plan will be sent to the executive layer to be executed. In the case that the sent plan is the same as the one running in the executive layer, it will be ignored and the layer will continue the execution of the running one.

As already mentioned, intention reactive rules will be treated differently. The applicability of the reactive rules will be computed at the start of the deliberative cycle and the first applicable rule (using the specification of the agent as order) will determine s the plan sent to execution.

Intention rules allow the developer to specify the situations where a plan is applicable in a declarative manner. Using the running plan as part of the conditions to determine that an intention rule is applicable allows the agent to use its internal state as part of the reasoning process, and to handle history. The reactive rules will allow the agent to work correctly in critical situations were fast computation is needed and a quick reflexive action is needed.

4.4 Deliberative Cycle

In this section we will show how the mental components described previously interact each other. In Figure 3, the component interaction and the connection with the executive layer is shown.

Next, we will show the deliberative cycle of the BDI architecture proposed in this work. The main difference with the basic BDI deliberative cycle is that in our approach the intention revision is handled inside the intention selection semantics.

1. update perceived elements ϕ in the $\mathcal{P}_b$
2. if a reactive intention rule is applicable using only *actual beliefs* and the *running plan*, goto step 5
3. obtain the *justified desires* from $\mathcal{P}_d$ using the *actual beliefs*
4. obtain the applicable intentions using the *justified desires*, the *actual beliefs* and the *running plan*.
5. select one intention from the previous step using a intention selection criterion.
6. send the selected intention plan to the executive layer and set it as the running intention.
7. goto step 1.

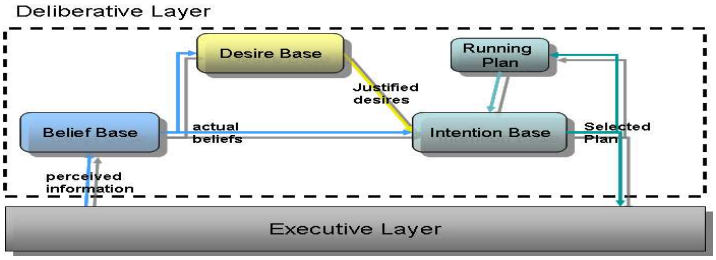


Figure 3: Mental components and Layer interaction

This deliberative cycle ensures a sophisticated goal oriented reasoning using the justified desires to determine the best strategies or game attitudes. The intention rules give different alternatives to achieve one of the justified desires. The definition of those elements and the deliberative cycle are combined in order to allow the agent developer to abstract of the reasoning mechanisms and concentrate on the match situation that trigger a game attitude and its respective plan. Furthermore, the combination of reactive intention rules, its selection policy and the fact that actual beliefs are computed "by demand" allows the agent to act reactively when needed, which is very important in robotics.

5 Conclusions and Future Work

In this work we have presented and formally defined a BDI architecture for high level control of mobile robots. This architecture is built on top of a layered system. This layered system gives the architecture the necessary level of abstraction to do cognitive reasoning. This design allows the abstraction and modularization of different aspects of the complex domain that robotic soccer represents.

The design of the knowledge representation and reasoning mechanisms was carried out using a logic-based model. Logic programming is a useful tool for the implementation of reasoning systems that offer the possibility of rapid prototyping of agents and declarative design. The desire oriented approach allows a declarative way to express high level strategies, and to select among them in an elegant way. The intention rules allow the developer to specify the situations where a plan is applicable in a declarative manner. Using the running plan as part of the conditions to determine that an intention rule is applicable, allows the agent to use its internal state as part of the reasoning process, and to handle history. The reactive rules will allow the agent to work correctly in critical situations were fast computation and reflexive action are needed.

Having built a functional team of agents using this model to participate in the CAFR 2008 competition, we are currently working on the development of more complex agents that incorporate a wide range of AI techniques developed within LIDIA from the fields of planning, argumentation, learning, belief revision, among others.

Acknowledgements

This research is partially supported by CONICET (PIP 5050), Universidad Nacional del Sur and Agencia Nacional de Promoción Científica y Tecnológica.

References

[1] http://www.uncoma.edu.ar/cafr2008/. Official webpage of the VI Argentine Championship of Robot Soccer.

[2] S. Achim, P. Stone, and M. Veloso. Building a dedicated robotic soccer system, 1996. In Proceedings of the IROS-96 Workshop on RoboCup.

[3] John Anderson and Jacky Baltes. Doraemon user's manual. http://sourceforge.net/projects/robocup-video.

[4] John Anderson, Jacky Baltes, David Livingston, and Elizabeth Sklar. Toward an undergraduate league for robocup. In *Proceedings of the RoboCup Symposium*, 2003.

[5] M. Bratman, D. Israel, and M. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(4):349–355, 1988.

[6] 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. The MIT Press.

[7] Michael Fisher, Rafael Bordini, Benjamin Hirsch, and Paolo Torroni. Computational logics and agents: A road map of current technologies and future trends. *Computational Intelligence*, 23(1):61–91, February 2007.

[8] Alejandro J. García, Gerardo I. Simari, and Telma Delladio. Designing an agent system for controlling a robotic soccer team. In *Proceedings of X Argentine Congress of Computer Science*, page 227, Buenos Aires, Argentina, 2004. Universidad Nacional de la Matanza.

[9] Kwun Han and Manuela Veloso. Automated robot behavior recognition applied to robotic soccer. Robotics Research: the Ninth International Symposium, pages 199–204. Springer-Verlag, London, 2000. Also in the Proceedings of IJCAI-99 Workshop on Team Behaviors and Plan Recognition.

[10] Benn Vosseteig Jacky Baltes and John Anderson. Robocup e-league video server. http://sourceforge.net/projects/robocup-video.

[11] Fernando Martín, Mariano Tucat, and Alejandro J. García. Soluciones a problemas de percepción y acción en el dominio de un equipo de fútbol de robots. In *Proceedings of X Argentine Congress of Computer Science*, pages 1895–1906, Buenos Aires, Argentina, 2004. Universidad Nacional de la Matanza.

[12] S. Rao and Michael P. Georgeff. Bdi agents: From theory to practice. In *In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95*, pages 312–319, 1995.

[13] Nicolás Rotstein and Alejandro J. García. Evasión de obstáculos con bajo costo computacional para un equipo de fútbol de robots. In *Proceedings of X Argentine Congress of Computer Science*, Buenos Aires, Argentina, 2004. Universidad Nacional de la Matanza.

[14] S. Russel and P. Norvig. *Artificial Intelligence: a Modern Approach*. 1995.

[15] Gerhard Weiss. Learning to coordinate actions in multi-agent systems. In *Reading in Agents*, pages 481–486. Morgan Kaufmann, 1998.