2009

Belief-Desire-Intention in RoboCup

Thopik Adianto

Belief-Desire-Intention in RoboCup

(*Thesis Format*: Monograph)

by

Thopik Adianto

Graduate Program in Computer Science

Submitted in partial fulfillment
of the requirements for the degree of

Master of Science

2

School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Abstract

The Belief-Desire-Intention (BDI) model of a rational agent proposed by Bratman has strongly influenced the research of intelligent agents in Multi-Agent Systems (MAS). Jennings extended Bratman's concept of a single rational agent into MAS in the form of joint-intention and joint-responsibility. Kitano *et al.* initiated RoboCup Soccer Simulation as a standard problem in MAS analogous to the Blocks World problem in traditional AI. This has motivated many researchers from various areas of studies such as machine learning, planning, and intelligent agent research. The first RoboCup team to incorporate the BDI concept is ATHumboldt98 team by Burkhard *et al.*

In this thesis we present a novel collaborative BDI architecture modeled for RoboCup 2D Soccer Simulation called the TA09 team which is based on Bratman's rational agent, influenced by Cohen and Levesque's commitment, and incorporating Jennings' joint-intention. The TA09 team features observation-based coordination, layered planning, and dynamic formation positioning.

Keywords: Belief-Desire-Intention, Rational Agent, Multi-Agent System, RoboCup, Soccer Simulation.

Acknowledgements

I would like to thank my supervisor, Dr. Mercer, for his guidance, patience, and encouragement throughout the journey of my studies which resulted in this thesis. Imbued with qualities of wisdom and judgment, Dr Mercer to me is a model mentor, and an embodiment of professionalism. Dr Mercer has taught me how to work independently and pragmatically in the noetic realm of artificial intelligence. He gave me confidence in myself as an academic researcher and as a professional.

My sincerest appreciation is extended to my parents for their support, wisdom, encouragement, and for instilling the importance of education and excellence in my life. I am thankful to my aunts for their support, kindness and good counsel. Finally, I am deeply thankful to Bora for her emotional support and for being the driving force that moves me forward.

Contents

List of Figures

Chapter 1

Introduction

The Belief-Desire-Intention (BDI) model of a rational agent [3] has been established as the most studied model for practical reasoning agents in Artificial Intelligence (AI). Recent interest in Multi-Agent Systems (MAS), particularly the concept of intelligent agents collaborating with other agents, have motivated many research works in Distributed Artificial Intelligence (DAI). In 1997, the RoboCup Soccer Simulator [18, 19] formalized the standard problem in a MAS. RoboCup, or Robot World Cup, is an attempt to foster AI and intelligent robotic research using a standard problem and a dynamic real-time soccer game for a wide range of AI and robotics research. The first RoboCup competition was held at IJCAI-97 Nagoya, Japan. Since the inception of the competition, many universities from around the world have been participating in the RoboCup annual competition representing various research areas such as real-time reasoning, machine learning, real-time planning, *etc.*

Although there are several generic BDI toolkits available such as Procedural Reasoning System (PRS)[?] and dMARS variant[?], none has been implemented in RoboCup. The first attempt to incorporate BDI in RoboCup was Burkhard *et al.* [5, 6] with their ATHumboldt98 team for the RoboCup 2D Simulation league. The BDI implementation in ATHumboldt98 is loosely modeled on Bratman's rational agent and focuses more on ad hoc low-level short-term planning purposes like ball kicking, ball intercepting, and kick direction (shoot or pass). Long-term planning is presented by intention fixing on a specific goal by repeatedly executing a short-term goal. The lack of cooperation has limited ATHumboldt98's BDI implementation to a single agent's perspective. The BDI architecture in ATHumboldt98 interacts mainly with RoboCup specific functionalities. The RoboCup specific functionalities make the architecture harder to be generalized for implementation into other real-time MAS applications.

In this thesis we present a novel implementation of Belief-Desire-Intention (BDI) in a RoboCup 2D Simulation team based on Bratman's [3] rational agent, influenced by Cohen and Levesque's [8] commitment, and incorporating Jennings' [16, 15, 17] joint-intention. Our BDI-based RoboCup team is named TA09. Existing source codes and research works, such as CMUnited99 [26] and FCPortugal2001 [20], are used as a foundation for TA09 to reduce development time.

A novel BDI architecture—with its related components such as *decision process, mode, plan, commitment,* and *observation-based intention recognition*—is implemented onto a stripped-down version of CMUnited99 to form TA09. The BDI layer is designed according to Bratman's philosophical approach of rational agent. *Beliefs, desires,* and *intentions* are governing a *decision process* in selecting which action is to be executed. Bratman's concept of partial planning is reflected in the form of *mode* and *plan. Mode* is an agent's state of mind when executing a series of low-level actions sent to the RoboCup Simulation Server. A sequence of *modes* forms a *plan* and a set of *plans* is stored in a plan library which is specialized into a *role. Commitment* to an adopted plan and persistency in executing the *plan*'s *mode* are directly influenced by Cohen and Levesque's [8] concept of intention and commitment. Jennings' *joint-intention* [16] has inspired a novel teammate *observation-based coordination via intention recognition* in TA09. This novel BDI architecture with its related components allows emergent behaviours of a rational agent described by Bratman.

Non-BDI functionality such as goalkeeper, roles, and formations are added to TA09 to form a fully functional RoboCup team. A basic version of goalkeeper is included in FCPortugal2001 and is adopted in TA09. Similar to a real soccer team, each player in TA09 is assigned to a specific role such as defender, midfielder, and striker. Each role will carry a set of plans specialized for that role. A team formation and positioning system similar to Lau and Reis' [20] SBSP (Situation Based Strategic Positioning) is added to TA09. Depending on ball direction, velocity, and position, TA09 players position themselves strategically.

BDI and non-BDI components are the internal machinery in decision making and planning of rational agents forming TA09. The glue for BDI and non-BDI components is the configurable elements such as the plan library, loadable parameters governing mode's behaviours, and dynamic goals which as a whole enable intelligent behaviours of rational agents to emerge from TA09.

The remainder of this thesis is comprised of:

- Chapter 2 that summarizes various research works from BDI, RoboCup, and agent related areas.
- Chapter 3 that discusses all aspects of BDI and non-BDI components implementation in TA09.
- Chapter 4 that discusses comparative and subjective evaluations performed to measure TA09's design and performance against CMUnited99, FCPortugal2001, and ATHumboldt98.
- Chapter 5 that discusses TA09's design and implementation, testing and evaluations, and future directions to overcome TA09's current limitations.
- Chapter 6 that concludes this thesis with our closing thoughts.

Chapter 2

Background Works

Central to this thesis the concept of Belief-Desire-Intention (BDI) influenced by Bratman [3] and other researchers is summarized in the next section. RoboCup initiated by Kitano *et al.* [19] along with RoboCup champions CMUnited99 by Stone *et al.* [19, 26, 27, 28] and FCPortugal2001 by Lau and Reis [20] are outlined in the RoboCup section along with BDI-based ATHumboldt98 by Burkhard *et al.* [5, 6]. Other research works related to plan recognition and machine learning are also discussed in Section 2.3.

2.1 Belief-Desire-Intention (BDI)

Bratman [3] in his book *Intention, Plans, and Practical Reason*, develops a planning theory of intention, a philosophical approach to intention as part of practical reasoning by desire-belief reasoning of action, future-directed intention, and partial plans. Bratman's ideas have spurred interest in the so-called Belief-Desire-Intention (BDI) model of intelligent agent. The four main theses presented as the foundation of BDI model are: *1) the methodological priority of intention in action*, *2) the desire-belief theory of intention in action*, *3) the strategy of extension*, and *4) a reduction of future-directed intention to appropriate desires and beliefs.* The main idea of practical rationality is that an agent's desires and beliefs at a certain time provide reasons for acting in various ways at that time. The agent's intentional action has to be at least as strongly supported by these desire-belief reasons as any of its supposed alternatives. A plan is typically partial, has a hierarchical structure, resists reconsideration, and eventually controls conduct in which the connection between deliberation and action is systematically extended over time. Bratman suggests the phenomena of partial plans and reasoning aimed at filling in such plans are central to our understanding of intentions. An agent's plans have to be both internally consistent and consistent with the agent's belief; in addition, agent's plans should be means-end coherent, *i.e.* filling the plan with means, preliminary steps, actions, and specification of ends that are globally consistent. Bratman's approach has motivated us in applying an extended version of his BDI model of intelligent agent in RoboCup 2D Soccer Simulation.

Although Bratman's philosophical approach is taken from a single agent perspective, it does not restrict its extension to a multi-agent platform. Jennings [17, 16, 15] extended the commitment of a rational agent with *joint-intention* and *joint-responsibility* in a Multi-Agent Systems (MAS) domain. Jennings introduced the joint responsibility model to specify a novel high-level architecture for cooperative problem solving. The mental notions of *belief*, *desire*, *intention*, and *joint-intention* play a central role in individual and group problem solving behaviour. This architecture has been implemented for the real-world domain of electricity transportation management and the CERN Proton Synchrotron. Jennings pointed out limitations with the individualistic approach: joint action is more than just the sum of individual action even with coordination and there is a difference between individual and group commitment. *Joint-intention* is defined by Jennings as joint commitment to perform collective actions during a period of shared mental state. Thus collaboration is not an intrinsic property of the actions but rather is dependent on the mental state of the participants. Features of shared mental states are: agents must agree on common goals, agree they wish to collaborate to achieve their shared aim, agree a common means (plan) of reaching their objective, acknowledge that actions performed by different agents are related, have criteria for tracking rationality of their commitments, and have behavioural rules which define how to behave locally and towards others both when joint action is progressing as planned and when it runs into difficulty.

Joint-responsibility extends the ideas of joint intention to include plan states. Responsibility means each individual should remain committed to achieving a common objective by the commonly agreed solution until one of these becomes true: the desired outcome of a plan step is achieved; following the agreed upon action sequence does not achieve the desired consequence; or one of the specified actions cannot be carried out.

Cohen and Levesque [8][13] explored and formalized principles governing the rational balance among an agent's beliefs, goals, actions, and intentions. Cohen and Levesque modelled intention as a composite concept specifying what the agent has chosen (desire) and how the agent is committed to that choice (intention). The agent's commitment is reflected in the agent's persistency to achieve its desire (goal) over a period of time if the goal remains achievable. Cohen and Levesque proposed a logic with four primary modal operations: BELief, GOAL, HAPPENS (what event happens next), and DONE (which event has just occurred). Agents can be characterized using those operators to perform actions that are intended to achieve their goals. The world is modelled as a linear sequence of events and by adding GOAL an agent's intentions can be modelled. This world model includes courses of events which consists of sequences of primitive events that characterized what has happened and will happen in each possible world. Possible worlds can relate one to the other in the semantics of BEL and GOAL. An agent is not guaranteed to execute a sequence of events without events performed by other agents intervening.

Rao and Georgeff [22, 23] introduced a family of multi-modal branching-time BDI logics with a semantics that is grounded in traditional decision theory and a possible-worlds framework categorized with sound and complete axiomatizations with constructive tableau-based decision procedures for satisfiability testing and formula validation. Building BDI system information on the state of the environment is called the system's *beliefs*; such a system can be implemented as a variable, as a database, as a set of logical expressions, or as some other data structure. Beliefs can be viewed as the *informative* component of the system's state. Information about the motivational objectives to be accomplished with priorities or associated payoffs is the system's *desires.* The system representation of a chosen course of action is called the system's *intentions.* The intentions of the system are captured by the *deliberative* component of the system. With possible-worlds semantics, Rao and Georgeff considered each possible world to be a tree structure with a single past and a branching future. Each tree structure denotes the optional courses of events that can be chosen by an agent in a particular world. Evaluation of formulas is with respect to a world and a state. Hence, a state acts as an index into a particular tree structure or world of the agent. The belief-accessibility relation maps a possible world at a state to other possible worlds. The *desire-*, and *intention-accessibility* relations behave in a similar fashion. Rao and Georgeff's possible world semantics [22] using decision trees is simple yet elegant and has influenced the use of a similar decision tree structure in this thesis.

2.2 RoboCup

Kitano *et al.* [19] initiated Robot World Cup (RoboCup) in an attempt to foster Artificial Intelligence (AI) and intelligent robotic research using a standard problem, a dynamic real-time soccer game, for a wide range of AI and robotics research. The first RoboCup competition was held at IJCAI-97 in Nagoya, Japan. Since that inception many universities from around the world have been participating in the RoboCup annual competition presenting various research areas such as real-time reasoning, machine learning, and real-time planning. The RoboCup Official Site [9] categorized five RoboCup competition leagues as *Simulation*, *Small-size*, *Middle-size*, *Standard Platform*, and *Humanoid.* The simulation league consists of 2D, 3D, 3D development, and Mixed Reality. 2D Simulation is the most common and popular league among all simulation sub-leagues and adopts real-world soccer game rules. The small-size league allows up to five robots with diameter maximum 18 cm playing with an orange golf ball as the soccer ball in a field of 6.5x4.5 metres for two 10-minute matches. The middle-size league allows up to six robots with diameter maximum 50 cm playing a real orange colored soccer ball in a field of 12x18 metres for two 15-minute matches. The standard platform used to use the Sony Aibo robotic dog and now uses the standard humanoid robot called Aldebaran Nao [21]. The humanoid league is introduced in 2002 using biped autonomous humanoid robots competing to perform challenges like penalty kicking and goal keeping.

Chen *et al.* [7] wrote an extensive RoboCup 2D Simulation Users Manual. RoboCup simulation is packaged with three components: *Server*, *Monitor*, and *Logplayer*. *Server* is a soccer game simulator with all clients of both teams being connected to it via UDP/IP client-server communication. A team can have up to 11 players including a goal keeper. Each player on a team is a separate process and connects to the server on a specified port. The players send requests to the server on actions they want to perform. The server receives those requests and updates the simulation world. Independent of this player-server communication, the server sends real-time sensory information to all connected players in discrete time intervals or cycles. For realistic simulation, noise is added to real-time sensory information by the server in such as way that the further an object away from the observer the more inaccurate the distance and the angle reported by the server. *Monitor* is a visualization tool that allows an observer to see what is happening to the simulation in the server during a game match. The information shown includes team names, score, positions of all players, and the ball. Monitor is not needed to run a game on the server and multiple monitors can be connected to a single server at the same time. *Logplayer* is a replay tool that allow the monitor to view pre-recorded simulation games using various features such as play, stop, fast forward, rewind, and jump to specific time in the game. To enforce real soccer game rules RoboCup 2D Simulator game is being judged by an automated referee and a human referee. The automated referee is built in the server and will enforce and regulate kick-off, goal, ball out of field, player clearance, play-mode control, half-time, and full time. A human referee is watching using Monitor to ensure that there are no rule violations not caught by automated referee such as flooding the server with messages, obstruction, blocking, and inappropriate behaviours. A player connected to the server can be either a regular player, goal keeper, offline coach (trainer), or on-line coach. A regular player is allowed to request for all actions except *catch_ball()* which is reserved for the goal keeper. Coaches are privileged clients used to provide assistance to the players. The trainer has extra capabilities such as announce play-mode (*e.g.* kick-off, corner kick, *etc.*), broadcast audio message, move players and balls to any location on the field including their directions and velocities, and receive noise-free information on a movable object from the server. The online coach is used during a match to observe the game and provide advice to the players. The online coach has more limited capabilities than the trainer and is only able to communicate with players and receive noise-free information from the server. Figure 2.1 shows the virtual markers on simulation field announced by the server to connected players and Figure 2.2 shows a kick-off screenshot of *Monitor*.

Further collaboration with Stone et al [19, 26, 27, 28] resulted in a new RoboCup 2D Simulation soccer client called CMUnited99. A stripped-down version of CMUnited99 source code has been made publicly available to foster research in RoboCup 2D Soccer Simulation. Novel concepts such as *layered learning* and *flexible team structure* are introduced in CMUnited99. Layered learning consists of two layers where the first layer utilizes a Neural Network (NN) for low-level individual skills such as ball interception, ball kicking, obstacle avoidance, and the second layer utilizes a Decision Tree (DT) for higher level "social" skills involving multiple agents such as passing ball to a teammate. Layered learning using a NN proved to be a successful method in improving low-level individual skills in the RoboCup environment where noise is added in the perception sensors from the server for realism. Flexible team structure centered around teamwork structuring and Periodic Team Synchronization (PTS). Teamwork structure consists of: flexible agent *roles* with protocols for switching among them, a

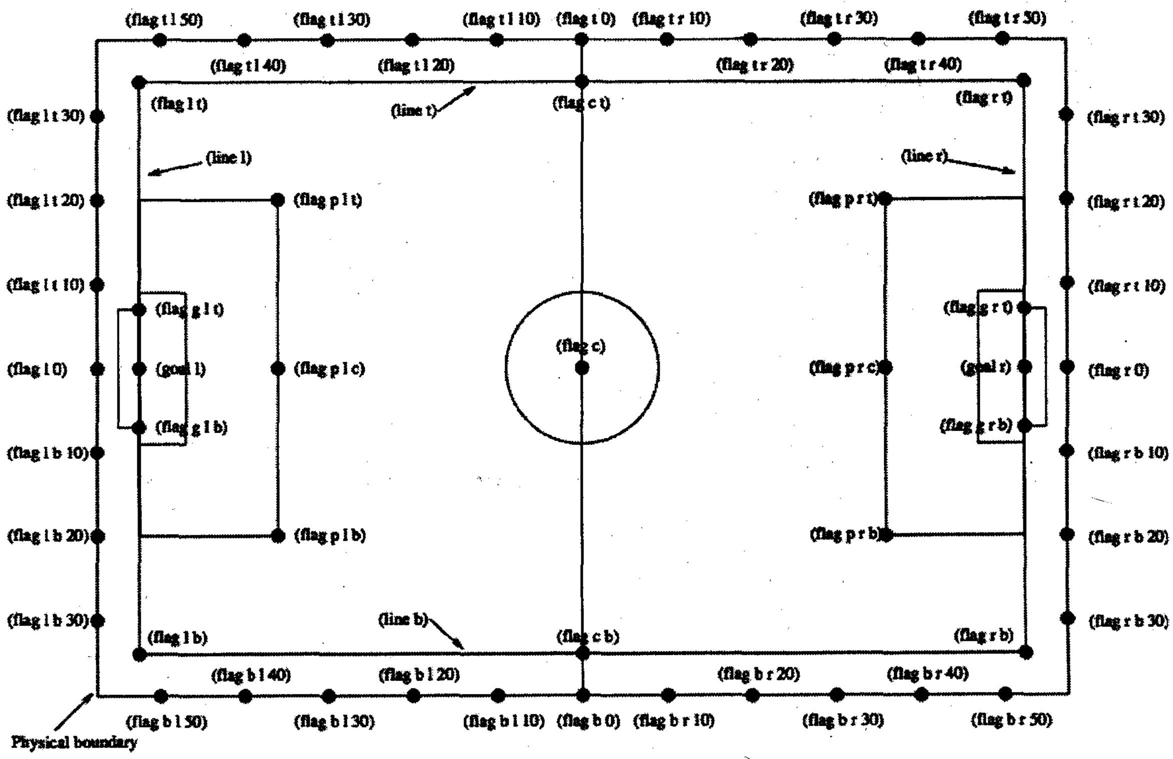

Figure 2.1: Fields markers from [7]

collection of *roles* built into team *formations*, and multi-step, multi-agent plans for execution in specific situations called *set-plays*. This role concept allows better grouping of different individual skills depending on the need of a particular agent on a team. An aggressive team will need more offensive players than defensive one. Similar mixes of roles are required for dealing with corner kicks or free kick close to the penalty box. PTS allows agents to periodically synchronize in a full-communication setting in a low communication environment. This communication allows agents to switch formations, roles, and set-plays whenever possible such as free-kick, corner-kick, and when within audio range. PTS is quite effective in distributing information in noisy and unreliable environments like the RoboCup Simulation Server. The main requirement of PTS is that the agents are within audio range. This limitation becomes important during game play when agents can be scattered over the field. During half-time, free kick, or penalty kick, most agents are positioned closer to each other and PTS can

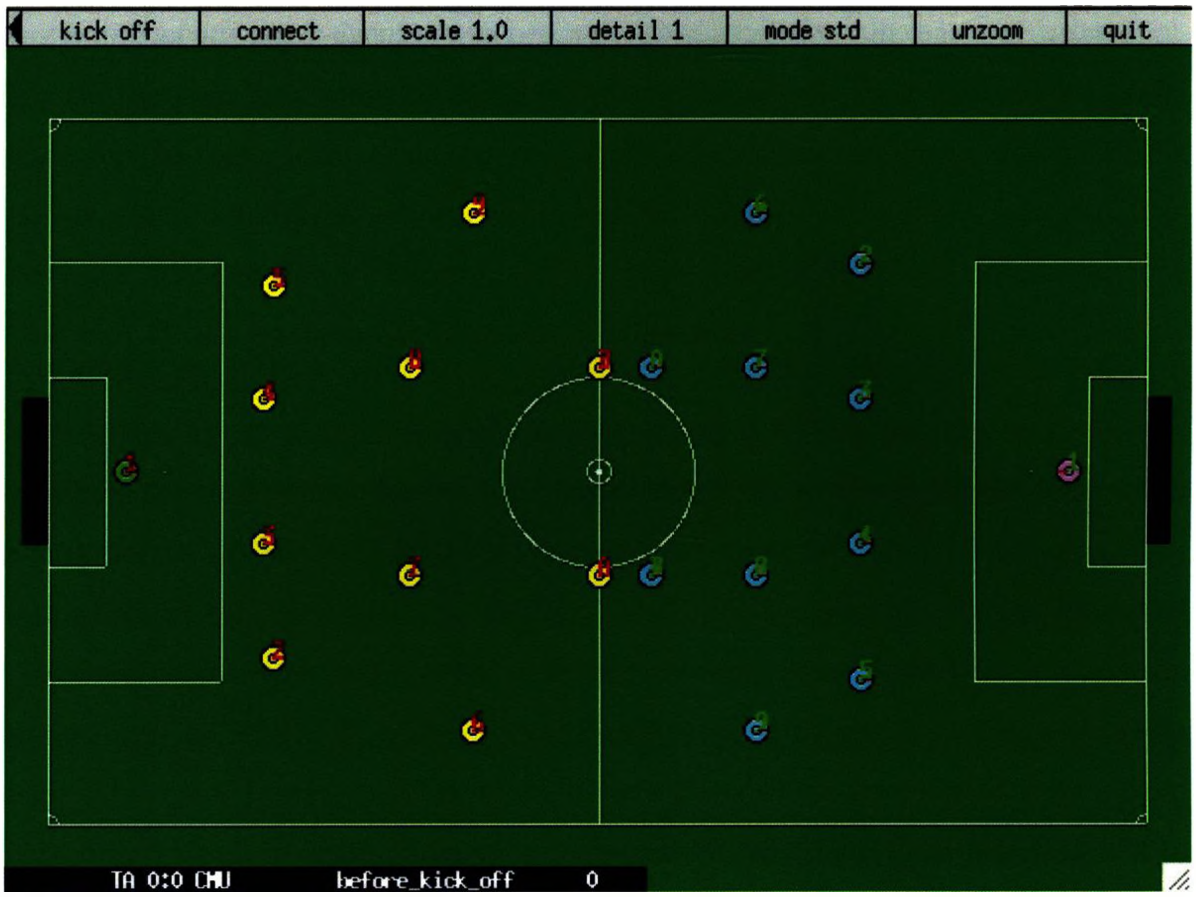

Figure 2.2: Match screenshot

be utilized effectively.

Based on the CMUnited99 source code Lau and Reis [20] implemented Situation Based Strategic Positioning (SBSP) and Dynamic Positioning and Role Exchange (DPRE) for formation and positioning as a whole team in FCPortugal2001. SBSP utilizes various policies to determine the best position for a player in a particular situation. DPRE allows players to switch roles and positions dynamically based on the adopted team formation during a match. Formation changes are triggered when the team adopts a new tactic. The tactic contains a set of predefined formations and each formation contains a set of roles. Roles define FCPortugal2001 players' behaviours. FCPortugal2001's SBSP is similar to CMUnited99 set-plays which relies on predefined roles associated with a set of behaviours along with the strategic positioning of involved agents to coordinate or interact on the basis of behaviour. This can be seen as a loose coupling of layered behaviours such as low-level skills, behaviours in roles, formation, and an overall group's convention in which SBSP's policies which determines actions to be executed. Like CMUnited99, Lau and Reis released a stripped-down FCPortugal2001 source code which does not include the SBSP and DPRE features due to competitive reasons.

Burkhard *et al.* [5, 6] incorporated BDI in their ATHumboldt98 team for the RoboCup 2D Simulation league. ATHumboldt98 has a World Model storing an agent's beliefs of its world based on information provided by the RoboCup Simulation Server (RCSS). Belief in the BDI approach matches the World Model of ATHumboldt98 where each player maintains its own world based on information provided by the RCSS. The World Model also projects future situations and tracks historical events. Desires in ATHumboldt98 are goals which are selected from a fixed goal library which is comparable to a decision tree selection. Goal selection has to be fast and Burkhard *et al.* discussed the trade-off between long-term reasoning and the speed of short-term reaction. ATHumboldt98 has *active options* that are used when the ball is in possession and *passive options* that are used when the ball is not in possession. *Active options* are GoalKick, DirectPass, ForwardPass, and Dribbling while *passive options* are InterceptBall, GoToHomePosition, and DefendGoal. All *options* are constrained by ConserveStamina and AvoidOffside. The last aspect of BDI, *Intention*, is divided into two stages of planning called *layered planning*. The first stage is choosing the best possible goal and fixing on it as the intention. This goal is a coarse long-term plan. It is prior to a more precisely defined execution and can be considered as partial planning. The second stage is the execution of a goal by selecting appropriate finer steps that fit into the selected goal. Burkhard *et al.* showed that the layered learning fits the criteria of bounded rationality where the second stage is short-term and reactive to the world's changes while still guided by the first stage. Burkhard *et al.* stated the commitment to intention is shown by both implicit and explicit persistence of goals and intentions in the ATHumboldt98 implementation.

Hexmoor and Zhang [12] examined the concepts of norms and roles in a multi-agent system for RoboCup soccer simulation. The definition of norms includes behavior constraint on agents, goals, and obligations. Hexmoor and Zhang formally defined norm as *Norm = (O, R, G, U)* where O is the content of the norm set, R is the sanction that may result from not following the norm, G is the agent's goal that invokes the norm set when the agent chooses to consider other agents, U is a utility function that considers the agent's gains and losses in terms of G. General utilitarian norm adoption that is based on maximizing goal achievement utilities can be used in single or multiple goal selection. The definition of roles ranges from the abstract representation of an agent function, service or identification, part of what an individual agent chooses to play, to service + policy. Hexmoor and Zhang defined role as *Role = (A, C, O, G, Ab, U)* where A is the adopting agent, C is the social context of A, O is the content of the role, G is the agent's goal that invokes consideration of other agents, Ab is the set of capabilities that are required for A, and U is a utility function that considers the agent's gains and losses in terms of G. Role adoption is similar to norm as it attempts to maximize goal achievement utilities. Hexmoor and Zhang discussed that a reciprocal relationship can exist between goal-based role selection and role-based goal selection where in the latter, a problem-solving agent selects the goal that maximizes fulfillment of its role. In the relation to RoboCup, a *positional role tree* is a hierarchy of decisions where the tree leaves are specific positions that can be occupied by an agent and the arcs "specialize" from general to more specific. The role of a player in a RoboCup team changes based on the formation decided as a whole team. Each role is associated with different abilities which follow similar specialization concept. A *group role* is a specialization shared by a group of agents that will also share a group norm. A group role is adopted by individuals in a role to maximize the group's utility as a whole.

2.3 Other Agent Features

Rao and Murray [24] provided algorithms for performing reactive plan recognition within the framework of agent's beliefs, desires, and intention with limited resources and in a continuously changing environment. This mental-state recognition and integrated reactive plan execution and plan recognition is applied to air-combat modelling to enable pilots to infer the mental-state of their opponents and choose their own tactics accordingly. The approach is based on using plans as recipes and Belief-Desire-Intention (BDI) to guide and constrain the reasoning processes of agents.

Huber and Durfee [14] developed plan recognition for coordination in a multi-agent environment. Their approach was to have agents infer the plans of each other by observing the actions or behaviours of the other agents without communication. A discrete time, two-dimensional, simulated grid world where an agent can only do basic motion of moving one grid north, south, east, west, or no motion is being used. There were two types of agents installed in the simulation: the observed agent and the observing agent. The observed agent was placed on the initial starting location, given a goal location, and that agent would plan the shortest straight-line path to the goal and then start moving toward it. Once the observed agent arrived at the goal location it will remain static. The observing agent using a belief network would employ several heuristics using observations of the observed agent to chase the observed agent. Several parameters being adjusted to find the best belief level threshold are average end time, average last move time, and average total moves. Huber and Durfee discussed the trade off of early or late commitment resulting in a trade-off between time and effort for the observing agent to reach its goal. The implication of their experiments is that an agent will have to be able to recognize certain characteristics of the environment and an agent's particular situation to be able to determine the relative cost of acting (chasing) versus perceiving (observing).

Guerra-Hernandez *et al.* [2] incorporated a learning capability into a BDI multi-agent system. The agents learn the context of their plans in order to know how and when to use it. The learning capability is using a first order method called Induction of Logical Decision Trees. Four components of the generic learning agent architecture: *1) A learning component responsible for making improvements by executing a learning algorithm; 2) A performance component responsible of taking actions; 3) A critic component responsible for providing feedback; and 4) A problem generator responsible for suggesting actions that will lead to informative experiences.*

Ahmadi and Stone [1] introduced an instance-based action model for action planning by capturing arbitrary distributions of action effects. They used RoboCup 4-legged goal scoring scenario using AIBO robots as a test bed. In a dynamically changing environment such as RoboCup Ahmadi and Stone implemented an on-line incremental re-planning method that modifies the transition model to account for the effects of other agents and then re-plans only for the affected states. A Markov Decision Process (MDP) is a model chosen to represent the planning problem. An MDP transition function is built with the help of the learned action model. A reward maximizing plan is generated using value iteration with state aggregation. In a static environment, the value iteration algorithm can be run offline and in a dynamic one it must be executed on-line. Ahmadi and Stone compared their instance-based model with parametric action model in their experiment. The instance-based approach uses each action effect from experiments, called a sample action effect, and stores it in the model. In a dynamically changing environment the transition value of MDP and the value function for the new MDP needs to be computed on-line. Ahmadi and Stone presented a fast re-planning algorithm using pre-computed learning values (Q-values) for a static environment. This resulted in a fast re-planning algorithm and the robot can distribute the value iteration steps over several decision cycles without missing any action opportunities.

Chapter 3

TA09 Implementation

A stripped-down version of CMUnited99 [26] without the layered learning and flexible team structure modules is publicly available. The stripped-down version of CMUnited99 has all the necessary modules required to build a more complex implementation. We decided to use CMUnited99 as the code base for the TA09 implementation. Following the same layered architecture in CMUnited99 [27], a novel BDI layer is added on top of CMUnited99. Other supporting functionalities are added to the base CMUnited99 to complement the BDI layer. Lau and Reis [20] also publish their FC-Portugal2001 source code albeit without full SBSP and DPRE functionalities. The FCPortugal2001 basic formation functionality is being adopted in our implementation along with a novel positioning algorithm. FCPortugal2001 source code is used to gain insights on how to use the base CMUnited99 code effectively and how to add new layers of complexity on top of CMUnited99.

CMUnited99 source code does not come with a goalkeeper while FCPortugal2001 provides a stripped down version of a goalkeeper. A goalkeeper is the last line of defence and an important factor in winning the game. Only the goalkeeper role is allowed to use the *catch()* command. Additionally, the goalkeeper's movement is configured and focused around and within the penalty box area. We used the FCPortugal2001 goalkeeper in TA09 to focus more on the BDI aspects of non-goalkeeper players. We adopted FCPortugal2001's *formations.conf* file and file loading mechanism into TA09.

RoboCup 2D Soccer Simulator (RCSS) version 11.1.0 which consists of rcssbase, rcsslogplayer, rcssmonitor, and rcssserver is used for this implementation on PowerPC based OS X 10.4 and 10.5. Other software libraries used to compile RoboCup Soccer Simulator are Boost version 1.33, X11 library, and GCC 3.3 compiler that comes with OS X 10.4/10.5 SDK. Python[10] programming language is used for an initial prototyping of a simple RoboCup client. Subversion [11] is used as the main source control repository system. In the next few sections we will discuss in-depth TA09's BDI and non-BDI components.

3.1 Layered Architecture

Like the Stone *et al.* [26, 27] layered approach, the TA09 architecture sits on top of the base CMUnited99 implementation. The TA09 general architecture diagram is shown in Figure 3.1. RoboCup 2D Simulation Server (RCSS) will send sensory information intermittently to each player connected to the server. Each player will parse sensory information to reconstruct player's perception of the world. Any changes to the player's world through sensory updates will influence the BDI layer that drives *decision process* which decides what *plan* to adopt and which *plan*'s *mode* to execute. *Mode* will determine a series of *actions*; *mode* translates the *actions* into commands RCSS will understand. RCSS executes the commands in its simulation and sends back sensory information as a result of those commands. *Parameters* are variable values that influence the behaviour of each *mode.* A list of *parameters* is loaded from the configuration file independently when each player is initialized. The plan library is a collection of *plans* which consist of a series of *modes.* Like *parameters*, the plan library is loaded independently when each player is initialized. Formations are a set of x and y coordinates for each player on a team to form the team formation for positioning during game play. The formations file is loosely adopted from Lau and Reis [20] FCPortugal2001 team with TA09's unique positioning system.

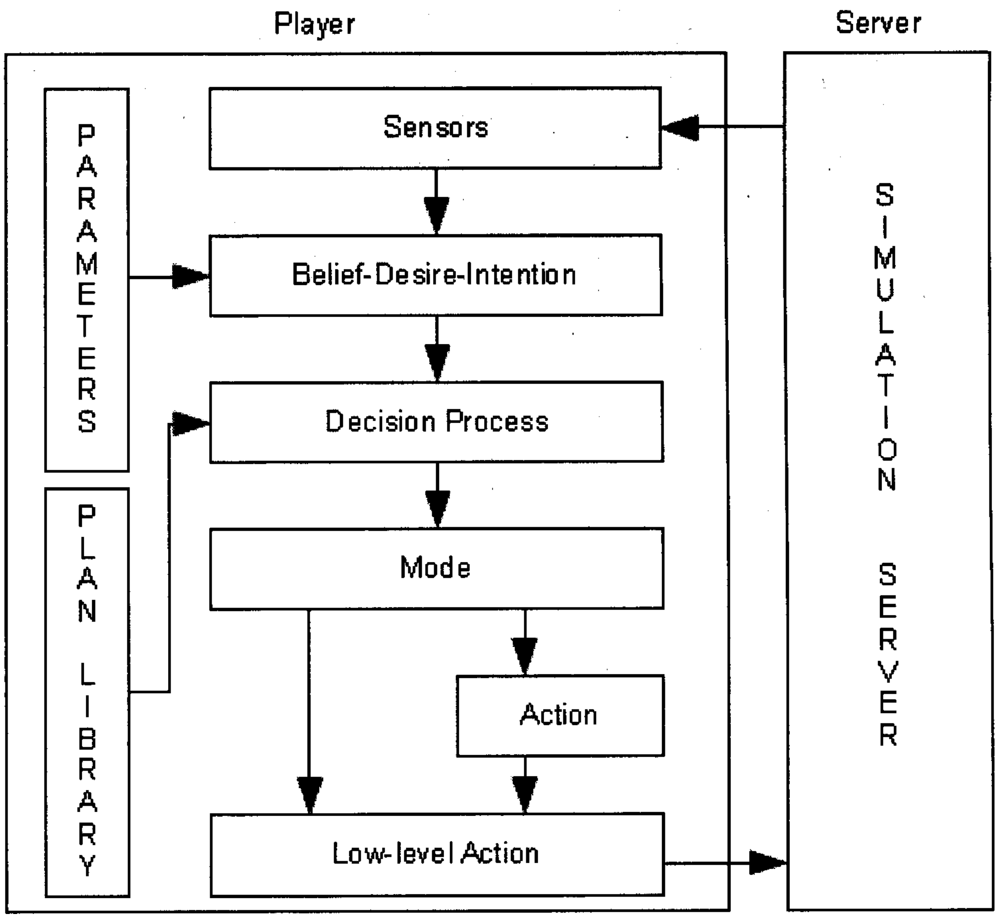

Figure 3.1: Layered architecture.

3.2 Mode

Mode is an agent's present state-of-mind reflecting the agent's intention. Currently, there are seven modes: *chase*, *dribble*, *pass*, *shoot*, *defend*, *position*, and *cover*. These *modes* form a basic working RoboCup team player in TA09. A *mode* consists of execution preconditions, a list of desired *modes i.e.* preferred *modes* to be considered for the next execution step after the executing current *mode*, a series of *actions* to be executed, and goals to be reached. A diagram of the *mode* architecture is shown in Figure 3.2.

Adoption of a *mode* means commitment to execute that *mode* until its goals are achieved or its commitment is dropped. If the preconditions are not met, the agent will not adopt that *mode.* When the agent commits to a mode, the agent will keep executing the mode until that mode's goals are reached as long as the preconditions continue to be met prior to each execution of that mode. Section 3.8 will elaborate how a mode is chosen and executed in TA09's *decision process.*

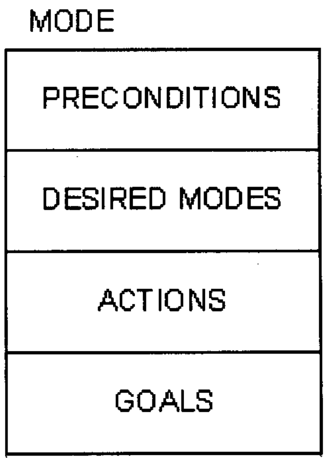

Figure 3.2: Mode diagram.

Each *mode* execution will generate a list of desired *modes* to be selected for the next execution in *decision process* and will be discussed in detail in Section 3.5.

Actions are a series of commands recognizable by RCSS such as *dash*, *kick*, *turn*, *etc.* grouped in a logical way to form a collective functional action.

The behaviours of each *mode* are defined by *parameters* loaded from configuration files independently by each player. The parameters file allows players' behaviours to be defined according to a player's role on a team *e.g.* the defender role will have parameters influencing *defend* mode to have a wider and deeper defensive zone compared to other roles. All of the *modes* in TA09 will be described in detail in the next few paragraphs.

3.2.1 Chase

Chase is using *get_ball()* from CMUnited99 [26] for obstacle avoidance and stamina conservation functionalities. The precondition for *chase* is the number of teammates closest to the ball. This configurable number is set to avoid stealing the ball from a dribbling teammate and to prevent players from swarming the ball. The main action of *chase* is *get_ball()* alone and the mode's goal is reached when the ball is within dribbling distance.

3.2.2 Dribble

Similar to *chase, dribble* consists of *get_ball()* but with the additional basic action *kick*(). *kick()* will kick at a low power to ensure that the ball stays within dribbling distance. Preconditions of *dribble* are: the ball is kickable, within kicking range, or the ball is within a configurable dribbling distance. The *dribble* action is to dribble to strategic areas in the opponent's field— such as the goal line, penalty box corners, and field corners—by considering the number of opponents in the vision cone between the player and these strategic areas. *Dribble*'s goal is reached when the ball is dribbled into the opponent's penalty box, or is past the opponent's penalty kick point, or is past beyond the player's positioning range.

3.2.3 Pass

Pass will pass the ball to the best teammate with the precondition of having the ball within the dribbling distance. *pass_ball()* from CMUnited99 [26] is used in the actual passing; *pass_ball()* calculates the angle and power of the basic action *kick()* to target the teammate position at the desired velocity. A finer decision mechanism in *pass* is the algorithm to prefer the least congested and the least unobstructed teammates in a forward position before falling back to teammates behind the ball *i.e.* pass backward. The last resort of *pass* is to kick to the opponent's left or right corner field. The goal of *pass* is to have the ball leave the dribbling distance to ensure that there is no passing failure or ball is not stolen by a near-by opponent. On an unsuccessful attempt, *pass* will chase the ball using CMUnited99 [26] *get_ball()*.

3.2.4 Shoot

Shoot will shoot the ball at the opponent's goal with the precondition that the ball is within the dribbling distance. The action of *shoot* is to shoot at the least obstructed angle between the shooter and the goal line. The goal of *shoot* is attained when the goal is scored or the ball strays out of the penalty box or the offensive zone.

3.2.5 Defend

Defend acts similarly to *pass* by trying to clear the ball from the defensive zone to the least congested and the least unobstructed teammate. There is no precondition for defend mode and can be executed at any time. *Defend* will persist on chasing the ball and clearing the ball until the ball is out of the defensive zone. The last resort of *defend* is to kick to the closest side line or mid-field side line.

3.2.6 Position

Position does not have any preconditions and can be executed at any time. The action in *position* is to move to the player's formation position and is successful when the player reaches the positioning zone. The positioning zone is calculated based on a configurable extraction and retraction distance from the formation's position. The positioning zone allows the whole team to retain a relative formation positioning to cover the field.

3.2.7 Cover

Cover does not have any preconditions and can be executed at any time. During execution *cover* will chase the ball if the ball enters the covering zone of the player's current position. Chasing will not happen if the ball is being dribbled by the player's teammate. The goal of *cover* is reached when the ball, which entered the covering zone, is secured within a dribbling distance.

3.3 Plan

A *plan* is a sequence of *modes* and an ordered list of *plans* forms a plan library used by an agent's *decision process*. Each *plan* carries a numerical weight associated for defence and offense. The numerical weight will influence a player's preference of one plan over the others depending on the player's analysis of the situation. A plan library is a list of plan definitions stored in the *plans.conf* file and loaded upon the start of each player. Figure 3.3 and 3.4 show plan library diagram and sample library respectively. Figure 3.5 shows the *plan definition* format along with samples. Similar to the parameters file, plan library is used in skills specialization for a specific role. Each player has a subset of plans from the defined plans in the plan library. This plans specialization allows a player to adopt behaviours according to its role. *Specialized plan* format and samples can be seen in Figure 3.6. In our implementation *plans-defense.conf* is used by the defender role, *plans-midfield.conf* for the mid-field player role, and *plans-forward.conf* for the striker role.

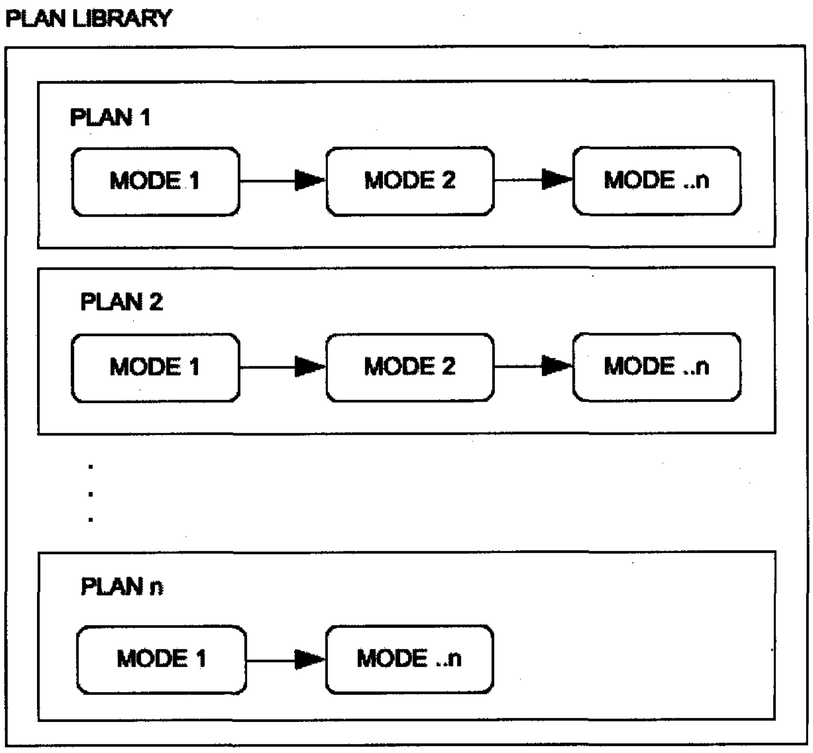

Figure 3.3: Plan Library diagram.

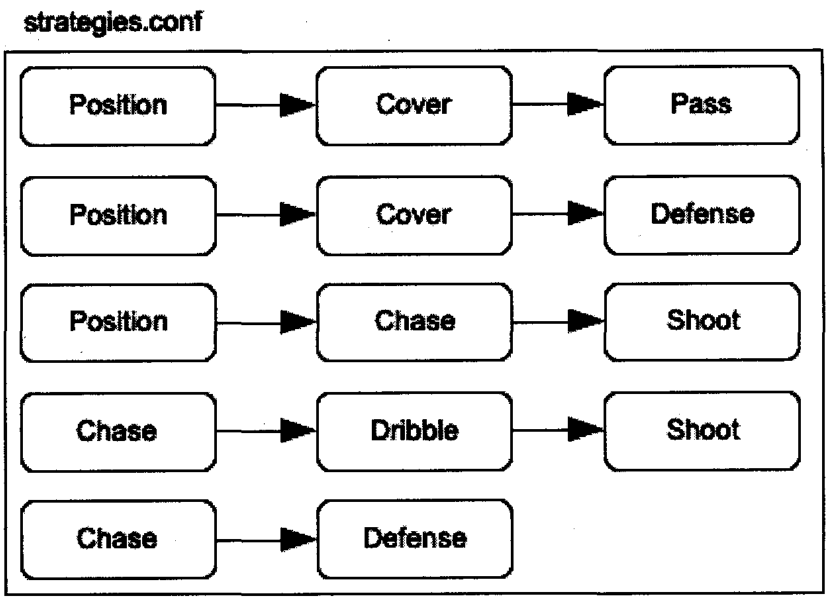

Figure 3.4: Plan Library sample.

```
<Name> <Mood> <ModeCount> <Mode1> <Mode2> ...
...

# Initial: Position , Cover , Chase , Pass
P-PCCP 4 6 7 0 3
# Cover , Chase , Dribble , Shoot
P-CCDS 4 7 0 2 4
# Position: Position , Chase , Shoot
P-PCS 3 6 0 4
```

Figure 3.5: Plan Definition format and sample.

Complement plan is a loose coordination method that allows a player to execute a predetermined plan when a player recognizes a plan being executed by a teammate. This association allows a player to adopt a *complement plan* upon detection of a teammate's *plan.* A *plan* may have one or more *complement plans.* If visual observation of another teammate suggests to a particular *plan* in plan library then its *complement* will be adopted as a means of recognizing the teammate's intention. This can be used to construct more complex coordinations while retaining the loosely coupled way of planning. Adopting a *complement plan* results in a strategic coordination where a player adopts a strategic *plan* depending on a teammate's active *plan. Complement plan* format and a sample are shown in Figure 3.6.

```
<Name1> <Weight> <ComplCount> <ComplPos1> <ComplName1> ...
<Name2> <Weight> <0>
...

# Attack: Chase & Pass then Position & Shoot
P-CP 0 1 0 P-PS
# Position , Shoot
P-PS 0 0
# Attack: Chase, Shoot
P-CS 0 0
```

Figure 3.6: Specialized Plan format and sample.

3.4 Formation

The stripped-down version of CMUnited99 [26] does not come with a formation feature; on the other hand, FCPortugal2001 [20] comes with a simple formation configurable in the formation file but without SBSP (Situation Based Strategic Positioning) and DPRE (Dynamic Positioning and Role Exchange) features. Our implementation follows the FCPortugal2001 approach by defining each formation in a formation file loaded by every player on initialization. Formations used are popular formation types found in a realistic soccer game such as *442* and *433*. Figure 3.7 shows the TA09 formation format and a sample of the *formations.conf* file. Our implementation uses dynamic extraction and retraction to allow players to move freely within the area of each player's position. This implementation is fairly simple compared to FCPortugal2001's SBSP but is usable for a fully functioning RoboCup team.

```
<Formation Name>
<P1x> <P2x> <P3x> <P4x> <P5x> <P6x> <P7x> <P8x> <P9x> <P10x> <P11x>
<P1y> <P2y> <P3y> <P4y> <P5y> <P6y> <P7y> <P8y> <P9y> <P10y> <P11y>

442OPEN
-45.0 -11.0 -12.0 -12.0 -11.0   8.0   2.0   2.0   8.0  20.0  20.0
  0.0  18.0   7.0  -7.0 -18.0  25.0  10.0 -10.0 -25.0  10.0 -10.0
```

Figure 3.7: Formation format and sample.

3.5 Belief-Desire-Intention

Belief-Desire-Intention (BDI) presented in TA09 is an abstract layer rather than a definitive data structure or implemented algorithm. Figure 3.8 shows the BDI diagram consisting of the three main components of *Beliefs*, *Desires*, and *Intentions*. The *Beliefs*, *Desires*, and *Intentions* are local to each player.

Beliefs are constructed from prior and run-time knowledge. Prior knowledge consists of parameters, formations, plan definition library, and specialized plans. Run-

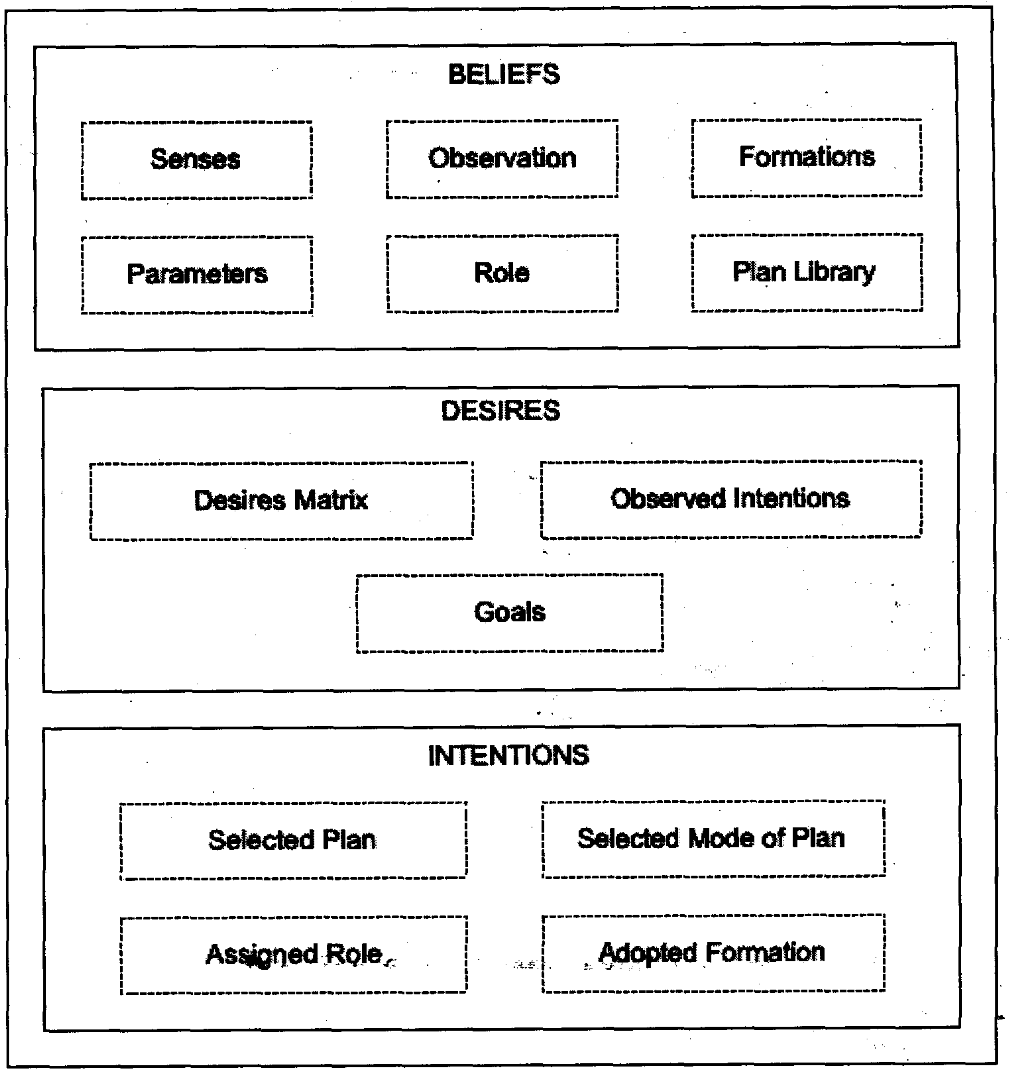

Figure 3.8: BDI diagram.

time knowledge is acquired from the RCSS, the player's observation, and the role which is derived from parameters, formation, and specialized plans. Compared to real-world soccer, prior knowledge is static training or strategy knowledge acquired prior to the match; on the other hand, run-time knowledge is acquired during the match from within the player itself or through communication among teammates.

Desires are represented by *modes* suggested by a previously executed *mode*, goals to achieve in the adopted *mode*, and observed teammates' intentions. Suggested *modes* are considered by the *decision process* in choosing the best suggested *modes* for the next execution. Figure 3.9 shows the *desires matrix* listing all desired *modes* suggested by each *mode* upon completion of a *mode*'s execution. There are various goals in each *mode*. A player's *desires* are extended to the goals in the suggested *modes* since the result of *mode* execution relies on its goals' fulfillment. Observed teammate intentions will also influence the player's desires in the decision process which will be discussed in detail in Section 3.8.

Intentions are reflected in a player's commitment to a *plan*, assigned role, and adopted formation. Commitment is reflected in player's persistence in trying to execute a *mode* until one of the goals of the *mode* is achieved. The role assigned to a player is also considered as an intention. Similar to role, a formation is adopted from a set of formations by all players in TA09. Both role and formation adoptions are considered as a player's intention and commitment to retain its role and formation until there is a need to do otherwise.

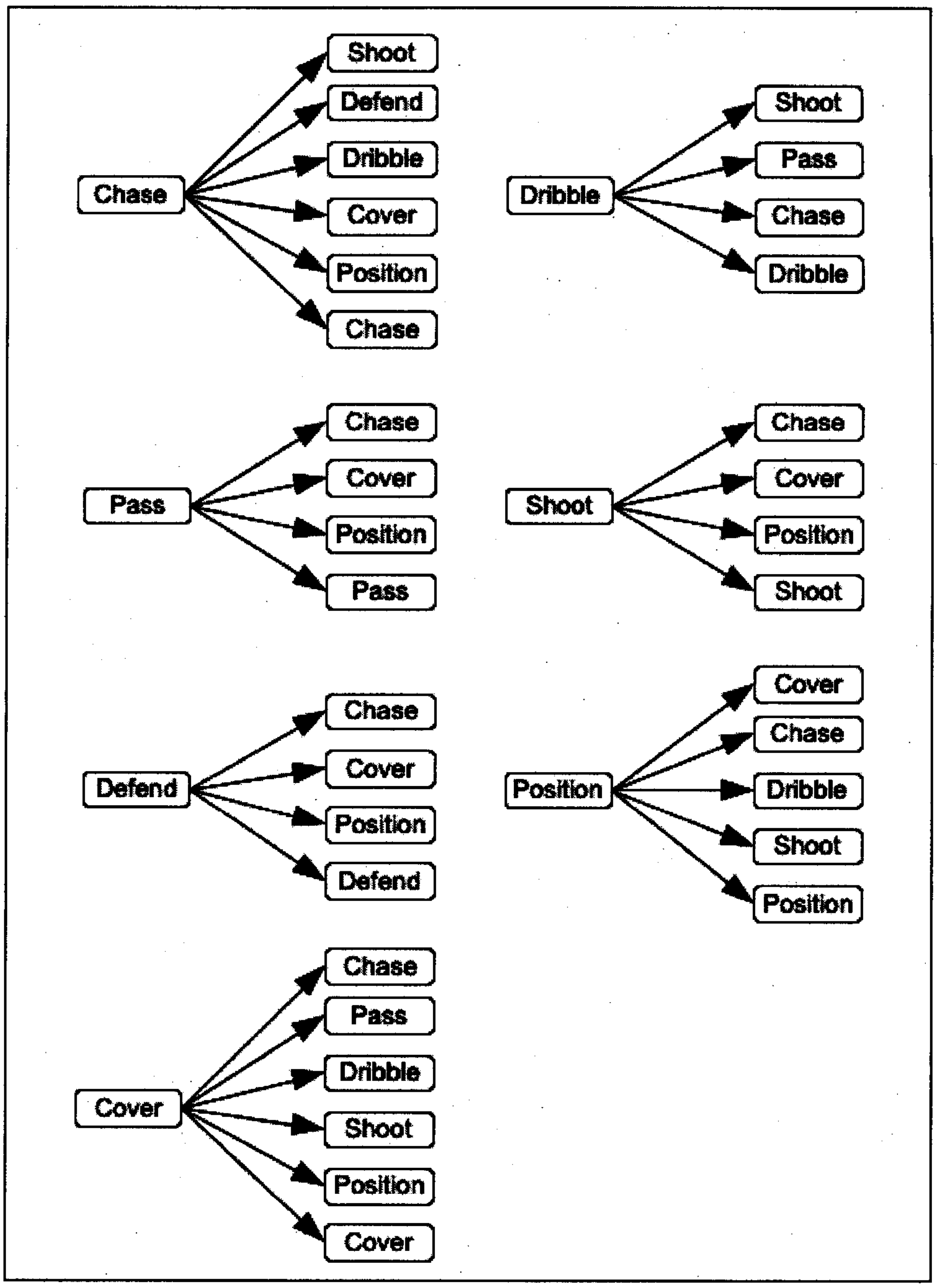

Figure 3.9: Desires matrix of each mode.

3.6 Observation-based Coordination

In the noise-added environment such as RoboCup Soccer Simulator a timely yet accurate communication is a challenge. Initially we looked into using direct audio communication and the *announce-propose* method for coordination but the result was neither favourable nor realistic since it requires a player yell to announce its plan or intention to other teammates periodically. This communication method can be easily eavesdropped although it can be secured by using simple message encryption. The current tested version of the RoboCup Simulator server does not drain a player's stamina for excessive audio communication use. Realistically a soccer player could lose stamina if the player needs to yell constantly during the course of the game. Instead we used an observation-based coordination approach to speculate for a teammate's intention via the teammate's distance to the ball; this approach relies on visual features and has greater range compared to audio communication approach, along with the benefit of not having to deal with eavesdropping. On a period of server cycle, each player will observe visible teammates and remember the distance to the ball in order to deduce the observed teammate's intention on the next observation. In a real soccer game a strategy or set-play is often being initiated by a player on certain situations with the implicit assumption that teammates will behave accordingly. During training, a coördinated strategy is being exercised repeatedly in a simulated situation in order to avoid misunderstanding and to improve coordination. The knowledge gained from training can be constructed as *plans* and associated *complement plans* in the plan library. If the distance between a teammate and the ball is within dribbling range then most likely that teammate is in *dribble* or *chase mode.* When the distance between the teammate and the ball is shrinking but greater than dribbling range then the teammate's possible modes are *chase*, *cover*, or *position mode*; when that distance is growing then that teammate is probably going back to its position in *position* or *cover mode.*

Figure 3.10 STEP 1 shows A, a midfield player, positioned around the center of

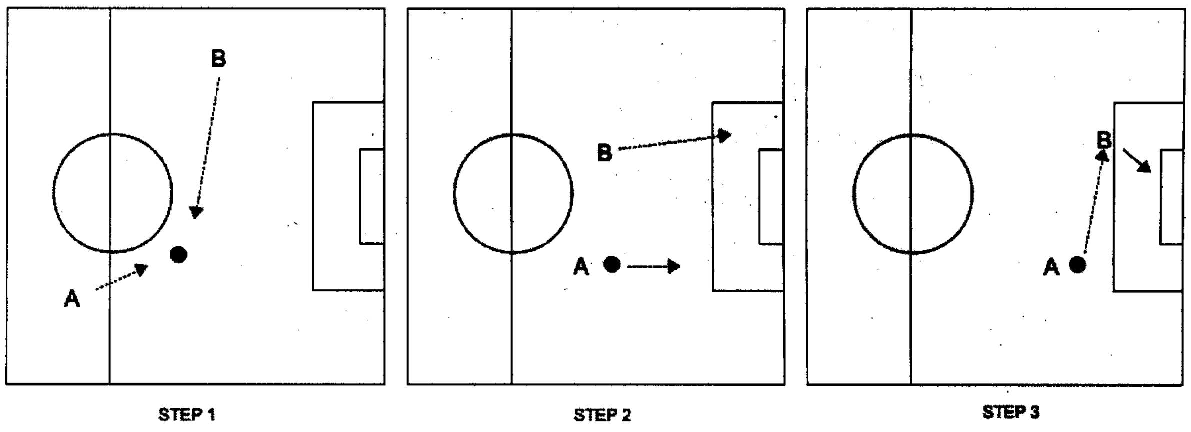

Figure 3.10: Observation-based coordination.

the field and *B*, a forward teammate, who has moved deep into the opponent's field. *A* has adopted *chase-dribble-pass plan* and executing *dribble mode* in STEP 1. *Dribble mode* for a midfield player is set in parameters to dribble to opponent's penalty box area. *B* observed *A* and deduced that *A* is in *dribble mode*. *B* then searched the plan library for a *complement plan* of any *plan* with *dribble* mode and found the *complement plan* of *position-cover-shoot*. Then in STEP 2 *B* adopts *position-cover-shoot* and positions itself to a strategic position as a forward player before entering into *cover* mode. After reaching goal of *dribble*, *A* will execute *pass* and look for the best teammate to pass to as shown in shown in STEP 3. Depending on the current position of both players and distance between them, *B* may be the best candidate for *A* to pass the ball to. If so, *A* will pass to *B* and *B* will chase the ball, as part of *cover mode*, before proceeding to shoot in *shoot mode* as shown in STEP 3. This coordination happens without any communication and relies solely on observation and inference.

3.7 Situation Analysis

Mood is a distinct state of behaviour that a player adopts based on a situation during the match. Ball position and direction are the driving factors in changing the *mood* of a player. The three *moods* used in TA09 are *defensive*, *offensive*, and *neutral mood*. *Defensive mood* is adopted when the ball is deep inside the home area and is configurable by defensive line *parameter*. *Offensive mood* is set when the ball is inside the opponent's field and is configurable by the offensive line *parameter*. Outside those two *moods* the player will default to *neutral mood*. The *mood* feature will influence a player's decision to prefer a certain *plan* over other *plans* in the plan library during *plan* changing.

3.8 Decision Process

Decision process regulates the interaction of all TA09 components into a working rational agent. Figure 3.11 shows *decision process*, the main component in the TA09 architecture for making decisions based on influences from BDI and non-BDI components. Initially a default *plan* is selected by an agent and the first *mode* of the selected *plan* is executed to produce a list of desired *modes*. For example, after *chase mode* is executed, the logical desired *modes* are *dribble*, *pass*, *kick*. Desired *modes* are tested for their preconditions and the first mode which passes its preconditions is selected for execution.

Intention observation is handled in the *intention explore* stage by comparing every *mode* of every *plan* in the plan library with the possible observed *mode*. If a *mode* of a *plan* is matched with one of the possible observed *modes* then it means there is a chance for a *complement plan* to coordinate with the observed teammate's plan. Each *complement plan* of the matched *plan* for a matching position is then checked. A matched position implies that the observer can execute the *complement plan* expecting the observed teammate to continue its current intended *plan*. This process is a *loosely-coupled observation-based coordination* and only the observer needs to react or plan to match with the observed teammate.

If there is no observed intention, *decision process* will check if the last *mode* was

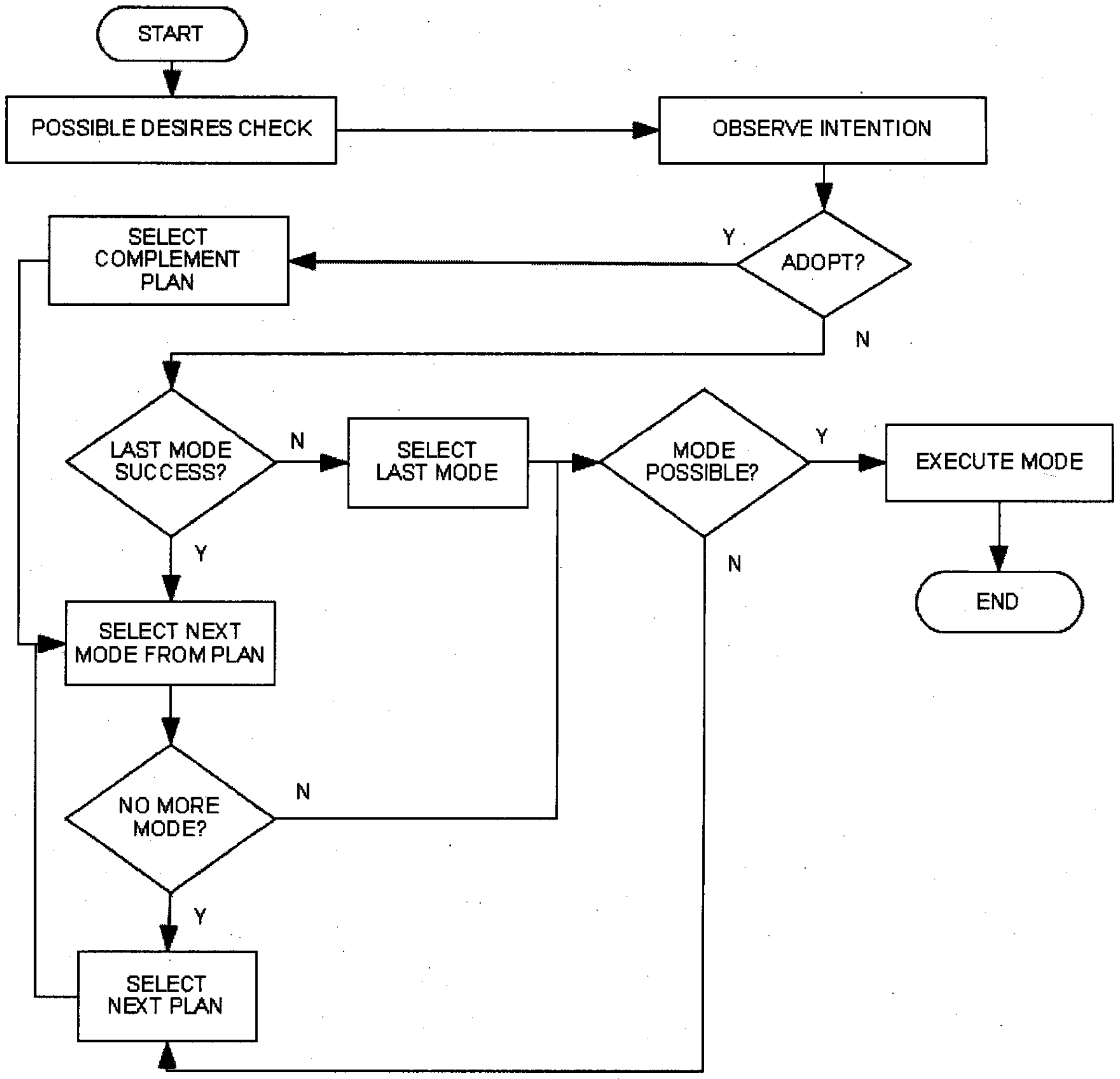

Figure 3.11: *Decision process* diagram.

executed successfully. If execution of a *mode* returns false, *i.e.* the *mode*'s goals have not been reached, *decision process* will keep executing that *mode* until it returns a true value. If a true value is returned, the next *mode* in the *plan* is selected for testing and execution. Commitment will ensure that a *mode* in a *plan* is always executed as long as it is still possible to execute it. If it is not possible by all means, the agent will drop the commitment on that *plan* and find another *plan* from the plan library for testing and execution. After the last *mode* in a *plan* is executed successfully the *plan* is dropped and the next *plan* from the plan library is adopted.

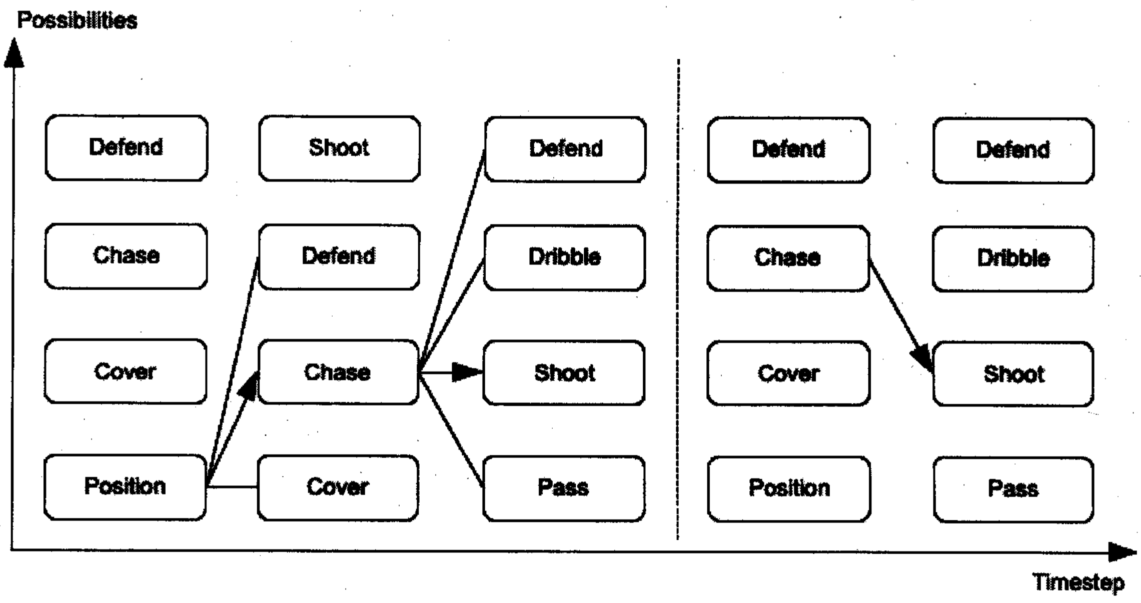

Figure 3.12: Decision path example.

Figure 3.12 shows a decision path where the y axis is listing all possible modes at that particular decision cycle and the x axis is listing all decision cycles. Possible *modes* are shown as regular lines and the chosen mode is shown as an arrow line. A disconnected line in the decision tree indicates the beginning of a new *plan* or execution of a default *mode* when none of the desired *modes* are possible to execute.

Chapter 4

Evaluations

Comparative and subjective evaluations are used to determine the extent to which a soccer simulation team is achieving its overall, predetermined BDI objectives. Comparative evaluation is used to compare the performance of one soccer simulation team against another soccer simulation team whilst subjective evaluation assesses the BDI and non-BDI aspects of TA09 team with respect to CMUnited99, FCPortugal2001, and ATHumboldt98.

4.1 Comparative Evaluation

In this study, a comparative evaluation is used to measure the performance of TA09 against CMUnited99 [28] and FCPortugal2001 [25] teams. CMUnited99 is matched against itself as a baseline experiment. Two 100 match experiments are performed for each team, one without a goalkeeper (NG) and another with a goalkeeper (G). Each match has two 300 second halfs with no extra halfs or penalty shootout. The goalkeeper is the last line of defence and is distinctly different from other non-goalkeeper roles such as defender, midfielder, or striker. Our thesis focus is on the non-goalkeeper aspects of RoboCup but a soccer team is not complete without a goalkeeper so we decided to use FCPortugal2001's goalkeeper in all teams in G matches.

4.1.1 CMUnited99 vs CMUnited99′

Default CMUnited99 game play for every players is chase the ball and then shoot (hard *kick*()) at the opponent's goal immediately without any passing or dribbling.

CMUnited99	Win	Lose	Draw	Avg Goals For	Avg Goals Against
Without Goalkeeper (NG)	46	38	16	6.52	6.43
With Goalkeeper (G)	34	41	25	1.19	1.37

Figure 4.1: CMUnited99 vs CMUnited99′ results.

In the NG experiment all players from both teams keep swarming around the ball whenever the ball is moved by the player which is closest to the ball. Player congestion is mostly in the mid-field and almost all the goals scored are from a stray ball during heavy kicking from mid-field or near the penalty box area. CMUnited99 won slightly more NG matches than CMUnited99′ but both teams have average goals scored close to each others average.

In the G experiment a similar pattern is shown but goalkeepers from both teams manage to save many balls shot intentionally or simply stray. The difference in number of goals between CMUnited99 teams in both NG and G experiments is fairly low which shows that both teams performed equally after taking into account simulation random variables such as wind factor. These baseline experiment results confirmed that our comparative evaluation setup is suitable for contrasting the performance of TA09 against CMUnited99 and FCPortugal2001 teams.

4.1.2 TA09 vs CMUnited99

TA09	Win	Lose	Draw	Avg Goals For	Avg Goals Against
Without Goalkeeper (NG)	67	22	11	4.72	2.66
With Goalkeeper (G)	49	23	28	1.11	0.63

Figure 4.2: TA09 vs CMUnited99 results.

In the NG experiment CMUnited99 players swarmed the ball and left many opportunities for the TA09 players to counter attack. Most goals from CMUnited99 were from stray ball shot as a result of congestion between CMUnited99 swarming players and TA09 defenders. TA09 attacks mostly started by passing among teammates in a spread-out formation—dribbling toward the opponent's area whenever possible—and shooting for the goal in CMUnited99's defensive zone. TA09 offensive attacks were effective and resulted in almost triple the number of matches won compared to CMUnited99 and almost double the average goals scored compared to CMUnited99.

In the G experiment, the CMUnited99 chase & shoot algorithm is shown to be quite effective scoring goals when used in the proximity of the penalty area. A sudden strong shot from one of CMUnited99 swarming players may sneak past TA09 defenders and leave TA09 goalkeeper incapable of catching the ball in time. Most CMUnited99 goals were scored from this type of situation. TA09 goals came from relentless attacks by forward players and "clever" passes from midfield players to forward players away from the congested CMUnited99 players.

4.1.3 TA09 vs FCPortugal2001

TA09	Win	Lose	Draw	Avg Goals For	Avg Goals Against
Without Goalkeeper (NG)	43	38	12	2.64	2.77
With Goalkeeper (G)	33	13	54	0.77	0.55

Figure 4.3: TA09 vs FCPortugal2001 results.

In the NG experiment, FCPortugal2001 players line up at the center line as their default positioning and one forward player is sent forward to attack and kick as hard as possible to the TA09 goal before returning to the center line. On defence all FCPortugal2001 players form a two-level zagged line at the middle of their field retracting from their default position from center line. If any ball passes the first line, the second line players intercept. This lining up strategy has proven really effective to block TA09 dribbling players and to nullify passes and even shots. TA09 kept trying to spread out players and passes followed by dribbling but the ball is almost always intercepted by FCPortugal2001 players. FCPortugal2001 offense was effective due to the non-existent TA09 goalkeeper and spread-out TA09 players. TA09 came out slightly better than FCPortugal2001 in the number of matches won but the average goals is a bit less than FCPortugal2001.

In the G experiment, goalkeepers on both team managed to prevent many goals. FCPortugal2001 goals came mostly from counter attacks when TA09 is all out attacking leaving only the goalkeeper and a few defenders to defend. TA09 goals were scored from successful passes from midfield to forward players and from a successful all-out attack beating FCPortugal2001 defenders and goalkeeper. Effective goalkeeping from both teams contributed to the high number of tied matches. FCPortugal2001 attacks and counter attacks utilizing a single forward player detached from the zagged line formation can be easily defeated by the TA09 goalkeeper. Most TA09 attacks however are built from several players shooting in the FCPortugal2001 penalty box area resulting in the higher number of won matches and average goals compared to FCPortugal2001.

4.2 Subjective Evaluation

A subjective evaluation is taken to analyze key features of TA09 in comparison with CMUnited99 [28], FCPortugal2001 [25], ATHumboldt98 [5, 6] and a typical human soccer team. Criteria for subjective evaluation are: adherence to Bratman's [3] philosophical approach (four theses for BDI and planning in particular); planning and reasoning; behaviours, roles and formations; coordination; and simplicity of overall design. The BDI portion of subjective evaluation will compare TA09 with mostly ATHumboldt98 team because ATHumboldt98 has an explicit BDI implementation. Non-BDI based CMUnited99 and FCPortugal2001 are included in the comparison whenever applicable.

4.2.1 Belief-Desire-Intention

Beliefs are presented in the World Model component of ATHumboldt98 [5, 6] in which all RCSS sensors are processed to construct an agent's world. TA09 differentiates an agent's knowledge or world model into prior knowledge and run-time information. Prior knowledge is constructed from parameters, formations, a plans definition library, and specialized plans. TA09 run-time information is exactly the same as ATHumboldt98 World Model component; it is also similar to CMUnited99 and FCPortugal2001 implementations. The inherent design of RoboCup Simulation requires each player to process periodic new sensor information and to construct a player's world which is used in planning and decision making. Practically all competitive RoboCup simulation teams have similar types of world models constructed which fit naturally into Bratman's belief model.

In ATHumboldt98 *desires* are possible goals selected out of a fixed goal library. These goals can be achieved by adopting an appropriate *option* which has its utility estimation and planner. Available accessible *options* are essentially desires in ATHumboldt98. The ATHumboldt98 concept of desire is similar to TA09 concept of desire—presented in the *desires matrix* and *decision process*—due to their common foundation on Rao and Georgeff [22] *possible worlds semantics* approach in which *desires* are presented by possible worlds in a branching tree structure.

Commitment to *intention* is being regulated by the ATHumboldt98 BDI Reasoning Process by fixing on intended *option* until *option*'s goal is reached. Once committed, ATHumboldt98 player will not plan on choosing another *option.* ATHumboldt98 intention is fixed on an *option* and TA09 intention is fixed on a *plan* as a series of modes. In TA09 commitment to a *mode* is shown by persistency in executing that *mode* until its goals are reached as long as the preconditions of that *mode* are fulfilled. In addition, TA09 has a *joint-intention* feature, a concept of collaborative intention introduced by Jennings [16], that does not exist in ATHumboldt98.

4.2.2 Planning and Reasoning

CMUnited99 [28] has low-level skills such as *get_ball()*, *pass_ball()*, and *kick_ball()* which can be considered as an agent's state of mind because they are committed on execution over a duration of time to achieve certain goals. FCPortugal2001 [25] has various *actions* such as *shoot*, *pass*, *dribble*, and *cover goal* grouped into *ball possession* and *ball recovery* modules of SBSP. These *actions* are committed and executed over a period of time. In ATHumboldt98 [5, 6] the *option* selected to achieve desired goals over a period of time is governed by Reasoning Process. In TA09, *mode* is similar to CMUnited99 low-level skills, FCPortugal2001 *actions*, and ATHumboldt98's *options* except *mode* is at a more abstract level which can be linked in a series as a *plan* to construct a plan library. TA09's model of intention is aligned with Cohen and Levesque's [8] *persistent goal* which models an agent's chosen and committed state of mind to a course of events over a period of time. Goals are embedded in each *mode* and collectively in the *plan*. TA09 *mode* is an agent's state of mind constructed with specific actions to achieve certain goals. Braubach *et al.* [4] classifies this type of goal as an *achievement goal* where action is performed until a certain condition is reached.

CMUnited99 [28] uses a trained Decision Tree (DT) to choose various low-level skills such as passing, dribbling, or kicking in appropriate situations. FCPortugal2001 [25] strategy information consists of several Tactics which activated and adopted by players depending on various policies. ATHumboldt98 [5, 6] Planning component is initiated whenever new sensor information is received by an ATHumboldt98 player and is done in two stages. The first stage is to choose the best possible desired goal and commits to it as the intention of the long term planning. In the second stage, the Planning component will execute partial planning in the Advanced Skills component to produce short-term planning that fits between two RCSS cycles. TA09 plan library specifies a series of general steps to execute while CMUnited99's DT relies on offline training to choose is the best action to take depending on the players' positioning and distance. FCPortugal2001 Tactic is driven by preset Situations chosen by situation analysis during the match.

The ATHumboldt98 [6, 5] BDI Reasoning Process is executed on every RCSS cycle. The first stage is checking on the continuation of intention for an *option*. If there is no commitment then constraints, desires, and intention determination stages will be executed followed by planning and execution of the chosen intention. If there is commitment to a particular intention, the reasoning process will skip the determination stages and continue planning and executing the chosen intention. The TA09 *decision process* will keep executing a *mode* until its goal is reached while still checking the precondition of that *mode* prior to execution. Commitment to a plan is dropped successfully when the last *mode* in the plan is executed successfully when its goal is reached or dropped prematurely when the intended *mode* cannot be executed due to its preconditions not met.

CMUnited99 [28] utilizes offline training with a Neural Network and a Decision Tree for low-level skills and higher "social" skills, *i.e.* passing, respectively. FCPortugal2001 [25] uses predefined strategy information in *strategy.conf* that will influence player behaviours depending on the match situation. ATHumboldt98 [5, 6] uses predefined parameter values in player's decisions and skills. CMUnited99 offline training is much more sophisticated and it can be trained against different RoboCup teams. FCPortugal2001, ATHumboldt98, and TA09 rely on offline hand-crafted parameters in fine-tuning player's behaviours.

TA09 planning is partial at *plan* adoption time and complete when the last *mode* in that *plan* is executed successfully. During the execution of a *plan*, a TA09 player resists reconsideration of other modes suggested by the *desire matrix* as long as that *plan* is still possible to be carried out according to the TA09 player's beliefs. These characteristics fit Bratman [3] description of planning by a rational agent and commitment of Cohen and Levesque [8].

4.2.3 Role, Formation, and Coordination

Role and positioning are treated as a single component in CMUnited99 [29]. Each role will have a different behaviour and will retain its relative position unless its formation is changed. For example, upon formation change via Periodic Team Synchronization (PTS), a midfielder player can switch to defender role or forward role. Formation change can be initiated by any player which later propagates to the rest of the team during the match. FCPortugal2001 [25] uses Situation Based Strategic Positioning (SBSP) by dynamically adjusting individual positions depending on various policies during the game. On the other hand, Dynamic Positioning and Role Exchange (DPRE) is used to calculate positioning based on role similar to CMUnited99. Both Lau and Reis [20] and Stone *et al.* [29] reported their success with dynamic formation changes during the match. TA09 positioning is similar to CMUnited99 formation positioning and FCPortugal2001 SBSP: a player's position in a formation is the starting point and, depending on the ball position and direction, a TA09 player extracts or retracts within its default formation position. The TA09 implementation of role is implemented using parameters and a specialized plan library to form defender, midfielder, and forward roles.

CMUnited99 uses a run-time formation-switching algorithm based on score difference and time left during the match. This allows CMUnited99 to switch formation and role that will influence players' behaviours. FCPortugal2001 analyzes various match conditions, such as ball possession, ball position-velocity, and all players beliefs to assign one of the preset Situations that will activate Tactic. ATHumboldt98 [6, 5] relies on utility estimation in each *option* during planning. The utility estimation analyzes the probability of a successful result from the *option* selection. TA09 uses ball position and direction to set the *mood* in each player to react to the situation. *Mood* will influence preference of plan selection in the *decision process* to adjust players' behaviours.

Periodic Team Synchronization (PTS) is used in CMUnited99 by utilizing direct audio communication to propagate formation information, ball location, and other useful information among teammates. FCPortugal2001 relies on mutual knowledge of decision rules by all players to allow them to predict each other's decision. In the role exchange case, audio communication is used to avoid conflicting prediction due to an agent's local perspective. Burkhard *et al.* did not mention any specific communication method for team coordination in ATHumboldt98. The TA09 *observation-based coordination* feature is influenced by Jennings' [16] specification of *joint-intention* where an agent acknowledges that actions performed by other agents are related to its intention. Although joint-intention can be done by collaborating agents all agreeing on a common goal, we chose to implement *one-sided* or *loosely-coupled* joint-intention to avoid communication overhead and complex negotiation.

Strategical positioning and coordination are key factors in winning a soccer game. CMUnited99 *set-plays* are predefined coordination specifying position and role triggered by specific situations typically announced by the referee such as corner kick, free kick, *etc.* In a set-play situation, a player role will change to a specific role to perform behaviours required for the *set-plays.* The *set-play* role is dropped after the execution of the *set-play.* FCPortugal2001 SBSP uses preset *Tactics* containing formations that determine players' positionings and roles to strategically mobilize players depending on various policies during the match. ATHumboldt98 does not have specific predefined multi-player coordination. TA09 relies on *complement plans* together with intention observation to execute a strategical plan. A complement plan does not rely on predefined positionings and roles like in CMUnited99 nor does it tie into *Formation-Tactic* found in FCPortugal2001. Teammate intention is the sole factor in triggering a complement plan thus allowing the flexibility of being executed at any time without complex coordination and communication.

4.2.4 Design and Architecture

CMUnited99 [29], FCPortugal2001 [25], and ATHumboldt98 [5, 6] are designed from the ground-up for RoboCup 2D Simulation; hence none of those team architectures can be easily adaptable to other real-time Multi-Agent System applications. On the other hand, TA09 architecture for BDI components such as *mode*, *plan*, and *decision process* can be generalized for other real-time MAS applications. The *mode* concept is suitable as an abstraction of partial planning and can be replaced with a specific target MAS implementation while still representing an agent's state of mind. Loosely coupled *modes* along with the *desires matrix* are components that would require implementation changes according to the target MAS application. Both the *plan* and the *decision process* concepts are generic enough to be used without much change. *Plan* connects a string of *modes* together and combined with commitment to a *plan* forms persistent planning. Commitment can be adjusted to be short-term or reactive by committing only on *mode* or longer term by committing on *plan*. Intention observation part can be omitted if joint-intention feature is not needed in target MAS implementation.

4.2.5 Conclusion

We created a successful RoboCup 2D Simulation team as shown in the Comparative Evaluation. In the Subjective Evaluation we showed that TA09 stays faithful to Bratman's model of rational agent with improvements from Jennings' joint-intention and Cohen and Levesque's commitment. The flexibility in TA09 allows behaviour modifications by simply changing configuration files. In the next chapter we will discuss all other aspects beyond the evaluations we did in this chapter.

Chapter 5

Discussion

In this chapter we will discuss issues encountered during TA09 design, implementation, testing, and evaluations. We will point out some of TA09 limitations along with suggested improvements for future work.

5.1 Design and Implementation

Existing works related to BDI, MAS, RoboCup, together with CMUnited99 [29] publicly available source code, provided the foundation for this project. The TA09 implementation started with a simple RoboCup client prototype using Python [10]. This prototype was replaced with an adaptation of CMUnited99 into basic TA09 with BDI and non-BDI components added. We spent approximately three man-month time on TA09 in BDI design, implementation, improvement, and testing. The bulk of the time was spent on implementing *mode*, plan library, various improvements via parameters adjustments during testing, and common features needed to compete in RoboCup competition at a reasonable skill level such as formation, positioning and roles.

5.2 Testing and Evaluations

Once the TA09 BDI components were implemented and tested, we started to add non-BDI components during testing to the point that the TA09 team demonstrates the play level of a typical RoboCup team and mimicks real soccer team play. Testing on the TA09 components are done via various test case scripts that simulate specific conditions, *e.g.* under attack or attacking, and with specific parameters and plan setups, and player positionings. We observed test matches by watching *Monitor* and verified the results from TA09 log outputs.

During TA09 vs CMUnited99 experiments, we adjusted parameters and *plans* to be able to counter the CMUnited99 chase & shoot behaviour. *Plans* in defender and midfielder roles are configured to pass more often in the defensive and neutral *mood*. This change to passing effectively made TA09 players to spread-out based on their formation positions to allow better congestion avoidance and strategical positioning, which in turn enabled TA09 to play better than initially against CMUnited99. Likewise in the TA09 vs FCPortugal2001 experiments, we did similar tweakings with TA09 *parameters* and *plans* to break FCPortugal2001's strong defense. With simple changes to *plans* and the configuring parameters, TA09 outperformed CMUnited99 and FCPortugal2001 teams.

5.3 Limitations

Although the TA09 implementation showed desirable results, there are limitations that could be overcome by future improvements and competing at RoboCup competitions. We believe that once these limitations are eliminated along with some further improvements the TA09 team will be able to compete at the current RoboCup competition level. In the next few paragraphs we will point out the TA09 limitations along with possible areas of improvements based on our experience during this research.

The *chase*, *dribble*, *pass*, *shoot*, *defend*, *position*, and *cover* modes can be enhanced further for improved performance. A better trajectory calculation for ball interception and obstacle avoidance could be beneficial to the *chase*, *dribble*, *pass*, and *cover* modes. *Shoot* mode can use online machine learning during the match to learn to beat the opponent's goalkeeper. Pattern recognition can help the *defend* mode to detect attack patterns or set-plays accurately. The *position* mode could be improved along with better formation and positioning which will be discussed in the next few paragraphs.

Currently the plan library is static and it is loaded once upon the start-up of each player. The plan library could be dynamic—*i.e.* player could add, change, or delete plan library during run-time—to allow greater flexibility and extensibility compared with the current offline configuration. Each specialized plan carries a weight assigned to each *mood*. *Plans* with the same weight will be treated equally with prioritization given to the listing order of the plans in the library. Real-valued weight values that can be dynamically adjusted on the basis of the success of the *plan* execution could improve the performance and accuracy of the *decision process* in selecting a *plan*. A feedback-based or machine-learning approach can be used in assigning real-valued weight values to each plan during the match or as a result of offline training. An approach similar to Guerra-Hernandez *et al.* [2] machine learning in BDI can be added to the *decision process* to learn the priority of a plan suitable for a given match situation.

Both Lau and Reis [20] and Stone *et al.* [29] reported on their success with dynamic formation changes during the match despite their differences in implementation. TA09 could use dynamic formation change based on match situation using various predefined rules. Ideally formation change is initiated by the captain or coach—similar to a real soccer game—and propagated to the rest of the team either via observation or audio communication means.

Role switching is particularly useful when a player needs to switch roles because the player is in a strategic position *i.e.* a defender is in an attacking position. Lau and Reis [20] and Stone *et al.* [29] reported successful implementation of flexible role assignment in their FCPortugal2001 and CMUnited99 respectively. Hexmoor and Zhang's [12] *positional role tree* could be implemented to allow a player skills specialization depending on player's position in a team formation. Role is typically related to a position in a formation hence ideally role switching, implemented using *positional role tree* would be coupled with dynamic formation change in TA09. The intended role can persist until the player drops the current intention due to changing preconditions, such as formation change or ball opportunities, that influenced the player to adopt the different role.

Any implementation of the audio proposal is handicapped by a limited audio range, a noisy environment, and the unrealistic periodical hollering of the player during the game. Instead of using the audio proposal TA09 uses intention observation for coordination. Auditory communication is an effective and precise method to use in close range situations. Stone *et al.* [26, 29] reported great success in audio communication in CMUnited99 PTS implementation. Information propagation via audio means could be a great complement to the TA09 observation component. Obscuring the message and error-checking method are needed to ensure messages are not eavesdropped and are properly received by all of the intended teammates. The intended plan, formation change, and role switching are a few areas that could benefit from close range audio communication. The use of audio should not be excessive and should be limited to short-range communication only. In the current RCSS version *say* command does not consume stamina but realistically it is hard for a player to keep hollering during ball chasing. *Say* command could drain player's stamina in the future RCSS version for a more realistic simulation.

Observation could be improved by tracking more teammates and opponents instead of just the ball position and direction. Plan prediction similar to Huber and Durfee's [14] *plan recognition* and Rao and Murray [24] *reactive plan recognition* could be used to better predict a teammate's plan compared to just the *mode* as in the current TA09 implementation. TA09 observation also sets player's *mood* as *defensive, neutral,* and *offensive.* Non-discrete *mood* values coupled with the plan's weight will allow smoother *mood* transition. A risk factor could be added as part of the observation component for better plan selection in the *decision process.* The risk factor will utilize system feedback via reward or penalty similar to Ahmadi and Stone's [1] instance-based action model for action planning.

Chapter 6

Conclusions and Future Work

In this paper, the Belief-Desire-Intention (BDI) approach— based on Bratman's belief-desire theory, influenced by Cohen and Levesque's [8] commitment, and incorporating Jennings' [16, 17, 15] joint-intention in a novel way—is implemented successfully in the TA09 for RoboCup 2D Simulation. Our BDI implementation addressed various facets of belief-desire theory beyond ATHumboldt98 implementation while extending single agent planning aspects into multi agent coordination via joint-intention. Comparative evaluation results show moderate improvements in TA09 over CMUnited99 and FCPortugal2001 in terms of won matches and average goal scored. In the subjective evaluation we showed several advantages of the TA09 design in the BDI and non-BDI components such as *mode, plan,* and observation-based coordination compared to ATHumboldt98, CMUnited99, and FCPortugal2001.

In order to compete at the RoboCup competition level TA09 requires further fine-tuning in *modes*, a role switching functionality, and improvements in players' observation; we considered those key areas to be the priority of the next iteration of the TA09 team in the future. TA09 BDI architecture is not tied to just RoboCup domain specifically but can also be implemented for real-time Multi-Agent Systems (MAS) applications. *Mode* is a domain dependent abstraction of an agent's state-of-mind hence it should be implemented according to the target MAS application along with a plan library for planning capabilities. Our future plan for TA09 BDI improvements are: a time-sensitive decision process, a dynamic plan library, and a feedback-based plan prioritization.

The performance of TA09 demonstrates that the BDI model of a rational agent can excel in RoboCup competition. Furthermore, the BDI approach in TA09 can be adapted for other real-time MAS applications as a flexible rational agent design.

Appendix A

Source codes

A.1 TA09 team

http://www.csd.uwo.ca/ mercer/TA09

A.2 CMUnited99 team

http://www.cs.cmu.edu/ pstone/RoboCup/CMUnited99-source.tar.gz

A.3 FCPortugal2001 team

http://www.ieeta.pt/robocup/documents/FCPAgent.tgz

Appendix B

RoboCup Simulation Server configuration

B.1 server.conf

```

/* server Configuration file */

##########################################################
# TA09 simulation experiment configs change
# from stock config file in CMU package:
#
# server::drop_ball_time = 40    # Faster kick-off.
# server::wind_random = true     # Set random wind factor.
#
##########################################################

# server::catch_ban_cycle
server::catch_ban_cycle = 5

# server::clang_advice_win
server::clang_advice_win = 1

# server::clang_define_win
server::clang_define_win = 1

# server::clang_del_win
server::clang_del_win = 1

# server::clang_info_win
server::clang_info_win = 1

# server::clang_mess_delay
server::clang_mess_delay = 50

```

```
# server::clang_mess_per_cycle
server::clang_mess_per_cycle = 1

# server::clang_meta_win
server::clang_meta_win = 1

# server::clang_rule_win
server::clang_rule_win = 1

# server::clang_win_size
server::clang_win_size = 300

# server::coach_port
server::coach_port = 6001

# server::connect_wait
server::connect_wait = 300

# server::drop_ball_time
#server::drop_ball_time = 200
server::drop_ball_time = 40

# server::freeform_send_period
server::freeform_send_period = 20

# server::freeform_wait_period
server::freeform_wait_period = 600

# server::game_log_compression
server::game_log_compression = 0

# server::game_log_version
server::game_log_version = 3

# server::game_over_wait
server::game_over_wait = 100

# server::goalie_max_moves
server::goalie_max_moves = 2

# server::half_time
server::half_time = 300

```

```
# server::hear_decay
server::hear_decay = 2

# server::hear_inc
server::hear_inc = 1

# server::hear_max
server::hear_max = 2

# server::keepaway_start
server::keepaway_start = -1

# server::kick_off_wait
server::kick_off_wait = 100

# server::max_goal_kicks
server::max_goal_kicks = 3

# server::nr_extra_halfs
/* Number if extra-time periods in a game if it is drawn */
#server::nr_extra_halfs = 2
server::nr_extra_halfs = 0

# server::nr_normal_halfs
/* Number of normal halfs in a game */
server::nr_normal_halfs = 2

# server::olcoach_port
server::olcoach_port = 6002

# server::pen_before_setup_wait
server::pen_before_setup_wait = 30

# server::pen_max_extra_kicks
server::pen_max_extra_kicks = 10

# server::pen_nr_kicks
server::pen_nr_kicks = 5

# server::pen_ready_wait
server::pen_ready_wait = 50

# server::pen_setup_wait
```

```
server::pen_setup_wait = 100

# server::pen_taken_wait
server::pen_taken_wait = 200

# server::point_to_ban
server::point_to_ban = 5

# server::point_to_duration
server::point_to_duration = 20

# server::port
server::port = 6000

# server::recv_step
server::recv_step = 10

# server::say_coach_cnt_max
server::say_coach_cnt_max = 128

# server::say_coach_msg_size
server::say_coach_msg_size = 128

# server::say_msg_size
server::say_msg_size = 512 # CMU

# server::send_step
server::send_step = 150

# server::send_vi_step
server::send_vi_step = 100

# server::sense_body_step
server::sense_body_step = 100

# server::simulator_step
server::simulator_step = 100

# server::slow_down_factor
server::slow_down_factor = 1

# server::start_goal_l
server::start_goal_l = 0
```

```

# server::start_goal_r
server::start_goal_r = 0

# server::synch_micro_sleep
server::synch_micro_sleep = 1

# server::synch_offset
server::synch_offset = 60

# server::tackle_cycles
server::tackle_cycles = 10

# server::text_log_compression
server::text_log_compression = 0

# server::auto_mode
server::auto_mode = false

# server::back_passes
server::back_passes = true

# server::coach
server::coach = false

# server::coach_w_referee
server::coach_w_referee = false

# server::forbid_kick_off_offside
server::forbid_kick_off_offside = true

# server::free_kick_faults
server::free_kick_faults = true

# server::fullstate_l
server::fullstate_l = false

# server::fullstate_r
server::fullstate_r = false

# server::game_log_dated
server::game_log_dated = true

```

```
# server::game_log_fixed
server::game_log_fixed = false

# server::game_logging
server::game_logging = true

# server::keepaway
server::keepaway = false

# server::keepaway_log_dated
server::keepaway_log_dated = true

# server::keepaway_log_fixed
server::keepaway_log_fixed = false

# server::keepaway_logging
server::keepaway_logging = true

# server::log_times
server::log_times = false

# server::old_coach_hear
server::old_coach_hear = false

# server::pen_allow_mult_kicks
/* Turn on to allow dribbling in penalty shootouts */
server::pen_allow_mult_kicks = true

# server::pen_coach_moves_players
/* Turn on to have the server automatically position players for
peanlty shootouts */
server::pen_coach_moves_players = true

# server::pen_random_winner
server::pen_random_winner = false

# server::penalty_shoot_outs
/* Set to true to enable penalty shootouts after normal time and extra
time if the game is drawn.
To have the game go straight into penalty shoot outs, set this to true
and nr_normal_halfs and nr_extra_halfs to 0 */
server::penalty_shoot_outs = false

```

```
# server::profile
server::profile = false

# server::proper_goal_kicks
server::proper_goal_kicks = false

# server::record_messages
server::record_messages = false

# server::send_comms
server::send_comms = false

# server::synch_mode
server::synch_mode = false

# server::team_actuator_noise
server::team_actuator_noise = false

# server::text_log_dated
server::text_log_dated = true

# server::text_log_fixed
server::text_log_fixed = false

# server::text_logging
server::text_logging = true

# server::use_offside
server::use_offside = off # CMU

# server::verbose
server::verbose = false

# server::wind_none
server::wind_none = false

# server::wind_random
#server::wind_random = false
server::wind_random = true

# server::audio_cut_dist
server::audio_cut_dist = 50

```

```
# server::ball_accel_max
server::ball_accel_max = 2.7

# server::ball_decay
server::ball_decay = 0.94

# server::ball_rand
server::ball_rand = 0.05

# server::ball_size
server::ball_size = 0.085

# server::ball_speed_max
server::ball_speed_max = 2.7

# server::ball_stuck_area
server::ball_stuck_area = 3

# server::ball_weight
server::ball_weight = 0.2

# server::catch_probability
server::catch_probability = 1

# server::catchable_area_l
server::catchable_area_l = 2

# server::catchable_area_w
server::catchable_area_w = 1

# server::ckick_margin
server::ckick_margin = 1

# server::control_radius
server::control_radius = 2

# server::dash_power_rate
server::dash_power_rate = 0.006

# server::effort_dec
server::effort_dec = 0.005

# server::effort_dec_thr
```

```
server::effort_dec_thr = 0.3

# server::effort_inc
server::effort_inc = 0.01

# server::effort_inc_thr
server::effort_inc_thr = 0.6

# server::effort_init
server::effort_init = 1

# server::effort_min
server::effort_min = 0.6

# server::goal_width
/* The width of the goals */
server::goal_width = 14.02

# server::inertia_moment
server::inertia_moment = 5

# server::keepaway_length
server::keepaway_length = 20

# server::keepaway_width
server::keepaway_width = 20

# server::kick_power_rate
server::kick_power_rate = 0.016 # CMU

# server::kick_rand
server::kick_rand = 0

# server::kick_rand_factor_l
server::kick_rand_factor_l = 1

# server::kick_rand_factor_r
server::kick_rand_factor_r = 1

# server::kickable_margin
server::kickable_margin = 0.7

# server::maxmoment
```

```
server::maxmoment = 180

# server::maxneckang
server::maxneckang = 90

# server::maxneckmoment
server::maxneckmoment = 180

# server::maxpower
server::maxpower = 100

# server::minmoment
server::minmoment = -180

# server::minneckang
server::minneckang = -90

# server::minneckmoment
server::minneckmoment = -180

# server::minpower
server::minpower = -100

# server::offside_active_area_size
server::offside_active_area_size = 5 # CMU

# server::offside_kick_margin
server::offside_kick_margin = 9.15

# server::pen_dist_x
server::pen_dist_x = 42.5

# server::pen_max_goalie_dist_x
server::pen_max_goalie_dist_x = 14

# server::player_accel_max
/* The max acceleration of players */
server::player_accel_max = 1

# server::player_decay
/* Players speed decay rate */
server::player_decay = 0.4

```

```
# server::player_rand
/* Player random movement factor */
server::player_rand = 0.1

# server::player_size
/* The size of the default player */
server::player_size = 0.3

# server::player_speed_max
/* The max speed of players */
server::player_speed_max = 1.0 # CMU

# server::player_weight
/* The weight of the player */
server::player_weight = 60

# server::prand_factor_l
server::prand_factor_l = 1

# server::prand_factor_r
server::prand_factor_r = 1

# server::quantize_step
server::quantize_step = 0.1

# server::quantize_step_l
server::quantize_step_l = 0.01

# server::recover_dec
server::recover_dec = 0.002

# server::recover_dec_thr
server::recover_dec_thr = 0.3

# server::recover_init
/* The intial recovery value for players */
server::recover_init = 1

# server::recover_min
server::recover_min = 0.5

# server::slowness_on_top_for_left_team
server::slowness_on_top_for_left_team = 1
```

```

# server::slowness_on_top_for_right_team
server::slowness_on_top_for_right_team = 1

# server::stamina_inc_max
/* The maximum player stamina increament */
server::stamina_inc_max = 35 # CMU

# server::stamina_max
/* The maximum stamina of players */
server::stamina_max = 3500 # CMU

# server::stopped_ball_vel
server::stopped_ball_vel = 0.01

# server::tackle_back_dist
server::tackle_back_dist = 0.5

# server::tackle_dist
server::tackle_dist = 2

# server::tackle_exponent
server::tackle_exponent = 6

# server::tackle_power_rate
server::tackle_power_rate = 0.027

# server::tackle_width
server::tackle_width = 1

# server::visible_angle
server::visible_angle = 90

# server::visible_distance
server::visible_distance = 3

# server::wind_ang
server::wind_ang = 0

# server::wind_dir
server::wind_dir = 0

# server::wind_force
```

```
server::wind_force = 0

# server::wind_rand
server::wind_rand = 0

# server::coach_msg_file
server::coach_msg_file = ''

# server::game_log_dir
server::game_log_dir = './'

# server::game_log_fixed_name
server::game_log_fixed_name = 'rcssserver'

# server::keepaway_log_dir
server::keepaway_log_dir = './'

# server::keepaway_log_fixed_name
server::keepaway_log_fixed_name = 'rcssserver'

# server::landmark_file
server::landmark_file = '~/.rcssserver-landmark.xml'

# server::log_date_format
server::log_date_format = '%Y%m%d%H%M-'

# server::team_l_start
server::team_l_start = ''

# server::team_r_start
server::team_r_start = ''

# server::text_log_dir
server::text_log_dir = './'

# server::text_log_fixed_name
server::text_log_fixed_name = 'rcssserver'
```

Bibliography

[1] Mazda Ahmadi and Peter Stone. Instance-based action models for fast action planning. In *RoboCup 2007: Robot Soccer World Cup XI*, pages 1–16, Berlin, Heidelberg, 2008. Springer-Verlag.

[2] Alejandro Guerra-Hernandez Amal, Ro Guerra-hernández, and Amal El Fallah-seghrouchni. Learning in BDI multi-agent systems. In *In Proceedings of CLIMA 2003*, pages 185–200. Springer-Verlag, 2004.

[3] Michael E. Bratman. *Intention, Plans, and Practical Reason.* Harvard University Press, 1987.

[4] Lars Braubach, Er Pokahr, Daniel Moldt, and Winfried Lamersdorf. Goal representation for BDI agent systems. pages 9–20. Springer, 2004.

[5] Hans-Dieter Burkhard, Markus Hannebauer, and Jan Wendler. At humboldt - development, practice and theory. In *RoboCup-97: Robot Soccer World Cup I*, pages 357–372, London, UK, 1998. Springer-Verlag.

[6] Hans-Dieter Burkhard, Markus Hannebauer, and Jan Wendler. Belief-Desire-Intention deliberation in artificial soccer. *AI Magazine*, 19(3), 1998.

[7] Mao Chen, Ehsan Foroughi, Fredrik Heintz, ZhanXiang Huang, Spiros Kapetanakis, Kostas Kostiadis, Johan Kummeneje, Itsuki Noda, Oliver Obst, Pat Riley, Timo Steffens, Yi Wang, and Xiang Yin. *RoboCup Soccer Server.* 2001.

[8] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artif. Intell.*, 42(2-3):213–261, 1990.

[9] The RoboCup Federation. Robocup official site. http://www.robocup.org, January 2009.

[10] Python Software Foundation. Python. http://python.org, July 2009.

[11] GNU. Subversion. http://subversion.tigris.org/, July 2009.

[12] Henry Hexmoor and Xin Zhang. Norms, roles and simulated RoboCup. In *In Proceedings of the Second Workshop on Norms and Institutions in MAS, 5th International Conference on Autonomous Agents*, 2001.

[13] Marcus Huber and Edmund H. Durfee. Deciding when to commit to action during observation-based coordination. In *AAAI-90.* AAAI Press, 1990.

[14] Marcus Huber and Edmund H. Durfee. Deciding when to commit to action during observation-based coordination. In *In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95*, pages 163–170. AAAI Press, 1995.

[15] Nicholas R. Jennings. *Cooperation in Industrial Multi-Agent Systems.* World Scientific, 1994.

[16] Nick Jennings. Specification and implementation of a belief-desire-joint-intention architecture for collaborative problem solving. *Journal of Intelligent and Cooperative Information Systems*, 2:289–318, 1993.

[17] Nick R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems, 1993.

[18] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. RoboCup: The Robot World Cup Initiative, 1995.

[19] Hiroaki Kitano, Milind Tambe, Peter Stone, Manuela Veloso, Silvia Coradeschi, Eiichi Osawa, Hitoshi Matsubara, Itsuki Noda, and Minoru Asada. The RoboCup synthetic agent challenge 97. In *In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 24–29. Morgan Kaufmann, 1997.

[20] Nuno Lau and Luís Paulo Reis. FC Portugal 2001 team description: Flexible teamwork and configurable strategy. In *RoboCup 2001: Robot Soccer World Cup V*, pages 515–518, London, UK, 2002. Springer-Verlag.

[21] Bruno Maisonnier. Aldebaran robotics. http://www.aldebaran-robotics.com/eng/index.php, July 2009.

[22] Anand 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.

[23] Anand S. Rao and Michael P. Georgeff. Formal models and decision procedures for multi-agent systems, 1995.

[24] Anand S. Rao and Graeme Murray. Multi-agent mental-state recognition and its application to air-combat modelling. In *IN PROCEEDINGS OF THE 13TH INTERNATIONAL DISTRIBUTED ARTIFICIAL INTELLIGENCE WORKSHOP*, pages 283–304, 1994.

[25] Luís Paulo Reis, Nuno Lau, and Eugenio Oliveira. Situation based strategic positioning for coordinating a team of homogeneous agents. In *Balancing Reactivity and Social Deliberation in Multi-Agent Systems, From RoboCup to Real-World*

Applications (selected papers from the ECAI 2000 Workshop and additional contributions), pages 175–197, London, UK, 2001. Springer-Verlag.

[26] Peter Stone, Patrick Riley, and Manuela M. Veloso. The cmunited-99 champion simulator team. In *RoboCup-99: Robot Soccer World Cup III*, pages 35–48, London, UK, 2000. Springer-Verlag.

[27] Peter Stone and Manuela Veloso. A layered approach to learning client behaviors in the robocup soccer server. *APPLIED ARTIFICIAL INTELLIGENCE*, 12, 1998.

[28] Peter Stone and Manuela Veloso. Using decision tree confidence factors for multiagent control. In *AGENTS '98: Proceedings of the second international conference on Autonomous agents*, pages 86–91, New York, NY, USA, 1998. ACM.

[29] Peter Stone and Manuela Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *ARTIFICIAL INTELLIGENCE*, 110(2):241–273, 1999.