Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games (CIG 2007)

Fuzzy Prolog as Cognitive Layer in RoboCupSoccer

Susana Muñoz-Hernandez
Dept. of Languages and Information Systems and Software Engineering
Technical University of Madrid, Spain
susana@□.upm.es

Wiratna Sari Wiguna
School of Computer Science
Technical University of Madrid
wiratna@gmail.com

Abstract— **RoboCupSoccer domain has several leagues which varies in the rule of play such as specification of players, number of players, field size, and time length. Nevertheless, each RoboCup league is a variant of a soccer league and therefore they are based on some basic rules of soccer. A layered design of agents system presented in [1] shows a modular approach to build control for a team of robots participating in RoboCupSoccer E-League. Based on this design, we propose a generalized architecture offering flexibility to switch between leagues and programming language while maintaining Prolog as cognitive layer. Prolog is a very convenient tool to design strategies for soccer players using simple rules close to human reasoning. Sometimes this reasoning needs to deal with uncertainty, fuzziness or incompleteness of the information. In these cases it is useful Fuzzy Prolog [2], [3], [4], [5]. In this paper we propose to use a combination of Prolog (that is crisp) and Fuzzy Prolog to implement the cognitive layer in RoboCupSoccer, which has the advantage of incorporating as conventional logic as fuzzy logic in this layer. A prototype of a team based on this architecture has been build for RoboCup Soccer Simulator, and we show that this approach provides a convenient way of incorporating a team strategy in high level (human-like) manner, where technical details are encapsulated and fuzzy information is represented.**

Keywords: Logic Programming, Constraint Logic Programming Implementation, Fuzzy Reasoning, Prolog Application, RoboCupSoccer, Cognitive Layer.

I. Introduction

The idea of robot playing soccer has been developed since early 90s [6]. Soccer environment is a dynamically changing environment which requires individual skill as well as team skill and therefore is an interesting research □eld on Arti□cial Intelligence and robotics. Prolog is a programming language that represent logic reasoning. Is is a perfect tool to represent human reasoning, so it seems to be a good choice for implementing the cognitive layer of soccer players that is a simulation of human behaviour related to this game. For example, applying the rule "if the goal keeper is not at the goal then kick to ball". But many of the most important decisions that are made by soccer players deal with non-crisp issues. They are related to fuzziness (e.g. "if other player of my team is FAR from me then don't pass him/her the ball"), uncertainty (e.g. "if I CAN get the goal then kick the ball"), or incompleteness (e.g. "if I cannot see the position of a player, by default I'm not going to pass him the ball"). Fuzzy Prolog is an attempt to introduce fuzzy reasoning into logic programming that also deals with uncertainty and incompleteness. It is aimed of this proposal to combine the advantages of these different types of programming and to show how to handle this combination.

There are many works that have been done on this research area related to RoboCup [1], [7], [8] and to Fuzzy Prolog [2], [3], [4], [5]. This work is the continuation of the research line of the project [9].

The rest of the paper is organized as follow. Next section gives brief overview on RoboCupSoccer and section 3 describes Fuzzy Prolog. Section 4 and 5 talks about our approach and its evaluation. Section 6 concludes this paper and mentions some further works.

II. RoboCupSoccer

RoboCup is an international annual event promoting research on Arti□cial Intelligence, robotics, and related □eld. The original motivation of RoboCup is RoboCupSoccer. As the nature of soccer game, autonomous robots participating in RoboCupSoccer should have individual ability such as moving and kicking the ball, cooperative ability such as coordinating with teammates, and of course, the ability to deal with dynamic environment.

RoboCupSoccer consists of several leagues, providing test beds for various research scale:

- Simulation League
  RoboCup Simulation League consists of a number of simulated soccer matches as the main event. There are no actual robots in this league, and the matches are shown on large screen. A team consists of several computer programs which act as players. The game consists of two 5-minutes halves.
- Small Size Robot League (F-180)
  In Small Size Robot League each team consists of □ve small sized robots, one of them might be the goalkeeper. The team might have a global vision system with one or two camera mounted above the □eld, or local vision for each robot. The game lasts for two equal periods of 15 minutes.
- Middle Size Robot League (f-2000)
  Each team in Middle Size Robot League consists of six mid sized robots with all sensors on-board, one of whom is the goalkeeper. The rule is changed continuously every year, aiming to be closer to a real soccer match. A match in RoboCup Middle Size Robot League lasts for two equal periods of 10 minutes.

- Four-Legged Robot League
  The Four-Legged Robot League teams consist of four Sony AIBO robots with all sensors on-board, one of whom is the goalkeeper. No modifications or additions to the robot hardware are allowed. Event organizers for this league will provide a computer for sending Game-Controller messages to the robots. The game consists of two 10-minutes halves. Apart from the soccer game, this league also includes challenges to encourage research within Four-Legged Robot League.
- Humanoid League
  Humanoid League has been included in RoboCup competition since 2002, but only at 2005 soccer game with two humanoid robots per team was held for the first time. Humanoid robots are autonomous robots with human-like body plan and human-like sense. They are grouped into two size classes, KidSize and TeenSize. Soccer match for this league lasts for two equal periods of 10 minutes. Penalty kick competitions and technical challenges are also included in this league.
- E-League
  The first E-League was held at RoboCup 2004. This league is aimed as a bridge from RoboCupJunior to RoboCupSoccer, to accommodate undergraduate students which are too old for RoboCupJunior but do not have enough resources to join leagues in RoboCupSoccer. This league is a simplified version of Small Size Robot League, where vision processing and communications are factored out, thus provided by the league. Each team in this league consists of four small sized autonomous robots, one of whom can be a goalkeeper. The match lasts for two equal periods of 10 minutes.
- RoboCup Commentator Exhibition
  Participants of RoboCup Commentator Exhibition are a number of systems which automatically generate soccer commentary for simulation league games. The comments are in natural language and they are real-time generated. In RoboCup Commentator Exhibition, the goal is to observe and comment.

Our work is part of a joint research project [9] on RoboCup-Soccer E-League with the National University of Comahue (Argentina). However as a preliminary work, we employ RoboCupSoccer Simulation League for the sake of simplicity.

III. Fuzzy Prolog

The Ciao Prolog System offers a complete Prolog system supporting ISO-Prolog. Its modular design allows restriction and extension of the language both syntactically and semantically. The Ciao Prolog Development System provides many libraries including a constraint logic programming system and interfaces to some programming languages. In Ciao Prolog terminology, a library is implemented as either a module or a package. Fuzzy Prolog described in [2] and [3], [5] is implemented as the package "fuzzy.pl", a syntactic extension of the CLP($\mathcal{R}$) system in the Ciao Prolog System. This is a continuous variant of Fuzzy Prolog.

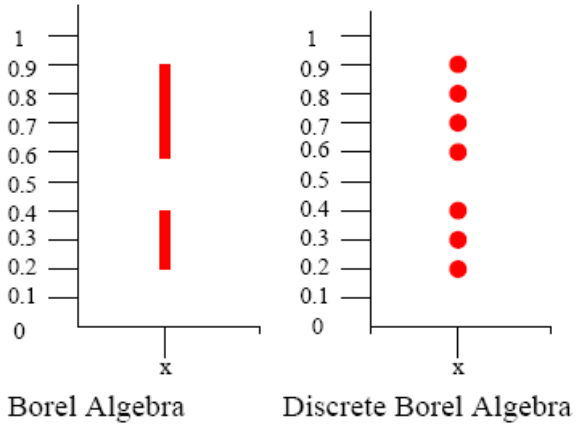

Fig. 1. Truth Value: Borel Algebra versus Discrete Borel Algebra [4]

We use in this work the discrete variant of Fuzzy Prolog that is implemented using CLP($\mathcal{FD}$) as described in [4]. It offers an implementation of a Fuzzy Prolog system with discrete (versus continuous) truth values. The next subsection summarizes the basic formal concepts that are described by the semantics of this language.

A. Discrete Fuzzy Prolog Language

The set of continuous subintervals on [0,1] is denoted by $\mathcal{E}([0,1])$. The Borel Algebra, $\mathcal{B}([0,1])$, is the power set of $\mathcal{E}([0,1])$. We talk about discrete instead of continuous interval when it is compound by a finite set of elements included in the corresponding continuous interval. The set of discrete subintervals on [0,1] is denoted by $\mathcal{E}_d([0,1])$. We call $\mathcal{B}_d([0,1])$ the Discrete Borel Algebra over the interval [0,1] for representing the set of finite unions of discrete subintervals on [0,1].

As defined in [4], truth values in discrete Fuzzy Prolog are elements of Discrete Borel Algebra over the interval [0,1]. Fuzzy sets are defined by functions of the form $A : X \rightarrow \mathcal{B}_d([0,1])$. The Fuzzy Prolog system with Borel Algebra as in [2] and [3], [5] is often referred as continuous Fuzzy Prolog system. Figure 1 illustrate the difference between Borel Algebra and Discrete Borel Algebra.

Notice that the truth value representation of Fuzzy Prolog and discrete Fuzzy Prolog is very general (union of intervals of real numbers) and it can seem to be little intuitive. The applications of this generality are disscussed in [2] and [3], [5]. Simple truth values (as a unique interval or a plain real number) can be more adequate to be used in RoboCup game. These simple values are particular cases of the general framework.

Definition 3.1 (discrete-interval): A discrete-interval $[X_1, X_N]_\epsilon$ is a finite set of values, $\{X_1, X_2, ..., X_{N-1}, X_N\}$, $0 \leq X_1 \leq X_N \leq 1$ such that $\exists\ 0 < \epsilon < 1$. $X_i = X_{i-1} + \epsilon, i \in \{2..N\}$.

As indicated in the definition, the set of values of a discrete-interval depends on the choice of ϵ. Smaller ϵ value represents more precision. For example, an interval $[0.3, 0.5]_{0.1}$ with $\epsilon = 0.1$ is a finite set {0.3, 0.4, 0.5} while the interval $[0.30, 0.50]_{0.01}$ with $\epsilon = 0.01$ is a finite set {0.30, 0.31, 0.32, ..., 0.48, 0.49, 0.50}.

Therefore we denote the algebras $\mathcal{E}_d([0,1])$ and $\mathcal{B}_d([0,1])$ incorporating the granularity to the notation substituting d by the particular ϵ (e.g. $\mathcal{B}_{0.1}([0,1])$).

Definition 3.2 (discrete-aggregation): Discrete-aggregation (in fuzzy sets) is the application of a numeric discrete-aggregation operator (or discrete-aggregation) of type $f : [0,1]^n \rightarrow [0,1]$. If it satisfies $f(0, ..., 0) = 0$ and $f(1, ..., 1) = 1$, and in addition it is monotonic.
Notice that the operator is only monotonic by definition, not continuous (that is why it is called "discrete").

Definition 3.3 (discrete-interval-aggregation): Given a discrete-aggregation $f : [0,1]^n \rightarrow [0,1]$, a discrete-interval-aggregation $F : \mathcal{E}_\epsilon([0,1])^n \rightarrow \mathcal{E}_\epsilon([0,1])$ is defined as follows:

$$F([x_1^l, x_1^u]_\epsilon, ..., [x_n^l, x_n^u]_\epsilon) = [f(x_1^l, ..., x_n^l), f(x_1^u, ..., x_n^u)]_\epsilon$$

where $0 < \epsilon < 1$.
Intuitively we can say that F provide a discrete-interval from the aggregation of n discrete-intervals.

Definition 3.4 (discrete-union-aggregation): Given a discrete-interval-aggregation $F : \mathcal{E}_\epsilon([0,1])^n \rightarrow \mathcal{E}_\epsilon([0,1])$ defined over discrete-intervals, a discrete-union-aggregation $\mathcal{F} : \mathcal{B}_\epsilon([0,1])^n \rightarrow \mathcal{B}_\epsilon([0,1])$ is defined over union of discrete-intervals as follows:

$$\mathcal{F}(B_1, ..., B_n) = \cup\{F(\mathcal{E}_{1\epsilon}, ..., \mathcal{E}_{n\epsilon}) | \mathcal{E}_{i\epsilon} \in B_i\}.$$

The alphabet of the fuzzy language consists of variables, constants, function symbols, and predicate symbols. A term is defined inductively as follows:

1) A variable is a term.
2) A constant is a term.
3) If f is an n-ary function symbol and $t_1, ..., t_n$ are terms then $f(t_1, ..., t_n)$ is a term.

An atom or atomic formula is defined as the following:
If p is an n-ary predicate symbol and $t_1, ..., t_n$ are terms, then $p(t_1, ..., t_n)$ is an atom.
A fuzzy program is a finite set of fuzzy facts and fuzzy clauses. Information is obtained from the fuzzy program through a fuzzy query.

Definition 3.5 (fuzzy fact): If A is an atom,

$$A \leftarrow v$$

is a fuzzy fact, where v, a truth value, is an element in $\mathcal{B}_\epsilon([0,1])$ and $0 < \epsilon < 1$.

Definition 3.6 (fuzzy clause): Let $A, B_1, ..., B_n$ be atoms,

$$A \leftarrow_F B_1, ..., B_n$$

is a fuzzy clause where F is a discrete-interval-aggregation operator of truth values in $\mathcal{B}_\epsilon([0,1])$, $0 < \epsilon < 1$, and F induces a discrete-union-aggregation as by definition 3.4.

TABLE I
DISCRETE FUZZY PROLOG SYNTAX FOR FUZZY FACT

Fuzzy Fact	Discrete Fuzzy Prolog Syntax
$p(john) \leftarrow 0.7$	`p(john,70) ::~.`
$p(peter) \leftarrow [0.4, 0.6]_{0.01}$	`p(peter,V) ::~ { V in 400 .. 600}.`
$p(joan) \leftarrow [0.2, 0.5]_{0.1} \cup [0.8, 1]_{0.1}$	`p(joan,V) ::~ { V in 20 .. 50}.` `p(joan,V) ::~ { V in 80 .. 100}.`

Definition 3.7 (fuzzy query): A fuzzy query is a tuple

$$v \leftarrow A?$$

where A is an atom, and v is a variable (possibly instantiated) that represents a truth value in $\mathcal{B}_\epsilon([0,1])$, where $0 < \epsilon < 1$.

B. Discrete Fuzzy Prolog Syntax

Constraint Logic Programming, CLP, is one of the most promising extensions of Logic Programming from the implementation point of view. There are many Prolog systems that implement it [10]. One of the most popular extensions is CLP($\mathcal{FD}$), Constraint Logic Programming over Finite Domains. It is related to constrain the set of possible values of the variables for efficiency. So, the possible values can be obtained according to a set of constraints that should be satisfied by each variable. In Ciao Prolog, CLP($\mathcal{FD}$) works on integer domain. We use it to represent the set of truth values of a fuzzy variable by a set of integer numbers. Therefore, given the ϵ as in definition 3.1, truth value is interpreted as a finite union of discrete subintervals on $[0,1]_\epsilon$. Hence, the discrete subinterval is a set of integers.

The interval $[X_1, X_N]_{1/k}$ is interpreted for CLP($\mathcal{FD}$) as the discrete interval $[X_1 * K * 10, X_N * K * 10]_1$

For example, the interval $[0.4, 0.6]_{0.01}$ is interpreted as the set {400, 401, 402, ..., 598, 599, 600} while $[0.4, 0.6]_{0.1}$ is interpreted as the set {40, 41, ..., 59, 60}. Table III-B shows the syntax for fuzzy facts. We use 0.1, 0.01, etc for simplicity, but any other value (e.g. 0.2, 0.34, ...) can be used also.

The fuzzy clause is defined as `Head ::~ Aggregator Body`. For example, the syntax for fuzzy clause

$slow\_dash \quad (Distance, Power) \leftarrow_{min} near(Distance), low\_dash\_power(Power)$

is in discrete Fuzzy Prolog

```
slow_dash  (Distance,Power,V) ::~ min
           near(Distance,V1),
           dash_power(Power,V2).
```

In Fuzzy Prolog syntax, the query is formulated as an atom A with v, the truth value, as additional parameter. The fuzzy query is defined similar with the ones in continuous Fuzzy Prolog. For example, the syntax for the fuzzy query that consults if running slowly is a good option taking into account the distance that the player want to cover and the power of his/her dash

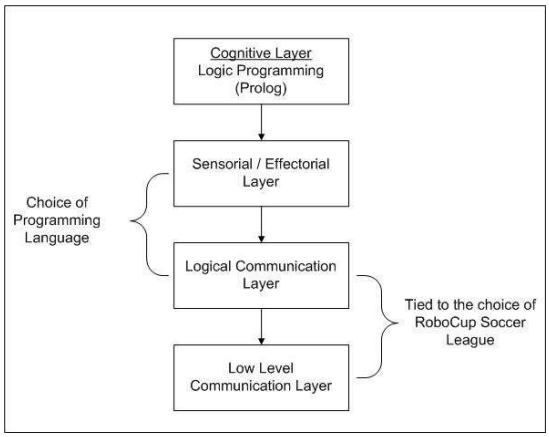

Fig. 2. Generic System Architecture

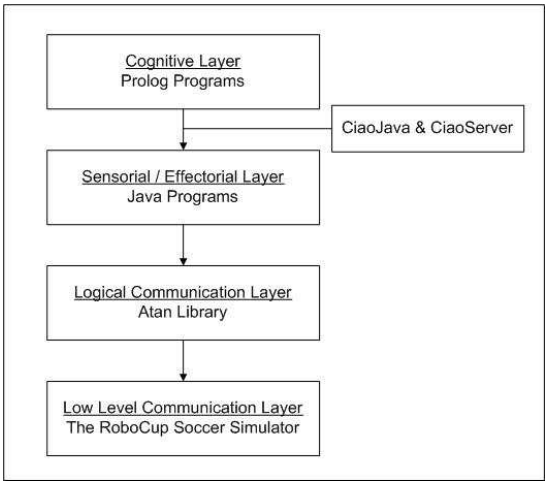

Fig. 3. System Architecture for RoboCup Soccer Server

$$v \leftarrow slow\_dash(Distance, Power)?$$

is written in Fuzzy Prolog:
`?−slow_dash(Distance,Power,V).`
The value of V is obtained from the aggregation (using the operator min) of the truth values $V1$ and $V2$. The value of V in Fuzzy Prolog with CLP($\mathcal{FD}$) ranges (values or intervals) between 0 and 100, 0 and 1000, ... (depending on the precision of the program and representing the corresponding truth values between 0 and 1 obtained dividing by 100, 1000, ... respectively). As CLP($\mathcal{FD}$) supports `labeling`, V could be instantiated to one or more values satisfying the constraints. This is the reason why discrete Fuzzy Prolog is more useful for this application than the continuous variant, because it provides constructive answers as values instead of providing constraints (as in the continuous variant).

Note that when more precision is needed, a suitable ϵ in the definition 3.1 could be chosen, which results in a larger set of integer between 0 and 1 for truth values.

IV. Architecture and Implementation Details

Based on agent system architecture proposed by [1], we propose a generic system architecture for RoboCup offering flexibility on choice of programming language and minimal modification to switch between leagues. This architecture is shown in figure 2. Prolog is proposed for cognitive layer, and in our work we use Fuzzy Prolog for implementing the cognitive layer. The system architecture of our implementation for RoboCupSoccer Simulation League is shown in figure 3. As it can be seen in the figures and as it is going to be described in this section, the generic architecture is customized in our implementation for Simulation League.

A. Low Level Communication Layer

As the name suggests, this is the lowest layer of our architecture. This layer includes all hardwares and softwares provided by the league. The robots, infrared transmitter, video camera, communication network, and vision systems belong to this layer. Different leagues in RoboCupSoccer are represented by different Low Level Communication Layer. E-League has the robots, Doraemon vision package, and communication server as part of this layer, whereas Simulation League has only The RoboCup Soccer Simulator as part of this layer.

B. Logical Communication Layer

This layer acts as the interface between low level communication layer and the upper layers. It is intended to hide physical structure of the environment from the upper layer. As long as the interface of the services offered by this layer remain unchanged, then the rest of the upper layer can also remain unchanged [1]. Basic services that should be offered for E-league are :

- Reading the packets generated by video server.
- Establishing communication with the communication server.
- Continuous sensing for the referee decision.

In our implementation for Simulation League, this layer is represented by a Java library called Atan [11] which provides following services :

- Connection to the simulation server via UDP.
- Side conversion to support the internal representation of the state of the world.
- Parsing of the output string from simulation server.
- Generation of a command string that can be understood by the simulation server.

C. Sensorial/Effectorial Layer

This layer serves as a bridging layer between logical communication layer and cognitive layer. It translates visual information into the representation needed by cognitive layer, and also translates output from cognitive layer into basic action to be performed by the robots. In our implementation for Simulation League which use Prolog programs as cognitive layer and Java library as logical communication layer, this

means translating visual information into prolog predicates and interpreting prolog query result. Let us show an example of the java translation code used when a player see a ball:

```
public void infoSeeBall
       (double distance, double direction) {
     infobuf.append("distance").
       append(PrologConnection.ATOM_SPLITTER).
     append("ball").
       append(PrologConnection.ATOM_SPLITTER).
     append(distance).
       append(PrologConnection.FACT_SPLITTER).
     append("direction").
       append(PrologConnection.ATOM_SPLITTER).
     append("ball").
       append(PrologConnection.ATOM_SPLITTER).
     append(direction).
       append(PrologConnection.FACT_SPLITTER);
}
```

If the player see a ball in distance x and direction y, then this information is translated into prolog predicates as `distance(ball,x)` and `direction(ball,y)`.

D. Cognitive Layer

Cognitive layer is where the strategy is implemented. It is the highest level layer. Our work is focused in this layer where we employ The Ciao Prolog System [12], and in particulas the Fuzzy Prolog library, to do reasoning over provided information. Our approach is providing the capability of handling fuzzy, uncertain and incomplete information to the cognitive layer. This information is very close to the human reasoning, so this framework is improving the human-like control of this layer. A strategy can be easily implemented on this layer without having to put effort on low level technical details more related to the machine than to the human mind.

V. EVALUATION

For testing our architecture (Figure 3) we have implemented a prototype using Ciao Prolog and its library that provide Fuzzy Prolog, a Java interface programs to connect to the Atan library and a RoboCupSoccer Simulator.

Some simple scenarios have been prepared for observation on difference between fuzzy and crisp approach on similar strategy.

For example, if the strategy is implemented as a prolog program, the program with crisp strategy takes only the best action and fails when there is none. The program with fuzzy strategy proposes the almost best action when there is no best action.

The program with crisp strategy (using classical prolog) looks like the following:

```
get_command  (Info,Command) :-
             update_info(Info),
             best_command(Command).
```

And the fuzzy approach for similar strategy:

```
get_command  (Info,Command) :-
             update_info(Info), !,
             best_command(Command,100).
get_command  (_,Command) :-
             best_command(Command,V),
             V .>. 80.
```

The strategy is coded into the predicate `best_command/2`. Below is the example of fuzzy approach to determine next action when the player has the ball, in other words, the ball is closed to him.

```
% play_on
% if i have the ball then
% - shoot to goal
% - dribble to goal
% - pass to teammate
best_command(Command,V) ::~ min
              play_mode(play_on),
              \+ player_role(goalie),
              ball_in_possesion(V1),
              goal_position(Dist,Dir),
              good_to_shoot(Dist,Dir,V2),
              shoot(Dist,Dir,Command).

best_command(Command,V) ::~ min
              play_mode(play_on),
              \+ player_role(goalie),
              ball_in_possesion(V1),
              not_guarded(V2),
              goal_position(Dist,Dir),
              dribble(Dist,Dir,Command).

best_command(Command,V) ::~ min
              play_mode(play_on),
              \+ player_role(goalie),
              ball_in_possesion(V1),
              is_guarded(V2),
              pass_to_teammate(Command).
```

In the above example there are many fuzzy concepts. Depending on the distance to the goal position the player will evaluate if it is a good for shooting to goal. In case it is, he will do it. Otherwise, he will evaluate if his possesion of the ball is save enaugh (no players of the other team are close to the ball). If not, he will dribble with the ball to skip the danger. Finally, if the player is not close to shoot to goal and is is not possible to dribble and then shoot to goal, then he will pass the ball to another player of his team.

This is only an example for representing the necesity of using fuzzy rules and fuzzy concepts in soccer control.

The evaluation includes power calculation using set of rules, rule based decision, and team play. Fuzzy rules enable fuzzy control implementation for power calculation. In rule based decision, there are certain cases where fuzzy approach offers better solution than crisp approach. These small differences in power calculation and decision leads to a slight better performance of fuzzy approach on team play with respect to the crisp approach. However, fuzzy approach requires more processing time than crisp approach and this could lead to poor performance if it is not managed properly. We will evaluate the efficiency of the two approaches (crisp and fuzzy, Prolog

and Fuzzy Prolog) in further work.

We have realized that none of the two alternatives (crisp versus fuzzy) are good for all scenarios. Indeed, it seems that a combination of fuzzy reasoning with crisp predicates could be the perfect combination and this is a promising result because Fuzzy Prolog [5] is able to deal with this combination.

VI. Conclusion

We choose RoboCupSoccer domain as our case problem to employs Fuzzy Prolog system (an approach to incorporate fuzzy reasoning into logic programming). This work is an initial step toward series of research in this area.

On the other side, we believe that logic programming is a perfect environment for dealing with the cognitive layer at RoboCupSoccer league as it is in general to implement cognitive and control issues in robotics.

Our goal is to provide a framework to employ Prolog in general and Fuzzy Prolog in particular for RoboCupSoccer. A generic architecture for RoboCupSoccer is given, with flexibility in changing leagues and programming languages while maintaining a combination of Prolog and Fuzzy Prolog as cognitive layer. For the implementation, we apply the framework to work on the RoboCup Soccer Simulator, by implementing a prototype player with Fuzzy Prolog as cognitive layer and adapting properly the sensorial/effectorial layer. Considering time and technical constraints, we choose the discrete Fuzzy Prolog system over the continuous approach, for this application, providing constructive answers as values (instead of constraints) is translated into more efficiency for the strategy (measurements will be studied in further work).

We observe that the fuzzy program is slower than crisp program for several reasons :

- Fuzzy program needs to be translated into CLP($\mathcal{FD}$)
- Instead of failing immediately when a predicate in a body clause is not satisfied, the evaluation in a fuzzy rule continues and the truth values are aggregated with the truth value 0.
- When there is no best action to be done, fuzzy program attempts to find an alternative action.

Therefore, the fuzzy program should be designed carefully by taking into account the different procedural semantic between a crisp prolog program and a Fuzzy Prolog program.

This work establishes a preliminary groundwork towards a series of research on employing Fuzzy Prolog in RoboCupSoccer E-League. The results of this work are:

- A prototype of discrete Fuzzy Prolog system, implemented as `dfuzzy` package in Ciao Prolog.
- A framework to employ Fuzzy Prolog in RoboCup Soccer, including a prototype of cognitive layer for RoboCup Soccer.
- A prototype of player client as application of the framework for the RoboCup Soccer Simulator League (sensorial/effectorial layer).
- Simple scenarios to demonstrate the utility of fuzzy reasoning for soccer players control.

There is room for improvement of this work. With regards to the cognitive layer, more advanced strategy and various proper aggregation operator can be applied, using either continuous or discrete Fuzzy Prolog system. We intend to distinguish at the players control, when is it better to use fuzzy reasoning (Fuzzy Prolog), and when is it faster to use crisp reasoning (Prolog). Our future work is to improve and employ the cognitive layer for RoboCup Soccer E-League. Another possible work in this research area is to use Fuzzy Prolog as cognitive layer in different RoboCup domain, for example in RoboCup Rescue or RoboCup@Home.

References

[1] A.J.García, G.I.Simari, and T.Delladio, "Designing an Agent System for Controlling a Robotic Soccer Team," 2004, argentine Conference on Computer Science (CACIC 2004). [Online]. Available: http://www.cs.umd.edu/ gisimari/publications/cacic2004GarciaSimariDelladio.pdf

[2] S.Guadarrama, S.Muñoz, and C.Vaucheret, "Fuzzy prolog: A new approach using soft constraints propagation," *Fuzzy Sets and Systems*, vol. 144, no. 1, pp. 127–150, 2004.

[3] S.Muñoz-Hernandez and C.Vaucheret, Eds., *Extending Prolog with Incomplete Fuzzy Information*, ser. Proceedings of the 15th International Workshop on Logic Programming Environments, 2005.

[4] S.Muñoz-Hernandez and J.M.Gomez-Perez, Eds., *Solving Collaborative Fuzzy Agents Problems with CLP($\mathcal{FD}$)*, ser. Proceedings of International Symposium on Practical Aspects of Declarative Languages 2005, California, 2005.

[5] S.Muñoz-Hernandez and C.Vaucheret, Eds., *Default values to handel Incomplete Fuzzy Information*, ser. IEEE Computational Intelligence Society Electronic Letter, ISSN 0-7803-9489-5, vol. 14. IEEE, 2006.

[6] M.Chen, K.Dorer, and E.Foroughi, *Users Manual RoboCup Soccer Server*, 2003.

[7] J.Anderson, J.Baltes, D.Livingston, E.Sklar, and J.Tower, "Toward an Undergraduate League for Robocup," in *In Proceedings of Seventh International RoboCup Syposium (RoboCup-2003)*, 2003.

[8] J.M.Santos, H.D.Scolnik, I.Laplagne, S.Daicz, F.Scarpettini, H.Fassi, and C.Castelo, "Uba-sot: An approach to control and team strategy in robot soccer," *International Journal of Control, Automation, and Systems*, vol. 1, no. 1, pp. 149–155, 2003.

[9] "AL05_PID_0040 project," http://faea.uncoma.edu.ar/materias/ia/Robotica.

[10] M. Hermenegildo, F. Bueno, D. Cabeza, M. García de la Banda, P. López, and G. Puebla, "The CIAO Multi-Dialect Compiler and System: An Experimentation Workbench for Future (C)LP Systems," in *Parallelism and Implementation of Logic and Constraint Logic Programming*. Commack, NY, USA: Nova Science, April 1999.

[11] "Atan library," http://atan1.sourceforge.net.

[12] "The Ciao Prolog," http://www.cliplab.org/Software/Ciao.