Abstracting multidimensional concepts for multilevel decision making in multorobot systems

Thesis

UNRESTRICTED

Abstracting Multidimensional Concepts for Multilevel Decision Making in Multirobot Systems

James A. Law MEng

Submitted in accordance with the requirements for the degree of

Doctor of Philosophy

The Open University

Department of Design and Innovation

Faculty of Technology

September 2007

The candidate confirms that the work submitted is his own and that appropriate credit has been given where reference has been made to the work of others.

DATE OF SUBMISSION 28 SEPTEMBER 2007

DATE OF AWARD 23 JUNE 2008

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Abstract

Multirobot control architectures often require robotic tasks to be well defined before allocation. In complex missions, it is often difficult to decompose an objective into a set of well defined tasks; human operators generate a simplified representation based on experience and estimation. The result is a set of robot roles, which are not best suited to accomplishing those objectives. This thesis presents an alternative approach to generating multirobot control algorithms using task abstraction. By carefully analysing data recorded from similar systems a multidimensional and multilevel representation of the mission can be abstracted, which can be subsequently converted into a robotic controller.

This work, which focuses on the control of a team of robots to play the complex game of football, is divided into three sections: In the first section we investigate the use of spatial structures in team games. Experimental results show that cooperative teams beat groups of individuals when competing for space and that controlling space is important in the game of robot football. In the second section, we generate a multilevel representation of robot football based on spatial structures measured in recorded matches. By differentiating between spatial configurations appearing in desirable and undesirable situations, we can abstract a strategy composed of the more desirable structures. In the third section, five partial strategies are generated, based on the abstracted structures, and a suitable controller is devised. A set of experiments shows the success of the method in reproducing those key structures in a multirobot system. Finally, we compile our methods into a formal architecture for task abstraction and control.

The thesis concludes that generating multirobot control algorithms using task abstraction is appropriate for problems which are complex, weakly-defined, multilevel, dynamic, competitive, unpredictable, and which display emergent properties.

Acknowledgements

Firstly, I would like to thank my supervisors, Jeff Johnson and Anthony Lucas-Smith, for their input and encouragement during the course of this research. Also to Richard, George, Charlie, and Angie for their technical and secretarial support. These are the people who helped make things happen, and without whom this research would not have been possible.

I would also like to thank the UK Mirosot community for the opportunities I have had during this work, above all to John, for all his assistance in getting the Mirosot system up and running. John was invaluable in enabling the practical experiments in this work.

On a more personal note, I would like to thank all my friends and family for their support during the course of this work. To Lindsey, Stacey and Kat, for the friendship they have provided, and to Christina for proof reading this thesis and all her support over the past months.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Coordinating teams of robots to cooperate in dynamic environments is an important research problem in the robotics community. Many target applications have been generated and are awaiting the arrival of a suitable implementation of the technology. However, as of yet, there has been no definitive solution to creating an architecture capable of satisfying the general needs of such a system. Most architectures are focused on a particular domain, whether it be a specific application or a particular control issue; such as task allocation. In this thesis we will demonstrate a novel approach, developed from fundamental principles, which should be applicable to a wide range of problems, both inside and out of the field of multirobot research.

Our interest stems from the problem of describing, analysing and simulating complex systems. A definition of what makes a system complex will be left until the following chapter, but for now consider it as a system with many interacting variables where the link between cause and effect is not clearly understood. Such systems exist in multidimensional and multilevel problem space. In a multirobot system, for example, the number of possible interactions is enormous. These can be inter-robot, robot-environment, or even inter-environment. To consider every possible option and plan for the best outcome using traditional deliberative control methods is impractical, if not impossible; there is a combinatorial explosion of decisions to be made by the control architecture at every step. To overcome the need

to search such vast spaces, many multirobot approaches rely on very simplified representations of the environment and task. Such systems are limited by the programmer's ability to identify and plan for specific events. If an event occurs which is unexpected, then the behaviour of the robotic system can be unpredictable, or even terminal.

The behaviour of such complex systems emerges from the interactions of the system at lower levels. For example, consider an orchestra playing a symphony. We know that the resulting audible music has been designed by instructing each individual musician to play a different sequence of notes. Each part, when played on its own, may be of little significance, but when played together the symphony emerges. Now consider that we know what the symphony should sound like, but we have no manuscript. In this case each musician would need to work out their own score. If we had vague ideas about melodies some of the musicians had at different times we could get them to play these, but the result would not be an accurate reproduction of the symphony. This is similar to the way in which many complex robotic problems are currently tackled. Engineers use their experience to estimate the sequence of robotic tasks required to complete an objective, and the outcome is a sub-optimal generalisation of the required solution. In our orchestral example, a better solution would be for each musician to try and work out their score by listening for their instruments in the full symphony. This is analogous to the method of analysis and control we present in this work. The problem lies in trying to distinguish the different parts and instruments from the finished piece, or identifying and reconstructing the actions of each robot required to generate an emergent behaviour sufficient for completing the objective.

1.1 Problem Definition

Our work will focus on a set of multirobot problems which are complex, weakly-defined, multilevel, dynamic, competitive, unpredictable, and which display emergent properties. We briefly define these terms below:

- Multirobot – A system comprising of more than one robot.
- Complex – Being dependent on many interacting variables.
- Weakly-defined – Having only limited information on how to achieve an objective.
- Multilevel – Being represented by a vertically graduated structure of elements, with abstraction occurring between levels.
- Dynamic – In an environment which is constantly changing, irrespective of whether a robot is performing an action.
- Competitive – Where forces are constantly in action to oppose the objective of the robots.
- Unpredictable – Where the future state of the system depends on factors which are unknown or unmodelable.
- Emergent – High level behaviour of the system is not directly controlled, but arises from the interaction of elements at a much lower level.

1.2 Research Questions

With regard to the problem set out above, we aim to provide an answer to the question:

How can we control a team of robots to perform a weakly-defined cooperative task in a complex, dynamic, unpredictable and competitive environment?

We will pursue the answer to this question by answering the following:

1. Can we identify and construct useful representations of complex, dynamic, unpredictable and competitive environments in ways which facilitate the use of robots?
2. Can we extract useful information on how to control a team of robots by recognising the occurrence of key structures in different teams operating in similar situations?
3. Can we use the same techniques to extract information about tasks in the environment at varying levels of representation?
4. Can we identify relations between interacting levels of representation?
5. Can we build our own representation of a set of tasks of varying complexity by combining information from different levels of analysis?
6. By combining information in this way, can we create an emergent strategy for controlling robots in the environment?
7. Can a team of robots, built upon these principles, function effectively in the given environment?

8. Can these ideas be combined to form an architecture whereby a robotic system can learn to perform in a given environment?

The following sections will lay out our argument for answering these questions. After introducing some fundamental ideas in robotics, as well as reviewing current architectures in chapter 2, chapters 3-5 will cover our experimental work, with conclusions drawn in chapter 6.

Published material arising from this thesis can be found in (Law, 2005; Law & Johnson, 2004, 2006, 2008; Robinson et al., 2004).

1.3 Research Strategy

This thesis is primarily concerned with the structural representation of complex systems. These systems have many elements, which interact on many levels, and through which the behaviour of the system emerges. Typically, these interactions are unknown, and control of the system is difficult, if not impossible. The approach described here is a novel technique for task abstraction and control. If the elements and their interactions, corresponding to specific behaviours (which we term *concepts*), can be identified, a detailed system description can be built. By reconstructing these interactions using a controller, the desired system level behaviour can be created. Listed below are the approaches which shall be used to tackle the questions given above.

1. We argue that there exists a set of measurable structures (spatial, temporal, multilevel, etc.), which are related to any given task. We focus on the game of robot football, in which the formational structures between groups of players, and relationships between individual players, the ball and the

goalposts, represent the state of the system. Using experimental evidence, we will show how spatial structures are integral to the success of teams, and how these structures can be identified by a robotic system.

2. By examining recorded data we will show that some structures appear frequently in robot football matches, and that these structures are descriptions of how to play the game. We will also show that there are structures which appear more often within winning teams than within losing teams, and that these structures describe how to play the game well. Similarly there will be structures which appear more often in losing teams, which will describe how to play poorly.

3. We shall describe robot football as a multilevel system and introduce a series of interacting layers which we will term *skills*, *tactics*, *plays* and *strategies*. We will abstract representations for key structures at each of these levels, which describe emergent behaviours in terms of interacting elements. This will show how the same analysis techniques can be used to describe the system from macro through to micro levels.

4. Emergence occurs when elements of a system at a lower level affect behaviour at a higher level. Examining our abstracted representations, we show how sets of elements map to a low level behaviour, and how these sets are also evident in higher level behaviours which subsume those lower level behaviours.

5. A robot football strategy, like other complex systems, contains a multilevel structure of behaviours, or subsystems. By abstracting a set of representations which match each of these subsystems, we can build an

entire system representation across multiple levels. We focus on generating a football strategy as a multilevel structure of skills, tactics, and plays.

6. Since each level of representation describes an emergent behaviour in terms of a set of interacting elements, and since sets of elements are shown to propagate across levels, the resulting strategy will be emergent. The entire strategy will be represented by sets of interacting low-level elements.

7. We build a controller to reconstruct the individual elements measured in our analysis. By generating the sets used to represent each behaviour, we aim to reproduce that behaviour. We reproduce the upper levels of a football strategy to show how a high level positional football behaviour emerges from the interaction of low level elements.

8. The task abstraction process is a learning mechanism, which robots can use to learn emergent behaviours. Although initially performed on a finite set of data, new data can be included, and the process repeated, to enable the robots to continue to develop. The controller described enables the robots to reproduce these learnt skills, and generate new data by playing other teams. By combining these two techniques, we generate an architecture which satisfies the problem, using a closed loop learning mechanism to generate desirable emergent solutions to complex problems.

1.4 Overview of Chapters

We begin in chapter 2 by introducing some important themes in robotic control, followed by a brief summary of the most popular multirobot control architectures to

date. We summarise the strengths and weaknesses of these systems and evaluate them in terms of our problem definition, highlighting the need for an alternative method of control. Chapter 2 continues with an introduction to robot football, the problems, typical controllers, and an explanation of the three types of robot football we will focus on in this thesis. It concludes with an introduction to some complexity theory, which will be used to develop the analysis method.

Chapter 3 examines spatial structures and their importance in robot football. It begins by comparing robot football to some traditional AI turn-based games, and showing that its complexity prevents it from being solved in a similar manner. Following the examples of chess and Go algorithms, we investigate the use of set pieces, and spatial structure, to confront this complexity.

Using cellular automata we examine simple strategies in a game to capture space. This leads into an analysis, using Voronoi diagrams, of how players can configure themselves to control useful areas on a football pitch. Further Voronoi analysis of robot football data shows how these areas are important in the game. Since areas are of importance, so are the configurations of players; this is the Voronoi-Delaunay dual property.

Having verified the importance of spatial structure in football, chapter 4 undertakes to abstract emergent behaviours from low level spatial configurations. We introduce our architecture for abstracting the properties which generate emergent behaviours. Our implementation requires identifying a multilevel structure of concepts (behaviours), which together describe a complete robot football strategy. Taking each concept in turn we search for commonly occurring relations of variables. Some relations appear (or do not appear) in many football matches.

These are likely to signify general conditions for playing the game. Some spatial structures appear more frequently in winning teams than in losing teams (and vice-versa), and these may describe why some teams perform better than others.

Chapter 4 also shows that relations of variables migrate between levels in the multilevel structure. By combining sets using logical operators, we can generate sets (or set fragments) on higher or lower levels of the structure. We illustrate these relations using Venn and cone diagrams. Common sets are also shown to exist between different types of robot football.

In chapter 5 we introduce an architecture for emergent control based on the reconstruction of representatives. We reconstruct the higher levels of a robot football strategy from results obtained in chapter 4. To support our thesis, five emergent strategies are tested experimentally: performance is measured for the controller alone, followed by implementation on simulated robots, and finally on the real robots. Desired behaviour and shortcomings are observed and commented upon. We combine the architectures for task abstraction and control into a single arrangement and describe its application to alternative systems.

Finally, in chapter 6, we draw our conclusions, and reflect back on the problems and questions posed in chapter 1. We close by considering ways to further the research presented in this thesis.

Appendix A describes benchmark experiments performed on our Mirosot robots. These benchmarks provide us with information on the limitations of our robots, and give a measurable comparison to other robotic systems. They are carried out prior to the work in chapter 5 to distinguish between failures in our abstracted controller, and failures in low level robot handling.

Chapter 2

Background

Before embarking on our investigation, we will introduce some important themes and ideas in robotics. We will define some common control-related topics, review relevant control architectures, establish our robotic task, and introduce some general complexity theory.

2.1 Control Issues

The following issues are all important to the field of multirobot coordination, and of particular relevance to the work undertaken in this thesis.

2.1.1 Deliberative and Reactive Control

These are the two main styles of decision making in robotics. *Deliberative* architectures follow the symbolic AI approach, using the sense, plan, act model (figure 2.1). In the sense operation, a robot begins by observing its environment and constructing symbols to represent its state. It then plans a sequence of actions using these symbols and its understanding of the task. Finally, in the act stage, it executes the plan. Deliberative architectures can be slow and ineffective in highly dynamic environments, when the computation time for the sense, plan, act cycle becomes significant. Such systems also fare poorly when encountering unexpected events. For a robot to work reliably in an environment, it must be programmed to handle information about every possible stimulus.

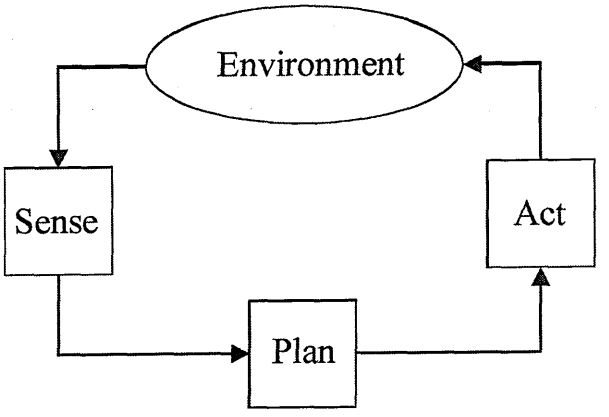


Figure 2.1 A Deliberative Control Architecture

Reactive architectures, including the well known branches of subsumption (Brooks, 1986) and behaviour based control (Arkin, 1987), remove the need for generating symbols by using stimuli to directly influence the robots actions. This follows the philosophy that "the world is its own best model" (Brooks, 1990). Walter (1953) and Braitenberg (1984) produced robots where the sensors were hard wired to the actuators, producing simple behaviours such as light following or avoidance. Due to the lack of deliberative decision making, these architectures respond quickly in dynamic environments. They also respond well to unexpected events, since they have no model of how the environment should be behaving, so no conflicts with their expectations occur. Due to this, they are well suited to complex and unpredictable environments. An extension of these ideas added *behaviours*, where sets of stimuli were mapped to sets of responses (Brooks, 1989). By building a number of behaviours, then switching between them, more complex actions can be performed. The subsumption architecture consisted of a number of behaviours built into a hierarchy (figure 2.2). Simple behaviours at the lower level are subsumed into more complex behaviours at higher levels. A major drawback of reactive architectures is their inability to plan and modify their goals, making them unsuitable for many applications.

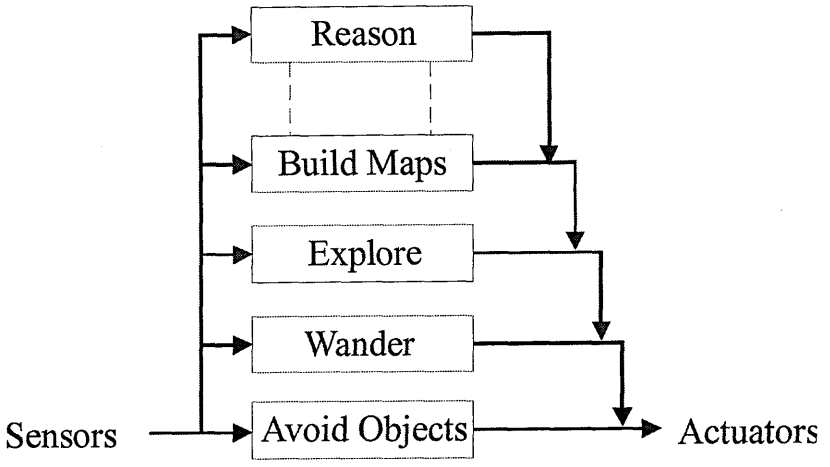


Figure 2.2 A Reactive Control Architecture

To address the dynamic requirements, reliability, and need for planning, hybrid architectures containing both decision making processes have been implemented (Arkin, 1990; Gat, 1992). These systems usually consist of three layers (figure 2.3). At the lowest level are the reactive behaviours, which deal with time-critical responses. At the uppermost level is the deliberative function, which handles forward planning. In between is a mediator layer, which handles the interaction between deliberative and reactive aspects of the architecture.

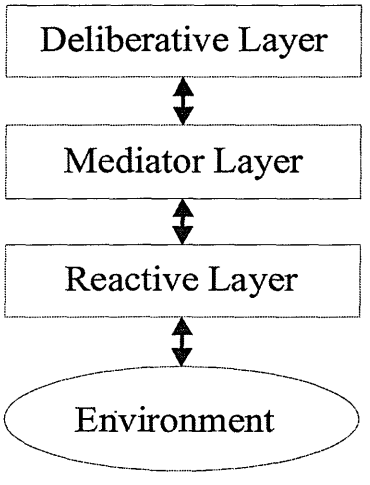


Figure 2.3 A Hybrid Architecture Containing both Deliberative and Reactive Elements

2.1.2 Centralised and Distributed Control

This refers to whether the decision making process is carried out in one place, or whether it is spread amongst the robots. In centralised systems, the team of robots can be considered as a single system with many degrees of freedom. The information provided by the sensors is fed back to a single robot, or computer (figure 2.4). With all the information at hand, the controller then calculates all the relevant actions and sends these out to all the corresponding robots, which then carry out these actions. This type of architecture includes leader-follower systems, where a single leader robot coordinates the actions of its followers, such as (Chaimowicz, Sugar, Kumar, & Campos, 2001).

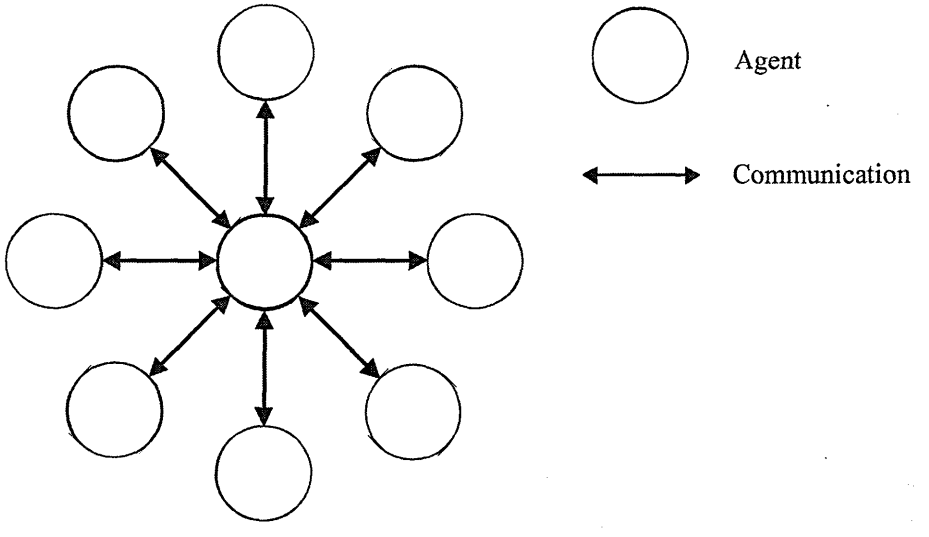


Figure 2.4 Representation of a Centralised Architecture

All agents communicate through a central node.

In a centralised system, passing all the relevant information to one processor allows for optimal plans to be made, provided the options are limited, and the environment is relatively static. However, in large, dynamic systems, the time taken to collate and process all the information can be significant. Furthermore, the

complexity of calculating the optimal response increases dramatically as more robots, stimuli, or action options are introduced. Having all decision making processes at a single point also affects the robustness to failure of the system. If the lead robot malfunctions, then the whole system is disabled.

In a distributed system, the planning element is decentralised and spread amongst the members of the team (figure 2.5). Usually this means each robot uses information from its own sensors to calculate its own set of actions. In this way, the communication requirements at a single point are reduced, the complexity of calculations is significantly diminished, and thus the system is able to respond much faster to changes in the environment. Also, since there is no single leader, the system is more robust to failures. The main drawback is that the limited information available to each robot may prevent optimal solutions from being found.

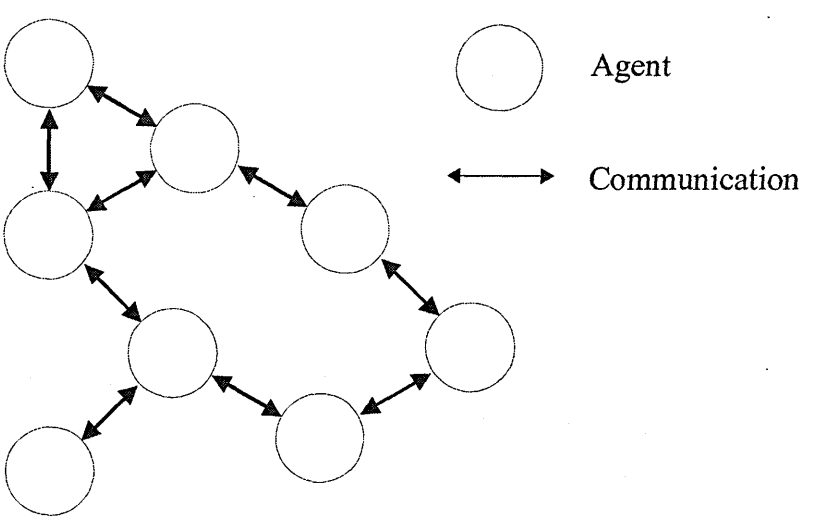


Figure 2.5 Representation of a Distributed Architecture

Agents communicate directly with one another.

In distributed systems teams are sometimes given the ability to form sub-groups, in which information is passed between robots to give them a better understanding of the situation, allowing them to form more optimal solutions (Dias,

2004). Similarly, centralised systems can incorporate distributed elements to reduce communication requirements and speed up response times (Caloud, Choi, Latombe, Le Pape, & Yim, 1990).

2.1.3 Dynamic Environments

We define a dynamic environment as one which may change without any interaction from a robot. This is important as it defines the level of dexterity and the computation speed required of a robot and its control architecture. Clearly, a robot which takes a minute to move ten meters is going to be of no use as a football player, but may be suitable for cleaning work. Correspondingly, there is a range in the rate of change within a dynamic system. The faster changes occur, the faster the response of a robotic system will need to be. Many multirobot systems have been generated to perform in environments which are not dynamic, or which have very little in the way of dynamic requirements. Applications such as box pushing (Mataric, Nilsson, & Simsarin, 1995; Stilwell & Bay, 1993) are not dynamic, whereas pursuit-evasion systems (Búason & Ziemke, 2003; Vidal, Shakernia, Kim, Shim, & Sastry, 2002) are. In this thesis we focus on highly dynamic systems, where robots are required to work in real time with strict limitations on the time allocated to decision making.

2.1.4 Cooperation and Competition

The majority of multirobot systems are designed to work as a team, with robots working together cooperatively to perform a task. By working together, a group of robots can achieve tasks which a single robot alone would be incapable of, such as the movement of large objects (Kube & Bonabeau, 2000; Sugar & Kumar, 1999), or

assisted cliff descent (Pirjanian et al., 2002). Groups of robots can also achieve some tasks faster than a single robot. Examples include collective construction (Wawerla, Sukhatme, & Mataric, 2002), site clearance (Parker & Hong, 2002), foraging (Sugawara, Sano, Yoshihara, Abe, & Watanabe, 1999), and exploration and mapping (Simmons et al., 2000). The opposite are tasks where robots compete against one another. Such systems have been demonstrated in predator-prey systems (Búason & Ziemke, 2003; Floreano & Nolfi, 1997). We are particularly interested in systems which display both, where a team of robots works together in competition against an opponent robot, or team of robots. These exemplify team games, or pursuit and evasion style security applications (Parker, 2002; Vidal et al., 2002).

2.1.5 Degree of Coordination

A recent interesting development in the field of team robotics has been the focus upon the degree of coordination required between team members. Consider a swarm of robots tasked with clearing an area (Brooks, Maes, Mataric, & More, 1990). Each robot moves around within a given area until it finds a rock. When it does so, it picks it up and carries it to the edge of the area where it is deposited. In this system the robots do not require any explicit coordination to achieve their task, although they are acting together to fulfil the same objective. In an alternative system, two robots are tasked with carrying a large object (Barnes, Ghanea-Hercock, Aylett, & Coddington, 1997; Sugar & Kumar, 2002). In this example, the two robots must closely coordinate their actions to ensure the load is not dropped, twisted, or broken. Emery et al. (2002) distinguish between the two types of teamwork by defining them separately as collaboration and coordination; "Collaboration occurs when multiple robots are working towards the same goal but do not explicitly coordinate their

actions", whereas "coordination involves more explicit protocols for deciding which robot will do what when".

Chaimowicz et al. (2001) introduce the concept of *tightly coupled cooperation* for tasks that cannot be completed by a single robot, and require real-time coordinated control between robots. Such problems are commonly ascribed to collective box pushing (Kube & Zhang, 1992; Rus, Donald, & Jennings, 1995), material handling (Sugar & Kumar, 2002), and coordinated security patrols (Kalra & Stentz, 2003). A problem with this description, is that the environments are not usually highly dynamic; robots are allowed time to consider and communicate their actions. For example, in the case of box pushing, the activities of robots do not need to be closely orchestrated as satisfactory results can be obtained through turn taking schemes, or even uncoordinated collective behaviours. Huntsberger et al. (2003) refine the definition to exclude such cooperation where there is time to consider and communicate actions. Their work emphasises "tasks that inherently require tight coordination under strict physical constraints" which are "characterized by constraints imposed on the activities of one robot as a function of the state of the others".

2.1.6 Weakly-Defined Tasks and Machine Learning

In the previous chapter we introduced our concept of a weakly-defined task as one in which we had little information on how to achieve the objective. In these situations, or those in which the optimal solution is not known, machine learning algorithms can be employed to try and solve the task, or improve the performance of the system.

An agent is said to learn from experience with respect to some task or performance measure, if its performance measure for the task improves with

experience (Mitchell, 2006). Therefore, it is necessary to define measurement criteria by which to asses the performance of the system. Learning systems usually consist of a skill to be learnt, a mathematical representation of that skill within the robot, an algorithm to tune the representation toward the actual skill, and some learning data. In the case of robot systems, this learning data is usually acquired by the robot repeating the skill with minor permutations. By sequentially selecting the best attempt, and modifying the representation, the robot is driven toward obtaining the required skill. This is known as reinforcement learning. Some multirobot learning systems are detailed in (Arkin, Endo, Lee, MacKenzie, & Martinson, 2003; Balch, 1999; Liu, Wang, Zhiqiang, & Zengqi, 2004; Stone & Veloso, 2000).

2.1.7 Emergence

Goldstein (1999), defines emergence as "the arising of novel and coherent structures, patterns and properties during the process of self-organization in complex systems". In a team of robots, the emergent behaviour of the system is that which is evident at a macro-level, but is not explicitly stated within the control structure. It arises from the interaction of elements of the control architecture at the micro-level. Hence the resulting behaviour is not programmed, but *emerges*.

Emergence is commonly seen in distributed systems, where robots have limited information, and act according to their local environment. When viewed on a global scale the interactions of the robots produce an emergent system behaviour, which is different to those exhibited and programmed into each robot. Examples of emergence in distributed systems are in swarm applications such as foraging (Sugawara et al., 1999) and aggregation (Garnier et al., 2005). A problem with emergent behaviours arises when trying to tailor them to a specific requirement.

Creating a 'useful' emergent system response can be difficult, especially for complicated tasks.

2.1.8 Prediction

Often, when discussing prediction in robot teams, we are interested in the effect actions will have on events far in the future. We want to be able to predict how decisions in the system, or changes in the environment, will affect our ability to reach our objective. Such prediction is not viable in the robot football systems described in this thesis. Instead, we are limited to short term prediction of some modelable physical properties. We can, for example, extrapolate the motion of bodies with some accuracy over short periods, but we cannot predict where opponent defenders are going to be after the time taken to conduct three passes.

2.1.9 Complexity

Complex systems are high-dimensional, non-linear, non-deterministic systems. The behaviour of such systems is emergent, usually due to the effect of self organisation. Although there is no agreed definition of what makes a system complex, there are a number of properties which tend to be evident in such systems. For example, they often have multilevel representations, with abstraction occurring between elements on adjacent levels. They contain large numbers of interdependent variables, or agents, which interact within the multilevel structure, to generate a combinatorial explosion in the space of possible interactions. These interactions give rise to an emergent behaviour, which is difficult to link to the underlying variables and agents. Such systems cannot be represented by systems of computationally reducible equations, or they may not have well specified boundaries.

An example of a complex system is that of a transport network. Consider a town where the inhabitants are the autonomous agents within the system. Each person behaves differently, and although each has a simple destination, the interactions between them, and the variables affecting their journey, give rise to endless possibilities in traffic flow. The times people start their journey, the weather, whether they take a car, bus, train, cycle or walk, whether they stop to meet people, and their destination, all affect how the system responds; a minor change in any of these variables will affect its emergent behaviour. Such behaviour may, for example, be the appearance of a traffic jam. If we are interested in resolving traffic jams in a particular area of the town, we need to try and understand the low level interactions that cause them before we can design a solution. The science of investigating such systems is that of *complexity science.*

A difficulty when dealing with complex systems is in trying to recreate, or change, the behaviour of the system in a desired way. Without a full understanding of the system, which is impossible, the outcome of any changes can never be fully known. In our example, building a new road may alleviate the congestion in our area of interest, but may have undesirable knock on effects. Simulations are often used to test the impact of such changes, but since they too contain limited information, they are not completely reliable. When trying to control complex systems to give a desired response, it is usual to attempt to simplify the representation of the problem. However, if the representation is over simplified, it loses its meaning, and any generated controller will fail to function as desired.

2.2 Multirobot Systems and Architectures

We now come to examine existing work in the field of multirobot research. Many tasks and solutions have been generated and demonstrated. Some of these are explicit solutions to particular problems, whilst others provide more general approaches to solving a variety of problems. The following is a brief overview of some of the most well known architectures for tackling general multirobot problems.

2.2.1 ACTRESS

Asama et al. (1989) introduce ACTRESS, an architecture aimed at distributed control of a heterogeneous robotic team. It is based on the Universal Modular ACTOR Formalism (Hewitt, Bishop, & Steiger, 1973) for information processing, in which the units for information processing are termed *actors*. In ACTRESS, robots are one subset of units called *robotors*, which are robotic forms of *actors*. Robotors are defined as autonomous components with the ability to make decisions, understand tasks, recognise their environment and communicate with other components. Examples of possible robotors are given as robots, computer systems (including simulators), and intelligent sensors. The architecture is said to be reliable, extensible, flexible, efficient, and adaptable, though this is stated in a way as to be applicable to any multirobot system. Stricter compliance to these values is considered in later architectures.

ACTRESS focuses on the organisation of communication, which is split into two layers. *Communication protocol* is the method of establishing communication between devices and guaranteeing data transmission, and is based on the hardware used. *Message protocol* is the method for understanding these communications, and

is split into five levels of abstraction from sensor and control data up to task descriptions.

Collaboration between robots, and a formal organisation of teams, is introduced in (Asama et al., 1994). Each robot initially selects a subtask from a human generated mission. Execution is then split into three parts: task planning, team organisation, and motion planning (figure 2.6). In the planning stage the robot refers to a knowledge base to divide the task into a set of actions. In the organisation phase it consults another knowledge base to obtain functions to undertake each action. If the robot has the required functionality it proceeds to undertake the task; if not, it becomes a coordinator and attempts to form a team to conduct the task. In the motion planning stage, the coordinator plans the motions for all the members of the team.

Team formation is conducted through a bidding process, using the contract net protocol (Smith, 1980). Free robots bid to become part of the team based on their ability to perform the tasks. A learning algorithm is included, which monitors the bidding process, and stores the results in knowledge base 3 (Figure 2.6). This speeds up subsequent rounds of bidding by enabling the coordinator to tailor the auction to the most suitable cooperators.

In (Asama et al., 1989), ACTRESS is used to control two simulated agents pushing boxes in a static environment. Further investigation covers efficient communication with an environment manager (Asama, Ozaki, Ishida et al., 1991), and collision avoidance (Asama, Ozaki, Itakura et al., 1991).

ACTRESS relies on strongly defined tasks and functions, as represented by the knowledge bases. It is demonstrated for movement and box pushing tasks.

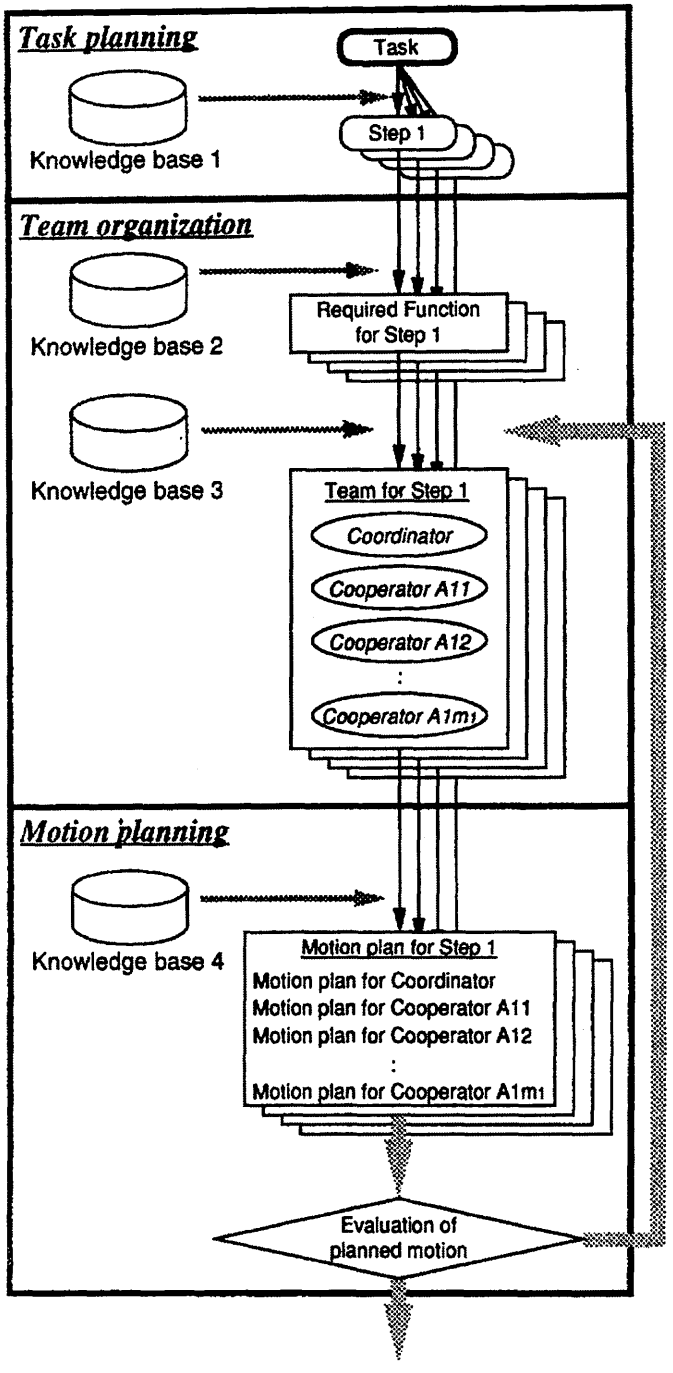


Figure 2.6 Cooperative Task Processing in ACTRESS

Task execution is split into three layers. A robot that fails to complete a task becomes a coordinator and organises a team to undertake it. The architecture makes extensive use of knowledge bases to describe the required procedures at each level. Diagram reproduced from (Asama et al., 1994).

2.2.2 GOFER

GOFER is a multilevel architecture for coordinating mobile robots in a mainly static indoor environment, with emphasis on the transportation of small objects (Caloud et al., 1990). It is comprised of four layers: task planning, task allocation, motion planning, and execution. Although distributed in its ability to perform tasks, GOFER utilises a centralised task processing system (CTPS), which has a global view of all tasks to be performed, and of the availability of robots.

In the planning stage, sequences of actions which are required to fulfil the mission are determined. Although put forward as a system that will generate a plan from a high level objective, without being told how to achieve it, there is little discussion of how this might work. Proposed methods include retrieving plans from a library, or using an unexplained constraint-based planner. Results show that on average it takes 7.3 seconds to create a plan, making it unsuitable for dynamic environments.

In the task allocation stage, the CTPS broadcasts a plan structure as a utility function to all available robots, devoid of values relating to robot properties. Each robot adds its corresponding values and returns the result to the CTPS, which then allocates the task to the best robot. Since robots will reply at different times, the CTPS can choose to wait for further responses before allocating a task if no robots are deemed suitable enough. Only robots which are waiting, and within communication of the CTPS, can request a task.

Once a robot receives a task, it leaves to perform it under its own initiative. Robots which are inactive, or finish a task, automatically seek work by broadcasting

their situation. It is stated that robots can also seek tasks from other robots, rather than the CTPS, but this is not demonstrated.

In the motion planning stage, robots generate a trajectory to follow based on a pre-compiled road map of possible routes in the environment. Execution of motion incorporates well defined rules on how to behave when encountering other robots using the same sections of the road map. Potential field techniques are used to avoid collisions with obstacles.

Le Pape (1990) explores the ability of individual robots to generate and prioritise their own goals and plans for execution. This ability is designed to generate a more distributed system, where robots can work independently as well as connect to the CTPS. Further discussion of the task allocation protocol explains how the CTPS can decide to wait, rather than allocate a task to the first available robot, based on utility functions measured for similar actions in the past. Maigret (1991) describes the hardware implementation, and low level control of individual robots.

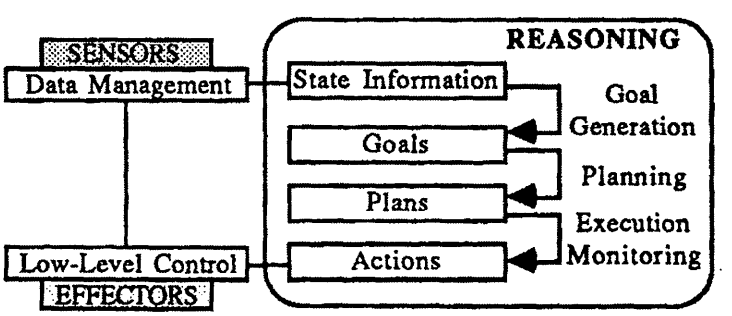


Figure 2.7 Architecture of a Single GOFER Robot

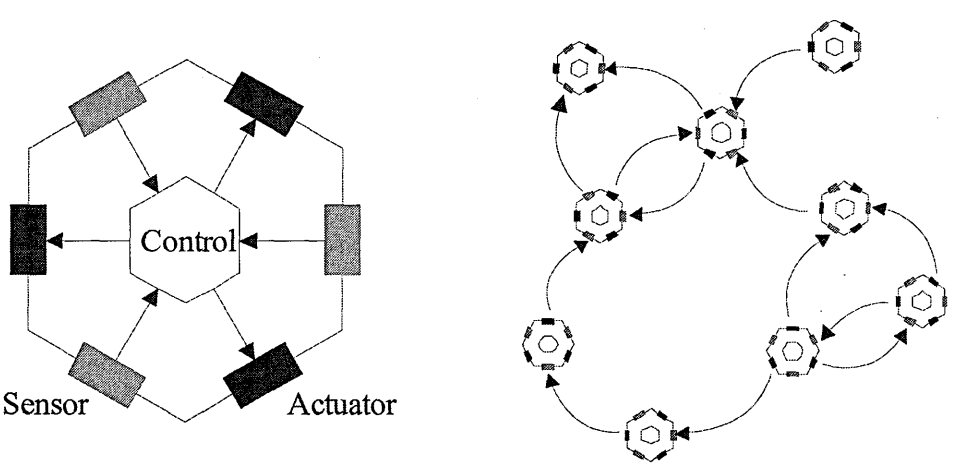
Diagram reproduced from (Le Pape, 1990).

GOFER is limited in its application to movement tasks, such as box pushing, item transport, and following, in static environments.

2.2.3 Swarm Robotics

Swarm robotics is an approach to multirobot systems which differs greatly from the other systems reviewed here. Although encompassing a large body of research, swarms are grouped together here as a comparison to the other architectures. As we will show in section 2.2.10, swarm robotics provides unique solutions to some of our listed problems.

Swarms, as investigated by Kai (1994), are distributed systems comprising of many simple, autonomous, usually homogenous, agents. They exhibit self-organising, emergent behaviour, which arises from simple interactions between agents and the environment. These agents are usually indistinguishable from one another, containing identical control structures, and acting solely in reaction to the state of their nearest neighbours.



(a) An individual swarm robot (b) Interactions in a robot swarm

Figure 2.8 Architecture of a Simple Robot Swarm

System behaviour emerges from the interactions of nearby robots.

The simple behaviours and local interactions make the approach scalable to very large numbers of robots. This is beneficial in situations which require large area coverage, such as distributed sensing (Hackwood & Wang, 1988). The often

simple mechanisms also make them ideal for miniaturisation; they are suited to microrobotic and nanorobotic applications (Flynn, 1987). The low cost and redundancy of such systems also makes them ideal for hazardous applications, such as mining or exploration (Buckland & Johnson, 1999).

Swarms are often used to generate or investigate spatial and temporal structures, such as flocking behaviour (Reynolds, 1987). The simple control schemes and lack of explicit communication make them ideal for highly dynamic and unknown environments, but also limits their application. Swarms display emergent behaviour, and designing and reliably predicting the behaviour of even simple systems can be difficult. This makes them inappropriate to many problems.

2.2.4 ALLIANCE

Parker (1994) introduces ALLIANCE, a distributed, fault tolerant architecture for multirobot control, based upon motivational behaviours. Each robot decides whether to perform a task based on internal measures of impatience and acquiescence.

Within the architecture, lower level behaviours handle survival tasks such as obstacle avoidance, whilst higher level behaviours handle goal oriented tasks such as map building and exploring. Contrary to other behaviour based approaches, ALLIANCE includes several *behaviour sets* which are either active as a group, or hibernating. Each behaviour set corresponds to a particular task, with only one set active at a time. Motivational behaviours select a behaviour set to be active, and suppress the output of other sets (Figure 2.9). Lower level behaviours, however, remain active at all times.

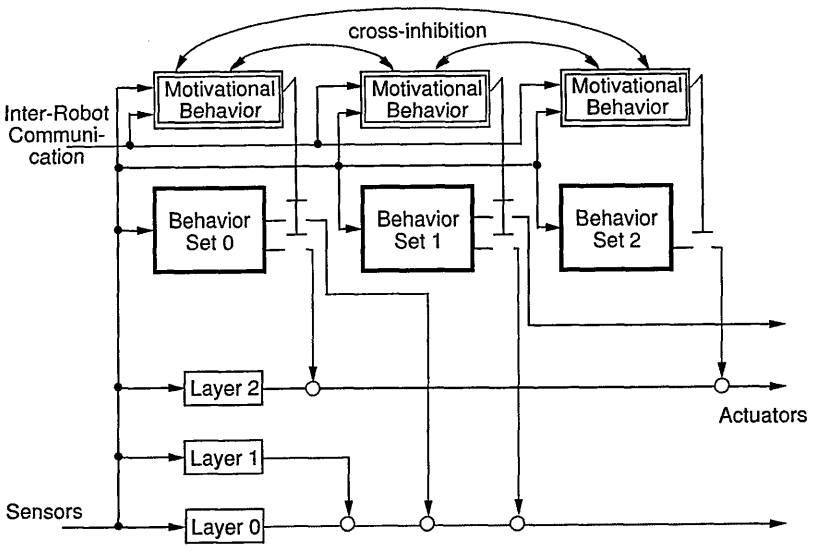


Figure 2.9 The ALLIANCE Architecture

Motivational behaviours enable or disable the output of task-achieving behaviour sets. Diagram reproduced from (Parker, 1998).

Robots routinely broadcast their actions so that other members can monitor their performance. Whilst a task is incomplete, a measure of impatience increases within each robot. If a task remains incomplete for a significant period, the impatience of a robot will grow to a level which causes its motivational behaviour to trigger the associated behaviour set. At this point, the impatient robot will take over the task.

A corresponding measure of acquiescence enables robots to measure their own performance. This enables robots to abandon tasks which they fail to complete. The combination of motivators enables robots to overcome failure in team mates, and prevent themselves from becoming a wasted resource.

The main downside to this architecture is that there is no potential for dynamically introducing new tasks into the system. All the robots have the tasks programmed into their behaviours from the outset. Robots can sit inactive between

tasks, and the time taken for impatience and acquiescence to operate means robots cannot dynamically respond to changes in the environment.

An extension in the form of L-ALLIANCE (Parker, 1997) implements reinforcement learning to improve behaviour set activation. In this incarnation, robots monitor the performance of their team mates, and evaluate their performance with relation to the task completion time, in either training or live scenarios. In training, robots are maximally patient and minimally acquiescent, giving them the maximum time to undertake a task. In live missions, robots update their motivation functions based on the perceived success of the robot performing the task. This allows them to take over or abandon tasks much more efficiently. During active missions, robots are still capable of learning, allowing them to change their perceptions, and respond dynamically to any changes in team mate performance.

ALLIANCE is demonstrated on heterogeneous teams for tasks including box pushing (Parker, 1994) and foraging (Parker, 1998).

2.2.5 M+

M+ (Botelho & Alami, 1999) is a distributed task allocation and coordination architecture based on the contract net protocol (Smith, 1980). It is used in conjunction with a mission planner as shown in figure 2.10. Embedded in the architecture is the ability to re-plan and re-allocate tasks. Each robot in the system is an autonomous agent with reasoning, decision and reactive capabilities.

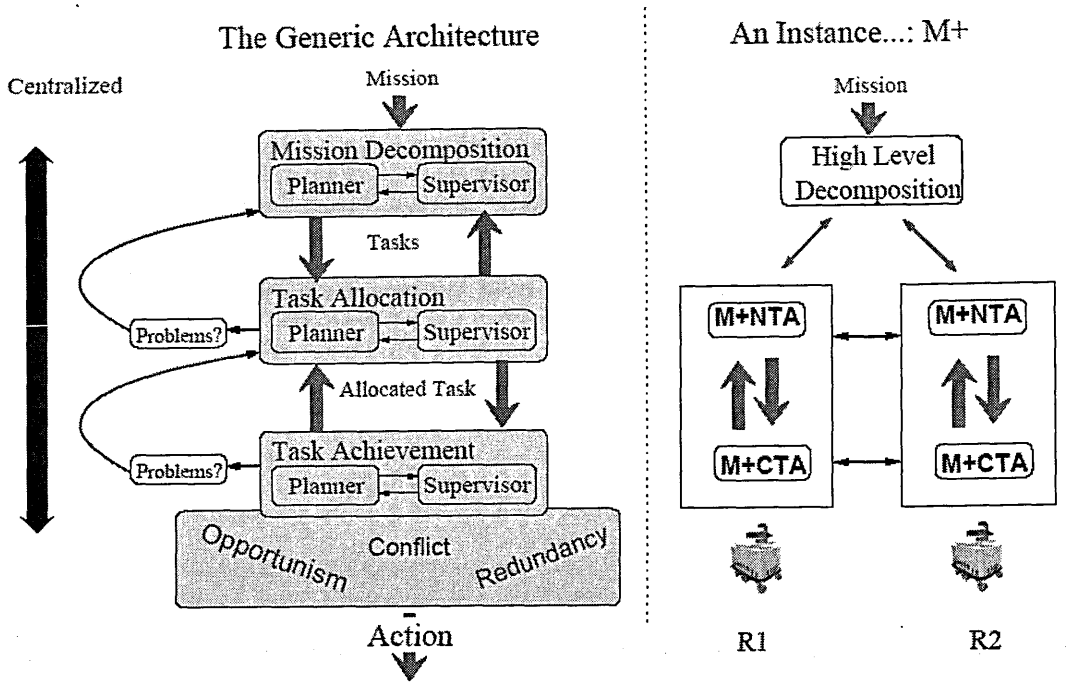


Figure 2.10 An Instance of the M+ Architecture

NTA stands for "Negotiation for Task Allocation", and CTA stands for "Cooperative Task Achievement". Diagram reproduced from (Alami, 2005).

Each level within the architecture consists of a reactive (supervisor) and deliberative (planner) element. Robots communicate with each other at all levels through corresponding elements. Reactive elements exchange signals or protocols, whilst deliberative elements communicate plans, goals, and data.

At the highest level, a mission is inputted to the system by a user. The mission is then split into a set of partially ordered tasks, which can each be performed by a robot. Each task is further described in terms of a set of goals to be achieved. Mission planning and decomposition is not handled by M+, but either by a dedicated centralised planner, or by hand.

At the allocation level, the mission is communicated to the robots in terms of the required tasks and goals. Using their knowledge of the state of the environment, each robot generates a sequence of actions, the execution of which satisfies the

goals. Tasks are then allocated to the robots using a negotiation process: Any robot can make a *first offer* to undertake a task, and in doing so takes responsibility for handling all communication related to the task. Until the moment at which the robot starts to perform the task, any robot can make a counter bid for the task. The robot with the best bid will be selected to undertake the task, although the robot making the first offer remains in charge of the communications. Any robot not bidding on a task remains idle until there is a change in the set of executable tasks.

The task achievement layer handles resource conflicts and efficiency issues, and is the only stage at which explicit cooperation between robots is evident. A novel contribution of M+ is the ability of robots to ask for assistance when they find they cannot complete a task: If a robot fails to fulfil its task, then it attempts to re-plan an alternative set of actions. However, if it cannot find a new plan, then the robot can call for assistance. Other robots then calculate whether they can assist, whilst still carrying out their current missions.

This functionality is implemented through a plan merging protocol. Robots can identify goals which conflict, are duplicated, or with which they can assist. By switching goals with other robots, or adding and removing goals from their own task, robots can coordinate to improve efficiency.

M+ is used within the LAAS architecture (Alami, Fleury, Herrb, Ingrand, & Robert, 1998), which includes tools for mission generation, planning, plan merging, navigation and manipulation. The architecture is extended to include coordinated navigation (Gravot & Alami, 2001), and improved efficiency through allowing multiple robots to undertake a single task (Alami & Botelho, 2002; Botelho & Alami, 2000). The plan merging operation is detailed in (Alami, 2005).

2.2.6 Leader-Follower

Chaimowicz et al. (2001) present an architecture for a team of heterogeneous robots performing a tightly coupled object manipulation task. The architecture is described as distributed, yet relies on a robotic leader being present at all times. The other robots in the team respond to the actions of, and orders from, the lead robot.

The demonstrated task is that of two robots carrying a large object. First, the leader generates a plan for the task, selecting a route to take. The leader then continuously broadcasts its position and velocity to the followers, which attempt to follow using their own trajectory controllers. This communication is shown by the data messages in figure 2.11. Control messages are also sent between robots. These handle role reassignment, task initialisation and task completion.

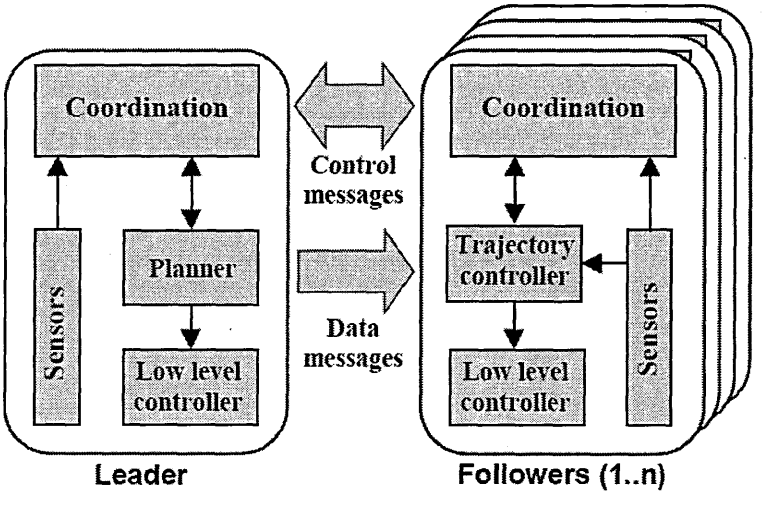


Figure 2.11 The Leader-Follower Architecture

Diagram reproduced from (Chaimowicz et al., 2001).

The architecture contains a role switching mechanism that enables robots to request and relinquish the leadership at any time. This enables the robot that is best suited to the role to take control as circumstances change. A change in leadership

can be instigated by either the current leader or by a follower: If a follower encounters a problem, such as an obstacle, it can request to take over leadership until the problem has passed, then return command to the original leader. Alternatively, a leader that encounters a problem can relinquish the role in the hope that another robot can find a solution.

The dynamic role selection algorithm is not robust; conflicts, such as no robot taking up the leadership role, or a robot failing to relinquish control at the request of a follower, can occur. These conflicts are resolved using a priority based approach, with intervention by a human operator to resolve deadlocks. There is also the possibility that a state of oscillation in leadership can occur. In these situations it is suggested that a set of negotiations may be required.

It is stated that a number of leaders can exist, operating in a hierarchy, with one overall leader. For example, a follower could be leader to another follower. However, this not demonstrated.

The architecture is demonstrated with two robots carrying a large box, in an unstructured environment. A third robot is introduced, acting as a remote sensor. In (Sugar & Kumar, 2002) all three robots are involved in the physical carrying operation.

2.2.7 MURDOCH

MURDOCH (figure 2.12) is an architecture based on intentional cooperation, with robots coordinating with one another explicitly using task related communication (Gerkey & Mataric, 2001). It is a distributed multirobot task allocation architecture, for a heterogeneous team, based upon the contract net protocol (Smith, 1980).

The notable difference of MURDOCH over other negotiation based architectures is its use of a publish/subscribe communication model. Messages are addressed in terms of their content, rather than their destination. A message is published by a broadcaster, labelled with a description of its content. Only robots that are interested in that content type, and who have subscribed to it, receive the message. In MURDOCH, task requests are published in terms of their requirements, such as the sensors and actuators required to achieve them. Only robots with those resources available will receive the task request. Handling communication in this manner promotes efficiency in the system.

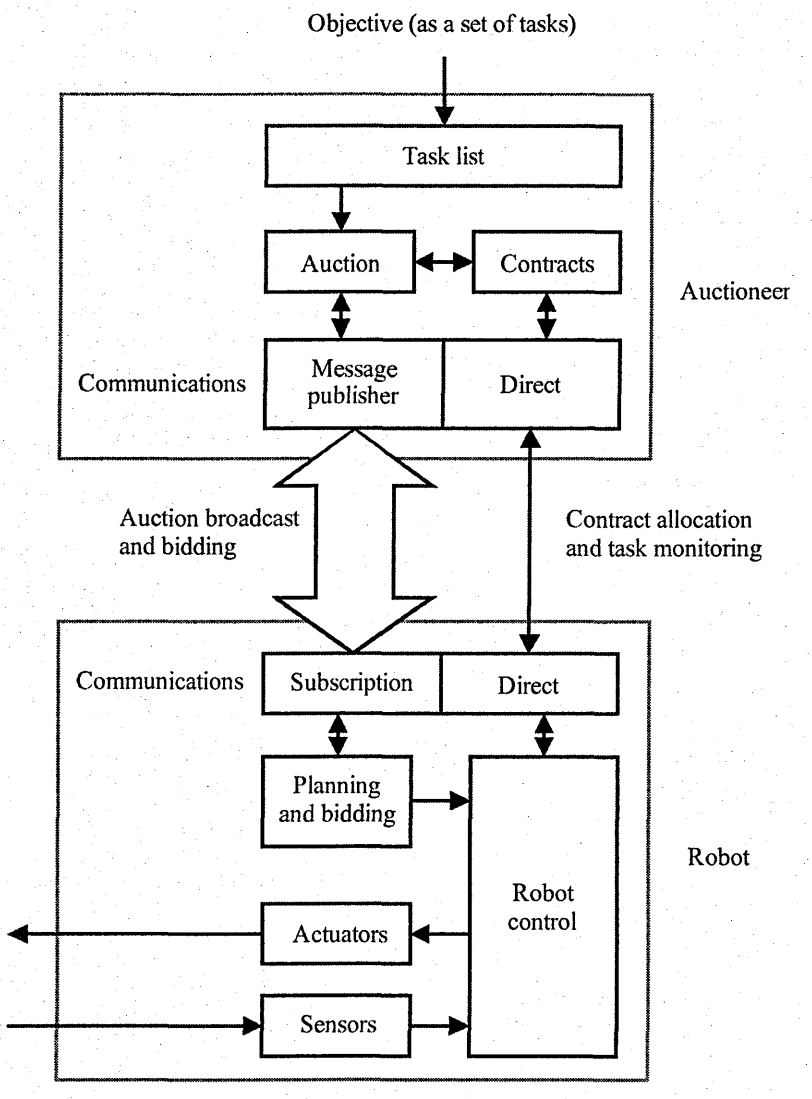


Figure 2.12 A Simplified Representation of the MURDOCH Architecture

An objective, described as a set of well defined tasks, is programmed into the system by the designer. Although capable of allocating a task tree, the structure must be defined by the designer, as must the tasks themselves. These tasks are then auctioned off to available robots.

An agent working on behalf of the user auctions each task, through its requirements, to the set of subscribing robots. The fitness of a particular robot to a task is measured in terms of one or more metrics. These may be evaluated individually, or as a function, and enable each candidate to provide a bid based on its chances of successfully completing the task. After a preset amount of time, the

auctioneer closes the auction and notifies the winning bidder, and issues it with a time limited contract. This contract is used to build fault tolerance into the system, with the auctioneer routinely contacting the winning robot to monitor its progress. If the task is not completed in the required time, the auctioneer re-auctions the task. Losing robots return to waiting for tasks.

The distributed nature is compromised by the use of an auctioneer, and problems may occur if failure occurs in that agent. Furthermore, auctions are limited to a single round of bidding and are given a fixed period in which to run (Gerkey & Mataric, 2002). The architecture is demonstrated on a loosely coupled set of tasks, as well as the more cooperative task of box pushing.

2.2.8 CAMPOUT

CAMPOUT (Control Architecture for Multirobot Planetary Outposts) is a distributed, behaviour-based architecture for executing tightly coupled tasks (Huntsberger et al., 2003). It provides both task allocation and execution mechanisms for use in complex environments.

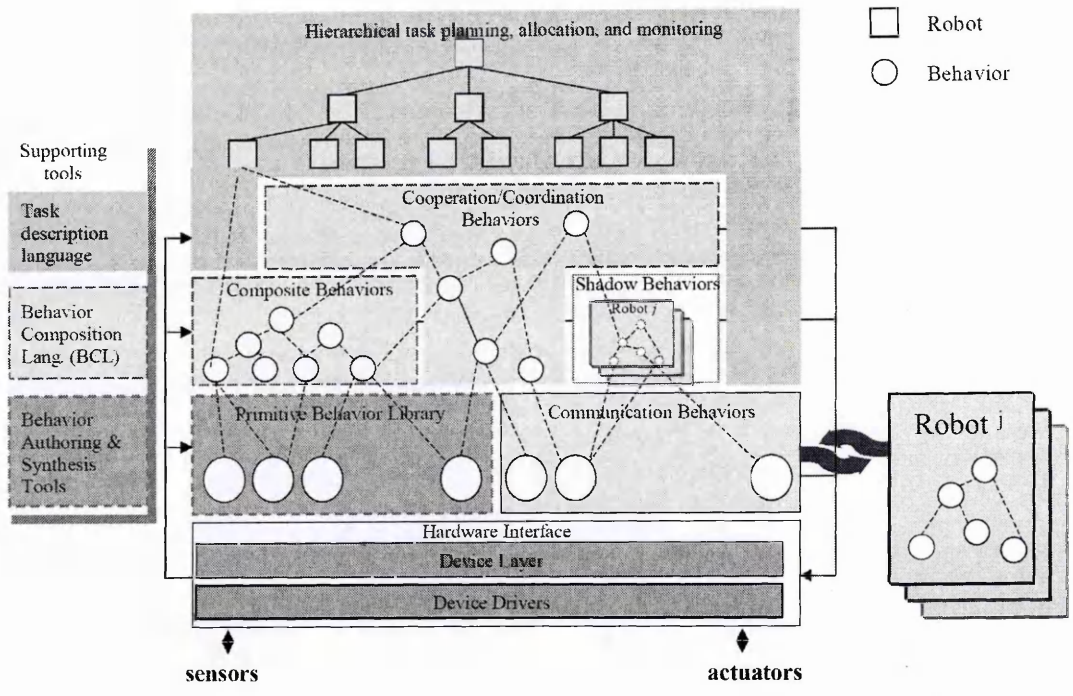


Figure 2.13 The CAMPOUT Architecture

Diagram reproduced from (Schenker et al., 2000).

The architecture contains mechanisms for representing behaviours as mappings of sensor data to actions in a preferential hierarchy, using finite state machines. High level functionality is achieved by combining and managing these low-level behaviours within the context of the higher level objectives. This is achieved through a behaviour coordination mechanism using two methods: arbitration and command fusion. Arbitration mechanisms select a single behaviour to take control of the system, for one control cycle, for use in situations where system resources are scarce. This arbitration is either priority based, where high priority behaviours overrule lower priority ones, or state-based, for behaviour sequencing. Command fusion mechanisms combine behaviours, allowing them to take control of the system in a cooperative, rather than competitive manner, and are used to control tightly coupled actions. They include voting mechanisms, fuzzy logic, and best trade-off

action selection, which aims to select the action that most closely satisfies the current objective.

The main attraction of CAMPOUT is its ability to share behaviours between robots to enable this tightly coupled coordination. These group behaviours are managed explicitly through communication behaviours, or implicitly through shadow behaviours. Explicit communication allows robots to share sensor data and whole functions based on the multi-valued output from behaviours. Implicit communication is achieved through the environment by shared attributes, such as the objects being interacted with, enabling robots to infer the state of other members of the team.

The downside of such a comprehensive and complex system comes in the difficulty of designing the behavioural hierarchy, and specification of the necessary interactions.

CAMPOUT is demonstrated for large object carrying (Huntsberger, Pirjanian, & Schenker, 2001) and cooperative cliff descent (Pirjanian et al., 2002).

2.2.9 TraderBots

Dias (2004) presents TraderBots, another distributed task allocation architecture built on the contract net protocol. A market economy is established using ideas of cost, revenue, and profit, with each robot acting to increase their own individual profit by undertaking lucrative tasks. Revenue is derived from achieving the team's objectives; by acting in a greedy manner, individual robots are actually contributing to the team in useful ways.

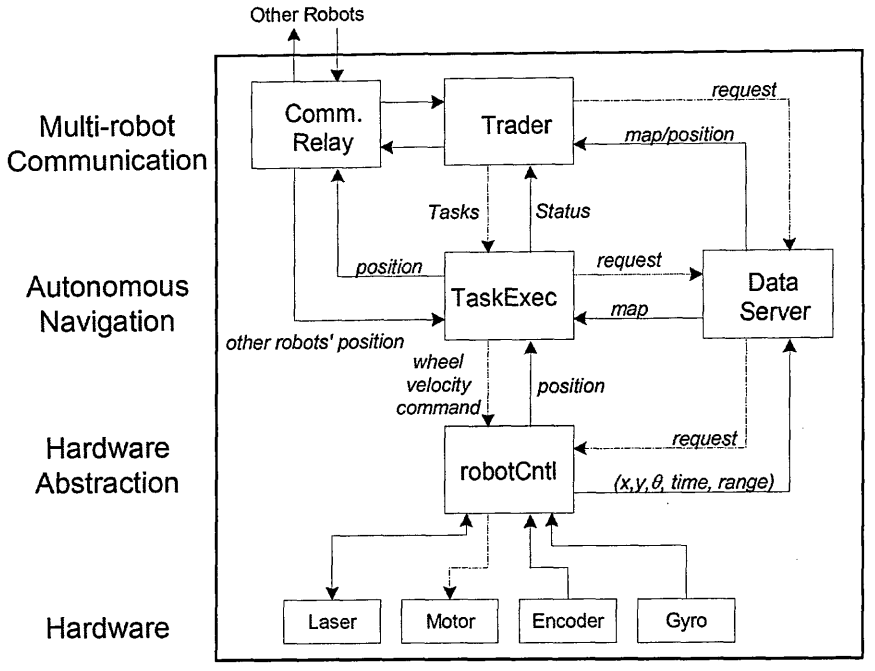


Figure 2.14 Architecture of a TraderBots Robot

Diagram reproduced from (Dias, Zlot, Zinck, Gonzalez, & Stentz, 2004).

Negotiation is a key factor within the architecture. Robots compete against one another to sell their services, whilst traders seek to sell tasks for the lowest prices. If costs are measured in terms of resources, then this negotiation helps optimise the response of the system.

A unique factor of the TraderBots approach is the ability for agents to act as consultants, and to contract out tasks and form work groups. For example, a robot with the ability to plan can buy a task, divide it into parts, and sell on those parts to other robots. In doing so, it may be able to reduce costs by enabling robots with less functionality to complete a task instead of it being undertaken by the most capable robot.

Robot failure is handled by reassigning tasks held by that robot to the remaining robots. In the case of a partial robot malfunction, measured as a problem with resource availability or an unexpected rise in cost, the robot attempts to sell off

its incomplete tasks. It will sell off all of its remaining tasks, even if it incurs a loss in profit. If a robot is incapable of trading, it is considered dead. In this case, the robot cannot sell off any of its remaining tasks. To overcome this, each trader keeps track of all awards it makes or receives. Once a robot death is identified, all traders which awarded tasks to the dead robot communicate with the remaining robots to find out if the tasks were sub-contracted. If so, the two robots renegotiate costs for the task. If not, the entire task is re-auctioned.

Coordinated task decomposition is introduced by Zlot and Stentz (2003). A mission is inputted into the system as a task tree. The tree is auctioned off, with robots able to bid on individual tasks (primitive tasks), or complete branches (abstract tasks). Some abstract tasks require all subtasks to be performed (AND tasks) whilst others only require one of several alternative subtasks to be performed (OR tasks). Bids for primitive tasks are made using the expected cost of the task, bids for abstract AND tasks are made using the minimum estimated cost for completing all of the subtasks, and bids for abstract OR tasks are made using the minimum estimated cost for the cheapest subtask. It is also possible that robots can come up with their own plans for completing an abstract task, in which case they can bid using an estimated cost based on their own plan. Although allocating a complex task in a more distributed way, there is no mention of how the task is initially decomposed into subtasks.

Coordinated task execution is studied by Kalra and Stentz (2003). This work examines the application of the architecture to a finer granularity of interactions. This extension is implemented by enabling robots to make shorter duration plans and hold more frequent auctions. A robot that identifies a coordinated set of actions,

which may be more profitable than the standard behavioural approach, can set up an auction to sell its plan. Although this enables plans to be changed more dynamically, and thus allows robots to be more responsive to each others actions, it does not solve the tightly coupled class of problems detailed in section 2.1.5.

TraderBots is demonstrated for distributed sensing (Dias, Zlot et al., 2004), the multi-depot travelling salesman problem (Dias, Zinck, Zlot, & Stentz, 2004), and perimeter sweeping (Stentz, Dias, Zlot, & Kalra, 2004).

For other multirobot architectures, see: overviews (Cao, Fukunaga, & Kahng, 1997; Parker, 2003), market architectures (Lemaire, Alami, & Lacroix, 2004; Stentz & Dias, 1999), voting (Sorbello, Chella, & Arkin, 2004), computational requirements (Gerkey & Mataric, 2003).

2.2.10 Architecture Comparison

For the problem stated in chapter 1, table 2.1 shows which of the above architectures are capable of tackling, or demonstrating, the associated characteristics (indicated by a '1' in the table). We have added an extra column headed *constant tasks*. This shows which architectures ensure robots are always active and productive, even between performing allocated tasks. Architectures which do not have constant tasks allow robots to become idle and unproductive. We also distinguish against architectures which only describe a single mission, and allow robots to simply carry on performing the same action, or stop, after the mission has been completed. Although this is not part of our problem, it is an issue which will distinguish our architecture from the rest.

Table 2.1 Comparison of Multirobot Architectures

Architecture reference	Constant task	Complex task	Weakly-defined task	Multilevel interactions	Dynamic environment	Competitive environment	Unpredictable environment	Emergent behaviour
ACTRESS	0	0	0	0	0	0	0	0
GOFER	0	0	0	0	0	0	0	0
Swarms	1	0	0	0	1	0	1	1
ALLIANCE	0	0	0	0	1	1	1	0
M+	0	0	0	1	0	0	0	0
Leader-Follower	0	1	0	1	1	0	1	0
MURDOCH	0	0	0	0	0	0	1	0
CAMPOUT	0	1	0	1	1	0	1	0
TraderBots	0	1	0	1	1	0	1	0

All of the architectures require tasks to be explicitly stated. In architectures that allow for high level missions to be specified, precompiled libraries, knowledge bases, and task trees are used as references to look up the sequence of achievable robotic tasks required to complete the mission. A key problem is how to decompose a high level, or complex, mission into a set of achievable tasks.

The approach outlined in this thesis decomposes a high level emergent behaviour into its constituent sub-behaviours using a learning mechanism. If the behaviours at the lowest level are each achievable, in their own right, by a robot, then the high level behaviour has been sufficiently decomposed. If by reproducing the low level behaviours, we can reproduce the desired system level behaviour, then the decomposition is successful.

2.3 Robot Football

Robot football (or soccer) was devised by Mackworth (1993), and gained popularity through the RoboCup initiative started by Kitano et al. (1997). It is a domain which encompasses all the requirements and attributes listed in table 2.1, giving us an ideal platform for undertaking this research. Robot football is the focus of a large and successful research community and, following the climax of Deep Blue beating Gary Kasparov at chess in 1997 (Campbell, Hoane, & Hsu, 2002), has been proposed as the new benchmark challenge for Artificial Intelligence (Kitano & Asada, 1998).

In essence similar to human football, robot football is a game played by two teams of simulated or physical robot agents on a rectangular pitch, whereby the aim is to transfer a ball into the opposing team's goal area. Many different leagues exist, each played in competition at international level, with research institutions battling it out to show their systems are the most advanced. These leagues range from one-on-one humanoid games, through large, distributed, 5-a-side wheeled teams, to small, fast, centralised, 11-a-side robot games, and simulated games.

Of these leagues, we focus on the highly dynamic, autonomous, centralised, 5-a-side and 11-a-side games. These leagues in particular exhibit the following qualities, which are central to our research:

- The game is weakly-defined – The objective, of scoring more goals than the opponent team, is a simple enough concept (subject to the constraints of the rules). However, the actions required to achieve that objective are not clearly defined.

- Controllers need to have a multilevel structure – A complete solution to playing the game requires precise control of robot movement and skills at a low level. At a higher level, robots must be able to work together, pass the ball, and set up opportunities for passing and shooting. At an even higher level, teams must be able to make plans, generate formations, and adapt to changing opponent strategies.

- The game is highly dynamic – The state of the environment is always changing, irrespective of the actions of the robots under our control. In the leagues studied in this thesis, controllers must act in time frames of milliseconds.

- The game is competitive – The opposing team will always be acting to disrupt play and prevent the home team from achieving its objective of scoring goals.

- Robot football is unpredictable – The number of factors affecting the state of the game, its sensitivity to them, and its high dynamicity, makes accurately predicting future states impossible.

- Controllers show emergent properties – Although they may be designed to perform in a specific manner, the complex nature of the game makes it impossible to design a solution for every possible event. Correspondingly, the low level interactions of the robots often generate interesting, and unintended behaviours.

- Robot football is complex – It has a multilevel structure, is non-deterministic, non-linear, and shows emergent properties. There are many

interacting variables ranging from the position, velocity, and acceleration of the ball and robots, to the structures and formations generated on the pitch.

Our interest in such systems stems from our involvement with the competitions of Mirosot (Robinson et al., 2004) and the RoboCup Simulation League (Iravani, 2005b).

2.3.1 Football Platforms

There are many robot football leagues, focusing on different technologies and aspects of the game. Each league has its own platform, which is often defined by the league rules. These may regard the form the robot, the number of players on a team, the types of sensors and actuators, or the control structure; whether it is centralised or distributed, for example. Much work has been focussed on the supporting technologies, such as machine vision (Lee, Hwang, Kim, Chung, & Kuc, 2005; Messom, Sen Gupta, & Sng, 2001; Weiss & Hildebrand, 2004; Yu et al., 2003), hardware (Aun, Lin, Quiang, & Seng, 2005; Novak, 2004), and control (de la Rosa, Oller, Vehi, & Puyol, 1996; Messom, 1998; Nitschke, 2006; Solc & Honzik, 2002).

In this thesis we shall be concerned with just three of the leagues: Mirosot, Simurosot, and the RoboCup Simulation League. All three of the above use teams of fast-moving, small, wheeled robots (or simulated equivalents), controlled from a centralised location. We propose that the games played in these leagues utilise space, and can be controlled, in similar ways. The following sections describe the platforms for each of these leagues in more detail, and highlight any additional information specific to their use in this thesis.

2.3.1.1 Mirosot

A Mirosot game consists of two teams of five robots, each managed by a centralised controller running on a host computer. The majority of feedback is provided by a downward pointing camera located over the centre of the pitch. Autonomous software analyses the image, locates positions of the robots and ball then passes this information to the controller, which determines the desired movements of the robots, sending commands to them over a wireless link. Secondary sensing is provided by encoders on the motors of each robot, and is used for velocity control. The robots are no larger than 7.5 cm × 7.5 cm × 7.5 cm, and usually have no mechanism for controlling the ball. Rather than 'kick', robots push the ball in front of them. The game is played over two 5 minute halves using a golf ball on a 1.7 m × 2.3 m pitch. A diagram of the system is given in figure 2.15.

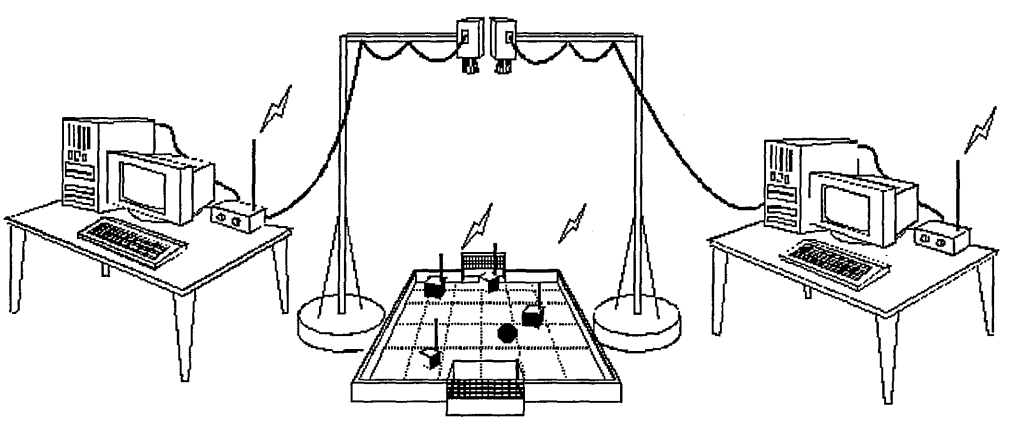


Figure 2.15 Mirosot System Diagram

Each team of robots is controlled by commands sent over a wireless link from a central computer. Information is gathered from an overhead camera. Diagram reproduced from the Mirost class rules (FIRA, 2002).

It is usual for a separate camera and computer to be used for each team, with unique software running on each. In our experiments we run both teams off one computer and vision system to remove the discriminatory effects of using separate equipment. Each team will, however, use a unique controller. Our Mirosot system has been developed over the course of this research in collaboration with the University of Warwick and the University of Plymouth. Details of the collaboration are given in (Robinson et al., 2004).

2.3.1.2 Simurosot

Simurosot is a simulation of Mirosot. Like Mirosot it is an international competition standard. Simurosot removes the image capture and processing issues of Mirosot, and passes simulated positions directly to the controller. Likewise, communication and noise are also removed from the system. We have designed our Mirosot software to be compatible with Simurosot so that controllers can be run on both platforms without any modification. This allows us to easily test controllers before running them on the real robots, and also to test the effect of noise and other real world influences on the system.

The Simurosot simulator, and consequently the Mirosot system, uses imperial units for its coordinate system. We will endeavour to use metric units for most measurements in this thesis, though it should be noted that this conversion will often result in peculiar ranges and accuracies.

2.3.1.3 RoboCup Simulation League

This is a simulated competition comparable to Simurosot but using eleven players on a team, rather than five. Teams are controlled in a centralised manner and use

similar rules to those used in Mirosot and Simurosot. The RoboCup simulation is more elaborate in that players are omnidirectional and can hold or kick the ball. This allows teams to use more intricate controllers, incorporating passing and dribbling, which are not possible in Simurosot or Mirosot. The RoboCup simulation also includes other features such as energy levels, meaning players tire out if they are particularly active.

Log files containing position information from competition matches are available for download from the internet, and have been used in previous work by Iravani and Johnson (2005). These files are used as a basis for our initial work examining structures in team games, and typically contain 6000-6500 frames worth of data. Conversely, the Mirosot and Simurosot platforms give us systems on which we can implement the controllers developed in this thesis.

2.3.2 Strategies, Roles, and Plays

The part of the robot football controller that handles the behaviours for playing football is called the *strategy*. In the case of Mirosot, Simurosot, and the RoboCup Simulation League, each strategy typically takes in the game state as a set of arguments, and returns wheel velocities or target positions to control movement of the robots. The strategy is responsible for making all the decisions regarding game play, and can be seen as the means for completing the objective; scoring goals. It is the parallel of the plans used in section 2.2 to achieve an objective. Strategies are usually compiled into a single file, and can be loaded into the robot football system. In this way, teams can prepare a number of different strategies and select the most appropriate based on the perceived weaknesses of their opponent.

Just as plans are decomposed into tasks, strategies are typically decomposed into *roles*. These roles contain achievable actions for each robot, such as positioning, movement, area boundaries, passing and shooting (Alvaro, Freedman, & Gonzalo, 2006; Fassi, Scarpettini, & Santos, 2003; Han, Lee, Moon, Lee, & Kim, 2002; Klancar & Matko, 2005; Veloso, Bowling, Achim, Han, & Stone, 1999). They are usually based on functional concepts relating to human football, such as goalkeeper, defender, or striker. Roles can be defined for the duration of a match, or be switched or assigned temporarily to extend functionality (Gerkey & Mataric, 2004; Kim, Kim, Kim, Kim, & Vadakkepat, 1998; McMillen & Veloso, 2006; Sng, Sen Gupta, & Messom, 2002). An example would be for a kick-off, where robots might be given a sequence of specific movements to perform before defaulting into their main roles. We call this approach to control a role based strategy.

An extension of these ideas by Bowling, Browning, Chang, and Veloso (2004) introduces an intermediate layer based on the idea of *plays*. A play is a plan for coordinating a set of robots in response to a particular game state. If a strategy is the plan for achieving the objective of scoring goals, then plays are plans for achieving sub-goals. These might be defending the home goal, attacking the opponent goal, promoting the position of the ball, taking a penalty or free kick, etc. Each play contains information regarding the validity period for that play, and the roles of any robots included in that play. Additionally the play might contain additional information regarding predefined action sequences, additional specific role information, and any sequencing or rules for switching roles.

A typical role based robot football controller uses three layers of abstraction based on the ideas of roles, plays and strategies. Figure 2.16 shows the relation

between roles plays and strategies in a robot football architecture, and highlights the links with tasks, plans and objectives in a typical task decomposition process.

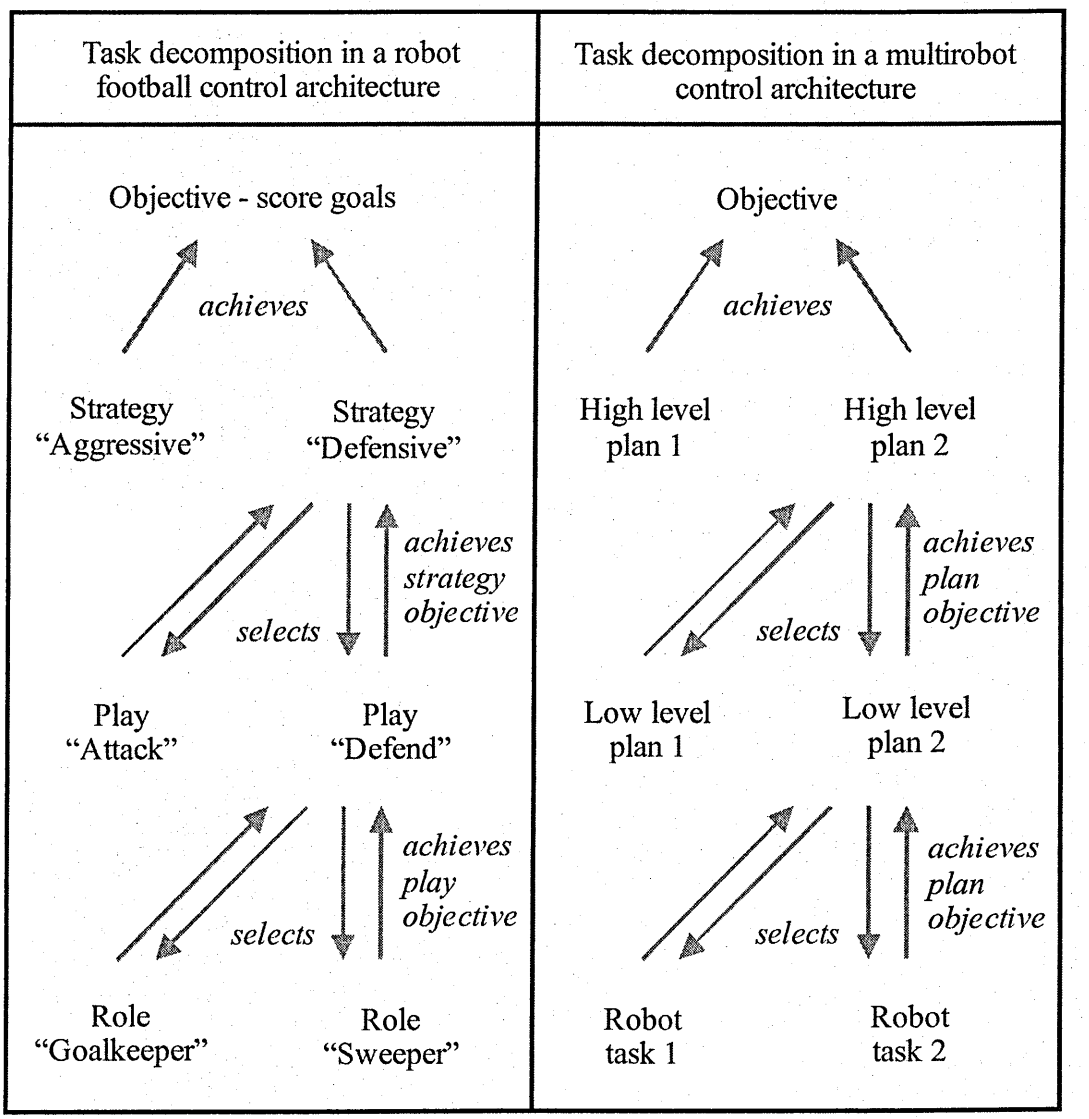


Figure 2.16 Typical Task Decomposition in Multirobot Systems

The number of roles plays and strategies depicted has been limited for clarity. Each play in a robot football architecture will usually consist of the same number of roles as there are players on the team.

At the highest level is the strategy layer, which contains a method for selecting between plays, based on the game state. Most team strategies utilise static play selection, whereby only one play is valid for any game state, though, as we will see

later, there are exceptions. At the intermediate level is the play layer (or play book), which holds all of the plays contained within the strategy. The two-play approach is common, which consists of a defensive play, for when the ball is in the home half, and an attacking play, for when the ball is in the opponent's half. Each play contains a set of roles which are allocated to the players when it is activated. Typically a defensive play will consist of roles which make robots take up positions around the home goal, attempting to block the movement of the ball. Attacking plays on the other hand will consist of roles causing robots taking positions further up the field, trying to push the ball closer to the opponent goal. Simple role, play and strategy structures are given in listing 2.1.

```
1:  Role: Goalkeeper
2:    X_target = goal_X_position
3:    If ball_y_position > goal_top
4:          Y_target = goal_top
5:    Else If ball_y_position < goal_bottom
6:          Y_target = goal_bottom
7:    Else Y_target = ball_y_position
8:    End
9:    GoTo(X_target, Y_target)

10: Play: Defensive
11:   Priority1 = nearest_robot_to_goal
12:   Priority3:Priority5 = available_robot_ID1:available_robot_ID3
13:   Role(Priority1) = Goalkeeper
14:   Role(Priority2) = Arc_Defender(origin, offset_angle1, radius)
15:   Role(Priority3) = Arc_Defender(origin, offset_angle2, radius)
16:   Role(Priority4) = Line_Defender(point1, point2)
17:   Role(Priority5) = Sweeper

18: Strategy: Basic
19:   If user_event = centre_kick
20:         Play = Kickoff
21:   Else If ball_X_position < centreline
22:         Play = Defensive
23:   Else
24:         Play = Attacking
25:   End
```

Listing 2.1 Pseudo Code Role Based Strategy

The strategy component selects appropriate plays from the playbook depending on the state of the game. The play then assigns roles to each robot.

Although sufficient for 5-a-side matches, this approach is not scalable, and deteriorates when additional players are incorporated. A team with 11 robots requires a much wider variety of roles and more complex plays. Implementing this many roles in a coordinated fashion is very difficult, and so roles tend to be replicated. Furthermore, this type of architecture is limited in its ability to adapt, and tends not to incorporate true cooperation. Any apparent cooperation is usually short lived and the effect of a pre-programmed set piece, such as a kick-off, where robots might be issued with a set sequence of passes and moves. At other times, players tend to work collaboratively, working toward the same goal, and supporting one another, but there is often little explicit cooperation between them. Finally, the generation of roles, plays, and strategies are themselves a concern; this is done by hand, as were the plan decompositions in section 2.2, and relies heavily on the experience of the designer.

2.3.3 Alternative Control Approaches

Machine learning is often used to improve the effectiveness of role based strategies. Typically these approaches focus on a single function, such as movement (Østergaard & Lund, 2003), or passing, and shooting behaviours (Hu, Kostiadis, & Liu, 1999; Wang, Yao, Wang, & Luo, 2005). Of particular note is the work undertaken by Bowling et al. (2004), which provides an investigation of using on-line learning algorithms to modify the high level strategy layer during competitions.

The architecture utilises a *playbook* holding a collection of traditional plays. For each conceivable situation there is a variety of applicable plays to choose from, each comprising of a set of robot roles, and rules for implementation. These rules

convey when the play is applicable, when it should be terminated, and special information regarding its execution.

Robot roles are dynamically assigned, and listed in a schedule. When a play is selected for implementation, the primary role is assigned first to the best qualified robot, followed by the remaining roles and robots. The roles contain a set of tasks to be undertaken in sequence, which are synchronised by the actions of the lead robot: when the lead robot completes its initial task, all robots in the play move to undertake the next action in their role.

The main contribution of this work is its ability to evaluate its own performance and dynamically adapt to an unknown opponent. Plays are initially selected at random, and assigned weightings based on their outcome. Subsequently, a selection algorithm ensures that successful plays, with higher weightings, will be selected more often, whilst unsuccessful plays are ignored. However, plays which do not get chosen also receive a weighting factor to maintain their chances of being selected in future situations.

The downside to this approach is that it is essentially a role based strategy, and the individual plays and roles still need to be generated by hand.

An alternative approach based on layered learning is introduced by Stone and Veloso (1998a). In this work, simulated agents first learn to perform low level tasks, using neural networks, which are then combined into higher level behaviours, learnt using decision trees. At the lower level, robots learn the task of intercepting a ball kicked with constant velocity toward a goal, by modifying a basic turn-and-run behaviour. Once learnt, this behaviour is then used in the higher level passing behaviour. In training, the ball is kicked to a randomly placed team mate, which

then attempts to intercept and return the ball. The positions of kicker and receiver in each test are then used to build a decision tree giving the probability of a successful pass based on these measurements. During testing, this decision tree is used to identify which, out of a set of possible team-mates, is most likely to intercept a pass. The two behaviours are combined in a carefully selected scenario to show how a sequence of passes might occur, although no experimental results are given.

Stone and Veloso (1998b) extended the work to include the low level behaviour of striking a moving ball at the opponent goal. In this example, the ball speed, trajectory, goal location, and position of striker are all incorporated into the learnt skill, making it much more applicable than the simple intercept behaviour.

The downside to this approach is that different learning methods are used for each layer, each requiring its own set of training data, leading to a long process for learning complete strategies. The procedure is not demonstrated on real robots.

Despite the novel approaches, both of the previous sets of work have relied on a hand coded decomposition of the football strategy. An alternative is provided by Luke et al. (1998), who use a genetic algorithm to evolve 11-a-side strategies from a set of basic functions including kicking and moving. In their approach, simplified game trees are constructed using if-then-else logic, whereby each measurable event on the pitch leads to a particular controllable robot action. To simplify the game strategy, two trees are created, one for moving and one for kicking. If a robot is in a position to kick the ball, the kick tree is called; otherwise the move tree is called. Beginning with randomly generated game trees, matches would be played between two teams and strategies evaluated using criteria including number of goals scored,

number of successful passes, and time spent in possession. At each evolution, sections of the trees were randomly replaced.

Two types of team were created: homogenous and pseudo-heterogeneous. In the first type, all players on a team used the same game tree. In the second type, teams were split into *squads*, with each having a unique game tree. This enabled single teams to develop specialised sub-groups such as attackers and defenders. An interesting aspect of the evolution of these pseudo-heterogeneous teams is the ability to swap whole game trees, and thus effectively trade good players and plays between teams.

Teams were co-evolved during this work, meaning that progress was only measured against other emergent strategies. No direct comparison to traditional strategies was made, so there is no measure of actual ability. Due to the complexity of the problem a number of constraints had to be made to increase the evolution speed. These included shortening game time and reducing the function set by hand.

A recurrent problem was the tendency for teams to evolve toward clustering strategies, when all players moved toward the ball. These were difficult to evolve from due to their success against other basic strategies. Such behaviour has also been seen in other evolutionary strategies, such as (Kobrin & Sinyavsky, 2006), in which neural networks were used to evolve team formations.

In summary, strategies built on roles require hand coded decompositions, based on the designer's experience. Those using a variety of learning algorithms to build behaviours from the ground up are development intensive, requiring many different skills to be identified and learnt using separate techniques. Those built using single methods to generate emergent strategies do not contain enough information to

produce competitive strategies, and doing so would require massive evolutionary processes.

Given these issues, is it possible to create a method of automatically decomposing the robot football task, using a single learning technique, which can be used to generate a competitive, emergent strategy?

This is a specific case of our problem. More generally, we are interested in generating an architecture that is capable of decomposing any complex system using a learning method, such that the resultant decomposition can be used to generate an emergent behaviour at the system level. By this, we mean that given a set of sensor data corresponding to a complex system, can we abstract a set of sense-response actions, which, when performed in a given sequence, cause the desired system level behaviour to emerge? Furthermore, can the abstracted actions be improved by analysing additional data as it becomes available?

We see the problem as largely one of representation. How can we represent the intricate requirements imposed upon a robot team undertaking a complex mission? Can we abstract structures which decompose the overall objective into comprehensible tasks? Can we generate these representations at a variety of levels of abstraction using a single technique?

2.4 Complexity Theory

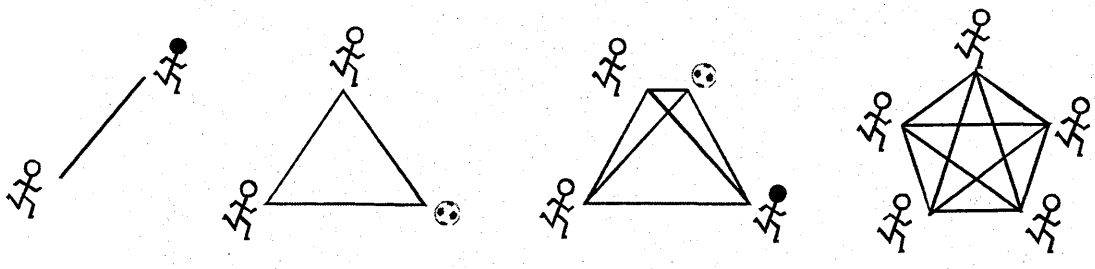
A method for analysis and representation which has proved itself for such complex systems is that of Q-analysis (Atkin, 1974). It is a multidimensional generalisation of network theory, which can be used to model relational structures between variables in a set. We base our work on extensions to this theory made by Johnson (2006) and Iravani (2005b). A very brief outline of some of the concepts we will use is given in the following sections.

2.4.1 Multidimensional Representation

The game of football has a multidimensional structure, which is one of the reasons it is so complex. To play the game requires a good understanding of some, if not all, of the relationships within its structure. Some relationships are global, existing in every game, such as are governed by the rules, whilst others may only appear in a single game or moment, being a trait of a particular team, or tactic. Some typical factors in these multidimensional structures may be the position of players, velocity of the ball, kick-off events, pitch edges, fouls and game time. To represent the relationships we will use the hypernetwork notation introduced by Johnson (2006).

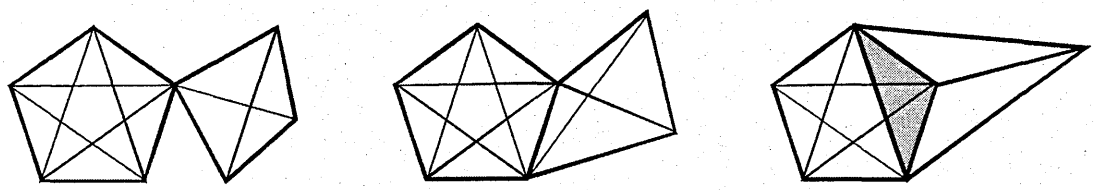
A *hypernetwork* represents structure between sets of nodes, a natural progression from a standard network, which represents structure between a pair of nodes. A network consists of agents related by lines, a 2-ary relation, whereas a hypernetwork can consists of agents related by lines, triangles, or any other polyhedron. A polyhedron with n vertices represents an n-ary relation and a polyhedron with $(p+1)$ vertices is called a *p-simplex.* Consequently, a set of simplices form a hypernetwork, with each simplex being an edge of the hypergraph.

Figure 2.17 shows some simplices representing possible structures in football, whilst figure 2.18 shows hypernetworks of connected simplices.



Mark Pass Two-on-one Formation

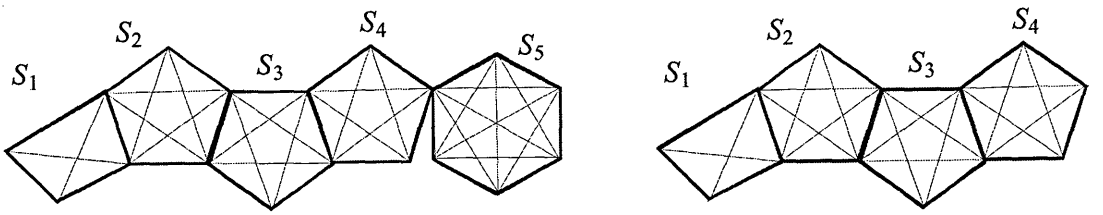
Figure 2.17 Some n-ary Relations in Football



1 Shared Vertex (0-near) 2 Shared Vertices (1-near) 3 Shared Vertices (2-near)

Figure 2.18 Hypernetworks of q-Near Simplices

Higher dimensional simplices can be decomposed into a set of lower dimensional simplices, called their *faces*. If two simplices share a set of $(q+1)$ nodes, then they will share a q-dimensional face, and are said to be *q-near*. Simplices sharing a single node are 0-near, while simplices sharing an edge are 1-near, and a triangle, 2-near (figure 2.18). Furthermore, two sets of simplices are said to be *q-connected* if there is a chain of *q-near* simplices joining the two. For example, figure 2.19 shows a hypernetwork of 5 simplices. Simplices 1 and 4 are 1-connected via simplices 2 and 3, but simplices 1 and 5 are 0-connected due to simplices 4 and 5 only sharing one common vertex. A set of connected simplices form a *simplical complex,* and mutually q-connected simplices are called *q-connected components*.

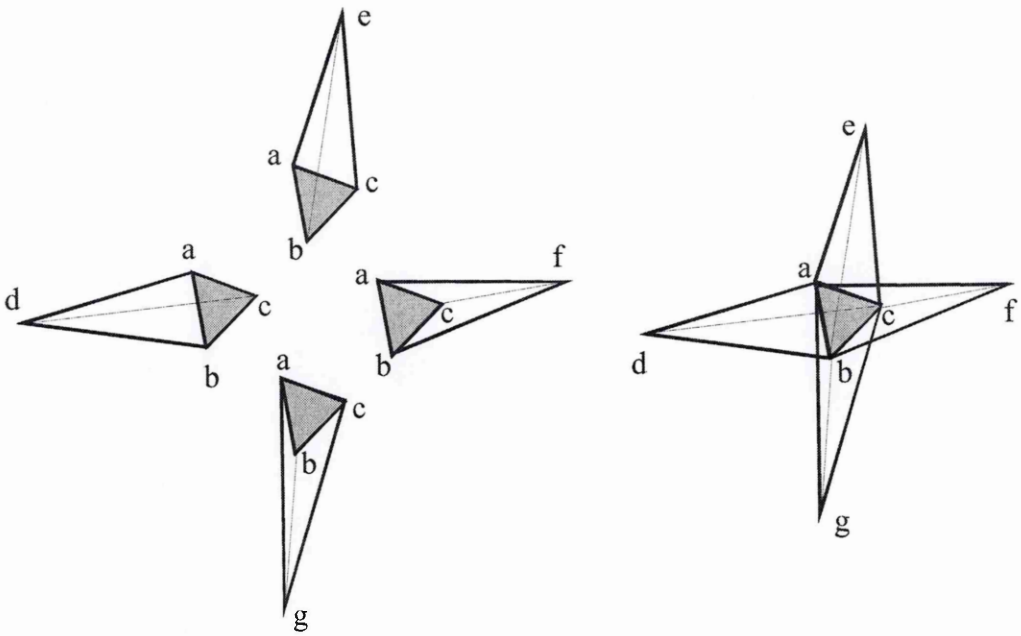


0-Connected 1-Connected

Figure 2.19 Chains of q-Connected Simplices

Simplices S_{1-4} are 1-connected, each sharing 2 vertices. S_5 is only connected to the chain by a single vertex, making the whole chain 0-connected.

The connectivity described above is based on shared faces of pairs of simplices. We can further this concept by considering shared faces between many simplices. Figure 2.20 shows four simplices ‹ *a*, *b*, *c*, *d* ›, ‹ *a*, *b*, *c*, *e* ›, ‹ *a*, *b*, *c*, *f* ›, and ‹ *a*, *b*, *c*, *g* ›, which all share the face ‹ *a*, *b*, *c* ›. The set of simplices is called a *star*, and the largest shared face is referred to as the *hub*. In this way, a hub signifies a strong correlation between the simplices. The more vertices contained in the hub, the stronger the link between simplices. Similarly, the more simplices forming a star, the more relevant the hub becomes in classifying those simplices. Therefore, hubs and stars can be used to identify strong links between sets of data.



(a) Simplices with shared faces

(b) Star of the simplices showing the common hub

Figure 2.20 A Star-Hub Configuration

We can tabulate the data given in a hypernetwork using an incidence matrix (table 2.2). By rearranging the rows and columns of this matrix, we can group the occurrences into blocks, or *maximal rectangles*, which correspond to the hubs of the hypernetwork. The *rectangle number* is the area of the maximal rectangle, and gives a value to the associated correlation. The larger the rectangle, the closer the correlation between simplices.

Table 2.2 Incidence Matrix for Figure 2.20

Simplex	Vertices						
	a	b	c	d	e	f	g
1	1	1	1	1	0	0	0
2	1	1	1	0	1	0	0
3	1	1	1	0	0	1	0
4	1	1	1	0	0	0	1

= Maximal rectangle

2.4.2 Multilevel Representation

As well as having a multidimensional structure, robot football is multilevel. We have already shown that traditional role based strategies for robot football use a multilevel structure of roles, plays and strategies, and this will be explored further in chapter 4.

In figure 2.17 we gave names to the simplices to identify what they represented. We can say that the simplex maps the set of nodes at one level to the named structure, which is a higher level of representation. These named structures are themselves elements in even higher level structures.

The conical structure shown in figure 2.21 represents the *Fundamental Diagram of Multilevel Systems*. The base of the cone represents a particular set of variables, whilst the sides of the cone represent a relation, which maps the set to a particular structure at the apex. If the set of variables lies at level *N*, the structure described by the relation lies at level *N+1*. In this way, the multilevel structure is closely linked to the idea of *emergence*. By applying a relation to a set of unstructured variables at level *N*, a structure emerges at level *N+1*.

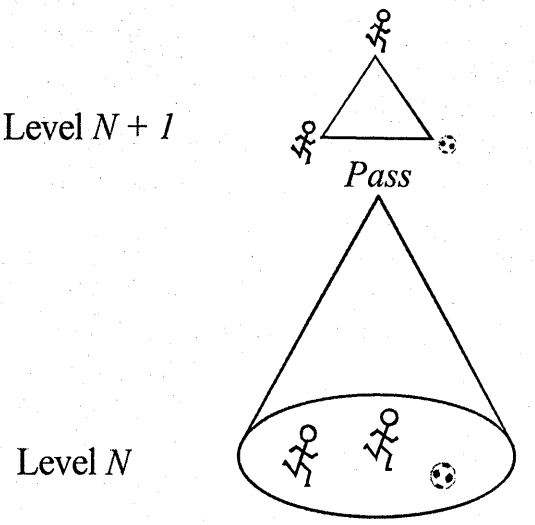


Figure 2.21 Mapping of Elements into Named Structures

The relationship described by the simplex is crucial. A set of elements configured in two distinct ways can have completely different meanings. Consider the sets shown in figure 2.22. Both show three players and a ball, $\{w_1, w_2, b_1, B\}$, though each has a different relationship, denoted R_1 and R_2. The relationship R_1 gives rise to the significant structure named *defenders dilemma*, whereas R_2 gives a separate configuration, which has no significant meaning, and has not been named. To distinguish between sets and structures, we use the notation $\langle w_1, w_2, b_1, B; R_i \rangle$ to represent the structure created by imposing the relation R_i on the set of elements $\{w_1, w_2, b_1, B\}$.

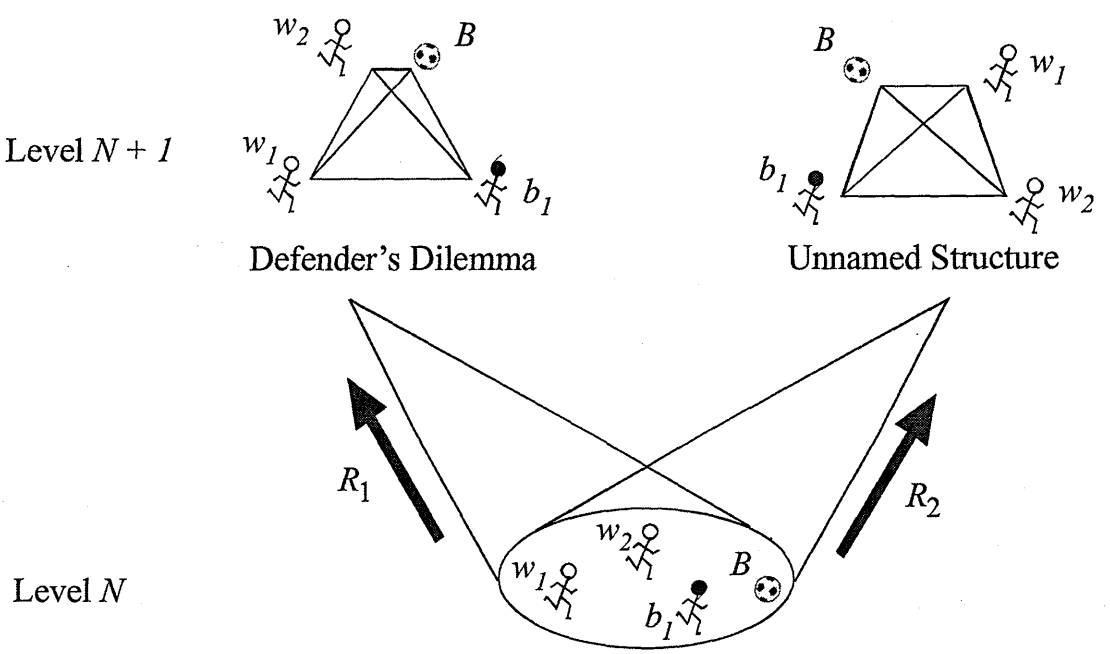


Figure 2.22 A Set of Elements Mapped into Two Distinct Structures

Figure 2.23 shows one possible multilevel representation of a role based robot football architecture. It depicts three distinct levels of a multilevel structure, with linking relationships. It can be seen that bases of cones can fully or partially overlap, but that when mapped to different relationships give rise to separate structures.

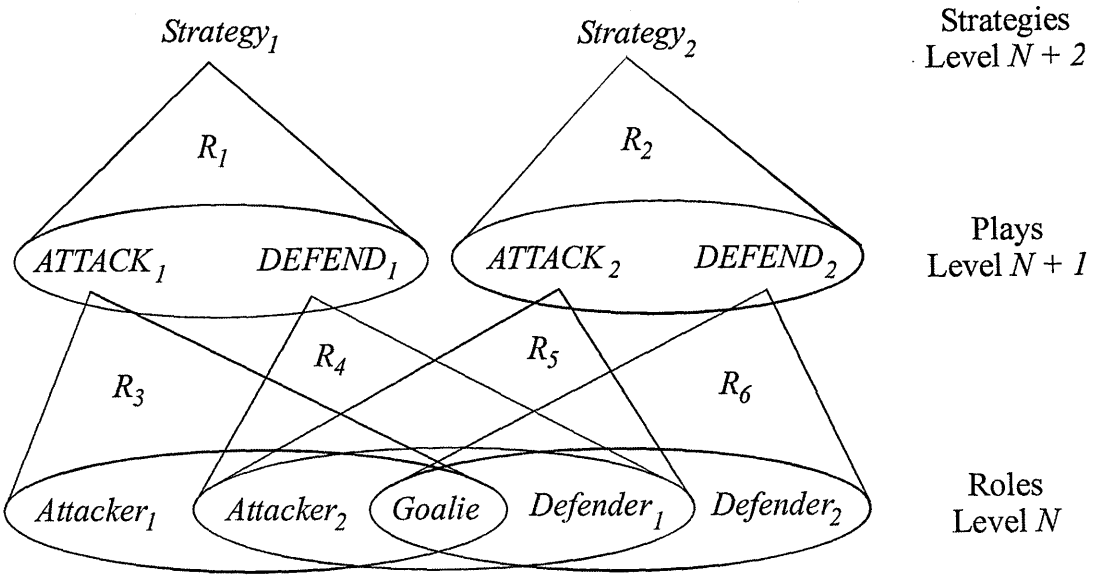


Figure 2.23 A Possible Football Structure

There are two varieties of aggregatory relationship, AND-aggregation and OR-aggregation. The majority of relationships we will consider in this thesis are AND-aggregated: the entire set occupying the base of a cone is required to generate the structure at its apex. Such a relation is shown in figure 2.21, where all the players AND the ball are required to generate the structure *PASS*. For the OR-aggregation, only one of the set is required to represent the apex structure. In figure 2.24 this is shown by the grouping of a set of plays into the structure *playbook*.

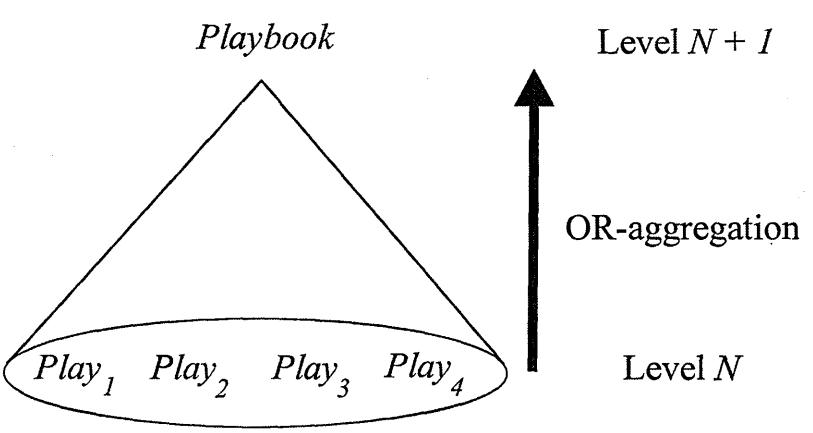


Figure 2.24 OR-Aggregation between Levels in a Multilevel Structure

2.4.3 Concepts and Concept Generation

Having defined notation to describe the multidimensional and multilevel structure of robot football, we now require a method for identifying the features which describe the game. Our work follows on from research conducted by Iravani (2005a), so will continue to use the foundations built on *concept generation.*

A *concept* is defined as "An abstract or generic idea generalised from particular instances" (Merriam-Webster, 2007). In this work a concept is an idea abstracted from a set of *primitives.* If a system is measured in terms of a set of *variables* or *properties*, then a primitive is a relation on a set of variables that describes some event or object. Primitives associated with similar events or objects will contain similar structures of variables.

We use simplices to represent primitives in this work. The primitive is drawn as a simplex, which depicts the relation between a set of measured variables. This structure of variables forms the primitive at level N, which can then be mapped onto the named concept, which appears at level $N+1$, as shown in figure 2.21.

Our approach is to take a set of primitives (simplices) associated with a concept and find their intersections. If the sets of simplices overlap to form a star, then the hub of this star gives us a possible *hypothesis* for relating the primitives. The hypothesis is that any simplex that contains that hub will be associated to the concept. For example, in figure 2.20, the simplices are all associated with some concept and share the face $\langle a, b, c \rangle$. We can, therefore, form the hypothesis that any primitive containing the structure $\langle a, b, c \rangle$ will also be associated the same concept.

If a set of simplices do not share a hub, then the associated primitives are members of separate concepts. Similarly, if stars form more than one hub, then the primitives involved are members of more than one concept. A hub that can be used to distinguish between two distinct concepts is called a *classifier hub*.

Iravani (2005a) distinguishes between two distinct varieties of concept. *Generalisation concepts* are described as concepts that represent a class of primitives. For example, three different ball passes in football can all be generalised to the concept *PASS*. A single pass is sufficient to be classed as part of the concept. The second concept is called a *relational concept*; relating a set of distinct primitives via some structure. In this case, the concept *PASS* could be made up of a ball, a passing player, and a receiving player, in a certain configuration. In this example all three primitives, and the structure, are required to generate the concept. Johnson (1983) defines generalisation concepts as being an OR-aggregation of primitives, whereas relational concepts are the result of an AND-aggregation.

To enable us to use concepts to drive behaviours in our footballers, we also have to link them with *representatives*, which are hubs characteristic of a particular set of simplices. These are representations of the concept, which can be used to generate command information for controlling the robots.

In a multilevel structure, the concept generation is performed at each level. In this way, concepts at one level become variables at a higher level of abstraction. This is shown in figure 2.25.

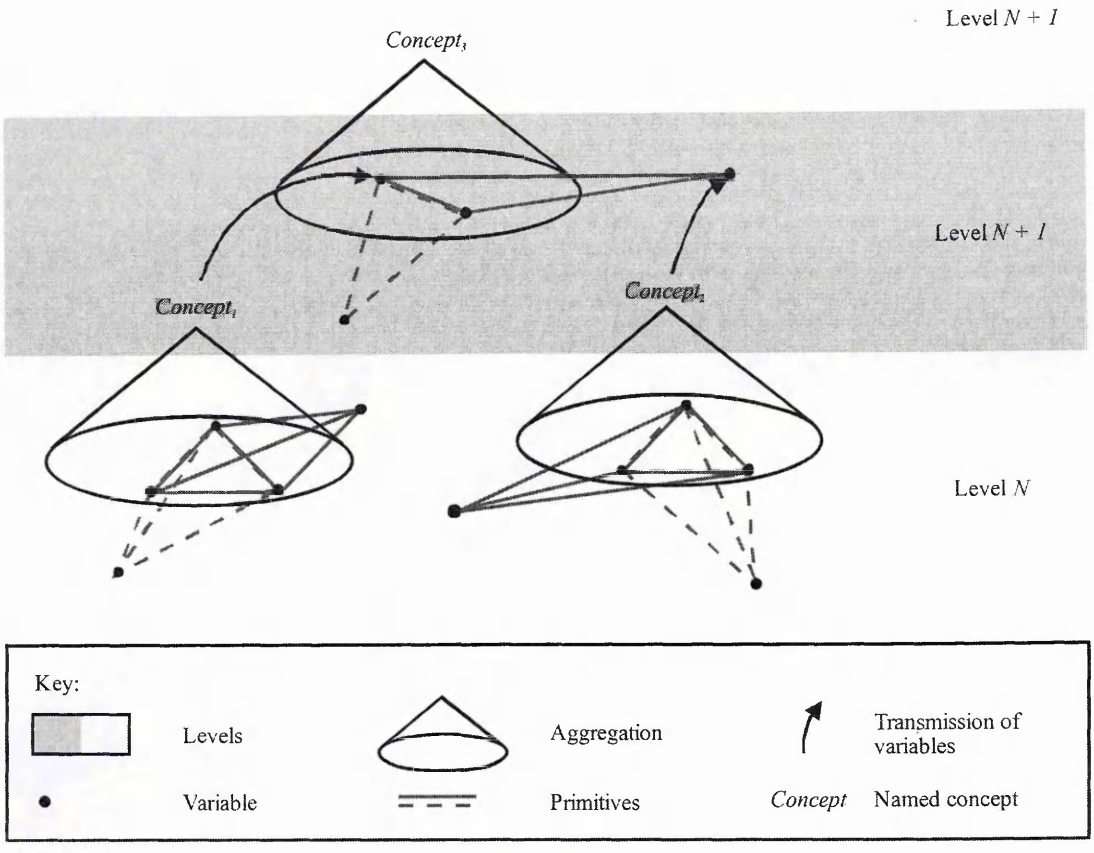


Figure 2.25 Multilevel Concept Generation

A structure of variables common to a number of primitives aggregate to a concept, which itself is a variable at a higher level.

Our objective in this thesis is to generate concepts relating to the many aspects of a complex system, measure the variables of associated primitives, and form hypotheses about their connections. By performing this analysis at various levels in the system, we can build a set of representatives of key concepts, which can then be used to control a set of robots to perform the complex task. Moreover, by identifying and programming individual concepts in this way, the resulting behaviour will be emergent, a function of the interactions of the individual concepts.

2.4.4 Summary

In the previous sections we have introduced a large number of ideas from complexity science. Table 2.3 and figure 2.26 summarise those which are integral to the work described later in this thesis.

Table 2.3 Comparison of Concept Generation and Hypernetwork Terminology

Concept generation terminology	Hypernetwork terminology
Concept – An abstract object generalised from a set of primitives bound together by a hypothesis	Hub – The largest face shared by a set of simplices. It represents a relation on a set of elements common to a number of simplices
Primitive – A measured relation of variables corresponding to a concept	Simplex – A structure formed by mapping a relation onto a set of elements
Variable – A measureable property of a system	Element – an object used to describe a system

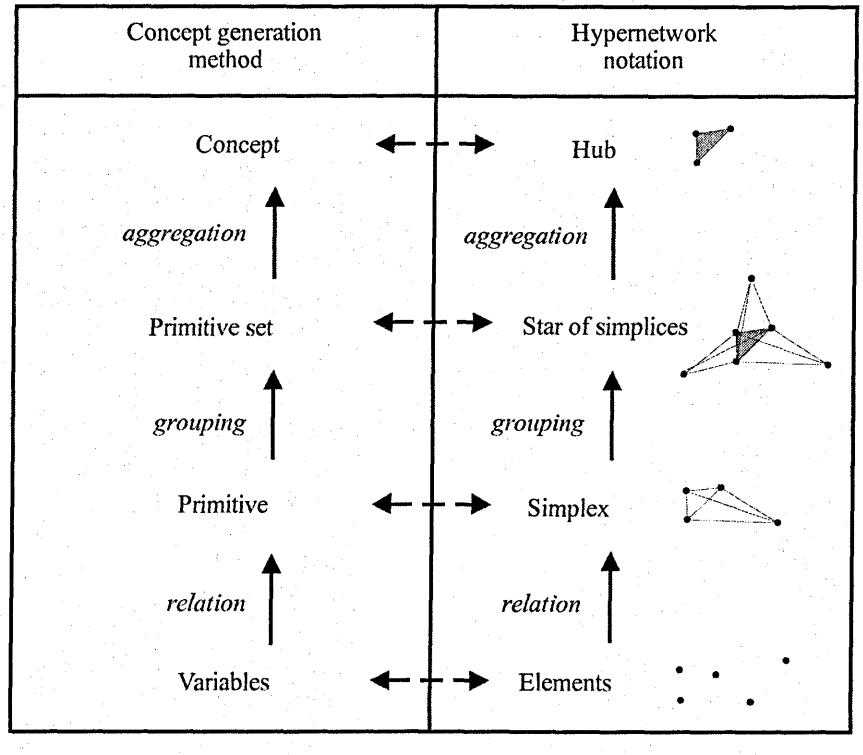


Figure 2.26 Relation between Concept Generation and Hypernetworks

Showing how the hypernetwork notation will be used to describe the process of concept generation.

2.5 Summary

This chapter has introduced some important themes in robotic control. It began with an introduction to some common control issues and problems relevant to the work in this thesis. This was followed by a brief summary of the most popular multirobot control architectures to date, including their strengths and weaknesses. Evaluating each of these architectures showed that they were not suited to our particular set of problems. Moreover, the majority of these architectures required well defined task descriptions; high level missions entered into these systems are inputted as a set of robot achievable tasks, which together fulfil that mission. None of the architectures showed the ability to autonomously decompose a complex mission into achievable tasks.

The third section introduced robot football as a complex system incorporating all of the problems highlighted in chapter 1. Three types of robot football: Mirosot, Simurosot, and the RoboCup Simulation League, were described, which will be used as experimental platforms in the later stages of this thesis. The traditional role based approach to robot football was discussed, showing how a strategy is composed of a set of plays and roles, which are described as follows:

- Strategy – A high level plan describing how to score goals against an opponent. It typically selects a play for implementation based on the state of the match.
- Play – An intermediate level plan, describing how to achieve a particular sub goal. It is only valid for a certain game state, and describes a set of roles to be implemented.

- Role – A low level behaviour assigned to a robot. It describes a set of tasks for the robot to undertake, usually parallels of human football player positions.

Problems with the approach are highlighted regarding its scalability, the lack of inherent cooperation, and a reliance on task decomposition by hand.

A set of alternative approaches to strategy generation, using learning techniques, are discussed. Those focusing on improving the performance of individual roles, or adapting play and role selection, still rely on roles and plays being defined by hand coded decompositions, and are greatly influenced by the experience of the designer. Where strategies are learnt from the ground up, by first learning individual behaviours, then mapping these into plays, a multitude of methods are required, making the process development intensive. Those strategies built using single evolutionary processes to generate emergent strategies do not contain enough information to produce competitive strategies, and doing so would require massive evolutionary processes. In keeping with our ideas, we suggest using a single learning technique to automatically decompose the robot football task, in such a way that the decomposition can be used to generate competitive, emergent strategies.

In the last section of this chapter we introduced some themes of complexity science, which will be used to develop the analysis method in chapter 5. It began with an introduction to multidimensional representation, describing how sets of related elements, and their structure, could be described by a simplex. A number of simplices form a hypernetwork, which describes connectivity in a multidimensional structure. Simplices in a hypernetwork can be disjoint, or connected through shared

faces. This leads to an important definition: that a hub is the largest shared face between a number of simplices, and represents a common relation between a set of elements. We also showed how simplices and hubs could be described using incidence matrices and maximal rectangles.

The idea of multilevel representation was introduced using cone diagrams. We showed that by applying a relation to a set of unstructured elements at level N, a structure *emerges* at level $N + 1$. We also showed how this could be represented in terms of simplices, and how two separate relations on the same set of variables generated two separate structures. Two types of aggregation were introduced, with OR-aggregation occurring when a structure can be described by a single element from a set, and AND-aggregation occurring when a set of elements is required to describe a structure.

Finally, we introduced the theory of concept generation. A concept was defined as an abstract object generalised from a set of primitives bound together by a hypothesis. These primitives are measured relations on sets of variables, which are themselves measurable properties of a system. Furthermore, a representative is the relation of variables used to represent the concept. The difference between generalisation and relational concepts was established as that between a concept representing a class of similar primitives, and a concept relating a set of distinct primitives via some structure. We also showed how concept generation could be performed using simplices and shared faces (stars and hubs), or by incidence matrices and maximal rectangles.

Chapter 3

Spatial Structures in Autonomous Goal Seeking Systems

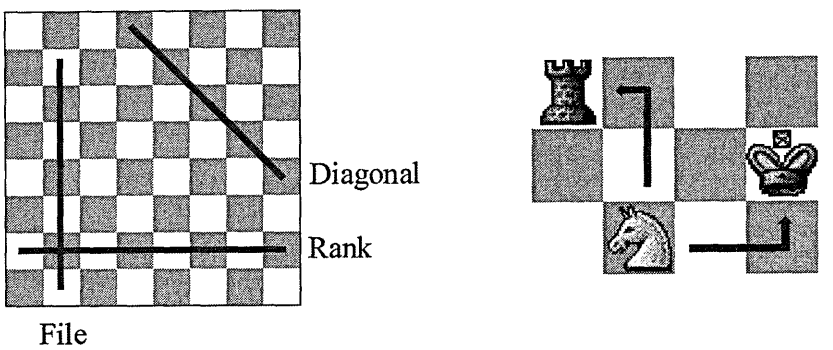
In a complex multiagent system there often exist a number of observable spatial relationships between agents which can be linked to the objectives of that system. Frequently these relationships are secondary by-products of many complex interacting rules which define a task, although they can also be governing rules themselves. For example, in a traffic system, drivers of vehicles maintain spaces between each other which are loosely based on the concepts of speed, thinking, and braking distances. These change whether the vehicles are following each other along a road, or emerging from a junction. On the other hand, figure skaters must coordinate to perform set moves and holds, which are the focus of their routines.

We are interested in these types of relationship, and suggest that they can be used to create powerful multiagent control strategies for use in complex and dynamic environments. The objective in this chapter is to examine the importance of spatial structures in competitive games, particularly in relation to robot football.

3.1 Complexity in Competitive AI Games

The research begins by considering the classical AI benchmark of computer chess, which is quintessentially concerned with structuring space (Atkin, Hartston, & Witten, 1976). This is illustrated in figure 3.1(a) where we give *names* to configurations of squares on the chess board. In figure 3.1(b) the spatial structure of the three pieces forms a structure called the *knight fork*, in which the knight checks

the opponent's king, and threatens the more valuable rook. These structures were known long before the invention of electronic computers, and the way that humans understand and manipulate them has long been held as an indicator of human intelligence.



(a) Structured space in chess (b) The knight fork

Figure 3.1 Structures in Chess

From the perspective of today, it can be seen that one of the very attractive features of chess for testing machine intelligence is the simplicity of its form and its rules. A grid of sixty four squares and thirty two pieces is a 'small' system. The rules of the system are also relatively straightforward, determining how the pieces can move, and what constitutes a win or draw. Crucially, the dynamics of chess are very simple from a modern viewpoint: (i) time in chess is governed by simple alternate move events (although human players are constrained to another time governed by the clock, bringing in an element of psychology), and, (ii) when a chess game is started from the same position, and the same moves are played, the same outcome will be observed as on previous occasions. On a higher level of complexity, and with more obvious reliance on spatial structures, is the game of Go. In Go, players take turns to place coloured stones on a 19 × 19 position grid until both players pass. The objective is to surround the opponent's stones, or to surround

contiguous sets of the opponent's stones, and to end owning the majority of territory once captured stones are accounted for.

Artificially Intelligent programmes have been generated to tackle both chess (Campbell et al., 2002; Hsu, Anantharaman, Campbell, & Nowatzyk, 1990) and Go (Churchill, Cant, & Al-Dabass, 2001; Müller, 2002) as well as the simpler games of checkers (Samuel, 1959; Schaeffer et al., 1992), othello (Buro, 1993) and nine men's morris (Gasser, 1996). All of these are goal seeking competitive systems played out with multiple pieces on a grid of squares, with turns taken in discrete periods of time. Robot football can be considered in a similar way, with the robots becoming pieces on a board divided up into pixels of the overhead camera, with turns taken in discrete portions of time divided up by the sampling rate of the camera.

The traditional approach to solving this type of problem using computational methods is to generate a game tree of all foreseeable moves, then search the entire space to find the sequence with the best chance of success. The size of the complete game tree is X^n, where X is the number of possible moves during any turn, and n is the total number of turns. Table 3.1 (adapted from (Bouzy & Chaslot, 2006)) gives estimated sizes of the search spaces for the above games.

Table 3.1 Estimated Game Tree Complexity for some Standard Games

Estimated game tree size			
Checkers	Othello	Chess	Go
10^{32}	10^{58}	10^{123}	10^{360}

If the pixels of the vision system are seen as squares on the playing field, with turns measured as frames, the complexity of robot football can be calculated in a

similar way: At each turn, a robot can move anywhere within a circle, with radius proportional to its velocity. For a camera with a resolution of 640 × 480 pixels capturing a 180 × 220 cm pitch at 30 frames per second, a robot moving at 1 m s^{-1} can move to any of approximately 600 squares. Therefore, with 10 robots on the pitch, 6000 possible moves can be made each turn. If a game lasts for two 5-minute halves, there are 18,000 turns, meaning the total complexity can theoretically reach $6000^{18,000}$. This is an estimate, as a number of factors, such as acceleration limits, obstructions and periods of inactivity, reduce the effective complexity of the game. A key difference between robot football described in this manner, and the board games described above is its non-deterministic nature. Due to noise in the sensing system, and the physical interactions of the robots and pitch, two sets of matches played by two identical sets of teams from the same start positions will never play out in exactly the same way. This makes predicting future states in the game extremely difficult.

For smaller games, such as noughts and crosses, nine man's morris, or othello, intelligent strategies can be successfully created based on brute force approaches to searching the game tree. In more complex games, such as chess and Go, the entire game tree is too large to generate, and techniques have to be employed to reduce the number or length of branches (Bouzy & Cazenave, 2001). Both of the latter have recognised set-piece openings, end games, and gambits, which can be used to this end. Even considering these set-pieces, the complexity of games such as Go can be too great, requiring game trees to be reduced to a limited number of turns in duration. Table 3.2 (adapted from (Bouzy & Cazenave, 2001)) indicates the success

with which some of the leading algorithms perform when compared to human players.

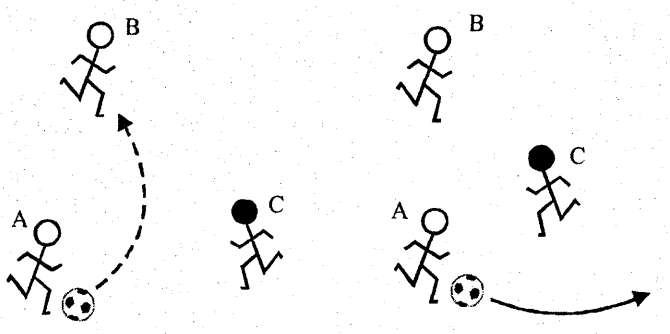
Table 3.2 Ability of AI Game Programs Compared to Human Players

Game	Relative skill levels
Checkers	Chinook (Schaeffer et al., 1992) > Human
Othello	Logistello (Buro, 1993) > Human
Chess	Deep Blue (Campbell et al., 2002) >= Human
19 × 19 Go	Strongest Go Program << Human

Football also contains many set-pieces that will help to reduce the substantial complexity of our representation. For example, human footballers are experts at mastering space. They demonstrate remarkable skills in movement and perception, well beyond the current state of the art in robotics. They base their game on the skills and set pieces they practice before a match, though the successful implementation of these tactics depends on the players' abilities to control space, to identify predefined plays from the positions of players around them, and create formations on the pitch to enable these plays. Players do not even need to touch the ball to be able to make a great contribution to their team.

Consider the well-known set piece described in figure 3.2. Players A and B are attackers from the same team. Player C is an opponent defender, who threatens to tackle player A for the ball. If player A feigns a pass to player B, player C must move to intercept that pass. In doing so, player C moves out of position, and player A can slip past. We say that player B has drawn player C out of position. Human players find it relatively easy to spot these spatial structures, which enable players to cooperate in useful ways. In contrast, these spatial configurations are difficult to

spot, and so, for the most part, overlooked in robot football. Many such structures occur frequently during human football matches, and they can be considered as building blocks of a strategy. In this work we shall endeavour to identify some of these structures, which can then be used to tackle the substantial complexity of robot football in a similar way to those used in computer chess and computer Go.



(a) Player C threatens player A, who feigns a pass.

(b) Player C moves to intercept the pass, allowing player A to slip past.

Figure 3.2 The Two-on-One Set Piece in Football

3.2 Competing for Space

We have established that structuring and controlling space is a key concern in chess. Similarly, we believe that structuring and controlling space is an underlying theme in football. Ownership of the ball, and opportunities to make plays, depends on the ability of players to control space through which that ball passes, or will pass.

Consider a pitch divided up into N distinct areas as shown in figure 3.3, where N is the total number of players. Each player controls an associated area defined by perimeters formed at points equidistant from the player and its closest neighbours. If all players can move at the same rate, then each region defines a set of points which the occupying player can reach before any opponent. We will call each area a

player's space. The combined area controlled by a set of players on one team shall be referred to as *team space.*

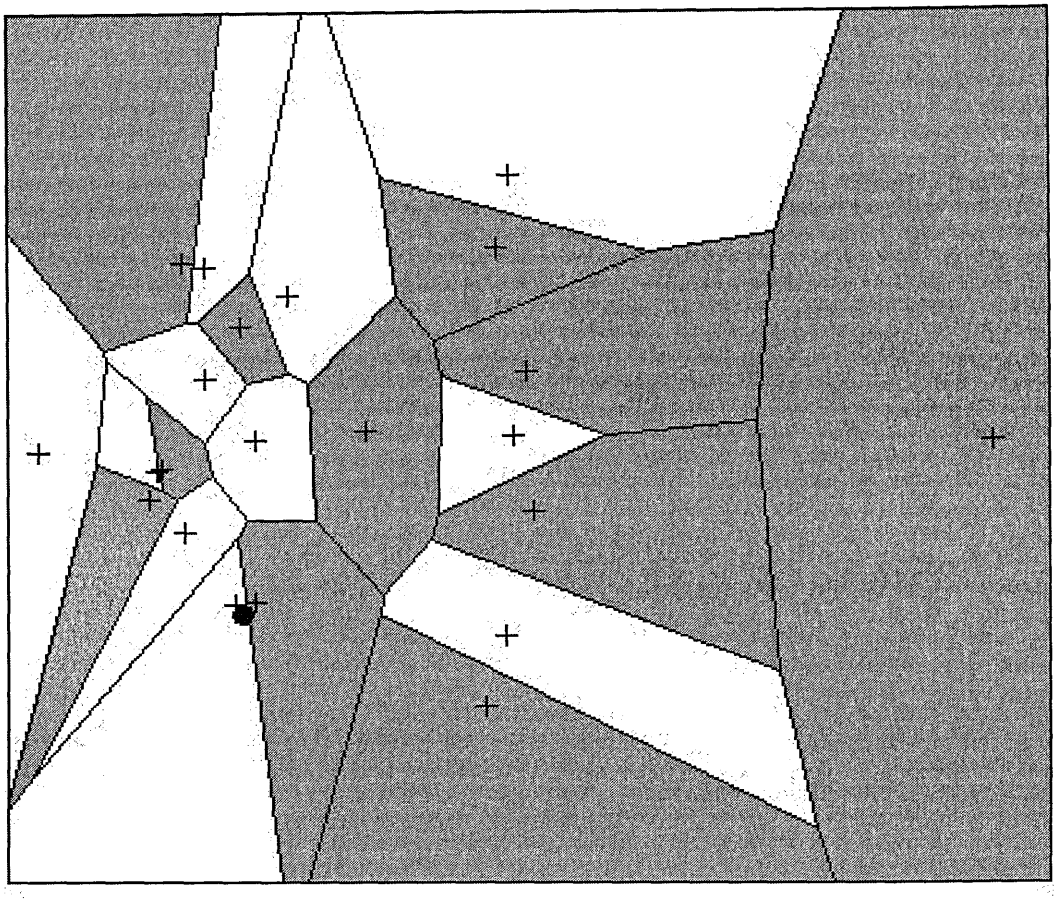


Figure 3.3 Spatial Ownership of a Football Pitch

Players control areas closer to themselves than any other player.

Now consider a static ball placed in this environment. If the ball lies within the perimeter of one of these areas, then the player occupying that area can reach the ball before any other player. If the ball lies on a perimeter or vertex, then two or more players will be able to reach it in the same time. To take control of a randomly placed ball it is advantageous to control a large area. Specifically, the player occupying the greatest area will have the highest probability of being able to reach a randomly placed ball first. By extension, the team with the largest combined area, or team space, will be the most likely to take control of the ball.

Contiguous areas of team space identify a 'safe passage' through which a sequence of ball passes can travel, so that it is always closer to home players than it is to opponents. By generating large and contiguous blocks of team space, players improve their prospects for generating sequences of passes. These ideas build on previous work to create a strategy as a string of tactics, such as ball passes (Johnson & Sugisaka, 2000).

This is a simplified view of spatial competition. Players will not always have equal movement characteristics, and will not be able to move in all directions with equivalent speed. The ball will also be moving, and will not do so in a random fashion. These factors will all affect the ability of players to control and interact with the ball, and a more complex definition of a player's controllable space will be required. However, initially a simple understanding of the concept is sufficient.

Our preliminary experiments use a cellular automaton. Such systems have been previously used in related work on path planning in robot football (Behring, Bracho, Castro, & Moreno, 2000; Bracho, Castro, & Moreno, 2001) and multiagent coordination (Barfoot & D'Eleuterio, 2001; Thangavelauthma, Barfoot, & D'Eleuterio, 2003).

3.2.1 Teamwork and the Space-Time Possession Game

In our initial experiment we desired to distance ourselves from the details of robot football, specifically tactics regarding moving the ball around in order to score goals, and focus on structuring space in useful ways. This led us to define the *Space-Time Possession Game* described below.

Let *G* be a grid of cells. For simplicity we'll assume G is composed of squares, although other planar tessellations are possible. *A* and *B* are two sets of *players* positioned on the grid such that each player occupies a single cell at any given time. The system has a discrete clock, and at each time frame any player can move to an adjacent unoccupied cell.

A player's claimed area is a function of distance. Each player possesses all the squares which are closer to it than any other player, such that the whole grid is owned by one or both teams (figure 3.4). Squares equidistant from either team are considered shared, with distances measured using chessboard distances.

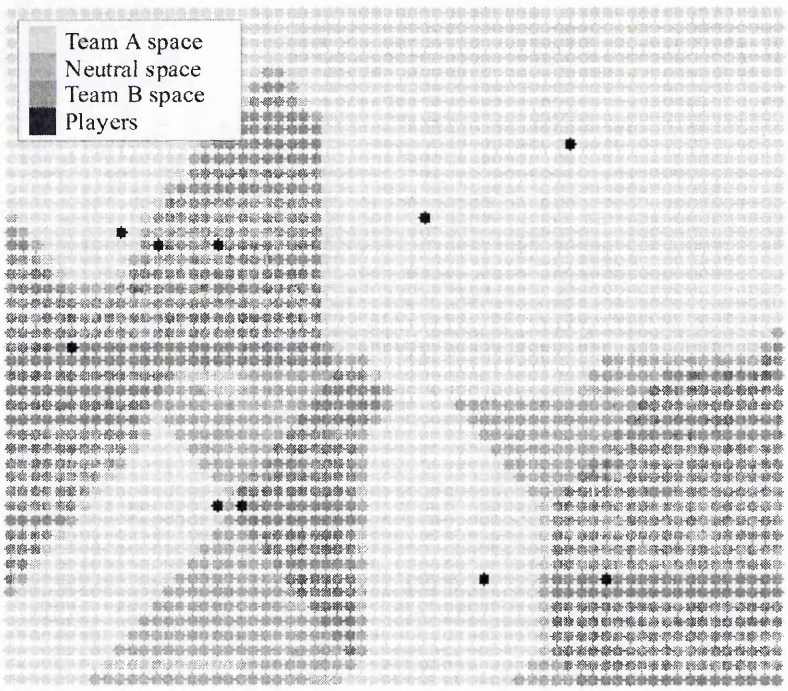


Figure 3.4 Team Space in the Space-Time Possession Game

Figure 3.5 shows the convention for pitch ownership. The two squares marked 'A' are players from one team and control areas of the grid marked 'a'. 'B' is an opposition member, and its controlled area is notated 'b'. Furthermore, cells marked 'c' are equidistant from both teams, and so jointly claimed by both sides.

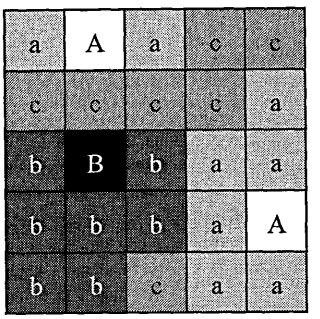


Figure 3.5 Cell Ownership in the Space-Time Possession Game

Team A players, marked 'A', control cells marked 'a'. The team B player, marked 'B', controls cells marked 'b'. Cells marked 'c' are in contention.

The objective of the game is to control strategic areas of pitch by outmanoeuvring the opposition. Initially, each team starts in an opposing half of the pitch, with players able to take up any position in their team's half. At every time step, the players are free to move a distance of one square in any direction, unless it is already occupied by another player. Player movements are controlled by a team strategy, in the same way as in robot football, with two strategy programmes playing against one another. There are a number of possible winning conditions, listed below, though only one will be applicable to each game:

i. The team that holds the largest distributed area after N clock ticks.

ii. The team that holds the largest contiguous area after N clock ticks.

iii. The first team to hold M distributed grid squares.

iv. The first team to hold M contiguous grid squares.

v. The first team to hold M distributed grid squares for N clock ticks.

vi. The first team to hold M contiguous grid squares for N clock ticks.

vii. The first team to link either end of the pitch with one contiguous set of claimed grid squares.

We will begin by focusing on developing control strategies for the first winning condition.

3.2.2 Experimental Results

We have programmed the Space-Time Possession Game in MATLAB, and implemented ten simple strategies. In previous work, teams took turns to move players, which gave an advantage to the team that moved last, resulting in limit cycles of possession (Law & Johnson, 2004). In the following results, players move in a randomly determined order. Each team consists of five players, controlled by a strategy, which runs run once at every time frame. The strategies are:

1. *Stationary* – The players remain stationary in their starting positions.
2. *Random* – The moving player is assigned a random direction to move in. This can be into any of its 8-neighbours, or it can remain stationary.
3. *Avoid* – Players move away from neighbours and pitch edges.
4. *Mark* – Each player moves toward an opponent player.
5. *Advanced Mark* – As *Mark*, but the players move to the position adjacent to their opponent which results in the largest possession score.
6. *Reinforce* – Players move toward the closest contended cell.
7. *Grow* – Players move to capture the maximum number of contended cells on the perimeter of their own area.
8. *Greedy* – The player's pitch possession is calculated for its current position, and for moves to every available 8-neighbour. The move which

results in the player owning the largest area is taken. If there will be no change, the player remains stationary.

9. *Team Grow* – As for *Grow*, but cells shared between players on the same team are ignored.

10. *Cooperative* – For each of the nine moves a player can make, the effect on the global team area is calculated. The player moves to the cell which gives maximum benefit to the team. If there will be no change, the player remains stationary.

We have experimented with playing each strategy against one another, using each combination of strategies to play ten games. There are forty five game combinations with each team adopting one of the ten strategies. Each match began with the players in the same symmetrical opening positions. Table 3.3 shows the number of games won in each set by the home team. Reading across the rows gives the scores for the home strategy against each opponent strategy.

Table 3.3 Scores for Strategies in the Space-Time Possession Game

Home strategy	*Stationary*	*Random*	*Avoid*	*Mark*	*Advanced Mark*	*Reinforce*	*Grow*	*Greedy*	*Team Grow*	*Cooperative*	Total
	Away strategy										
Stationary		2	0	0	0	0	0	0	0	0	2
Random	8		1	5	4	5	1	3	0	1	28
Avoid	10	9		4	6	10	1	3	0	0	43
Mark	10	5	6		7	10	5	5	4	1	53
Advanced Mark	10	6	4	3		10	1	3	3	2	42
Reinforce	10	5	0	0	0		2	1	2	1	21
Grow	10	9	9	5	9	8		1	3	2	56
Greedy	10	7	7	5	7	9	9		0	0	54
Team Grow	10	10	10	6	7	8	7	10		4	72
Cooperative	10	9	10	10	8	9	8	10	6		80

This table shows that the *Cooperative* team strategy wins most matches, with a success ratio of 80 out of 90. The *Stationary* strategy comes out bottom, winning only 2 out of its 90 games. However, the distribution of winning scores does not accurately represent the ability of each strategy. Table 3.4 gives the average area held by each home strategy.

Table 3.4 Average Areas Held by Strategies in the Space-Time Possession Game

Home strategy	Away strategy: *Stationary*	*Random*	*Avoid*	*Mark*	*Advanced Mark*	*Reinforce*	*Grow*	*Greedy*	*Team Grow*	*Cooperative*	Average
Stationary		48.20	45.22	22.68	22.39	25.98	42.74	47.74	28.10	28.10	34.57
Random	50.27		41.73	37.09	35.92	31.92	42.06	43.18	25.34	19.73	36.36
Avoid	53.41	55.83		38.69	39.99	67.18	45.41	47.60	17.70	17.08	42.54
Mark	70.02	42.19	39.34		48.68	66.88	37.29	45.56	40.10	29.31	46.60
Advanced Mark	71.39	42.23	38.09	40.18		59.55	34.52	43.51	39.10	36.25	44.98
Reinforce	65.21	55.78	25.52	13.27	22.85		37.87	39.12	35.25	31.73	36.29
Grow	54.88	56.12	52.44	42.81	45.43	47.33		44.45	37.46	35.21	46.24
Greedy	51.56	55.25	51.05	50.16	52.05	49.88	53.52		30.66	30.76	47.21
Team Grow	70.05	70.68	70.43	56.81	57.48	48.63	53.60	66.82		49.84	60.48
Cooperative	70.05	74.85	69.59	67.00	60.38	52.86	54.43	66.96	48.18		62.70

This second table shows that the difference in ability of the top two teams is much smaller than table 3.3 indicates. Although there is a difference of eight wins between the *Cooperative* and *Team Grow* strategies, on average there is only a 2.2% difference in the areas each controls. Similarly, there is a much smaller difference in controlled area between the worst teams. This shows the fickle nature of the game, and the difficulty in generating a winning team; two teams can be very evenly matched in their ability to control space, but will play to much wider degrees of success.

Although some of the strategies were developed with the aim of capturing space, others were based on simple structures; their ability to play the game emerged

from their interactions with other players. The following paragraphs highlight some of the more interesting behaviours noticed during the games.

Herding behaviours emerged in some of the matches. This was noticed between the strategy combinations *Cooperative* and *Random*, *Grow* and *Avoid*, *Mark* and *Reinforce*, and *Advanced Mark* and *Reinforce*. In each of these pairings, the former herds the latter into one or more small pockets at either the edge, or the centre of the pitch. This behaviour has different causes in each case. In the first instance, *Cooperative* players move adjacent to *Random* players and block one of their possible moves. The *Random* players then have a greater likelihood of moving away from the *Cooperative* players, and are eventually forced to the edge of the pitch. In the second instance, *Avoid* players are repulsed by their opponents, team mates, and pitch edges. As *Grow* players move to enlarge their areas, *Avoid* players are repelled toward the centre of the pitch. In the final two pairings, the movement of the opponent team toward the *Reinforce* players removes contended cells between the two teams on every other turn. Consequently, the *Reinforce* team makes more moves away from their opponents, in the direction of the pitch edges.

A second interesting emergent property was the result of players competing for space within their own team. Both the *Greedy* and *Grow* strategies foster this kind of competition, which has dramatically different effectiveness against different opposing teams. Against teams such as *Avoid*, where players remain at a distance from their opponents, the home team competes against itself for area; team mates pair up against each other and try to capture each others space. This leaves their opponents free to occupy the majority of pitch unopposed. Against teams such as *Cooperative*, *Mark*, or *Advanced Mark*, where players actively seek out their

opponents, they fare much better. In these situations, the opponent players seek out the competing pairs on the home team and split them apart; each *Greedy* or *Grow* player is now competing against one opponent, rather than a team mate. In this situation, Greedy and Grow teams are at their most successful.

Our final point relates to the stability of captured space. There was a tendency for the areas to fluctuate rapidly in games where opponent players occupied adjacent cells. In this configuration a single move can result in a shift to or from ownership of a number of contended cells, occasionally being significant enough to affect which team wins. We call this space *weakly controlled.* The player that moves last is often able to make significant gains, and, as the movement order is randomised, this could be either team. The spaces occupied in these games are, therefore, very unstable and liable to change to a great degree. Conversely, opposing players which are spaced apart hold much more stable areas. We call these *strongly controlled.* In these games contended cells are much fewer in number, meaning dramatic shifts in control are unlikely within the space of one turn.

The top two strategies both use *strong marking* approaches, whereby players move closely adjacent to opponents. As we will show later, this is one of the best methods for capturing large areas from the opponent. However, it is also risky, in that the areas are not stable. From the previous results, we can see that although they control much larger areas on average, they can be beaten by weaker strategies which are lucky in their final turn. This was evident in one match between the *Cooperative* and *Random* strategies, in which the latter won by moving last.

In this section we have introduced the idea behind controlling space, and separated ourselves from the confines of football. We have shown several general

strategies (both emergent and deliberative) for playing a space-capturing game, and in doing so, have shown that cooperation within teams is, as would be expected, the best method for playing competitive games. We have introduced the idea of strongly and weakly controlled space as those which are stable and unstable respectively, and have shown that the winning strategies appear to be those that take risks by weakly controlling space. We will now look for specific configurations of players, rather than strategies, which exhibit strong and weak control of areas.

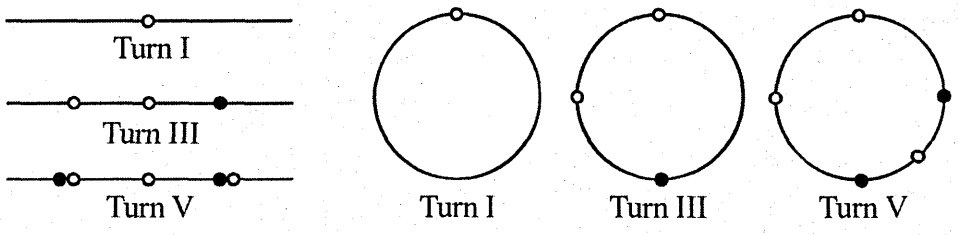
3.3 Voronoi Games

In the previous experiments we used a cellular automaton to segment areas on the pitch. In (Law, 2005) we showed that the processing time for the analysis of our cellular automata increased dramatically with pitch size. Specifically, that to use the same techniques on an image provided by our robot football camera would be impractical given the strict speed requirements of football. Our research to find an improved method for calculating areas in our possession game introduced us to the *Voronoi diagram* (Voronoi, 1907), and a similar set of problems called the *Voronoi games*.

A standard 2-dimensional Voronoi diagram is composed of n tessellating *convex hulls*, or *Voronoi cells*, which are defined as follows: Consider a set of points in a plane, $P = \{ p_1, p_2, ..., p_n \}$. For any point p_i there exists a locus of points (x, y) in the plane that are closer to p_i than any other point in the set P. These loci form the Voronoi cells, which we have referred to previously as a player's space. In ordinary Voronoi diagrams, boundaries between external points in P stretch off to

infinity, and have infinite area. However, it is straightforward to implement a boundary to produce an enclosed Voronoi diagram as was shown in figure 3.3.

The 1-dimensional Voronoi game was introduced by Ahn et al.(2001) as a variation of the hotelling process (Okabe, Boots, Sugihara, & Chiu, 2000). In this work, where the game is used as a model for competitively placing facilities along a road, players take turns to place n facilities on a line or circle (figure 3.6). This game is composed of n rounds, each player placing one site in each turn. At the end of the game, the arena is subdivided into sections according to the nearest neighbour rule, and the player with the largest area wins. Ahn et al. provide a set of rules for placing sites, which enables the second player to force a win in every game.



(a) Line game (b) Circle game

Figure 3.6 1D Voronoi Line and Circle Games

A modification to the line game is also introduced, whereby players are given one turn each to place their n sites on the circumference of a circle. A tactic is devised which enables the first player to win every game.

The one-round Voronoi game is extended by Cheong et al. (2002) to two or more dimensions. In this game, which is similar to our Space-Time Possession Game, a piece controls the area of pitch P closer to it than any other piece. Player one, white, places a set of pieces W, followed by player two, black, placing a set of pieces B. When all pieces are placed, the Voronoi diagram of $A \cup B$ is constructed, and the player which owns the largest area of P is declared the winner. Cheong et al.

show that given certain criteria, the second player can always steal at least half of the pitch. This proof is extended by Fekete and Meijer (2003) to show that for a rectangular pitch of aspect ratio ρ, black has a winning strategy for $n \geq 3$ and $\rho > \sqrt{2/n}$, and for $n = 2$ and $\rho > \sqrt{3/2}$. White can win in all remaining cases. It should be noted that these strategies all require white to place its pieces on a rectangular grid (figure 3.7).

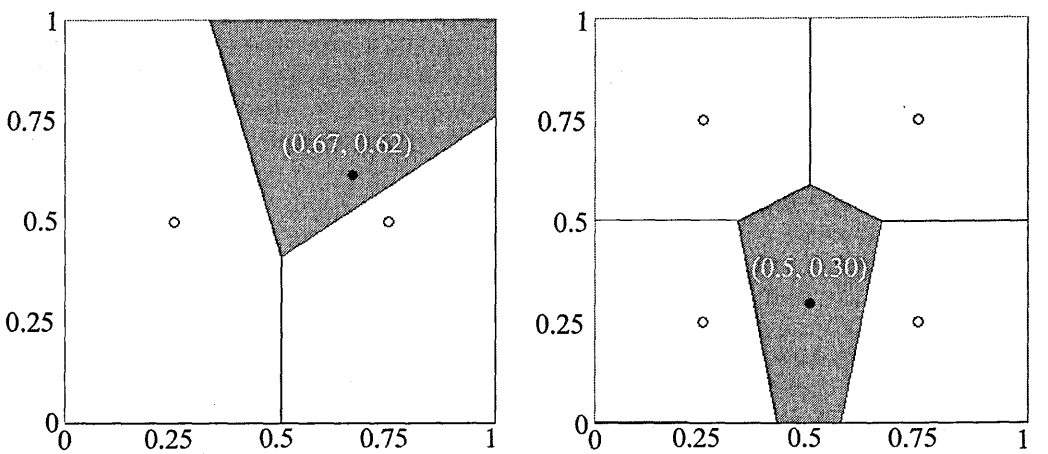


Figure 3.7 Grid Positioning Strategies in the Voronoi Game, Showing the Best Opponent Move

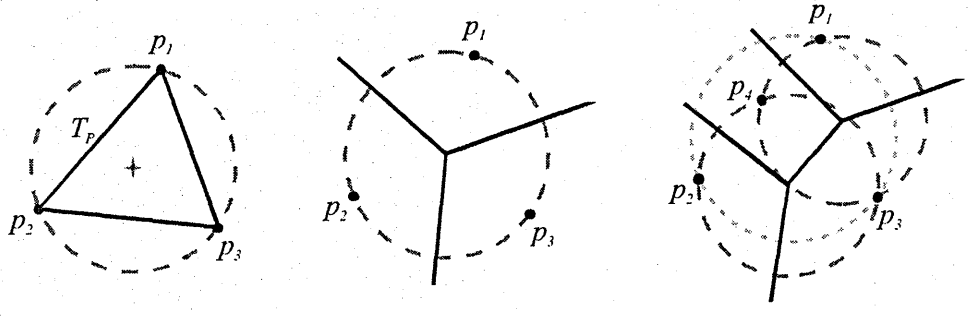
It follows that our space-time game can be considered as an extension to the 2-dimensional Voronoi game, with the introduction of multiple turns. Our multi-round Voronoi game has some additional constraints: players place pieces simultaneously, with only knowledge of their opponents' previous positions, and pieces have a limited movement between turns; they must remain within moving distance of their last position. Although there are some solutions to the games described above, none are applicable to our space-time possession game. We will instead undertake an experimental approach to attempt to find some simple structures which give appropriate solutions.

Based on our previous work, and ideas taken from the Voronoi games, we propose some useful spatial formations, and examine their application to the one-round Voronoi game. Movement of pieces across multiple rounds will be examined in chapter 5.

3.3.1 Spatial Structures

At this point we introduce some useful spatial structures, and introduce Voronoi diagrams in more detail, specifically how they are constructed and how they can be altered by the addition of new points, or players. We will not conduct a full mathematical examination, but briefly outline concepts and relationships which will be useful in understanding the forthcoming work. A more formal introduction can be found in (Okabe et al., 2000).

Consider three points in a plane, $P = \{ p_1, p_2, p_3 \}$, as shown in figure 3.8(a). They are related by a *Delaunay triangle*, T_P, as shown. A Delaunay triangle exists wherever three points describe an empty circle, which contains no other points in the plane. The centre of such a *circumcircle* defines a *Voronoi vertex*. As such, any points lying on an empty circle will create neighbouring Voronoi cells. The Voronoi diagram for the three points is shown in figure 3.8(b).



(a) Delaunay diagram of three points

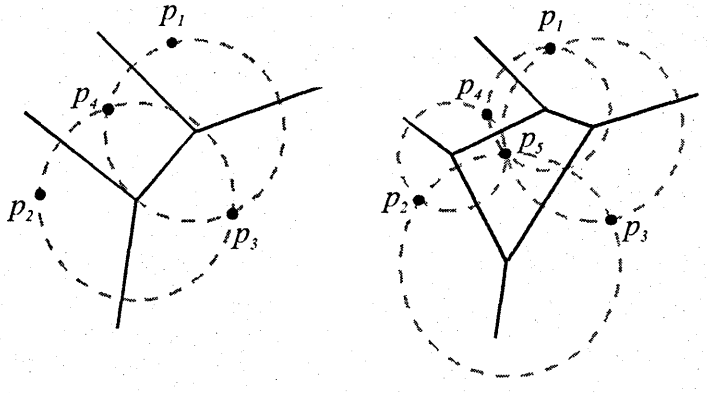
(b) Voronoi diagram of three points

(c) Construction of the four point Voronoi diagram

Figure 3.8 Constructing the Voronoi Diagram

If a new point is placed within the empty circle, it will split the existing Delaunay triangle in two. There will now be two circumcircles, and the Voronoi diagram will change accordingly, as shown in figure 3.8(c). By definition, the new point will neighbour each of the points on the original circle.

If two empty circles overlap (figure 3.9(a)), a point placed in the overlapping segment will neighbour all points on both of the original circles, as shown in figure 3.9(b). It should also be noted that placing a new point only alters the existing Voronoi cells of neighbouring points, not those elsewhere in the plane.



(a) Four points creating an overlapping segment

(b) Effect of placing a point in the overlapping segment

Figure 3.9 Localised Effect of Point Placement on the Voronoi Diagram

Consider a set of points $P = \{ p_1...p_{10}, q_1...q_5 \}$, and a subset $S = \{ p_1...p_8 \}$. If we wish to place a further point r which will create a Voronoi cell neighbouring only points in S, then we must ensure it is placed in an empty circle described by all points in S, but not $P \neq S$ (figure 3.10).

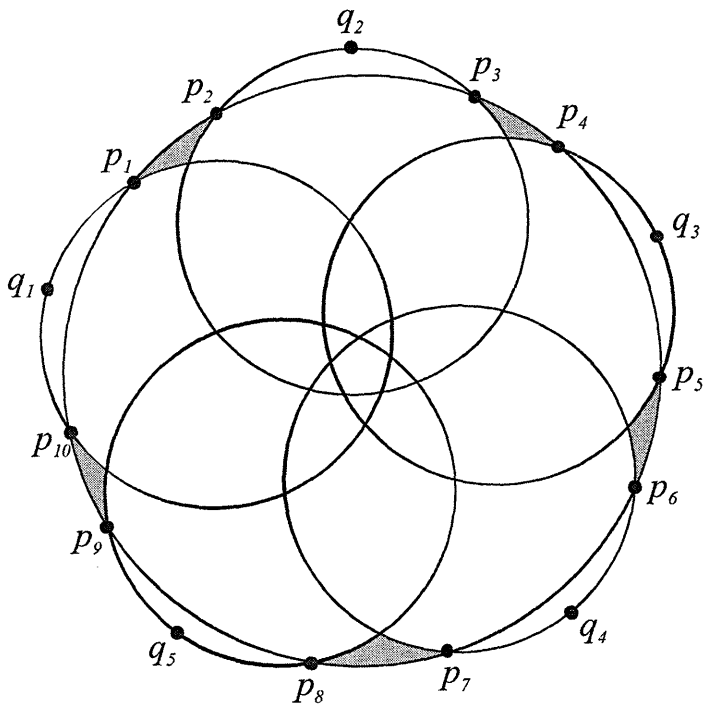
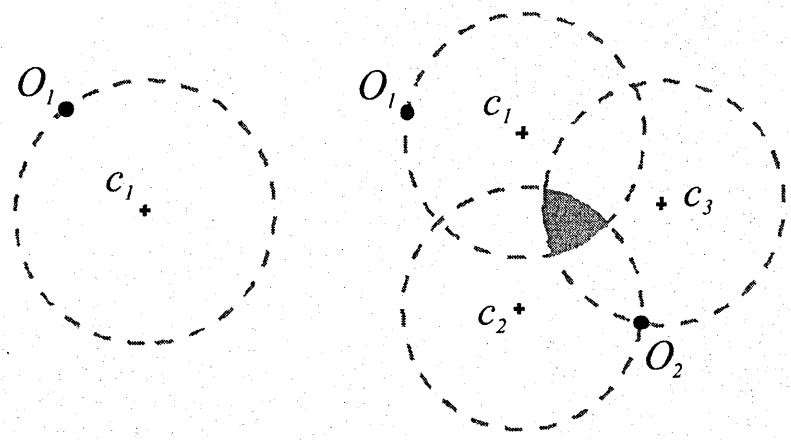


Figure 3.10 Creating a Voronoi Cell to Neighbour Specific Points

A point placed in a shaded segment will neighbour points in S

If we desire to place a new point such that its Voronoi cell will enclose a coordinate (x, y) in the plane, we must place it within the circle defined by the coordinate and the nearest existing point, or opponent player, as shown in figure 3.11(a). Similarly, we can form a Voronoi cell to enclose a number of distinct coordinates, provided the largest empty circles described by these coordinates intersect to form a common segment (figure 3.11(b)). Our point must then be placed within this segment.

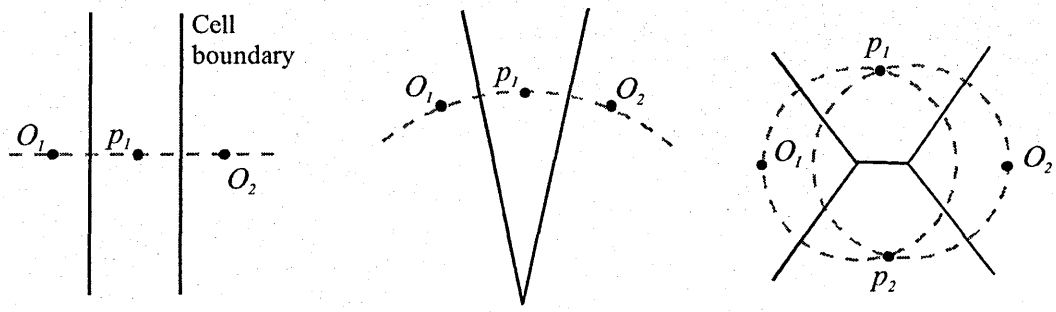


(a) To capture c_1, a point must be placed inside the circle

(b) To capture $c_{1\text{-}3}$, a point must be placed within the shaded segment

Figure 3.11 Capturing Coordinates in the 2D Voronoi Game

To separate existing neighbouring cells, a point can be inserted directly between those forming the existing cells. If the three points are collinear, they describe a circle of infinite radius, thus the outer cells will never meet (figure 3.12(a)). However, in the space-time possession game this configuration will be unstable. If the inserted point is offset slightly, as shown in figure 3.12(b), the two outer cells will converge. To prevent this, a second point should be added such that a second empty circle is created, with the two new points describing the overlapping segment (figure 3.12(c)).



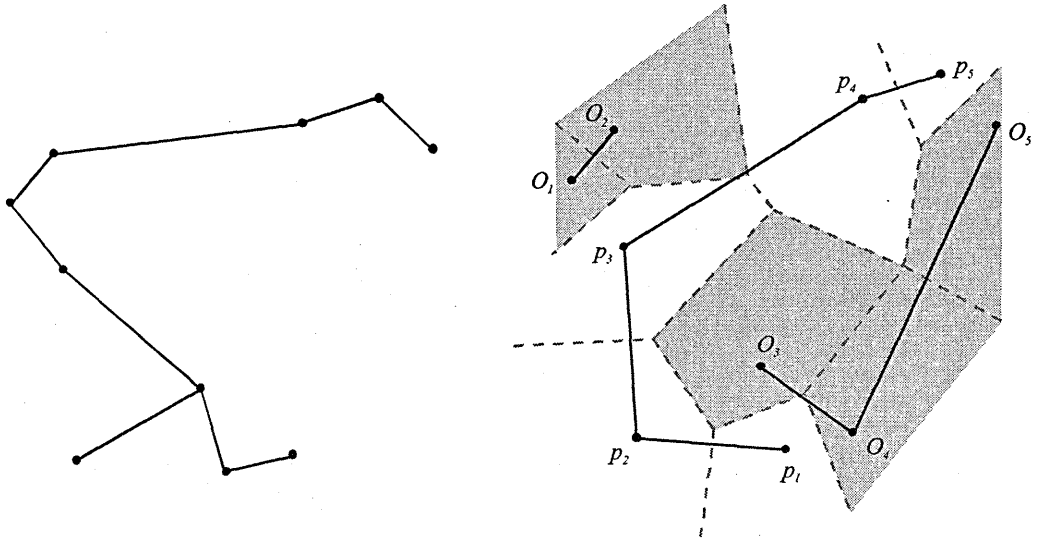
(a) Separation of two opponent cells by placement of a single collinear point, p_1

(b) Poor separation of two opponent cells by placement of a non-collinear point, p_1

(c) Separation of two opponent cells by placement of two points, p_1 and p_2

Figure 3.12 Separating Neighbouring Cells by Point Placement

The final structure which will be of use to us is the concept of *minimum spanning trees* (MST). The MST is the shortest path between points, and is composed of the edges of the Delaunay graph for these points (figure 3.13(a)). We introduce a variant we shall call a *team tree*, which will map how players of one team neighbour each other along edges of the Delaunay graph (figure 3.13(b)). This gives us useful information on the linkage of team areas, as well as how pockets of players are surrounded and separated from their team mates. Similar ideas have previously been suggested by Johnson and Price (2003).



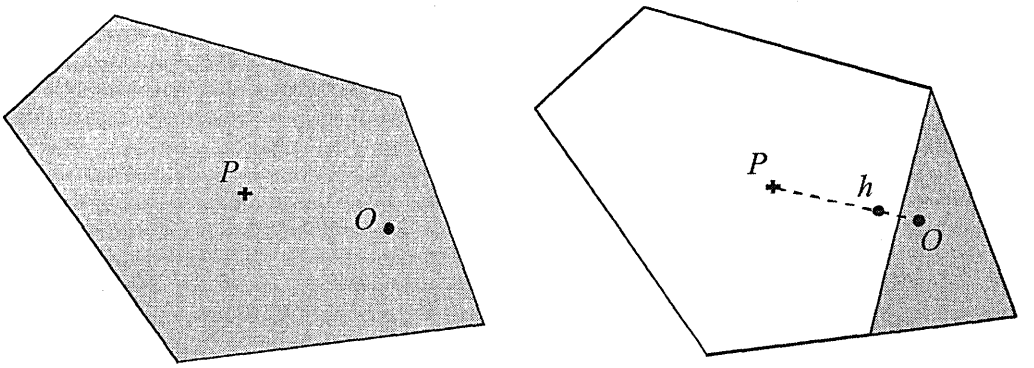
(a) The minimum spanning tree for a set of points

(b) Team trees for two sets of points

Figure 3.13 Tree Diagrams

3.3.2 Spatial Tactics

If all opponent pieces have been placed in a Voronoi Game, we can capture the largest area by stealing more than half of each opponent player's space. Figure 3.14(a) shows the Voronoi cell of an opponent piece, O, with area A and centroid P. In figure 3.14(b), h is the home piece we desire to position. The contrasting regions show how the existing cell will be divided by placement of h.



(a) Voronoi cell of opponent O with centre P and area A

(b) Placement of h to capture majority of A

Figure 3.14 Capturing Cells by Placing a Single Player

If P and O are coincident, then there is no position for which the portion of A closer to h is greater than that closer to O. However, if P and O are not coincident, then placing h on the line between P and O will cause h to capture to a larger proportion of A, as shown. This can be considered as a strong marking strategy, whereby players are placed closely next to individual opponent pieces. Provided opponents do not lie on the centre of their Voronoi polygons, then it is always possible to steal a slightly greater area of the pitch from the opponent team using this technique.

This is called a *takeover* and a variation is proposed by Cheong et al. (2002), whereby two home pieces are allocated to the $n/2$ opponent pieces holding the largest areas. By placing two pieces close to, and on opposite sides of, O, the home player captures almost the entire area of A (figure 3.15). Provided the areas of all the opponent pieces are not similar, a strategy based on this principle will capture at least half of the playing field.

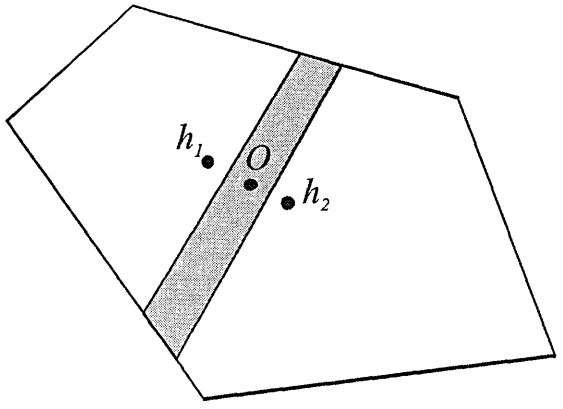


Figure 3.15 Cell Takeover by Two Players, h_1 and h_2

These strong marking strategies are suitable for marking n opponents with m pieces if $m \geq n$. However, if $m < n$ (say we have already allocated two home pieces to the largest opponent cell), the best strategy for the remaining m players may be to *weakly mark* multiple opponents. By this we mean placing a player between opponents, in such a way that it can capture area from a number of cells. In general, the more circumcircles enclosing a point, the more neighbouring cells that point will have. Also, the larger the radius of those circles, the further away the neighbours will be, and the larger the Voronoi cell associated with those neighbours. We propose that placing players in segments caused by many overlapping circumcircles may capture significant space.

3.3.3 Experimental Results

We generate ten strategies (described below) based on the structures described in the previous section. Each was played second in 100 one-round Voronoi games, using teams of 5 players on a 64 × 48 unit arena, against 5 randomly positioned opponents. The seeds for the random position generator were reproduced for each set of games, ensuring all strategies were played against the same set of random opponents. As a benchmark, games were also played using a random strategy, and a brute force best-

position search algorithm. A brief description of each strategy and its outcome is given below, with the statistical results shown in figure 3.16. Tabulated results for the mean and median scores are given in table 3.5.

1. *Random* - Pieces are placed at random. As would be expected, there is a normal distribution of area captured over 100 games, with a mean of 51.9%.

2. *Optimal* - A brute force search of all integer coordinates for positions which give the greatest returns. This strategy always wins in our tests, with a confident margin over the opponent. However, the lengthy computation makes it impractical for real-time applications or large pitch sizes. It is included as a benchmark for our other strategies.

3. *One-on-one* - A strong marking strategy with each piece paired with a single opponent. This is a very competitive strategy, giving results with a mean within 0.7% of our optimal benchmark strategy, yet using a much simpler algorithm. The spatial structures employed here are very different from those observed in the optimal strategy, but produce very similar outcomes. The drawback is the instability of captured space. These particular structures perform most competitively in situations where each opponent piece controls a similar sized area.

4. *Two-on-one* - A strong marking strategy, using two pieces to mark each of the strongest opponents. The remaining piece is allocated to the 3rd strongest opponent using a one-on-one strong marking style. Again, this is a strong strategy, consistently winning all 100 games, and with a mean falling within 4.2% of that of our optimal benchmark strategy. These

structures perform best against opponents where space is not evenly distributed between players.

5. *Radius* - A weak marking strategy. Pieces are placed in overlapping segments of Delaunay circumcircles with the largest cumulative radius. Effectively pieces are placed in large but highly neighboured spaces. This strategy does not perform as well as the strong marking strategies but, with a mean area of 60.3%, out performs the other weak marking strategies. A main benefit of this structure is its flexibility. The two strong marking strategies require pieces to be very close to the opponents at all times. To change between the one and two marker strategies requires single pieces to make relatively large movements, which will take time to perform. In comparison this, and the other weak marking configurations, place pieces are well distributed amongst the opponents, allowing an easy switch between strategies.

6. *Overlap* - Another weak marking strategy. Pieces are placed at the centre of the most overlapped circumcircle segments. Effectively these configurations place pieces to neighbour the maximum possible number of opponents. Although not as competitive as the strong marking strategies, this approach still wins in 81 of the games.

7. *Exclusive Radius* - This is similar to the standard radius strategy, with one key difference. As more pieces are placed, radii of circles used in previous placements are discarded from any new placement calculations to avoid overly populating one area of pitch. Despite this, it is less successful, winning 10 fewer games.

8. *Exclusive Overlap* - This is similar to the standard overlap strategy, with one key difference. As more pieces are placed, circles used in previous placements are discarded from any new placement calculations to avoid overly populating one area of pitch. Again this is less successful than its counterpart, winning 9 fewer games.

9. *Vertices* - Players are placed at the furthest points from all opponent pieces, i.e. on the most remote Voronoi vertices. This is a control experiment to demonstrate a poorly abstracted spatial structure. Intuition may suggest that by simply moving pieces far from their neighbours, they will occupy large empty spaces. Loosing 95 of the games indicates that this is not the case.

10. *Grid* - The first player strategy proposed by Fekete and Meijer (2003) for $n \geq 3$. It is indicated that positioning pieces on regular grids minimises the gains of an opponent. Here we implement the $1 \times n$ grid, and demonstrate its performance on a pitch of aspect ratio $\rho > \sqrt{2/n}$ (it is proposed as a winning strategy only if $\rho \leq \sqrt{2/n}$). The outcome is much worse than our strong marking strategies, with only 73 wins and an average possession of 53.9%, making it more comparable to our weak marking strategies. A drawback of this style of play when applied to an N-round game is its inability to adapt to the changing configurations on the pitch.

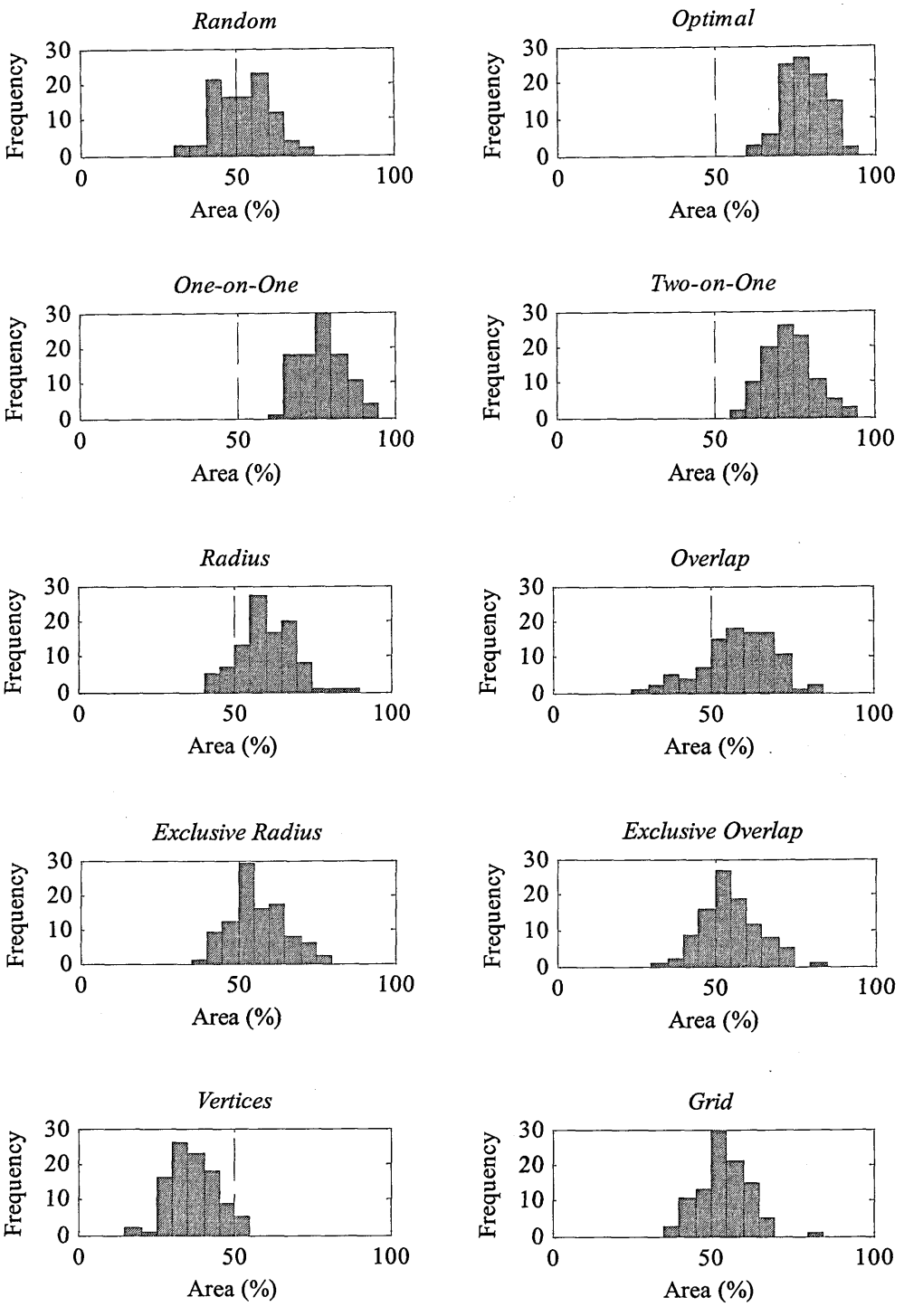


Figure 3.16 Strategy Results for the One-Round Voronoi Game

Bars show the number of games concluding with the named team occupying the given area.

Table 3.5 Mean and Median Scores for Positioning Strategies in the One-Round Voronoi Game

Strategy	Number of wins	Mean score (%)	Median score (%)
Random	57	51.9	51.1
Optimal	100	78.1	77.9
One-on-one	100	77.4	77.7
Two-on-one	100	73.9	73.6
Radius	88	60.3	59.5
Overlap	81	58.6	59.7
Exclusive Radius	78	56.3	54.9
Exclusive Overlap	72	55.2	54.2
Vertices	5	36.6	35.9
Grid	73	53.9	53.2

In accordance with our results from section 3.2.2 we have again shown that closely marking the opponent can be a successful strategy. Although close to the optimal solution in terms of score, the positions occupied by each are very different; the optimal solution places players further from the opposition, resulting in a more secure area ownership than that of the strong marking strategies. The brute force computation required to find such positions is not feasible in the real-time world of robot football, and we can see by the scores that there is only minor benefit in implementing more sophisticated algorithms over our simple marking strategies. Another interesting feature is the favourable results of the standard, over exclusive, weak marking strategies. These indicate that, in general, it is better to focus more resources on larger opponent areas than spreading them over a wider region.

Both the strong and weak approaches to marking have their advantages. Though the former excels in its ability to capture space, the latter has benefits in its

flexibility and stability. We expect that both types of structure will appear in football games.

3.4 Space in Football

Having investigated some spatial structures and strategies, we now return to the game of football to investigate the structure of team space during a match. Our aim is to find whether a relationship exists between the distribution of team space and the states of play during a football match. Similar work undertaken by Kim (2004) examined player space with relation to victory conditions in a simulation of real football. It was concluded that to win a team did not necessarily have to control the largest area on average during a match, but in order to score a goal a team did need to be in control of a larger area of pitch at that moment. Our experiments differ in that we are not only examining victory conditions, but searching for relationships which exist throughout a match.

We based our own tests on data from the RoboCup Simulation League. Using our possession game we examined ten different matches, representing a variety of winning conditions, and observed the changes in team space, player space, the goals scored, and the position of the ball during play.

3.4.1 Analysis of Team Space

We began by measuring the amount of pitch owned by either team in each of the ten matches, and compared their average ownership to the number of goals scored. The results are shown in table 3.6. In match 5, we observed the third largest goal difference of any match, and the largest average margin in pitch possession by the

winning team. However, matches 2 and 9, which have the largest goal differences, also have 2 of the smallest average possession margins. Examining relations between the goal difference and pitch margin for the remaining matches, it is difficult to suggest that controlling the majority of the pitch is sufficient to win a match. Neither is there a significant relationship between goals scored and the maximum team possession scores.

Table 3.6 Scores and Area Possessions Measured in RoboCup Simulation League Matches

Match	Score (A – B)	Average team possession as % of pitch		Average possession difference	Maximum team possession as % of pitch	
		A	B		A	B
1	0 – 0	44.89	55.11	10.22	74.42	78.50
2	10 – 0	51.73	48.27	3.46	71.54	78.59
3	1 – 2	48.39	51.61	3.22	75.41	75.86
4	0 – 0	54.48	45.52	8.96	74.95	79.13
5	0 – 6	42.14	57.86	15.72	76.01	79.07
6	4 – 3	54.23	45.77	8.46	80.36	74.68
7	3 – 0	51.28	48.72	2.56	76.76	74.81
8	4 – 3	44.68	55.32	10.65	65.82	78.70
9	1 – 8	50.68	49.32	1.35	76.87	77.64
10	2 – 0	44.29	55.71	11.43	86.29	84.24

We furthered this research by analysing the change in possession throughout game 5. By monitoring the changes in ball possession, and the variation between attacking and defending plays, we formed relationships between our definition of team space and the constantly changing state of the game. Figure 3.17 shows how often team A controlled specific quantities of pitch.

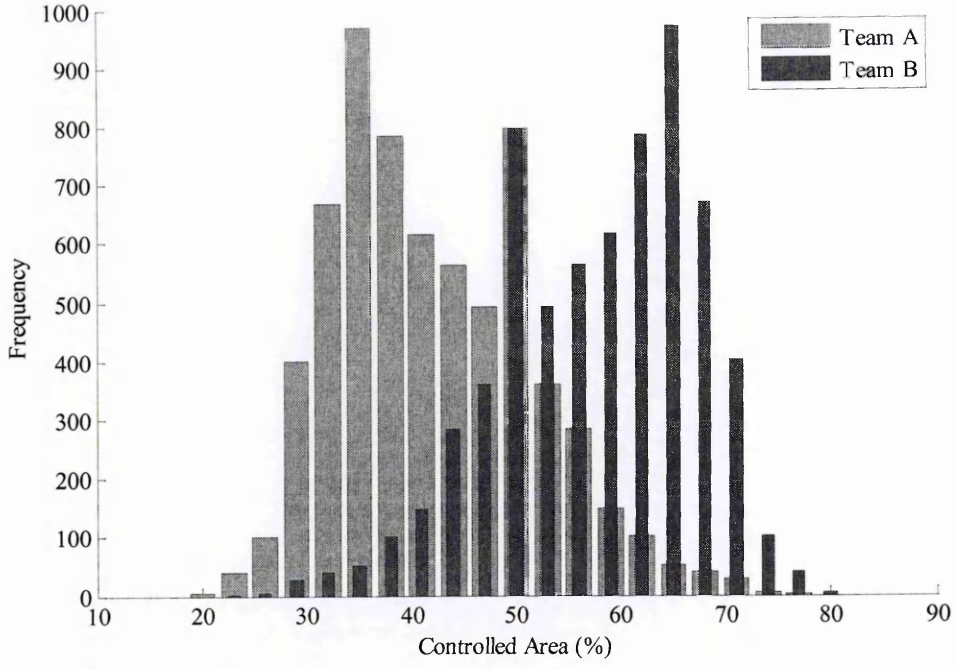


Figure 3.17 Possession Frequencies during a Robot Football Match

Each team controls the specified area of pitch for the indicated number of frames.

The total area of the pitch is 7140 units. Team A mainly controls only a fraction of this, around 35%. This low ownership is due to team B being in possession of the ball for 78% of the match, with team A playing defensively. It should be noted that the feature relating to a possession score of 50% is an effect of the time spent in the kick-off position after each goal is scored, and is not a proportional representation of either team's influence during standard play.

From the simulations, we observe that larger team spaces are usually linked to attacking plays, and smaller ones to defensive plays. In terms of spatial configurations, a large team space facilitates easier passing and movement to intercept stray balls, which is desirable in an attacking formation. In contrast, small team and player spaces indicate tight configurations of players, which are better for protecting a small area and intercepting passes and shots within that region.

However, as concluded earlier, it is not sufficient to state that by controlling more space a team is more likely to score goals. Neither is it appropriate to state that a team in control of a larger area will be on the attack. For either of these to hold any merit, the team in question must be in possession of the ball.

3.4.2 Movement on the Ball

We continue by examining the relationships between team space and ball position. Figure 3.18 shows ball position data and space distribution from a portion of match 5. The area plot is taken from the perspective of team A, with the x-axis for ball position defined as the line drawn the length of the pitch, passing through the centre of both goal mouths.

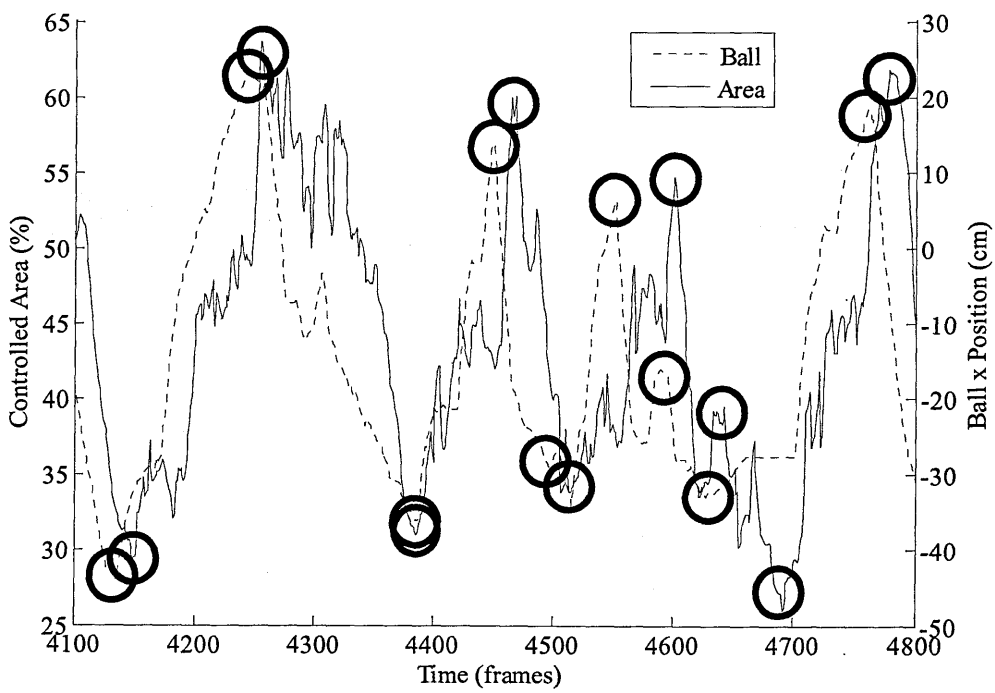


Figure 3.18 Comparison of Ball Position and Controlled Area in a Robot Football Match, Highlighting Similar Features

A relationship can be seen between the two plots, both having similar major features. These features appear in close phase to one another, with the lag varying between 20 and 50 frames. The area owned by team A roughly follows the same trend as the position of the ball, supporting our observations about spatial configurations in attacking and defending plays. As the ball moves along the length of the pitch, each team expands, or contracts, its area accordingly to control more pitch, or defend key areas. This relationship between ball position and team space is also evident throughout each of the other simulated matches.

We speculate that once a team has possession of the ball it adopts a broad spatial configuration, which facilitates passing and safe movement about the pitch. At the same time, the opposition forms a much tighter spatial configuration to protect specific areas of pitch, or block opponent players. As the ball is moved further toward the goal, the attacking players increase their control over the pitch, whilst their opponents form tighter, more defensive structures around their home goal. The phase lag between the signals in figure 3.18 is due to the reaction times of the robot football system.

An example of this spatial structuring is shown in figure 3.19. Here, team B (controlling the area in grey) has the ball (black dot) and is attempting to shoot at team A's goal, on the left hand edge of the image. Team A (controlling the area in white) has responded by forming a tight defensive structure around the goal. In this configuration they control the majority of pitch between the ball and goal, whilst also blocking shots from the most threatening opponent players. In contrast, the spatial configuration of team B, although controlling more area overall, has much less influence in this crucial region.

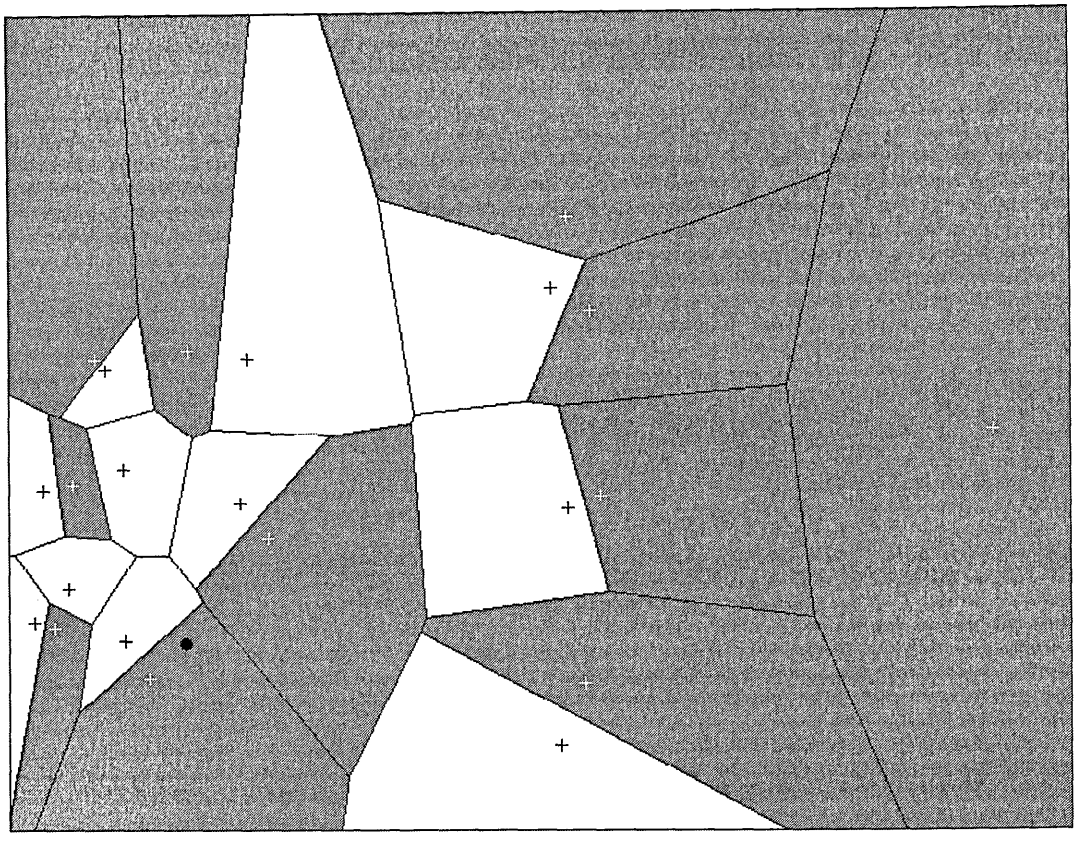


Figure 3.19 Defensive and Attacking Structures in Robot Football

From these results we can see how controlling space is important in robot football. Although we cannot give strict relationships regarding how to structure this space, we have shown some more general connections. Due to the duality of the Voronoi and Delaunay graphs for a set of points, we can also state that the structure between players is an important part of football.

3.5 Summary

Spatial structuring is an essential part of football. In human football games, players try to structure the pitch by taking up positions to improve their team's chances of success. Although players take up these positions without explicitly communicating their intentions to their team mates, extensive training allows the players to recognise tactical opportunities based on these positions alone (Johnson, 2000).

Furthermore, players can use their positions to weaken their opponent's by using feints, just as in the game of chess.

We have established a connection between the digital representation of robot football and some more traditional board games which are concerned with structuring space. We have briefly discussed the AI techniques used to solve these games, and have shown that the complexity of robot football is too great to rely on this standard approach.

Through our knowledge of football, we have identified the significance of controlling areas of pitch, and have created an abstracted generalisation of football in the form of an *N*-round Voronoi game we call the Space-Time Possession Game. Results from this work showed that a team in which agents cooperated outperformed a team composed of non-cooperating individuals. We also showed how some combinations of simple strategies evolved interesting emergent behaviours.

From our knowledge of human football, strategies for the one-round Voronoi game, and analysis of Delaunay and Voronoi structures, we have identified a set of spatial structures which correspond to ideas we consider to be useful in spatial competition. Using the one-round Voronoi game as an experiment, we show how our spatial configurations respond to a set of opponent positions. The results indicate that the structures we have identified are, at best, near-optimal, at worst, above average, and all more competitive that some arbitrarily chosen configurations. We hypothesise that while our strong marking strategies perform best in these games, a combination of the strong and weak marking configurations will be more appropriate for the *N*-round game and, by extension, robot football.

We continued in our analysis of spatial competition by using a bounded Voronoi diagram to analyse the change in team space during a simulated robot football game. The results showed a correlation between the motion of the ball and the overall area each team controlled on the pitch. From these findings, we conclude that robot football can be represented as a game of spatial competition. Furthermore, the duality of the Voronoi and Delaunay transforms provides evidence that positional structure between players is fundamental to the game of football.

In all these experiments, we assumed that every player was omnidirectional and could move with the same velocity and acceleration. In real systems this is not the case, and so a weighted Voronoi diagram is required. However, the principles under investigation relate to both types of diagram, and so we examine the more general case. Having identified the relevance of spatial possession and positional structure in robot football, we continue by searching for specific structures to form the basis of an abstracted team strategy.

Chapter 4

Task Abstraction using Concept Generation

In the previous chapter we highlighted the importance of area possession and hence spatial structure in robot football. This work also identified some structures and playing styles which facilitated competitive spatial ownership. In continuation, we will now attempt to formalise a set of explicit rules, based upon this new knowledge, which describe the game of football, and how it should be played by a team of coordinated robots.

In this chapter we shall focus on generating an abstracted definition of the robot football objective using the techniques outlined in section 2.4. Although we only demonstrate the methods on our robot football data, it should be recognised that the same methods could be implemented to abstract task descriptions for other complex systems.

4.1 An Architecture for Abstraction

Figure 4.1 depicts our analysis architecture. This describes the process of decomposing a complex task from sets of available data. We input data corresponding to the system we desire to abstract, a list of concepts, which we suppose are sub-structures within the system, and a list of variables, which we will use to describe these concepts. The analysis produces a list of hubs, which are measured structures of the variables, which appear frequently in the related concepts. From these hubs, we generate hypotheses and representatives, which can be used

respectively to classify further data, or recreate the concepts. The following sections describe the core parts of the architecture.

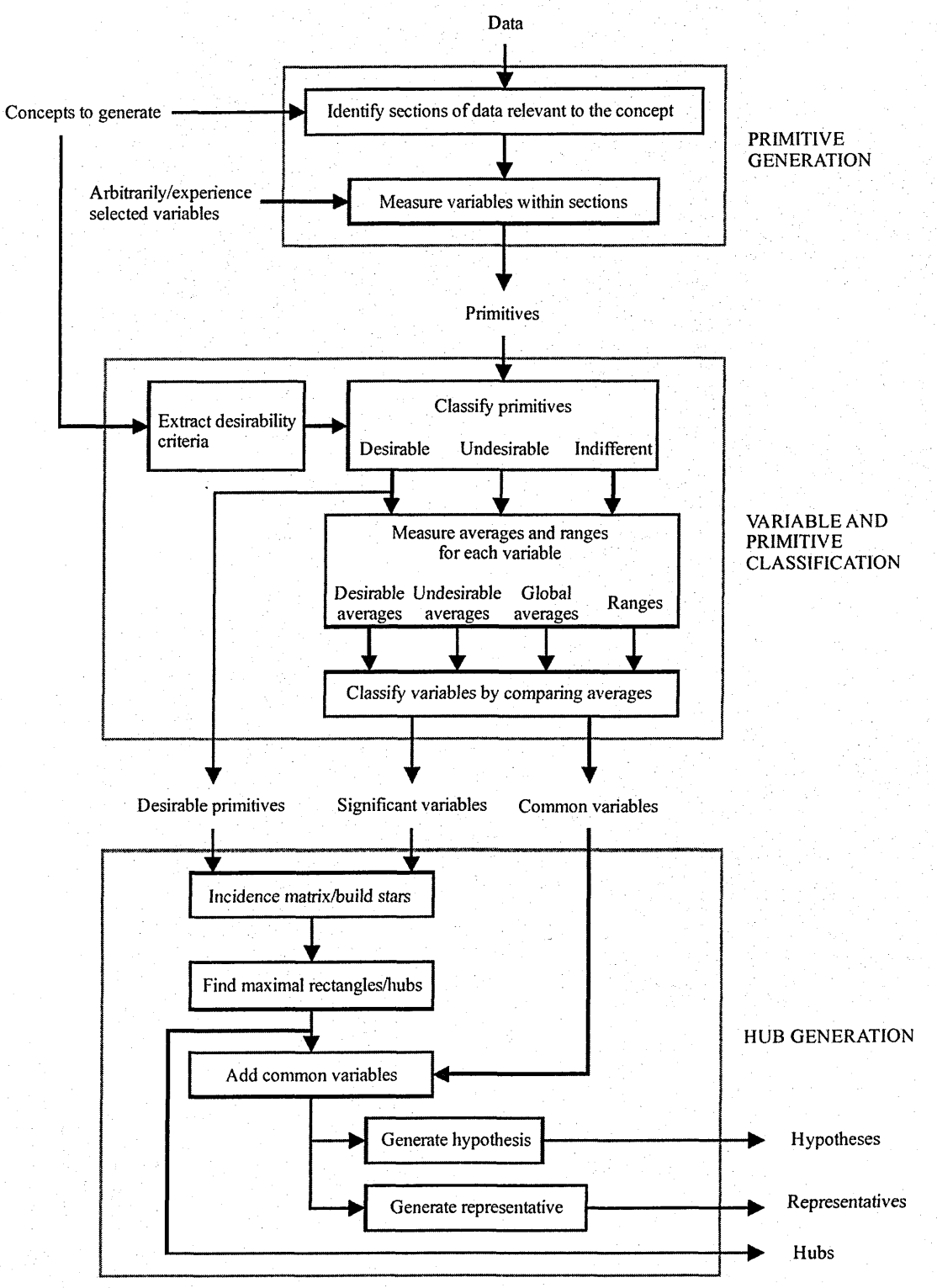


Figure 4.1 Block Diagram of the Proposed Abstraction Architecture

4.1.1 Primitive Generation

The section of the architecture relating to primitive generation is reproduced in figure 4.2. Recall the definition of a primitive in section 2.4.4, where we stated that it was a measured relation on a set of variables relating to a concept. This block takes in a set of data, a set of concepts to be generated, and a set of variables. The output is a set of primitives for each concept. Each primitive contains the value of each variable in the data over a period relating to the concept.

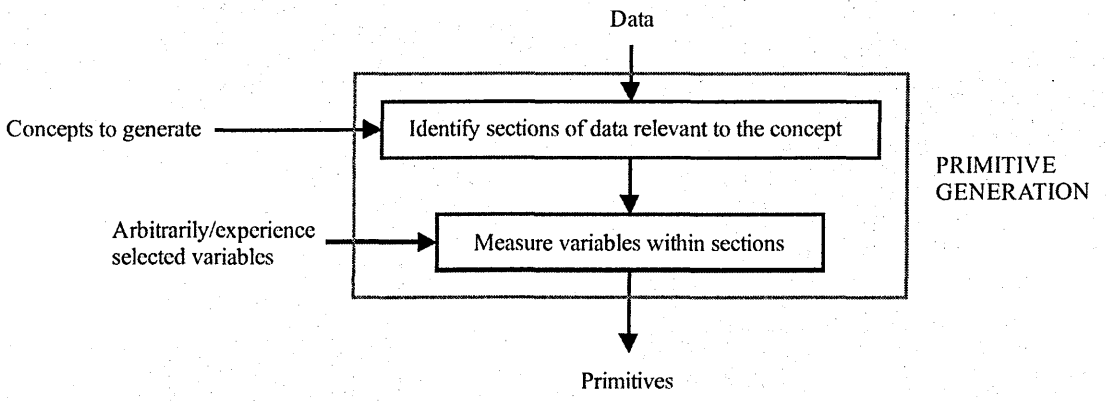


Figure 4.2 Block Diagram of the Primitive Generation Component

We begin with a set of recorded data from systems of, or similar to, the type we are interested in. In the case of our robot football problem, we use log files recorded from simulated matches. These files contain data on the positions and orientations of the robots and ball at every frame taken during the match. We consider these to contain sufficient data to describe strategies for playing football. Multiple data sets are required to improve the accuracy of the concept generation. This may require whole sets of data, though in the case of low level concepts, which are evident multiple times in one data set, a single set of data may be sufficient.

The data does not necessarily need to be from a system identical to the one of interest, provided it displays the same properties which we desire to conceptualise.

In this way, we can build a representation of a complex system by examining properties in from a variety of different systems. For example, we could examine a movement task from one set of data, and pick up and put down tasks from another. We could then combine these together to build a representation of an object transportation system. In this work we will examine data from simulated robot football games, but use it to control real robot footballers.

The primitive generator also requires a set of concepts to generate, and a set of variables with which to measure them. These are both currently generated by hand, although it would be preferential if these could be automated in later instances of the architecture. The concepts to be generated are the tasks we wish to abstract from the data. These can be at any level, and relate to any measurable feature in the data. For example, we could choose the overall objective of the system as our concept to generate, or we could choose some minor low level task, such as moving between two points. In practice, we chose concepts relating to all the tasks within a system, to generate the most accurate task decomposition of the objective. It does not matter if these tasks are irrelevant to the objective, as this will be discovered when performing the analysis at a higher level. We will give some examples of how to choose concepts for generation in section 4.2, when we select concepts relating to robot football.

The variables to use in the concept generation can be any measurable property of the system. These could be simple descriptors, such as distance, or the achievement of some related subtask. Recall that lower level concepts may be used as variables for higher level concepts. The more variables, the more accurate the concept generation becomes. Even variables which we may not consider to be

important may have some unseen effect on the concept, and this will be highlighted in the measured hubs.

The primitive generation begins by identifying sections of the data relevant to each concept. For example, if we were interested in generating a concept relating to the task of moving an object from point A to point B, a primitive would be described by a subset of the data where an object is in transit between those two points. Each transit would be recorded in a separate primitive, starting from the time the object left point A, and ending at the time the object arrived at point B.

The variables are then measured for the duration of each primitive. A powerful ability of the concept generation method is the ability to examine structures through time. For example, in our transport example, the transit itself may be made up of the event sequence pick up, move, put down. The alternative sequence of the same events pick up, put down, move, would not accomplish the task. To simplify matters, we will generally focus on structures which exist through the entirety of a concept, and not their dynamics.

The output of the primitive generation block is a set of primitives for each concept. Each primitive is made up of the values of the measured variables over a portion of the data relevant to the concept. Attached to each primitive is a relation, stating how the variables are associated. In our example, the variables in each primitive are related through the undertaking of the transport task.

4.1.2 Variable and Primitive Classification

In this block, desirable sets of primitives and variables are identified. The block diagram is reproduced in figure 4.3. It takes in the measured primitives and the list

of concepts to be generated, and outputs sets of primitives and variables which correspond to the desirable instances of the concept. It also outputs those variables which are common to both desirable and undesirable versions of the concept.

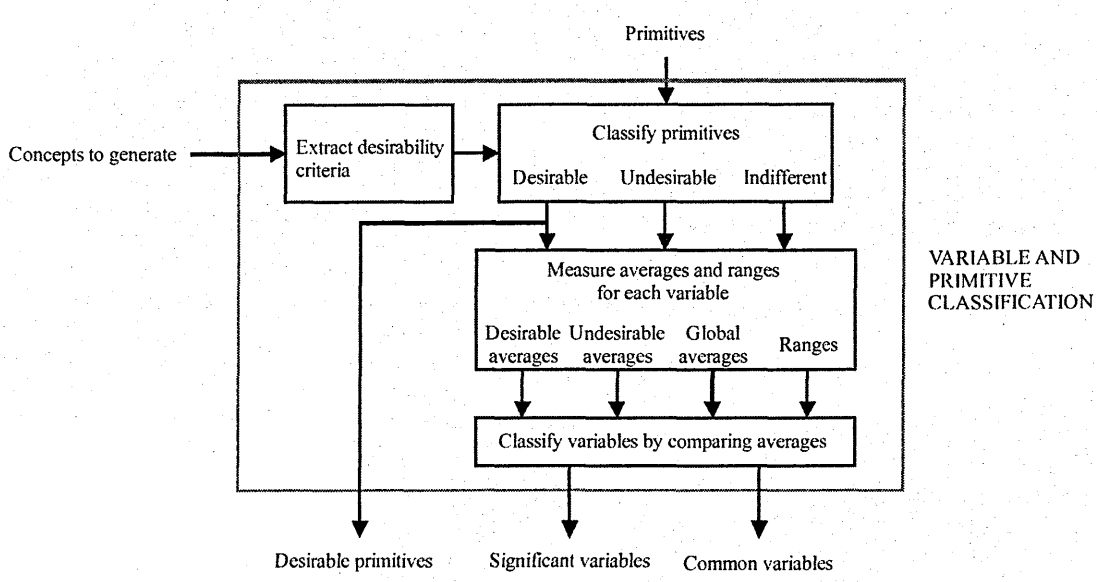


Figure 4.3 Block Diagram of the Classification Component

The process begins by extracting the criteria for classifying the primitives; some will be more desirable than others. For example, in our transport task, it may be that we are only interested in the fastest transfers. In this case the transfer duration would be the measure of desirability. This is used to classify the primitives into desirable and undesirable sets. For some concepts we may also be able to classify a set of indifferent primitives, which are neither desirable nor undesirable. The relations attached to each primitive are modified to reflect this classification. In our example, the relation for the desirable primitive associates its variables through the undertaking of a desirable transport task. Hence, all the desirable primitives will carry the same measured relation, based on the contained variables being part of a fast transport task.

In the next stage we classify the variables themselves. This begins by taking three averages for each variable: average over desirable instances, average over undesirable instances, and the global average over all instances. The actual classification compares these three values for each variable. If there is a significant difference between the average value for the desirable instances and the average for the undesirable instances, and if these averages are on opposite sides of the global average, then the variable is classed as significant. This means it is likely to be a good classifier for differentiating between desirable and undesirable primitives. If the separation of the desirable and undesirable averages is negligible, in terms of the range of values, then the variable is classed as common. This means a variable is common to both sets of primitives, and may contain useful information to describe the general system. The actual significance of this method of classification is calculated in section 4.3.4.

For an example of this classification, consider again the transportation task. If we classify the task in terms of the duration of transit, then we split our primitives into two sets: the desirable set for fast transits taking less than t seconds, and the undesirable set for slow transits taking more than t seconds. Now consider one of our variables relates to passing through a point C at some distant location away from both A and B. The variable is true if the object in transit passes through C, and false if it does not. In all of our fast primitives this variable is false, as it takes longer than t seconds to get from A to C and back to B. In some (but not necessarily all) of the slow primitives, the variable is true. Assigning values to truth and falsehood allows us to take averages for the variables across the fast and slow transits. There will be a distinct difference between the averages over each set of primitives. Whether this

difference is deemed significant will depend on the implementation of the classification. We discuss this further in section 4.3.

The outputs from the classification block are the set of desirable primitives, the set of significant variables, and the set of common variables. The significant variables output also contains the global average for each variable, and an indication of whether the desired value is greater than, or less than, this value.

4.1.3 Hub Generation

In this block, reproduced in figure 4.4, we measure the hubs relating to the concepts. These are structures of variables which are common occurrences in our desirable primitives. We also generate hypotheses and representatives based on this information.

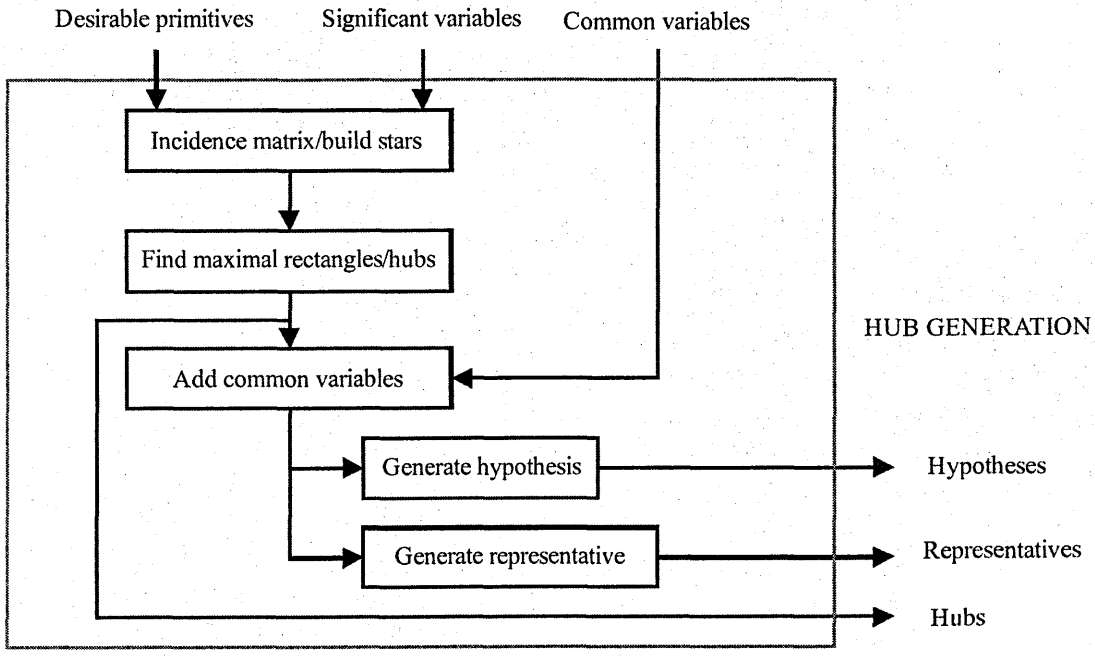


Figure 4.4 Block Diagram of the Hub Generation Component

The hub generation begins by compiling the desirable primitives into simplices of significant variables, from which stars can be formed, and the hubs extracted. In

practice, we use an incidence matrix with each row representing a primitive, and each column representing a significant variable. Any insignificant variables are stripped from the primitives. Each cell is labelled true if the variable in the primitive has a value occurring on the same side of the global mean as the desired value.

The next stage is to find the hubs of the stars. This can be done by finding the intersections of all the primitives with every other primitive, or by finding all the unique maximal rectangles. By this we mean rearranging the rows and columns of the matrix to find the largest rectangles which are not contained within any other.

Since all of the desirable primitives carried the same measured relation, this relation is attached to the incidence matrix as a whole. Every hub also carries this relation, which indicates how we have measured the variables to be associated. For example, a hub measured for our fast transport task will contain variables which are related through all being present during a transport task, of duration less than t seconds. Furthermore, the relation will also now state that these variables occur to a greater or lesser degree than the given global mean.

To generate hypotheses and representatives we can either select a single hub, or generate some kind of average of a number of hubs. In this example, the hypothesis could simply be that a transport task is desirable if the duration is less than t. However, the power of the technique is in being able to classify primitives which are less obviously related. In this case we would use a sample of primitives classified into desirable and undesirable sets by hand. A hypothesis distinguishing the sets could then be used to classify further primitives. In this work we select the largest and most frequently occurring hubs to be our representatives. The representatives and hypothesis may also contain the common variables.

Recall that a hypothesis is a generalisation of a set of primitives into an associated concept, and a representative is a relation on a set of variables which can be used to represent a concept. A hypothesis is used to classify future data. For instance, in our transport example, we could use the hypothesis to identify fast transits in other data without having to identify the start and end conditions, or measure the time. This is particularly powerful when the concept is something that is difficult to automatically identify in a set of data. In this situation, the primitives in the initial analysis must be identified by hand. The hypothesis can then be used to automatically identify the concept in further data. An example of this is given in (Iravani, 2005a). The representative, on the other hand, is used to create an instance of the concept. For example, in our transport task, rather than generate the subtasks (pickup, carry, drop, etc.) by hand to build the controller, we would endeavour to create a representative complete enough to identify these tasks for us. The power of the representative is in creating a list of required subtasks to fulfil some complex mission.

4.2 Multilevel Structure in Robot Football

Robot football is, as we have already stated, a complex multilevel and multidimensional system. In the last chapter we showed that the areas controlled by players, and hence the structures between them, are significant aspects of the game. We also showed a number of specific player configurations which had significant meanings. Based on this knowledge, we will now attempt to identify a set of concepts to input into our abstraction architecture.

Recall figure 2.16. Here we showed how a typical robot football controller has a multilevel structure, consisting of low level roles and mid level plays, combined to form a high level strategy. Figure 4.5 shows our own alternative decomposition of the robot football task based on these ideas.

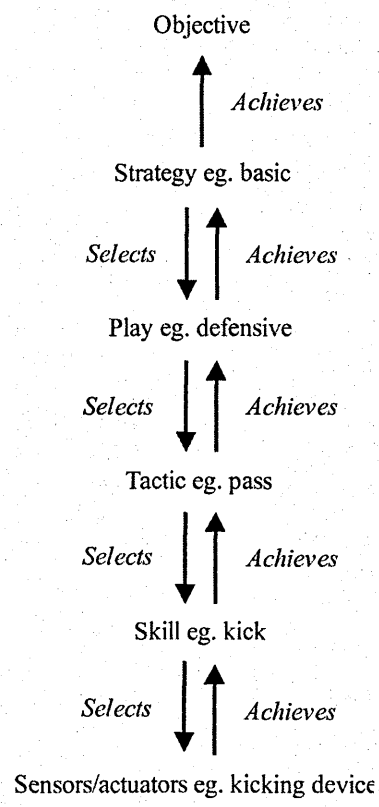


Figure 4.5 Robot Football Control Decomposition into Strategies, Plays, Tactics and Skills

To achieve the objective of scoring goals we implement a strategy. This strategy is a sequence of plays, with some kind of trigger to select plays according to the state of the match. Each play aims to achieve some objective, which in turn promotes the state of the strategy toward scoring a goal. Furthermore, each play is composed of a string of events, consisting of interactions between players or between a player and the ball. The actions causing these events we term *tactics*. At an even lower level, the abilities by which players perform these tactics we shall call

skills, which are themselves sets of sensor-actuator mappings. An example of this decomposition is given in figure 4.6.

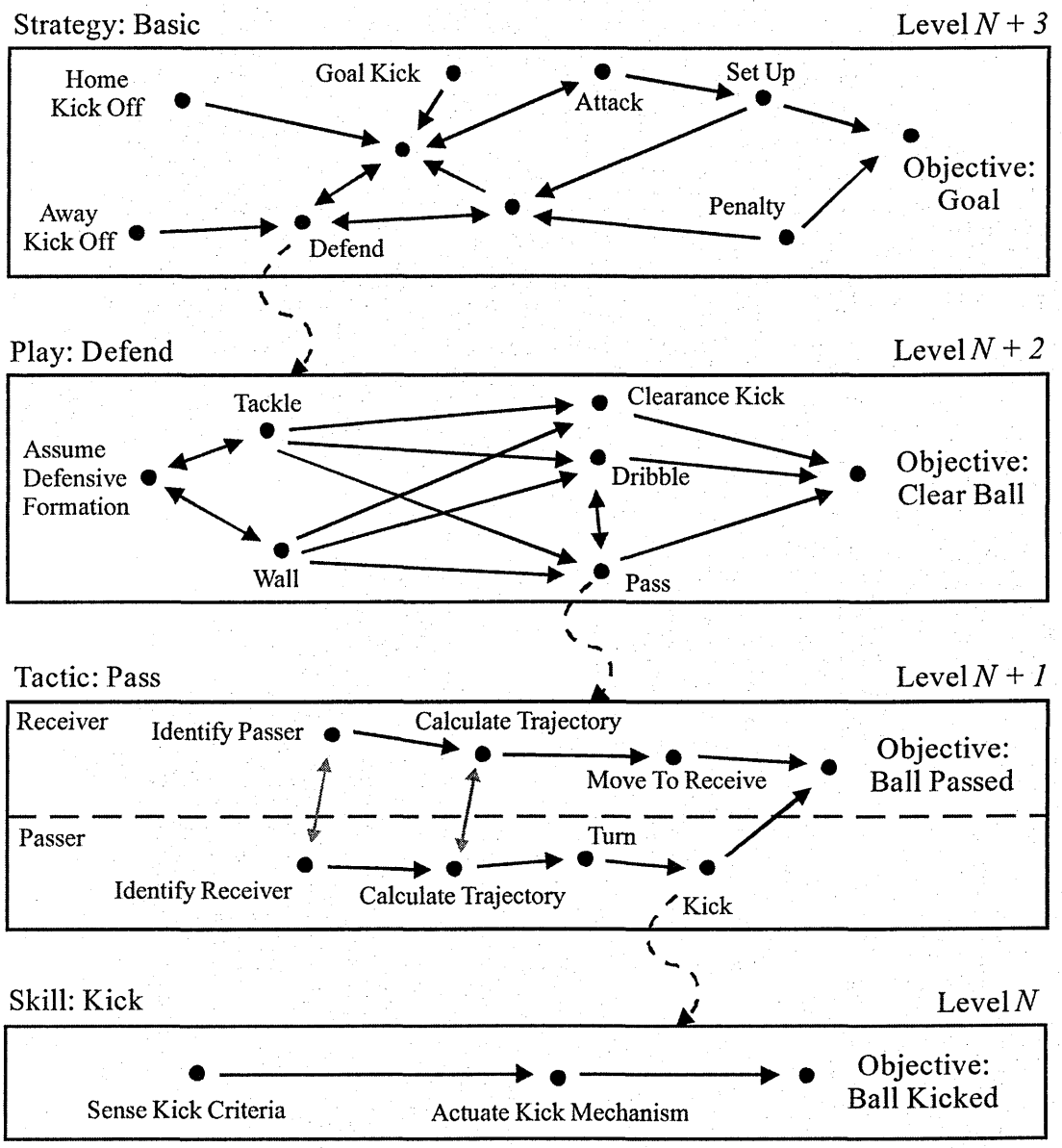


Figure 4.6 An Example of Robot Football Decomposition into Strategies, Plays, Tactics and Skills

Although we use the ideas of strategies and plays in a similar way to traditional architectures, we remove the restricting roles, and replace them with more flexible sets of tactics and skills. These allow us to focus on extracting emergent structures from the lowest level sensor and actuator combinations, all the way up to the high

level strategy. Roles may still exist in some form, but in our architecture they will be emergent structures formed by skills, tactics, and other variables. A diagram showing these ideas, using the multilevel abstraction notation, is shown in figure 4.7.

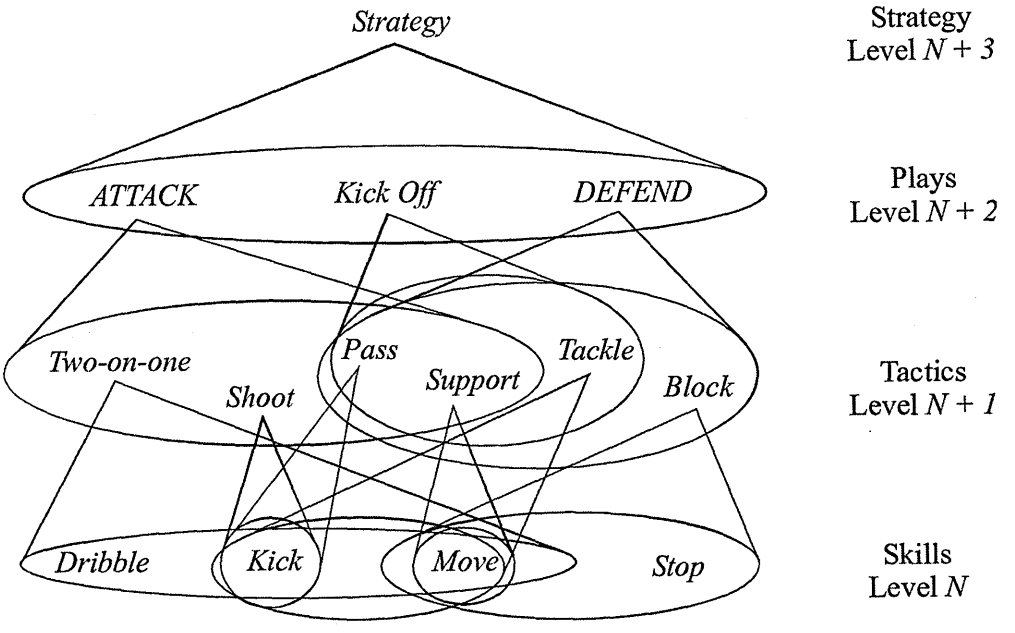


Figure 4.7 A Multilevel Football Strategy Structure Consisting of Plays, Tactics and Skills

A robot football team using these ideas may have three different strategies at its disposal, each of which has its own benefits. One might be stronger against a fast moving inaccurate opponent, one may be better suited to counter a slow but deliberate opponent, and one may work well against an opponent who uses brute force. Each strategy will be made up of a number of plays. There might be attacking plays, defending plays, midfield plays, kick-off plays, or formational plays. These are then subdivided into tactics: a pass, pass sequence, shot on goal, tackle, or some positional gambit, such as the 2-on-1. The skills we referred to are a player's ability to kick or dribble the ball, or simply to move into a new position. Obviously we could deconstruct this further to look at the physical characteristics of the robot, or even the signals sent to actuators.

If we compare these ideas back to the game of chess, we can say that plays are the duals of the set pieces we referred to in section 3.1, and each represents a sequence of tactical moves. For example, a particular defensive sequence might include castling to protect the king. This would be seen as a tactical move, requiring the movement skills of both the king and the rook. Robot footballers are less constrained than pieces on a chess board, so there are much greater possibilities for combining skills, tactics and plays.

The named structures (*Dribble*, *Kick*, *Pass*, *Shoot*, *Kick Off*, *Attack*, *Strategy*, etc.) in figure 4.7 are all concepts; they each represent some idea which we desire to abstract from the recorded data. Although these concepts, like the problematic task decompositions it robot architectures, are generated by hand from experience, they do not suffer the permanent effects of poor choice. If some concepts are poorly chosen, they will simply not appear as part of the concept on the level above when the analysis is performed.

We are interested in robot, rather than human, football. Therefore, the structures we will use in our analysis will be tailored so that they can be easily represented in mathematical terms. For example, it is common for a traditional robot football strategy to comprise of two plays: *ATTACK* and *DEFEND*. The switch to activate one or other play is the position of the ball. If the ball is in the opponent's half, the strategy switches to the attacking play. If the ball is in the home half, then the defending play is activated. The position of the ball relative to the half way line is straightforward to calculate.

Other possible plays and switches could be activated by certain areas of pitch, or whether a home or opponent robot is closest to the ball. There are also specialist

plays for kick-offs, goal kicks, free balls and penalties. In this work, we will focus on a simple set of plays: *IN HOME*, *IN AWAY*, *IN POSSESSION*, and *OUT OF POSSESSION*. The switches to activate these plays are respectively: ball in home half, ball in away half, home player in possession of the ball, ball not possessed by a home player. We will assume a player has possession of the ball if it is within a kickable distance.

It is probable, given these play descriptions, that two plays may come into conflict, with both equally suitable to the state of the game. From our list, both of the two possession plays will conflict with both of the attacking and defending plays. For example, if the ball is in the opponent half, and is controlled by a home player, then either *IN AWAY*, or *IN POSSESSION* could be selected. We avoid this conflict by generating additional plays to cover these joint possibilities. In this case, the new play has characteristics of both the existing plays, as shown in table 4.1. We see this as just another level in our multilevel structure, as shown in figure 4.8.

Table 4.1 Play Concept Descriptions

Concept	Description
IN HOME	Play active when the ball is in the home half of the pitch
IN AWAY	Play active when the ball is in the opponents half
IN POSSESSION	Play active when a home team player is in control of the ball
OUT OF POSSESSION	Play active when the opposition controls the ball
IHIP	Ball in home half and in possession
IAIP	Ball in away half and in possession
IHOP	Ball in home half and out of possession
IAOP	Ball in away half and out of possession

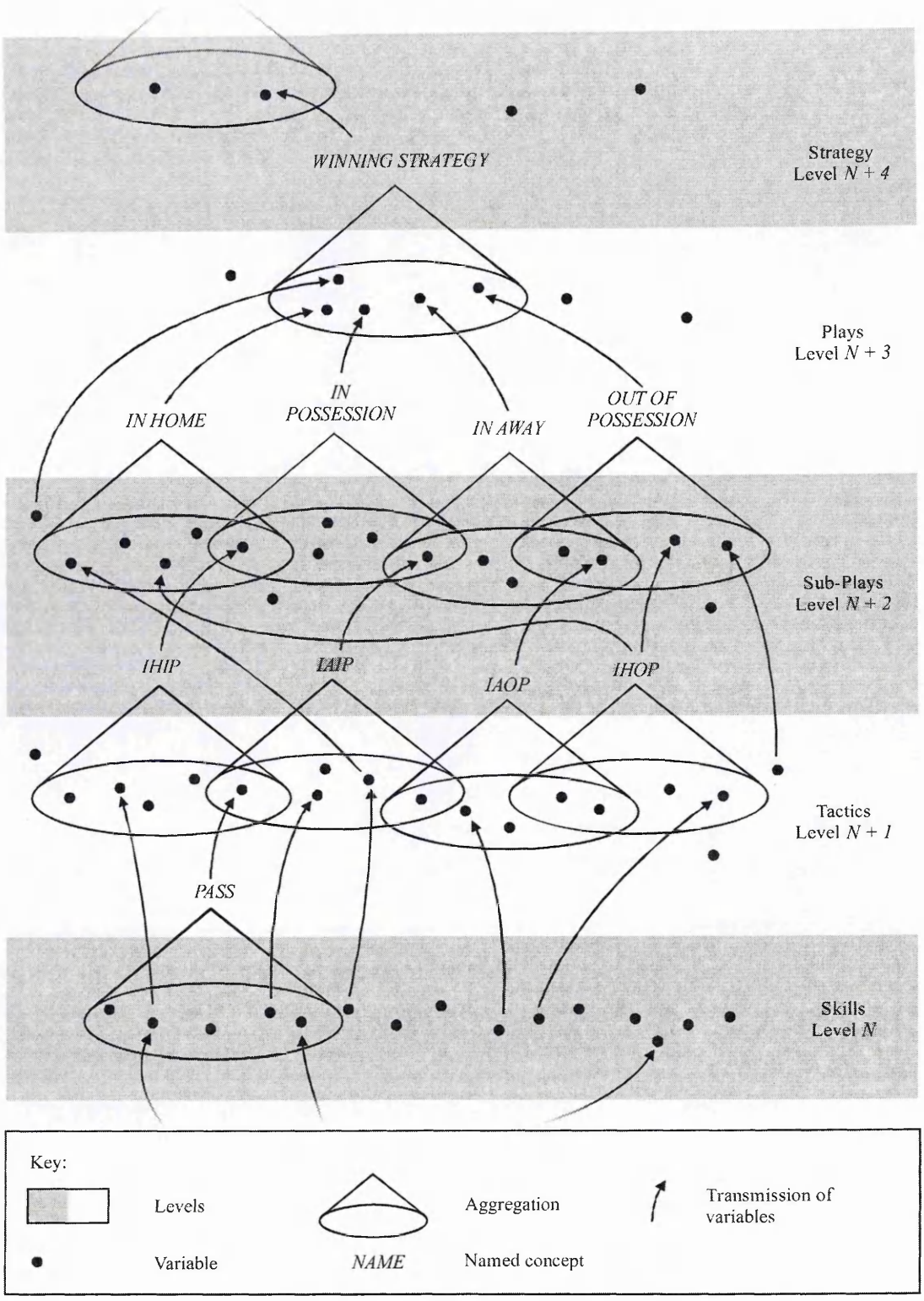


Figure 4.8 A Multilevel Strategy Structure Showing the Aggregation of Concepts

Each named structure in figure 4.8 is a concept which we will attempt to abstract in the following work. Skills become variables aggregated into tactical concepts; tactics become variables aggregated into sub-play concepts; sub-plays become variables aggregated into play concepts; and plays become variables aggregated into strategy concepts. It should be noted that the emphasis in this work is not on the concepts themselves. Robot football undoubtedly has a much more complex structure than the one described above, and far surpasses the few simple concepts we will examine. What is important is that there is a structure, and that it can be explained and analysed using the techniques described in section 2.4. The concepts we will generate are simply used as examples to prove the theory.

4.3 Architecture Implementation

In this section we shall introduce the particulars regarding applying our new architecture to the problem of abstracting the emergent properties of robot football strategies.

4.3.1 Primitive Generation

Our input data consists of log files from ten RoboCup simulation league matches. With each file describing the strategies of two teams, this gives us twenty team strategies to analyse.

In section 4.2 we identified possible concepts to analyse. These were named *WINNING STRATEGY*, *IN HOME*, *IN AWAY*, *IN POSSESSION*, *OUT OF POSSESSION*, *IHIP*, *IHOP*, *IAIP*, *IAOP*, and *PASS*. Later in this chapter we will take each in turn to show how our method is applicable on multiple levels, and to see

how concepts on different levels relate to one another, in reflection of the relationships shown in figure 4.8.

In this work we are focussing on generating representatives of robot football strategies, in an attempt to reconstruct a strategic controller. Because of this, the concepts used here have been chosen to enable easy definition and measurement of primitives. Consider the concept of the play *IN AWAY*. This play is described by any set of variables including the property *ball in away half*. The hypothesis could simply be that the primitive describes the *IN AWAY* play if it contains the property *ball in away half*. However, for the purposes of describing the play, we also want information on the positions and actions of the players. By analysing all the primitives containing *ball in away half*, we may also find other common configurations or activities, which we can also use to describe the concept *IN AWAY*. The architecture can, alternatively, be used to generate hypotheses to classify primitives which are not so clearly defined. In this case, the initial primitives would be classified by hand, and the architecture would generate hypotheses to classify future primitives.

Aside from the concepts and recorded data, the method also requires a list of variables that will be used to describe the primitives and concepts. These should be properties which can be easily described mathematically, such as distances between players, or frequencies of occurrences of events. We generate a large set of arbitrary variables, which will contain both useful and useless structures, from which we will extract the most relevant to describe our concepts. A selection of these is described in table 4.2.

Table 4.2 A Selection of Arbitrarily Chosen Variables

Variable description
Duration of primitive in frames
Percentage of primitive spent in possession of the ball
Number of passes made in the duration of the primitive
Average area owned by the home team
Number of shots taken at the opponent goal
Average number opponents controlling areas neighbouring that of player X
Number of players on the longest team tree
Number of players closer to the ball than player X
Number of opponents marking player X
Number of players directly between the ball and the home goal

These variables could be any measurable property of the system, from the distances between objects, to the occurrences of higher level concepts. The variables chosen can be different for each concept, or each level, with lower level concepts becoming variables in higher level concepts. We will often reuse the same set of low level variables to show how they migrate through the multilevel structure. These variables have been intentionally chosen to include structures which experience decrees are significant, as well as those which are more obscure, in an attempt to find the cause of the emergent behaviours. We could focus on measuring only variables relating to structures we know exist, for example by measuring variables relating to the role of each player, but then we would only be recreating the original strategy.

The first process is to identify primitives relating to the concept. We find primitives relating to both desirable and undesirable features. For example, a pass that reaches a team mate is a good pass, whereas a pass that reaches an opponent is a bad pass; a strategy that wins a match is a good strategy, whereas a strategy that

loses a match is a bad strategy. By logging both types of pass, or strategy, then distinguishing between them, we can later identify those variables which specifically generate a desirable version of the concept. Pseudo code relating to this process is given in listing 4.1.

```
Identify sections of data relevant to the concept
1:  input data
2:  input the validity criteria for each primitive
3:  for each frame in the input data
4:       if the criteria appears in the current frame and was not
           present in the last frame
5:            create new empty primitive
6:            store this frame number as the start of the primitive
7:       end
8:       if the criteria is not present in the current frame but
           was present in the last frame
9:            store the last frame number as the end of the active
                primitive
10:      end
11: end
12: output primitives
```

Listing 4.1 Pseudo Code for Identifying Primitives

Finally, we measuring the set of primitives. These are composed of the set of variables measured over a period of significant frames. For example, for the concept *PASS*, each primitive will be a string of values relating to the average of each variable over the set of frames during which the ball is being passed. When we look at the concept of an entire strategy, the primitives will be strings of values corresponding to the average of the variables over an entire match, i.e. the whole period over which the concept is valid. Pseudo code relating to this process is given in listing 4.2.

```
Measure variables within sections
1:  input primitives
2:  load functions for measuring desired variables
3:  for each primitive
4:       for each variable
5:            measure variable and store in the primitive
6:       end
7:  end
8:  output primitives
```

Listing 4.2 Pseudo Code for Measuring Variables

4.3.2 Variable and Primitive Classification

Variables and primitives are *desirable* or *undesirable* depending on how they relate to the concept. Primitives that are neither desirable nor undesirable are *indifferent*, and variables that appear in both desirable and undesirable primitives are *common*. For example, consider a *WINNING STRATEGY* concept; a strategy primitive which results in a win is classed as desirable, a loss is undesirable, and a draw is indifferent. In this case, by distinguishing between variables in the desirable and undesirable sets, we find the structures that influence whether a strategy wins or loses.

To classify the primitives, we first need to extract the desirability criteria from the concept. In the majority of cases we will classify desirable and undesirable primitives based on whether or not the strategy in which they occur won or lost. Ranking our twenty RoboCup teams by goal difference, as shown in table 4.3, we can see that there are 8 winning, and 8 losing teams, with the remaining 4 scoring draws. In this work we hard coded the desirability criteria for each concept as shown in listing 4.3.

Table 4.3 RoboCup Team Ranking by Goal Difference

Team ranking	Goal difference	Team ranking	Goal difference
1	+10	11	0
2	+7	12	0
3	+6	13	-1
4	+3	14	-1
5	+2	15	-1
6	+1	16	-2
7	+1	17	-3
8	+1	18	-6
9	0	19	-7
10	0	20	-10

```
Extract desirability criteria
1:   desirable criteria = goal difference > 0
2:   undesirable criteria = goal difference < 0
3:   output desirable criteria
4:   output undesirable criteria
```

Listing 4.3 Pseudo Code for Extracting the Primitive Desirability Criteria

Primitives measured in data relating to teams 1-8 are classified as desirable, 9-12 as indifferent, and 13-20 as undesirable. Pseudo code relating to this process is given in listing 4.4.

```
Classify primitives
1:  input primitives
2:  input desirable criteria
3:  input undesirable criteria
4:  create an empty list of desirable primitives
5:  create an empty list of undesirable primitives
6:  create an empty list of indifferent primitives
7:  for each primitive
8:       if primitive matches desirable criteria
9:            add primitive to list of desirable primitives
10:      else if primitive matches undesirable criteria
11:           add primitive to list of undesirable primitives
12:      else
13:           add primitive to list of indifferent primitives
14:      end
15: end
16: output desirable primitives
17: output undesirable primitives
18: output indifferent primitives
```

Listing 4.4 Pseudo Code for Classifying Primitives

The next stage is to measure variable averages. Three averages are generated for each variable within the primitives: average over all primitives, average over desirable primitives, and average over undesirable primitives, as shown in listing 4.5. We also measure the range of values each variable takes.

```
Measure averages for each variable
1:  input desirable primitives
2:  input undesirable primitives
3:  input indifferent primitives
4:  for each variable
5:       desirable average = average value across the desirable
           primitives
6:       undesirable average = average value across the
           undesirable primitives
7:       global average = average value across all primitives
8:       variable range = range of values across all primitives
9:  end
10: output desirable averages
11: output undesirable averages
12: output global averages
13: output variable ranges
```

Listing 4.5 Pseudo Code for Calculating Variable Averages

These averages are then compared to determine whether the variable is important to the related concept. If the average values for a variable recorded for the desirable and undesirable primitives are on opposite sides of the global average, then we define the variable as being *significant*; it is a possible classifier for differentiating between the two types of primitive. If the two averages fall on the same side of the global mean, then the variable is *insignificant*. This is possible, since indifferent primitives are used in calculating the global mean. An example, based on data from RoboCup matches, is shown in figure 4.9. The significance of these variables will be examined in section 4.3.4.

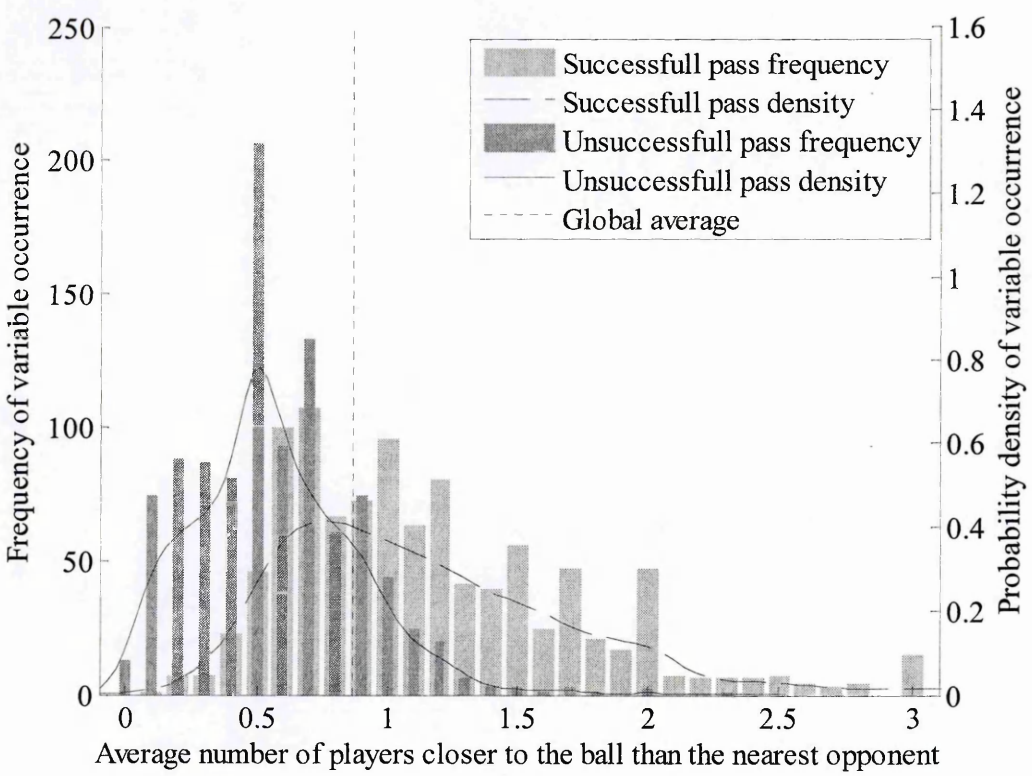


Figure 4.9 Classification of Variables by Average

In this example, the variable is used to differentiate between successful and unsuccessful passes. Out of 2036 passes, it correctly classifies 629 successful passes, and misclassifies 161 unsuccessful passes.

We denote the global average for variable i as Ma_i, the average of the desirable primitives as Ga_i, and the average of the undesirable primitives as Ba_i. The value of variable i is denoted x_i. Any insignificant variables (for which $Ga_i < Ma_i > Ba_i$ OR $Ga_i > Ma_i < Ba_i$) are discarded from further analysis; they provide no useful data for analysing our concepts, as they appear in both the desirable and undesirable primitives. Correspondingly, significant variables (for which $Ga_i > Ma_i > Ba_i$ OR $Ga_i < Ma_i < Ba_i$) are retained for analysis, as they are important to our analysis. Any variables for which the difference between Ga_i and Ba_i is less than 5% of the entire range are retained separately. Whereas the significant variables can be used to classify desirable and undesirable primitives, these *common* variables describe structures which are prevalent in both. In terms of a football strategy primitive, the significant variables give us information on how to play well, or play badly, whilst the common variables give us information on the fundamental aspects of the game. Pseudo code for variable averaging and classification is given in listing 4.6.

```
Classify variables by comparing averages
1:  input desirable averages
2:  input undesirable averages
3:  input global averages
4:  input variable ranges

Find common variables
5:  set threshold value for separation of variable averages below
      which variables are considered common
6:  generate an array to hold the list of common variables
7:  for each variable
8:       if the range is less than the threshold
9:            add the variable to the list of common variables
                along with its average value
10:      end
11: end
12: output the list of common variables

Find significant variables
13: create an empty list for holding significant variables
14: for each variable
15:      if (desirable average > global average and
           undesirable average < global average)
16:           add [variable identifier, greater than, global
                average] into significant variable list
17:      end
18:      if (desirable average < global average and
           undesirable average > global average)
19:           add [variable identifier, less than, global average]
                into significant variable list
20:      end
21: end
22: output list of significant variables
```

Listing 4.6 Pseudo Code for Classifying Variables

Our analytical method based on comparison of averages does have a possible drawback. During the course of this work, it was considered that each variable may only be useful over a particular range. For example, it may be that having one robot close to the ball is a better strategy than having none near the ball, but having five is worse than both. In future work upper and lower bounds should be imposed to limit the range of useful variables.

4.3.3 Hub Generation

In the next stage of the analysis, we generate an incidence matrix of the significant variables and desirable primitives. If the primitives are entered as rows, and the variables as columns, then for each primitive the variables will be valued as '1' if (x_i < Ma_i AND Ga_i < Ma_i AND Ba_i > Ma_i) OR (x_i > Ma_i AND Ga_i > Ma_i AND Ba_i < Ma_i), i.e. if the variable in that primitive occurs on the same side of the global average as the desirable average. This is shown in listing 4.7.

```
Incidence matrix/build stars
1:  input desirable primitives
2:  input significant variables
3:  create an empty incidence matrix with one row for each
      primitive and one column for each significant variable
4:  set every element in the matrix to 0
5:  for each primitive (i)
6:       for each significant variable (j)
7:            if (both the significant variable and
                 the corresponding variable in the primitive
                 are greater than the global average)
                 or (both the significant variable and
                 the corresponding variable in the primitive
                 are less than the global average)
8:                  set the incidence matrix element (i,j) to 1
9:            end
10:      end
11: end
12: output incidence matrix
```

Listing 4.7 Pseudo Code for Creating the Incidence Matrix

The next stage of the process is to perform the star-hub analysis. From the incidence matrix, we can form the stars relating to the set of primitives. A number of hypotheses, and possible representatives of the concept, can be generated by finding the hubs of the stars. We achieve this by first finding the intersections of all stars with one another, to give the hubs between pairs of simplices. If a hub of *dimension n* is a hub containing $n + 1$ vertices, then we will term a hub of $m + 1$ intersecting simplices an *intersection* of dimension m. We then iteratively repeat the

process finding the intersections of these hubs with one another. At each iteration we remove any intersections which contain no variables, as these represent disjoint stars. The outcome of this process is a list of hubs occurring in the star combinations. Generally hubs with large m will have small n, and vice-versa. Listing 4.8 details the recursive algorithm we use for performing the star-hub analysis.

```
Find maximal rectangles/hubs
1:  input incidence matrix
2:  call reduction function on incidence matrix
3:  call recursive function on reduced incidence matrix
4:  return resulting matrix

Recursive function
5:  input matrix
6:  if there is more than one row in the matrix
7:       create an empty matrix to hold the row intersections
8:       for each row in turn
9:            generate the intersection of the row with every
                remaining row in the input matrix and save in a
                temporary matrix
10:           call the reduction function on the temporary matrix
11:           add the reduced matrix into the intersection matrix
12:      end
13:      if the intersection matrix is not empty
14:           call the recursive function on the intersection
                matrix
15:      else
16:           return nothing
17:      end
18:      add the returned matrix to the end of the input matrix
19:      return the input matrix
20: else
21:      return the row
22: end

Reduction function
23: input matrix
24: for each row in the matrix
25:      check remaining rows for duplicates
26:      remove duplicate rows
27:      if row is empty
28:           remove row
29:      end
30: end
```

Listing 4.8 Pseudo Code for Performing the Star-Hub analysis

The process begins on line 1 by taking in the incidence matrix constructed in listing 4.7. Line 2 calls the reduction function on line 23, which removes any repeated or empty rows from the matrix. Line 3 calls the recursive function on the reduced matrix, and line 4 returns the list of hubs in a matrix form. Each row in the output matrix is a hub, and details which variables are present within it.

The recursive function, which begins on line 5, finds the hubs common across rows in the input matrix. On the first call, the function finds the intersection between every pair of rows, and stores the resulting hubs in a matrix on line 11. This matrix of hubs in then passed back into the recursive function on line 14. On each subsequent call, the intersections between hubs are found. As more calls are made, the dimension of intersection of the hubs increases. Eventually the hubs being passed back into the function on line 14 will be disjoint, and the list of intersections will be empty. This occurs at the deepest call to the recursive function, and causes it to return on line 15. Each function call then returns, adding the intersections it found to a single list. This is the list of hubs, which is then returned to the calling function. The return statement on line 21 handles the case when all the hubs in the deepest function call intersect fully. In this instance, there is only one intersection, and no more calls to the function are required.

It should be noted that this method does not generate all the possible hubs; that would require finding the intersection of every combination of simplices with every other combination of simplices. By finding the intersection of hubs, we dramatically reduce the search space, and uncover just the *maximal* hubs. By this we mean the largest dimension of hub joining a set of simplices.

In this work, we construct the relation described by a hub by hand, based on the concept, the primitive classification criteria, and the output of the variable classification.

In general, large numbers of hubs are produced. We count the number of desirable primitives in which each hub occurs (the dimension of intersection), and the number of variables contained within it (the hub dimension). For the purpose of displaying measured hubs, we tabulate only the hub with the largest dimension for each order of intersection.

These measures are also useful in selecting representatives and hypotheses. Although these are products of the abstraction architecture, we will leave a description of how they are formed until chapter 5, when we deal with generating representatives to form a controller. For the remainder of this chapter we will focus on finding and examining the hubs relating to our concepts.

4.3.4 Significance of Analysis

Hubs measured using the aforementioned methods indicate structures which are likely to be more prevalent in desirable situations. These can be used to identify, for example, when a pass is likely to succeed or fail. An earlier statistical examination of the ability of star-hub analysis to classify data is given in (Iravani, 2005a).

Here, we analyse the significance of our method of classifying variables by comparing averages. Once we have identified variables which occur more frequently in desirable situations, we can begin to examine the occurrence of combinations of these variables. By using these combinations to classify desirable and undesirable situations, we increase the chances of a successful classification

over using a single variable alone. We illustrate this by demonstrating how multiple variables combine to improve the chances of identifying a good pass situation.

From our RoboCup data, we randomly select a training set of 25 successful and 25 unsuccessful passes. A pass is defined as the sequence of a ball leaving one player and arriving at another. It is successful if the receiving player is on the same team as the passer, and unsuccessful if it is on the opposing team. This training set is then used to identify possible classifiers from a set of 100 variables relating to spatial structures. The occurrences of these possible classifiers are then measured in a randomly selected set of 1018 successful and 1018 unsuccessful passes. The results are shown in figure 4.10.

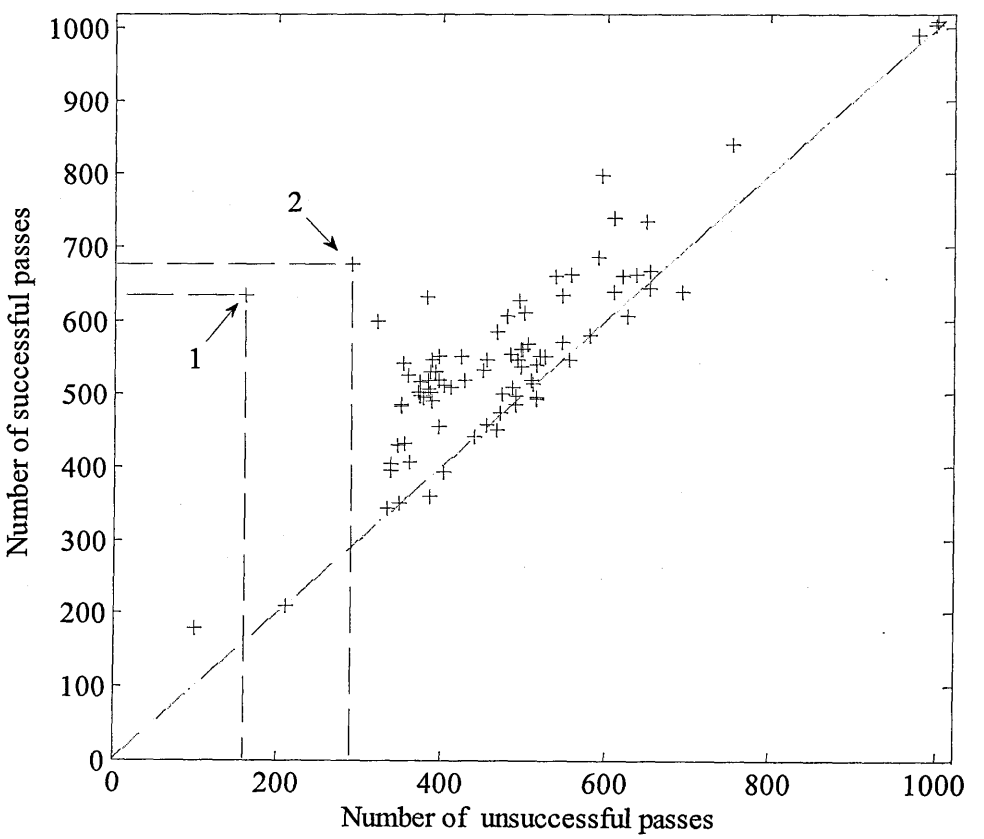
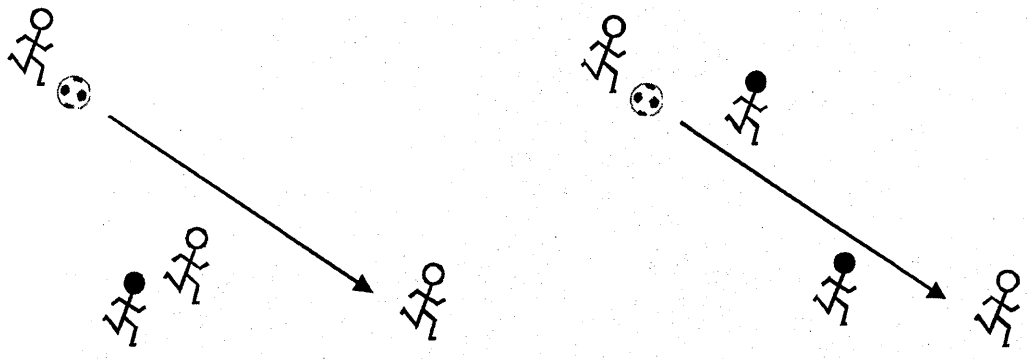


Figure 4.10 Occurrences of Variables in Measured Passes

Performing the analysis on the training data identifies 85 possible classifier variables. When measured in the full data set, 73 occur more often in successful passes, 11 occur more often in unsuccessful passes, and 1 occurs equally in both. These 11 poor classifiers all occur in similar numbers of successful and unsuccessful passes, with the worst only misclassifying a pass on 52% of occasions. Conversely, the 73 good classifiers have a much wider spread, with the best correctly classifying a pass on 80% of occasions. The statistical significance of these results is verified using the sign test.

If the method for classifying variables did not provide valid results, we would expect to measure their occurrence in the same number of successful and unsuccessful passes. This forms the null hypothesis; that the probability of a variable occurring is equal in both successful and unsuccessful passes. From our data in figure 4.10, we measured 11 variables which occurred more often in unsuccessful passes out of a total of 84 (ignoring the 1 variable which occurred with equal frequency in both types of pass). Using the one-sided sign test on our null hypothesis, we calculate the probability of 11 or fewer variables, out of 84, occurring more often in unsuccessful passes as 1.12×10^{-12}. This is a minute probability, which rejects the null hypothesis, and validates the classification technique.

The best classifier variable, labelled '1' in figure 4.10, corresponds to the number of home players closer to the ball than the nearest opponent during the course of the pass. The analysis indicates that in successful passes, there is, on average, always more than 0.87 home players closer to the ball than the nearest opponent. This is illustrated in figure 4.11. Notice that it is therefore possible for the ball to be closer to an opponent for part of the pass, whilst still being successful.

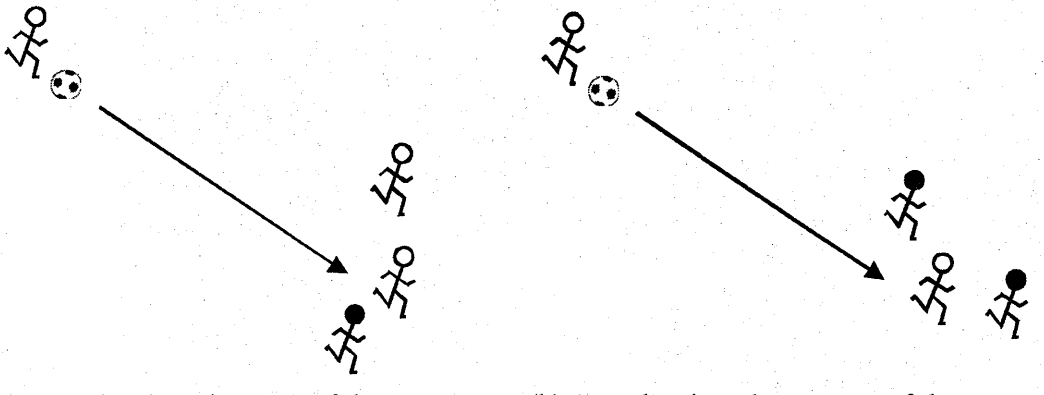


(a) A predominantly successful pass configuration with one home player always closer to the ball than the nearest opponent

(b) A predominantly unsuccessful pass configuration with opponent players closer to the ball for the majority of the pass

Figure 4.11 Illustration of a Successful Classifier Variable

The second best classifier from figure 4.10, labelled '2', corresponds to the number of opponent players in spaces neighbouring the receiver. It correctly classifies a successful pass on 70% of occasions. The analysis indicates that in successful passes there is, on average, fewer than 1.33 opponent players in spaces neighbouring the receiver. This is shown in figure 4.12.



(a) A predominantly successful pass configuration with only one opponent player adjacent to the receiver

(b) A predominantly unsuccessful pass configuration with two opponent players adjacent to the receiver

Figure 4.12 Illustration of another Successful Classifier Variable

We now examine the effect of using pairs of variables to classify the passes. The 85 classifier variables can be combined to form 3570 unique pairs. Each of these forms a hub of dimension 1. The occurrence of each hub in both successful

and unsuccessful passes is measured, and the results shown in figure 4.13. Those hubs containing the variables described above have been highlighted.

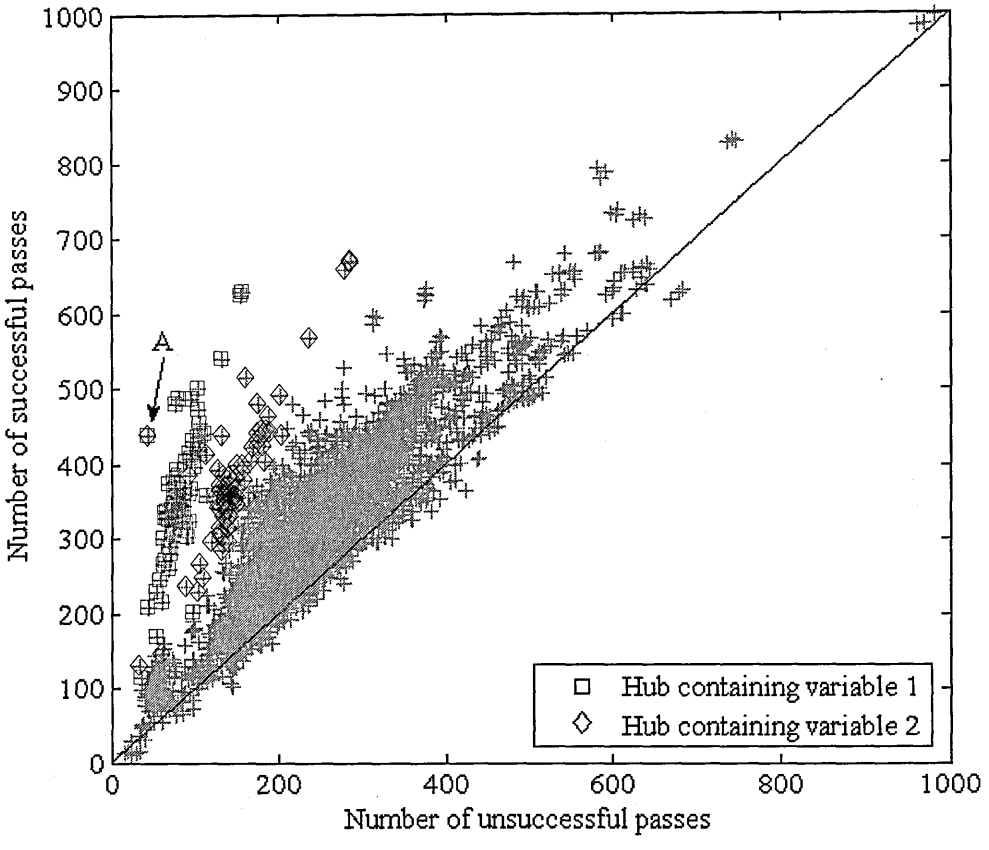


Figure 4.13 Occurrences of 2 Variable Hubs in Measured Passes

3344 hubs appear more frequently in successful passes, whilst 211 hubs appear more frequently in unsuccessful passes. The best classifier hub in figure 4.13 is marked 'A', and is a combination of the two best classifier variables from figure 4.10. By combining the two variables, the chance of successful classification of the pass data has risen to 90.7%.

Continuing on to hubs of dimension 2, there are 98770 unique combinations of 3 variables. Figure 4.14 shows the number of passes in which these structures occur.

Those hubs including the best variable pair from the previous analysis are highlighted.

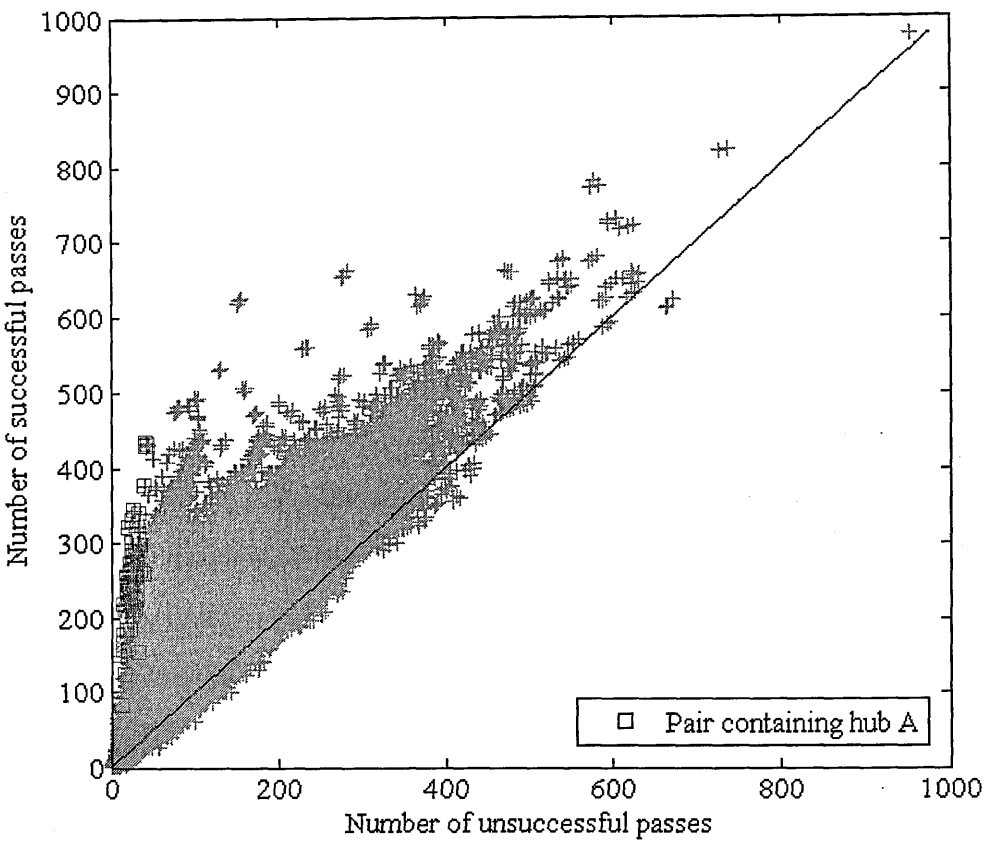


Figure 4.14 Occurrences of 3 Variable Hubs in Measured Passes

94794 hubs appear more often in successful passes, whereas 3596 appear in unsuccessful passes. The best classifier hub contains the 'A' pair, and can now be used to classify the passes with a 93.9% chance of success.

As we increase the number of variables in the hub, they become more descriptive. Therefore, the chance of the hub correctly classifying a primitive increases, provided the variables have been well chosen. However, the consequence is that the hub becomes more specific, and excludes more of the successful passes.

Iravani (2005a) uses measures of specificity and broadness to measure the hub significance. The specificity is the maximum probability that a primitive will be classified as type *C* when it contains hub *H*. For example, the probability of a pass being successful if it exhibits the 'A' hub is 437/482, because the hub is shared by 482 passes, of which 437 are successful. The broadness is the maximum probability that a primitive will contain hub *H*. In the case of hub 'A', the broadness is 482/2036, since the hub is shared by 482 primitives out of a possible 2036.

4.3.5 A Comparison to Earlier Work

The techniques introduced in the previous sections are a modification of those used in (Iravani, 2005a). In that work, variables took logical values, and all variables were used within the hub generation. This has three drawbacks:

Firstly, allowing all variables to be used in the hub generation creates a very large search space. As we shall show in section 4.11, computation time increases dramatically with the number of variables included.

Secondly, logical variables are less accurate in terms of their meaning. Consider the distance measured between two players. Iravani and Johnson (2005) use 4 binary variables to represent a single distance. Each variable is assigned an arbitrary range, so that together they cover all values between 0 and infinity. The variable containing the measurement is given the value '1'. The boundary values may have no significance in robot football, and it may be more appropriate to use a different range. The technique proposed here has the benefit of having much greater meaning. The average, Ma_i, is a measure of a valid separation between desirable and undesirable primitives, rather than an arbitrary value.

Thirdly, using multiple binary variables to segment a single continuous one results in redundancy. Iravani (2005a) examines primitives consisting of 50 binary variables. 44 of these relate to 8 quantitative measurements: angle, direction, position, and 5 distances, which could be represented by just 8 variables using the method described in the previous sections.

4.4 Strategy Generation

In this section we will use our architecture to search for hubs to relate structures and events on a football pitch to our concept of *WINNING STRATEGY* (figure 4.15). Considering our structural diagram shown in figure 4.8, this may seem strange, since we are mapping variables at the lowest level directly to the highest level concept, bypassing the intermediate levels. Given that our choice of multilevel structure is arbitrary, it is possible to choose as many or few levels as we like. The more levels and concepts we insert into our structure (provided they are well chosen), the more accurate our representation of robot football will become. However, there may still be some direct relationship that can be drawn between these two levels. We will examine later, in section 4.9, how these relationships map across multiple levels.

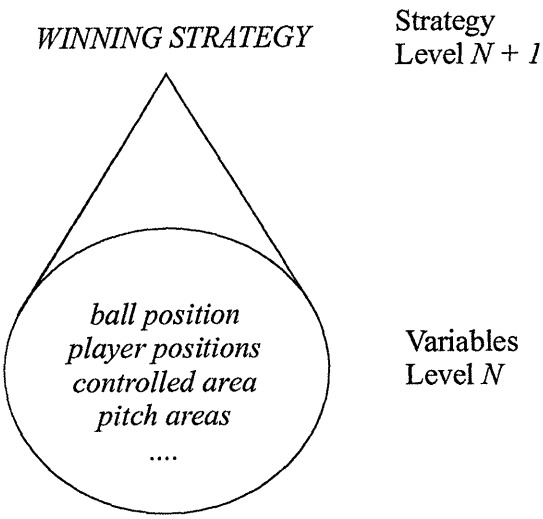


Figure 4.15 Mapping of Variables into a Strategy Concept

For this concept we will measure 66 arbitrarily chosen variables, which are briefly described in table 4.4. The variables may themselves appear at different levels of the multilevel structure, with some deserving their own concepts, such as the number of passes. Some have obvious significance to the game, whereas others may seem entirely irrelevant.

Table 4.4 Variables Selected for Concept Generation

Variable	Description
x_1	Match duration in frames
x_2	Duration of primitive in frames
x_3	Percentage of primitive spent in possession of the ball
x_4	Number of passes made in the duration of the primitive
x_5	Percentage of home ball ownership time spent passing
x_6	Percentage of the primitive played with the ball in the home half
x_7	Percentage of player turns spent closer to the home goal than the ball
x_8	Percentage of player turns spent further from the away goal than the ball
x_9	Percentage of player turns spent inside the triangle formed by the ball and home goal posts
x_{10}	Average area owned by the home team
x_{11}	Standard deviation of player areas as a percentage of total area
x_{12}	Standard deviation in number of neighbours
x_{13}	Percentage of passes which were successful
x_{14}	Percentage of opponent passes which were unsuccessful
x_{15}	Number of home passes
x_{16}	Number of shots at the opponent's goal
x_{17}	Percentage of shots becoming goals
x_{18}	Standard deviation of number of neighbours on the opposing team
x_{19}	Average number of neighbours
x_{20}	Average number of opponent neighbours
x_{21}	Number of players on the largest segment of the team tree
x_{22-32}	Number of players closer to the ball than each opponent player
x_{33-43}	Number of players closer to the home goal than each opponent player
x_{44-54}	Number of players closer to the opponent goal than each opponent player
x_{55}	Percentage of player turns spent unmarked
x_{56-66}	Percentage of player turns spent in a one-on-one to eleven-on-one marking configuration

We will attempt to extract information relating to our *WINNING STRATEGY* concept from the RoboCup log files. In section 4.3.2 we showed how these could be grouped into winning losing and drawing strategy sets. This classification is ideal for examining our *WINNING STRATEGY* concept, in which the objective is to score more goals than the opponent.

It should be noted that although we have chosen to rate strategies on their goal difference, there may be other acceptable criteria by which to rate the primitives. In this example, it may also be acceptable to rate strategies by the number of goals scored, which would give a slightly different set of results.

For our 20 primitives, the average primitives *Ma*, *Ga*, and *Ba* are shown in table 4.5.

Table 4.5 Average Variable Values Measured in Strategy Primitives

Variable	*Ma*	*Ga*	*Ba*	Variable	*Ma*	*Ga*	*Ba*
x_1	6349.00	6413.50	6413.50	x_{34}	4.06	3.97	3.95
x_2	5650.40	5620.00	5620.00	x_{35}	4.90	4.78	4.86
x_3	50.00	52.72	47.27	x_{36}	5.91	5.80	5.81
x_4	353.40	342.00	342.00	x_{37}	6.48	6.37	6.40
x_5	57.73	57.99	56.32	x_{38}	7.23	7.20	7.12
x_6	49.58	39.67	59.33	x_{39}	9.10	9.08	9.00
x_7	63.14	67.96	57.75	x_{40}	9.58	9.57	9.52
x_8	57.26	62.31	51.78	x_{41}	10.00	10.02	9.96
x_9	13.32	14.14	12.55	x_{42}	10.45	10.48	10.46
x_{10}	50.00	50.72	49.28	x_{43}	10.98	10.98	10.99
x_{11}	4.68	5.04	4.41	x_{44}	0.01	0.01	0.00
x_{12}	1.28	1.27	1.28	x_{45}	0.22	0.24	0.22
x_{13}	68.60	72.16	66.17	x_{46}	0.57	0.60	0.67
x_{14}	31.35	33.73	27.82	x_{47}	1.10	1.20	1.18
x_{15}	176.85	186.00	156.38	x_{48}	1.74	1.89	1.83
x_{16}	5.70	8.50	4.75	x_{49}	3.49	3.56	3.64
x_{17}	31.09	63.05	14.69	x_{50}	4.50	4.62	4.62
x_{18}	0.76	0.75	0.77	x_{51}	5.21	5.29	5.30
x_{19}	4.82	4.82	4.82	x_{52}	6.52	6.63	6.45
x_{20}	0.87	0.84	0.91	x_{53}	7.39	7.45	7.33
x_{21}	9.09	8.88	9.36	x_{54}	8.45	8.46	8.49
x_{22}	0.69	0.69	0.71	x_{55}	57.97	55.48	55.47
x_{23}	1.63	1.53	1.71	x_{56}	38.74	40.98	40.97
x_{24}	2.64	2.52	2.74	x_{57}	3.11	3.35	3.38
x_{25}	3.63	3.47	3.73	x_{58}	0.16	0.18	0.17
x_{26}	4.63	4.41	4.81	x_{59}	0.01	0.01	0.00
x_{27}	5.53	5.33	5.70	x_{60}	0.00	0.00	0.00
x_{28}	6.63	6.41	6.91	x_{61}	0.00	0.00	0.00
x_{29}	7.42	7.24	7.67	x_{62}	0.00	0.00	0.00
x_{30}	8.33	8.17	8.53	x_{63}	0.00	0.00	0.00
x_{31}	9.13	8.96	9.34	x_{64}	0.00	0.00	0.00
x_{32}	10.24	10.06	10.38	x_{65}	0.00	0.00	0.00
x_{33}	3.10	3.04	3.00	x_{66}	0.00	0.00	0.00

Using our analysis technique, we find that 8 of the variables are common across all the primitives. These are shown in table 4.6, along with their respective values. Variables $x_{60} - x_{66}$ are all valued at zero, indicating that the structures they represent do not turn up in any of the team strategies. These are therefore structures which should be avoided. Variable x_{43} turns up on every team with a value of approximately 11. This represents the number of home players that are closer to the home goal than the furthest opponent. In descriptive terms, this means that home players should never venture further out than the opponent goal keeper. These 8 common variables all give useful information on fundamental structures relating to robot football and are therefore all part of the *WINNING STRATEGY* concept.

Table 4.6 Common Variables in Strategy Primitives

Variable	Value
x_{43}	10.98
$x_{60\text{-}66}$	0

Of the remaining 58 variables, we identify 34 significant variables using our selection method. For our concept, we are interested in the variables which occur in winning strategies. We therefore need to know the average values for the variables over all the primitives, and whether they occur more, or less in winning primitives. This is shown in table 4.7.

Table 4.7 Significant Variables and Values for Strategy Generation

Variable	Relation	*Ma*	Variable	Relation	*Ma*
x_3	>	50.00	x_{23}	<	1.63
x_5	>	57.73	x_{24}	<	2.64
x_6	<	49.58	x_{25}	<	3.63
x_7	>	63.14	x_{26}	<	4.63
x_8	>	57.26	x_{27}	<	5.53
x_9	>	13.32	x_{28}	<	6.63
x_{10}	>	50.00	x_{29}	<	7.42
x_{11}	>	4.68	x_{30}	<	8.33
x_{13}	>	68.60	x_{31}	<	9.13
x_{14}	>	31.35	x_{32}	<	10.24
x_{15}	>	176.85	x_{41}	>	10.00
x_{16}	>	5.70	x_{43}	<	10.98
x_{17}	>	31.09	x_{44}	>	0.01
x_{18}	<	0.76	x_{45}	>	0.22
x_{20}	<	0.87	x_{52}	>	6.52
x_{21}	<	9.09	x_{53}	>	7.39
x_{22}	<	0.69	x_{59}	>	0.01

We now build the incidence matrix for the 8 winning strategies. This is shown in table 4.8. Each entry in the matrix is a '1' if and only if the variable it represents falls the desired side of the global mean as shown in table 4.7 above.

Table 4.8 Incidence Matrix for the Winning Strategy Primitives

Variables	Primitives								Variables	Primitives							
	1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8
3	1	0	1	1	0	0	1	1	23	0	1	1	1	0	1	1	0
5	0	0	1	1	0	1	0	0	24	0	1	1	1	0	1	1	0
6	0	1	1	1	1	1	1	1	25	1	1	1	1	0	1	1	0
7	0	1	1	1	1	1	1	0	26	1	1	1	1	0	1	1	0
8	1	1	1	1	1	1	0	0	27	1	1	1	1	0	1	1	0
9	0	1	1	1	1	1	0	1	28	1	1	1	1	0	1	1	0
10	1	0	1	1	0	1	1	0	29	1	1	1	1	0	1	1	0
11	1	1	1	1	1	0	1	0	30	1	1	1	1	0	1	1	0
13	1	0	1	1	0	1	1	1	31	1	0	1	1	0	1	1	0
14	1	0	0	1	1	0	0	0	32	1	0	1	1	0	1	1	0
15	1	0	1	1	0	1	1	0	41	0	1	1	0	1	0	0	1
16	1	1	1	0	0	0	1	0	43	0	1	0	0	1	1	1	1
17	1	1	1	1	1	1	0	1	44	1	1	1	0	1	0	1	1
18	1	1	1	1	1	0	1	0	45	1	1	0	1	0	1	0	0
20	0	1	1	0	1	0	1	0	52	0	0	0	0	1	1	0	1
21	1	0	1	1	0	1	1	0	53	0	1	0	0	1	1	0	1
22	0	1	1	1	1	1	0	0	59	0	0	0	0	1	0	1	0

Performing the star-hub analysis on this data set produced 91 unique maximal hubs. Table 4.9 shows a selection of the maximal hubs ordered by size and frequency of occurrence. Measures of specificity and broadness have been included to show the significance of each hub. There is no hub which occurs across all 8 desirable primitives.

Table 4.9 Maximal Strategy Hubs

Hub	Spec.	Broad.
$\langle x_{17} ; R \rangle$	7/9	9/16
$\langle x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30} ; R \rangle$	6/8	8/16
$\langle x_{10}, x_{13}, x_{15}, x_{21}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32} ; R \rangle$	5/6	6/16
$\langle x_{6}, x_{7}, x_{10}, x_{13}, x_{15}, x_{21}, x_{23}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32} ; R \rangle$	4/4	4/16
$\langle x_{5}, x_{6}, x_{7}, x_{8}, x_{9}, x_{10}, x_{13}, x_{15}, x_{17}, x_{21}, x_{22}, x_{23}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32} ; R \rangle$	3/3	3/16
$\langle x_{3}, x_{5}, x_{6}, x_{7}, x_{8}, x_{9}, x_{10}, x_{11}, x_{13}, x_{15}, x_{17}, x_{18}, x_{21}, x_{22}, x_{23}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32} ; R \rangle$	2/2	2/16
$\langle x_{3}, x_{5}, x_{6}, x_{7}, x_{8}, x_{9}, x_{10}, x_{11}, x_{13}, x_{15}, x_{16}, x_{17}, x_{18}, x_{20}, x_{21}, x_{22}, x_{23}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{41}, x_{44} ; R \rangle$	1/1	1/16

The relationship R is the same for every hub. In this case, it represents each variable occurring in a winning strategy with an appropriate average value over the duration of the match. In this instance, a hub containing 15 variables can be used to perfectly classify half of the winning strategies. From the specificity, we can see that as the hub size increases, the hub represents a higher proportion of desirable to undesirable primitives. Furthermore, we can add the 8 variables common to all teams to each hub, giving greater definition to the representatives of our concept. Given the small number of primitives used in this example, it is inappropriate to suggest how significant these hubs may be when measured in a larger number of matches. The significance to a larger population will be shown in section 4.7.

Exploring these results in more depth we find that variable x_{17}, which appears in 7 of the 8 winning strategies, relates to the percentage of shots on the opponent goal which are successful. Our analysis shows that in winning teams $x_{17} > 31.09$. This seems logical, since it relates directly to the score of each team. In the only winning primitive, p_7, where $x_{17} < 31.09$, 10 shots were taken, with only 2 being

successful. In this case, the high number of attempts was sufficient to score a win. Variable x_6 also occurs in 7 of the 8 winning strategies with a value of < 49.58, and is the most common variable, occurring in 58 of the 91 hubs. This is the percentage of time the ball spends in the home players half, and indicates that these winning strategies spend more time on the offensive, which is a sensible assumption. Conversely, variable $x_{59} > 0.01$ only appears in 2 of the winning strategies. This represents the number of instances 4 home players team up to mark an opponent player. This is obviously a rare occurrence, but its appearance in winning teams only suggests it could be a useful tactic. However, it is not the occurrence of the variables on their own that is of interest to us, but their occurrence in the emergent combinations.

4.5 Play Generation by Goal Difference

Having identified structures of variables to define the concept of a winning strategy, we now move on to show how the same technique can be applied to concepts at a lower level in the multilevel structure.

Recalling table 4.1, we identified eight possible play concepts based on ball position and possession. These concepts span levels $N + 2$ and $N + 3$ of the proposed multilevel structure given in figure 4.8. In this section we shall generate representatives for all eight plays using the method shown in the previous section. We have discussed how primitives can be classed as desirable or undesirable, based on a variety of criteria. Here, we shall use the same classification as used in the previous section, i.e. plays are deemed desirable if they are part of a strategy that

wins. We will examine the effect of using a different classification criterion in section 4.6.

Using the same 66 variables from section 4.4, we identify primitives relating to the above concepts in the following way. For every play concept described above, we search each of our 20 recorded strategies for the frames which satisfy the concept. We then measure all the variables over these sets of frames and take an average. This gives us 20 primitives for each concept. We then classify the primitives as before by goals scored, into 8 winning, 8 losing, and 4 drawn.

Using the method shown in section 4.4, we generate the variables common to desirable and undesirable primitives, as well as the maximal hubs. Those for the *IN HOME* concept are detailed in the following section. The remaining 7 play concepts have also been analysed, and the results will be summarised later in this thesis.

4.5.1 Example: *IN HOME*

For the *IN HOME* play concept we will again classify primitives as desirable or undesirable based on the goal difference of the strategy they are measured from. In this way we are finding the similarities in plays performed by winning teams.

The 12 variables common to both desirable and undesirable primitives formed under this concept are shown in table 4.10. Variable x_6 is the percentage of time the ball spent in the home half. This is the criteria for this concept, and so is justifiably 100% for every primitive measured. Variable x_{17} is the percentage of shots taken on the opponent goal. Although shots were taken, none were successful in any primitive. Variables x_{32}, and x_{43}, are the number of home players closer to the ball, and to the home goal, than the opponent goalkeeper. Variable x_{44} is the number of

home players closer to the opponent goal than their goalkeeper, and variables x_{60-66} are uncommon marking configurations.

Table 4.10 Common Variables in Play Primitives

Variable	Value
x_6	100
x_{17}	0
x_{32}	10.97
x_{43}	11
x_{44}	0
x_{60-66}	0

Of the remaining variables, 33 were found to be significant to the concept, generating 73 maximal hubs. A selection of these is given in table 4.11.

Table 4.11 Maximal Hubs for Plays Analysed by Goal Difference

Hub	Spec.	Broad.
$\langle x_4 ; R \rangle$	8/10	10/16
$\langle x_2, x_4, x_{15} ; R \rangle$	7/9	9/16
$\langle x_2, x_4, x_7, x_8, x_{15} ; R \rangle$	6/7	7/16
$\langle x_2, x_4, x_7, x_8, x_{15}, x_{28} ; R \rangle$	5/6	6/16
$\langle x_2, x_4, x_7, x_8, x_9, x_{15}, x_{16}, x_{20}, x_{51} ; R \rangle$	4/4	4/16
$\langle x_2, x_4, x_7, x_8, x_9, x_{15}, x_{16}, x_{20}, x_{34}, x_{35}, x_{36}, x_{51} ; R \rangle$	3/3	3/16
$\langle x_2, x_4, x_7, x_8, x_9, x_{10}, x_{15}, x_{16}, x_{20}, x_{22}, x_{30}, x_{31}, x_{38}, x_{39}, x_{41}, x_{46}, x_{51} ; R \rangle$	2/2	2/16
$\langle x_2, x_4, x_7, x_8, x_9, x_{10}, x_{15}, x_{16}, x_{18}, x_{20}, x_{22}, x_{30}, x_{31}, x_{32}, x_{34}, x_{35}, x_{36}, x_{37}, x_{38}, x_{39}, x_{41}, x_{46}, x_{51}, x_{53}, x_{54} ; R \rangle$	1/1	1/16

Variable x_4 is a strong classifier, occurring in all 8 of the desirable primitives, and consequently all 73 hubs. It relates to the number of passes made by the home team, and has a value of < 176.25. This shows that winning teams pass the ball fewer times in their own half. This may be important in itself, or may indicate that winning teams spend less time in their own half, or that too much passing is a bad defensive strategy.

Here, a hub containing 9 variables can be used to perfectly classify half of the desirable primitives.

4.6 Play Generation by Comparison

We will now describe an alternative approach to play generation. In the previous section we were interested in how to generate concepts relating to desirable plays, based on the overall success of a strategy according to the goal difference. Another possible approach would have been to rate each play individually based on whether it resulted in a goal, a loss of possession, a clearance, or some other event. In this section we shall examine play concepts when considered against their bipolar opposite.

We have already collated primitives relating to the times when the ball is in the home half, the away half, when it has been in possession, and out of possession. Consider an alternative *IN HOME* concept. We still desire to find how our variables perform when the ball is in the home half of the pitch, but instead of comparing configurations between good and bad plays, we will compare them to the configurations recorded when the ball is in the away half.

Using the same 66 variables, we form a new classification between the primitives measured with the ball in the home and away halves. In this instance we have 20 desirable and 20 undesirable primitives: one of each from every team. There are no 'drawn' or indifferent primitives since the ball is always in the home or away half.

Focussing on the concept play of *IN HOME*, we find 8 variables common to both the *ball in home half* and *ball in away half* primitives. These exactly match those in table 4.10 above. This is expected since variables common to both sets of primitives in section 4.5.1 must be part of the encompassing *STRATEGY* concept, and therefore part of both primitives here.

Continuing the analysis we find there are 56 significant variables, forming 2951 maximal hubs. The largest and most frequently occurring are given in table 4.12.

Table 4.12 Maximal Hubs for Plays Analysed by Comparison

Hub	Spec.	Broad.
$\langle x_6, x_{11}, x_{17}, x_{25}, x_{26}, x_{29}, x_{30}, x_{31}, x_{32}, x_{43}, x_{44} ; R \rangle$	20/20	20/40
$\langle x_6, x_{11}, x_{17}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{43}, x_{44} ; R \rangle$	19/19	19/40
$\langle x_6, x_{10}, x_{11}, x_{17}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{43}, x_{44} ; R \rangle$	18/18	18/40
$\langle x_6, x_{10}, x_{11}, x_{17}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{43}, x_{44} ; R \rangle$	17/17	17/40
$\langle x_6, x_{10}, x_{11}, x_{17}, x_{23}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{43}, x_{44} ; R \rangle$	15/15	15/40
$\langle x_6, x_{10}, x_{11}, x_{12}, x_{17}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{43}, x_{44} ; R \rangle$	14/14	14/40
$\langle x_3, x_6, x_{10}, x_{11}, x_{15}, x_{16}, x_{17}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{43}, x_{44} ; R \rangle$	13/13	13/40
$\langle x_3, x_6, x_{10}, x_{11}, x_{15}, x_{16}, x_{17}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{43}, x_{44} ; R \rangle$	12/12	12/40
$\langle x_6, x_{10}, x_{11}, x_{12}, x_{15}, x_{16}, x_{17}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{43}, x_{44} ; R \rangle$	11/11	11/40
$\langle x_6, x_{10}, x_{11}, x_{12}, x_{15}, x_{16}, x_{17}, x_{21}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{43}, x_{44} ; R \rangle$	10/10	10/40
$\langle x_6, x_{10}, x_{11}, x_{17}, x_{19}, x_{21}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{39}, x_{43}, x_{44}, x_{45}, x_{47}, x_{48} ; R \rangle$	9/9	9/40
$\langle x_3, x_6, x_{10}, x_{11}, x_{12}, x_{17}, x_{19}, x_{21}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{39}, x_{43}, x_{44}, x_{45}, x_{47}, x_{48} ; R \rangle$	8/8	8/40
$\langle x_3, x_6, x_{10}, x_{11}, x_{12}, x_{15}, x_{16}, x_{17}, x_{19}, x_{21}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{39}, x_{43}, x_{44}, x_{45}, x_{47}, x_{48} ; R \rangle$	7/7	7/40
$\langle x_3, x_6, x_{10}, x_{11}, x_{12}, x_{15}, x_{16}, x_{17}, x_{19}, x_{21}, x_{23}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{39}, x_{43}, x_{44}, x_{45}, x_{47}, x_{48} ; R \rangle$	6/6	6/40
$\langle x_3, x_6, x_9, x_{10}, x_{11}, x_{12}, x_{15}, x_{16}, x_{17}, x_{19}, x_{21}, x_{23}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{39}, x_{43}, x_{44}, x_{45}, x_{47}, x_{48} ; R \rangle$	5/5	5/40
$\langle x_3, x_6, x_{10}, x_{11}, x_{12}, x_{15}, x_{16}, x_{17}, x_{19}, x_{23}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{33}, x_{38}, x_{43}, x_{44}, x_{47}, x_{48}, x_{49}, x_{50}, x_{51}, x_{57} ; R \rangle$	4/4	4/40
$\langle x_3, x_6, x_7, x_8, x_{10}, x_{11}, x_{12}, x_{15}, x_{16}, x_{17}, x_{19}, x_{21}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{34}, x_{35}, x_{36}, x_{37}, x_{39}, x_{43}, x_{44}, x_{45}, x_{47}, x_{48}, x_{52}, x_{56} ; R \rangle$	3/3	3/40
$\langle x_3, x_5, x_6, x_7, x_8, x_{10}, x_{11}, x_{12}, x_{15}, x_{16}, x_{17}, x_{18}, x_{19}, x_{21}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{34}, x_{35}, x_{36}, x_{37}, x_{39}, x_{43}, x_{44}, x_{45}, x_{47}, x_{48}, x_{52}, x_{53}, x_{54}, x_{56} ; R \rangle$	2/2	2/40
$\langle x_5, x_6, x_7, x_8, x_{10}, x_{11}, x_{12}, x_{13}, x_{15}, x_{16}, x_{17}, x_{19}, x_{20}, x_{23}, x_{24}, x_{25}, x_{26}, x_{27}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{33}, x_{35}, x_{38}, x_{40}, x_{41}, x_{42}, x_{43}, x_{44}, x_{47}, x_{48}, x_{49}, x_{50}, x_{51}, x_{55}, x_{56}, x_{57}, x_{58}, x_{59} ; R \rangle$	1/1	1/40

Eleven variables exist as strong classifiers in this example, being present in all 20 of the home half primitives. An interesting inclusion is that of x_{11}, which is the standard deviation in home players' areas. This is greater in plays where the ball is in the away half, and bigger changes in area occur.

Examining these hubs, and comparing them to those in table 4.11, we can see there is little correlation between the two, even though they both correspond to play structures appearing when the ball is in the home half of the pitch. The difference is due to the alternative ways we have chosen to describe what makes a desirable primitive. This affects the relation R, and results in two different concepts.

The high specificity and broadness recorded for these hubs indicates a much stronger relational structure than that given in section 4.5.1. This reflects the reality that there is a much greater difference between attacking and defending plays than there is between good and bad examples of the same play. The strongest link between the two play concepts comes via variables x_{15} and x_{16}. This pair occurs in at least half of the primitives measured in each.

4.7 Tactic Generation

So far we have measured each primitive over the whole course of a match, but have also suggested measuring over the duration of specific play instances, and classing their success by the individual outcomes, rather the general goal difference. We will now demonstrate this method on an even lower level in the multilevel structure, that of tactics, and passing in particular.

For this analysis we extended our list of variables to 100 to include additional parameters which may be relevant specifically to passing events. These incorporate structures which directly relate to the passing and receiving players, and to the pass itself. Rather than list all 100 variables, we will reveal a shortened of the most significant variables list later on.

We analyse 2036 passes randomly selected from across the ten matches: 1018 successful and 1018 unsuccessful. These form our two sets of primitives. Each pass has a duration beginning when the ball leaves the passing player, and ending when it is received by another player. We define a successful pass as one in which the passer and receiver are on the same team. An unsuccessful pass is one in which the passer and receiver are on opposing teams. Some variables are measured and averaged over the pass duration, whilst others are measured purely at the moment the ball is kicked, or received.

By increasing the number of primitives and variables, we drastically increase the number of possible hubs and stars. The size of the search space is too great for our algorithms, and highlights what is possibly the main drawback in the analysis technique. The more primitives used to generate each concept, the more reliable the representation will become; however, given the number of possibilities for variables in concepts as complex as those studied here, the search spaces can easily become unmanageable. In this thesis we have used very simple analysis techniques to demonstrate the principles. It would not be unrealistic to assume that with optimised algorithms, and more resources, larger problems could be tackled in this way. It should also be noted that the size of this search space is still much smaller than that of the game tree described in section 3.1.

Rather than try to generate all the possible hubs for this set of data, we use a training set to generate a shortened list of hubs, and then search the primitives to find their significance. In our experiment, we initially analysed 25 primitives chosen at random from each of the desirable and undesirable sets. These produced 9240 hubs, which we then compared to all 2036 primitives. The specificity and broadness

relating to a selection of these hubs is given in table 4.14. Descriptions of the variables forming these hubs is given in table 4.14.

Table 4.13 Maximal Pass Hubs

Hub	Spec.	Broad.
$\langle x_{73} ; R \rangle$	681/971	971/2036
$\langle x_1, x_{27} ; R \rangle$	483/561	561/2036
$\langle x_1, x_{27}, x_{89} ; R \rangle$	376/422	422/2036
$\langle x_1, x_{27}, x_{76}, x_{89} ; R \rangle$	325/356	356/2036
$\langle x_1, x_{27}, x_{40}, x_{76}, x_{78} ; R \rangle$	118/126	126/2036
$\langle x_1, x_{27}, x_{40}, x_{46}, x_{47}, x_{76} ; R \rangle$	79/83	83/2036
$\langle x_1, x_{27}, x_{40}, x_{41}, x_{47}, x_{52}, x_{76} ; R \rangle$	28/28	28/2036

Table 4.14 Variables Selected for Pass Generation

Variable	Description
x_1	Duration (in frames) of the pass
x_{27}	Number of players closer to the ball than the closest opponent
x_{40}	Number of players closer to the home goal than the 3rd closest opponent
x_{41}	Number of players closer to the home goal than the 4th closest opponent
x_{47}	Number of players closer to the home goal than the 10th closest opponent
x_{73}	Number of opponent players in spaces neighbouring the receiving player
x_{76}	Area controlled by the receiving player
x_{89}	Number of players closer to the receiving player than the 10th closest opponent

The variable x_{73} is particularly significant in the results, being able to differentiate between successful and unsuccessful passes on 70% of the occasions where it occurs. It corresponds to the number of opponent players in spaces neighbouring the receiving player, and has a value of < 1.26. Interestingly, x_{73} does not appear in the other significant hubs. It occurs in hubs with a slightly lower

specificity for every size of hub than those listed. In all hubs of dimension > 0 the pair $\langle x_1, x_{27} \rangle$ predominates with a combined specificity of 86%. In comparison, the pair $\langle x_{73}, x_{74} \rangle$ has a specificity of 75%.

In this case, the small training set did not generate some of the more significant hub combinations. However, the method has still identified combinations which give similar specificity and broadness, by adding extra variables to the hub.

As the specificity of a hub increases, its broadness decreases. The hub becomes more detailed and less representative of the general case. Hubs begin to describe particular instances of the concept, which may be distinct from one another. For example, there might be passes which are used to promote position, passes which are used to set up particular events, passes that are forced, or passes used to move a ball away from a dangerous position. Each will have its own hub with both high specificity and broadness. This, in effect, identifies further concepts.

4.8 Statistical Analysis of Hub Occurrence

As we have stated, hubs tend to arise in relatively small frequency or size. One concern is that some or all of the results may appear at random. Can we rely upon the results obtained via this method to contain useful data?

In an attempt to answer this question, we will endeavour to determine the statistical likelihood of the hubs obtained in the pass concept analysis occurring at random. A major problem with this is that many of the variables are linked to one another, and it is difficult to assess the probability of their occurrence in isolation, let alone in sets. Hubs also appear with many permutations of variables, and it is

unlikely that two dissimilar hubs of equal dimension will have similar probabilities of occurring.

To tackle this, we estimate the probability of each variable occurring by measuring its appearance in the desired amount, over 90 pass primitives. Where there is obvious reliance between variables, we measure the probability of each set of linked variables occurring together.

Searching through the list of abstracted hubs we find the most frequently occurring, and use these to represent the most probable variable combinations for each size of hub.

Using our estimates for the probability of each variable, and set of variables, occurring, we then generate a probability for each of our most likely hubs. For the most common hubs of dimension 0-39, we have estimated the probability of them naturally occurring. This is plotted as the solid line in figure 4.16.

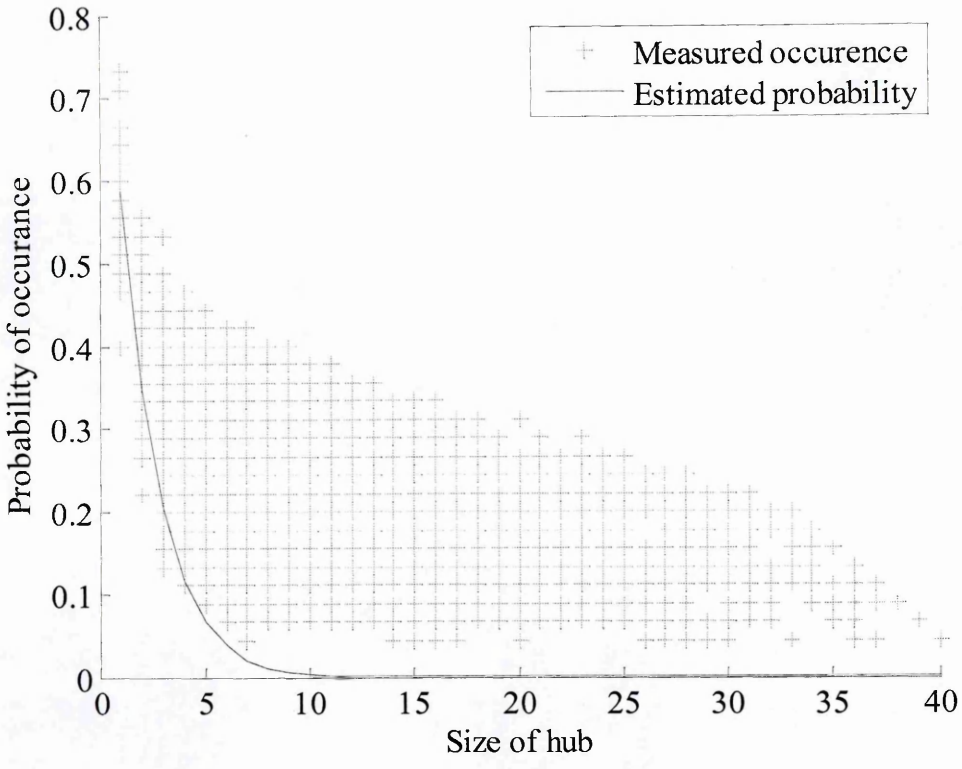


Figure 4.16 Probability of Hubs Occurring Naturally

Each cross represents one or more measured hubs, with those above the line being unlikely to occur naturally.

Superimposed on this plot are all the maximal hubs measured in section 4.7, plotted as points corresponding to their size and frequency of occurrence, measured as a fraction of the total number of desirable primitives.

It is clear from this estimation, that the majority of measured hubs are unlikely to appear naturally. Some small, infrequent hubs may appear by random, but we are not focussing on these. Our measured hubs have more relevance the further they lie above the estimated probability curve.

It should be noted that many more hubs will appear below the curve. These do not show up, due to the heuristics used in our search algorithms.

4.9 Propagation of Variables and Hubs across Levels

Having identified relations of variables to define a series of concepts, we now return to our multilevel structure to investigate the propagation of variables across related concepts.

Recall our ideas of structure in football plays generated in section 4.2. Figure 4.8 showed how the eight play concepts are related across two levels. We have shown that concepts at each level can be generated from a set of variables, and that the same variables can be reused at multiple levels. If there is a link between concepts within the multilevel structure, then surely there must also be a link between the variables associated with those concepts. More importantly, there should be a link between the sequences of variables appearing in each concept. If a concept at level N is related to a concept at level $N + 1$, then the set of variables describing the concept at level N should also be part of the description of the concept at level $N + 1$. In other words, the hubs used to describe the concept must flow through the multilevel structure. The exception to this case will be when multiple concepts with conflicting variables combine to form a higher level concept. However, even then, it is likely that some of the lower level hubs will emerge in the higher level concept.

Consider figure 4.17. This is a Venn diagram of our robot football structure. It shows the relation between variables and concepts. The circle represents the strategy concept. It is split into half vertically to represent the two opposite plays *IN HOME* and *IN AWAY*, and split horizontally to represent the plays *IN POSSESSION* and *OUT OF POSSESSION*. The four quarters represent the combination plays *IHIP*, *IHOP*, *IAIP*, and *IAOP*.

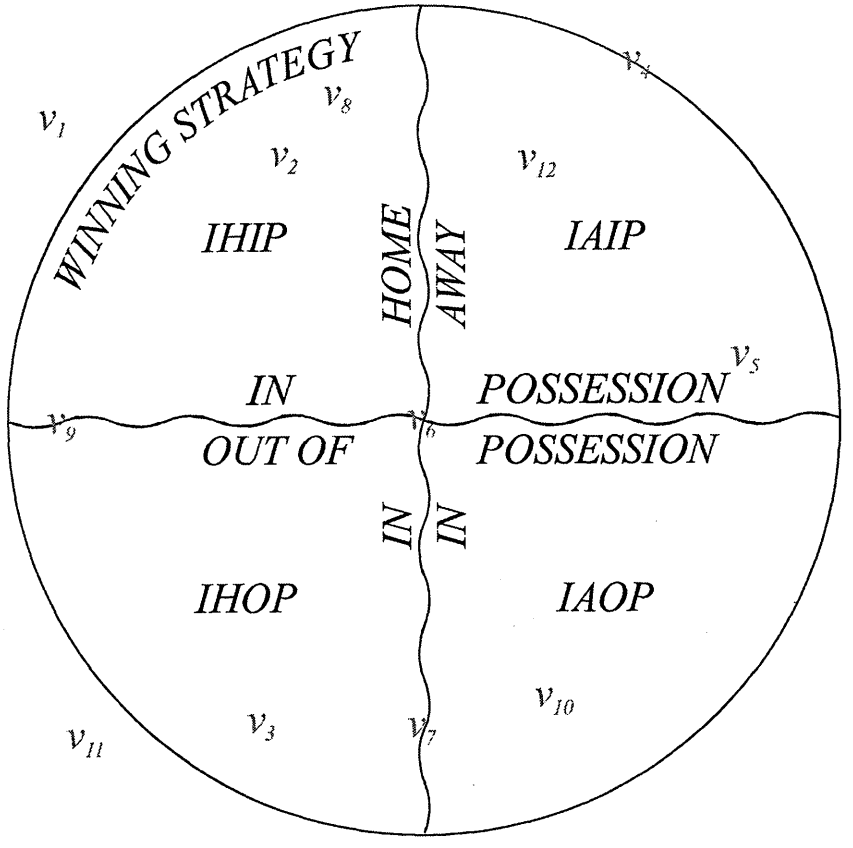


Figure 4.17 Venn Diagram Showing the Relation between Variables and Concepts in the Football Structure

Variables are placed on the diagram in accordance with their relation to each concept. If a variable lies within the region bound by a concept, then it is significant to that concept. If it falls on a boundary line, then it is significant to concepts on either side of that line. Consequently, a variable which is significant to all the concepts will fall at the centre of the circle, and cross into each segment.

Representing the concepts and variables in this way highlights a logical relation between concepts. Variables, or rather sets of variables, occurring in *IN HOME*, also appear in *IHIP* OR *IHOP*. Also, variables and sets occurring in *IHIP* appear in *IN HOME* AND *IN POSSESSION*. We can state that for this particular set of concepts, the relation mapping pairs of level N concepts to a level $N + 1$ concept is an OR-aggregation, whereas the mapping between two level $N + 1$ concepts down

to a single level N concept is an AND-aggregation. We test this hypothesis experimentally.

4.9.1 Experiment

We begin by selecting the eight plays as our concepts of interest. In this experiment we will investigate the hypothesis that there is a logical OR relation between pairs of level N hubs, and hubs present at level $N + 1$. To simplify matters, we will only focus on our classifier hubs, i.e. those which differentiate between desirable and undesirable primitives, not those common to both types. The hubs for both levels have already been found in earlier sections of this thesis.

We will call the two level N concepts α and β, and the level $N + 1$ concept γ.

Firstly we check all the available variables for their significance. By this we mean those variables which appear in hubs corresponding to each of the three concepts. Variables which appear in only one of the level N hubs may, or may not, appear at level $N + 1$.

Next, we test these significant variables for any conflicts. By this we mean any variables which occur on the same side of the global mean in hubs of α and β, but the opposite side in γ. If a variable is on opposite sides of the mean in α and β, then we assume it is feasible for it to occur either side in γ. However, if a variable occurs to the same side of the mean in both α and β, then we presume it should also occur that side in γ. Any conflicts indicate that variables can change as they migrate through the multilevel structure, and show a flaw in the hypothesis.

We continue by creating all the possible OR-aggregated combinations of hubs of α and β. According to our theory, these should correspond to the hubs in γ. We

then search for the aggregate hub with the closest match to each hub appearing in γ. The sets of variables common to each are used as a measure of the similarity of the hubs across the two levels. Here, we shall measure the similarity as a percentage of variables matched between the hubs. For example, a γ hub containing five variables is compared to a 6 variable aggregate hub. If the aggregate hub contains all five variables from the γ hub, then there is an 80% match, having a one variable, or 20% error. A γ hub of 6 variables paired with a 3 variable aggregate hub, all of which appear in the γ hub, is a 50% match. The closest match between aggregate hub and γ hub is recorded.

A problem with this method arises when we fail to include level N concepts, which are related to our higher level concept. These can cause additional variables and sets to be introduced, or cancel out variables and sets which already exist. We limit this in our experiments by choosing bipolar plays, which together describe the whole of a strategy.

4.9.2 Results and Discussion

Table 4.15 shows the initial findings of the variable analysis. We split the 66 variables introduced in section 4.4 into three groups. For each set of concepts, the insignificant variables are ones which do not turn up in any of the concepts. These identify structures which have no discernable effect on whether the primitives are desirable or undesirable. The significant variables are variables which are used to classify primitives in all three concepts. At this stage, we ignore whether they are valued above or below the global mean. The partially significant variables are those which turn up as classifiers in only one or two of the concepts.

Table 4.15 Mapping of Variables through the Multilevel Structure

Level $N+1$ concept (γ)	Level N concept (α)	Level N concept (β)	Number of insignificant variables	Number of partially significant variables	Number of significant variables	Number of conflicting variables
IN HOME	*IHIP*	*IHOP*	21	19	26	0
IN AWAY	*IAIP*	*IAOP*	13	21	32	0
IN POSSESSION	*IHIP*	*IAIP*	16	22	28	0
OUT OF POSSESSION	*IAIP*	*IAOP*	18	18	30	0

For our hypothesis to hold, we would expect that hubs turning up at level $N+1$ would be mainly composed of the significant variables, but that some partially significant variables may also exist. It is interesting to note that for each set of concepts, the number of significant variables is the largest, indicating that we have selected a good set of variables to describe these concepts. The final column in table 4.15 shows the number of conflicting variables. We find no conflicts, indicating that each significant variable falls the same side of the mean in γ as it does in at least one of α or β. This indicates our hypothesis is still valid at this point.

The preliminary results for our hub comparison are shown in table 4.16. Each row represents the level $N + 1$ hubs corresponding to the named concepts. The columns represent the percentage of each hub matched by the closest aggregate hub formed from the appropriate level N hubs. The values in each cell indicate how many of the level $N + 1$ hubs can be matched with that precision.

Table 4.16 Accuracy of Measured Aggregate Hubs

Concept	Number of γ hubs described with the specified accuracy (%)						
	50-60	60-70	70-80	80-85	85-90	90-95	95-100
IN HOME	0	0	0	0	0	7	66
IN AWAY	0	0	1	1	2	17	99
IN POSSESSION	1	2	9	12	14	16	10
OUT OF POSSESSION	0	1	3	11	8	18	56

We can see from these results that the majority of level $N + 1$ hubs can be almost perfectly described by OR-aggregations of level N concepts. In all, 228 of the 355 level $N + 1$ hubs can be described with 100% accuracy. The largest discrepancies are found in the hubs relating to the *IN POSSESSION* play. Only 9 of its 63 hubs can be exactly produced by combinations of lower level hubs. A possible explanation for this could be that our variables are simply not good at describing this play. This is supported by the fact that the play concept has the fewest hubs associated with it, and many of the variables used are only partially significant. If we ignore the partially significant variables in our analysis, the results are as shown in table 4.17.

Table 4.17 Accuracy of Refined Aggregate Hubs

Concepts	Number of γ hubs described with the specified accuracy (%)			
	80-85	85-90	90-95	95-100
IN HOME	0	0	5	68
IN AWAY	1	4	0	115
IN POSSESSION	0	1	1	61
OUT OF POSSESSION	0	0	6	91

As we can see from these results, removing the partially significant variables has done little to affect the hubs in the *IN HOME* and *IN AWAY* concepts, but has dramatically improved the results pertaining to the two possession concepts.

The partially significant variables appear mainly from 2 sources. The first is from the indifferent primitives. In this case, the global mean can be affected such that the variable does not register as significant in one of the concepts. The second is when a variable is a classifier for all primitives within a concept, and it appears as common to desirable and undesirable sets. Since we have ignored common variables, the occurrence of either of the above will mean a variable is ignored in the associated concept. If a variable is missing from one of the hubs, then this will obviously prevent it from matching a hub where the variable is existent.

In general, there appears to be a strong link between the level N and $N + 1$ hubs, which supports our hypothesis. There are, of course, many combinations of level N hubs which do not describe any level $N + 1$ hubs. These combinations indicate plays which should not be combined, or do not work well together. There are also some hubs which cannot be fully described. This may be due to poorly selected variables, chance occurrences in the plays, or measurement errors.

Although these results support our hypothesis about the migration of hubs between levels of the multilevel structure, they do raise an interesting point. In this analysis, we have begun with hubs already measured for each concept. What we cannot do at this stage is use information about the hubs at level N to generate, from scratch, the hubs at level $N + 1$. This would require knowing the relation describing the association between the hubs of the concepts.

If we can migrate hubs up this multilevel structure using an OR-aggregation, it is probable we can also migrate down, as we have postulated, using an AND-aggregation. This should be possible using our concepts due to the way in which they have been defined, and their relationship according to the Venn diagram shown in figure 4.17.

Finally, we should reiterate that all the concepts at level N that interact to form a concept at level $N + 1$ are required to be able to generate possible hubs at level $N+1$. In this example, we have made this possible by selecting bipolar concepts at level N which between them describe the whole of the concept at level $N + 1$.

4.10 Applicability of Variables and Hubs across Leagues

Consider figure 4.18. Mirosot and RoboCup are two distinct types of robot football, just as *IHIP* and *IHOP* are two types of *IN HOME* plays. We hypothesise that there may exist concepts common to both RoboCup and Mirosot, which can be used to describe a new concept of *ROBOT FOOTBALL*.

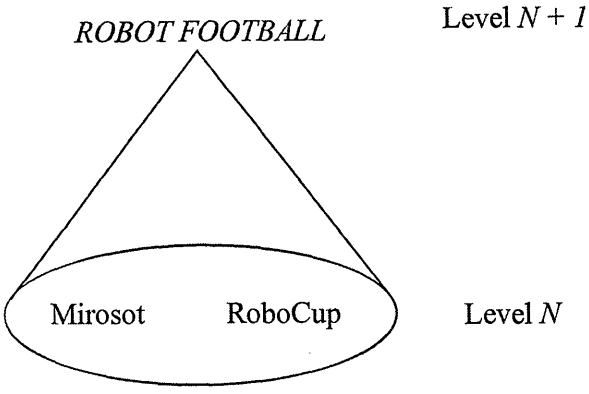


Figure 4.18 Mapping of Robot Football Leagues into a General Robot Football Concept

Although each are played with different numbers of players, on different sized pitches, both are team games played on a 2-dimensional rectangular pitch with the aim of putting a ball into the opponent goal. Some types of structure should, therefore, exist in both game concepts. Taking this a step further, we can see that these structures may also appear in human football and other games.

Plato philosophised that there existed a set of *forms*, which described perfect examples of every object and concept we could conceive (Ross, 1951). He supposed there was a form for 'beauty', and a form for 'circle', and that every time we recognised a particular feature, object, or meaning, what we were actually doing was recognising the similarity of that thing to one of the forms. There is an ideal form for everything, and we identify objects and concepts in the real world by relating them to these ideas. What is more, there may be forms with a greater or lesser level of description, for similar ideas. As well as being a form for 'circle', there may also be one for 'blue circle'.

Our ideas regarding a multilevel structure of concepts ties in with this philosophy. Each concept is itself a form, describing a particular idea in robot football. We are trying to find a measurable representation of these forms, by

identifying variables which can be used to describe them. It should not be difficult to see how RoboCup and Mirosot are both forms of robot football, which itself is a form of football, itself a form of team game, itself a form of competition, and so on.

In this final section, we will endeavour to show links between Mirosot and RoboCup using the same techniques we have demonstrated throughout this chapter. If we can successfully achieve this, then it should support our argument for extension of the multilevel structure beyond the confines of our limited RoboCup football data.

We reiterate that the two games of RoboCup Simulation League and Mirosot are very different. The RoboCup game is a simulation, and therefore less complex, than Mirosot, which is played with real robots. The RoboCup game uses 11 omnidirectional robot agents each of which can control the ball by holding or kicking it. Mirosot, on the other hand, is a 5-a-side game using differential drive robots, which can exert only limited control over the ball. The robots can only push the ball using a shallow scoop, which makes passing, catching, or kicking the ball in a precise direction very difficult.

In this experiment we will endeavour to find relations of variables which are relevant to similarly defined concepts in both types of robot football. Since passing is very infrequent, and difficult, in Mirosot, we shall examine the play concepts of *IN HOME* and *IN AWAY*. From our experience in robot football, we know these are commonly used to define strategies in both games.

4.10.1 Experiment

Until now we have confined our experiments to the readily available RoboCup Simulation League data. Before proceeding, we need to collect a set of data relating to Mirosot robot football. Data from Mirosot matches is not recorded and published, as is the RoboCup simulation data, so we must generate our own.

We begin by generating seven Mirosot strategies, and playing and recording ten games. Four of these strategies are inherited from previous competitions, whilst the remaining three are built from scratch using the traditional role based approach. Using the results, we repeat the analysis described in section 4.4 to generate hubs relating to the two concepts, but removing or scaling variables which correspond to the number of players on a team.

To allow a direct comparison between the hubs for these concepts across both types of robot football, we must scale down any variables corresponding to 11 players per team in the RoboCup data to their equivalent 5-a-side values. This should be acceptable, as it is comparable to using a ratio based variable of players to team size in the initial analysis, which would provide the same results.

As in section 4.4, we compare each set of variables and remove any which fall on opposite sides of the mean in each type of football. These indicate specific differences between the two game types. It should be noted that the mean for each variable will be different for each type of football. If we valued our variables over a limited range, as discussed in section 4.3.2, then we would need to look for overlaps in the ranges to determine whether each variable was significant to both game types. However, since we have measured the significance of our variables over infinite ranges, they will overlap if both occur to a greater or lesser extent.

We conclude by comparing the hubs occurring for each game type, for each concept. Any variable sets which occur in both the RoboCup and Mirosot data are recorded and their sizes and frequencies measured.

4.10.2 Results and Discussion

For the *IN HOME* concept, there are 8 desirable primitives in the RoboCup data, and 6 desirable primitives in the Mirosot data. These give rise to 73 RoboCup hubs, and 44 Mirosot hubs. Cross examining these, we find 82 unique hubs common to both types. A selection of these is given in table 4.18.

Table 4.18 Common Defensive Play Hubs across Football Leagues

Dimension of intersection	Maximum hub dimension	Largest hub
13	0	$\langle x_4 ; R \rangle$
11	1	$\langle x_2, x_4 ; R \rangle$
10	2	$\langle x_2, x_4, x_{18} ; R \rangle$
6	3	$\langle x_2, x_4, x_{18}, x_{31} ; R \rangle$
4	4	$\langle x_2, x_4, x_{18}, x_{28}, x_{32} ; R \rangle$
3	5	$\langle x_4, x_{18}, x_{20}, x_{28}, x_{31}, x_{32} ; R \rangle$
2	6	$\langle x_2, x_4, x_{18}, x_{20}, x_{28}, x_{31}, x_{32} ; R \rangle$
1	7	$\langle x_2, x_4, x_{17}, x_{18}, x_{20}, x_{28}, x_{31}, x_{32} ; R \rangle$

Variable x_4 appears in all of the primitives, making it a perfect classifier. The set $\langle x_2, x_4, x_{18} \rangle$ is also very strong, appearing in the largest maximal rectangles. Given that only nine of our measured variables are common to both the Mirosot and RoboCup hubs, these relationships are considerably strong. The significant variables and their values are shown in table 4.19. Here the average values listed are

the larger of the RoboCup and Mirosot values for the greater than relationships, and the smaller of the two for the less than relation.

Table 4.19 Common Defensive Variables and Values across Football Leagues

Variable	Meaning	Relation	Value
x_2	Number of frames analysed	<	2229.25
x_4	Total number of passes	<	37.8
x_{17}	Percentage of unsuccessful opponent passes	>	70.9
x_{18}	Number of home passes	<	22.5
x_{20}	Number of shots at opponent goal	<	0.375
x_{28}	Players closer to ball than closest opponent	<	0.87
x_{30}	Players closer to ball than third closest opponent	<	2.88
x_{31}	Players closer to ball than fourth closest opponent	<	3.60
x_{32}	Players closer to ball than fifth closest opponent	<	4.14

From these results it is interesting to note that in defensive plays, the winning teams do not force as many players around the ball, as do the losing teams.

There are also 8 desirable primitives in the RoboCup data, and 6 desirable primitives in the Mirosot data, for the *IN AWAY* concept. These give rise to 121 RoboCup hubs, and 46 Mirosot hubs, respectively. Cross examining these, we find 182 unique hubs common to both types. A selection of these is given in table 4.20.

Table 4.20 Common Offensive Play Hubs across Football Leagues

Dimension of intersection	Maximum hub dimension	Largest hub
11	0	$\langle x_{14} ; R \rangle$
10	1	$\langle x_2, x_{21} ; R \rangle$
9	3	$\langle x_8, x_{21}, x_{30}, x_{31} ; R \rangle$
8	4	$\langle x_8, x_{18}, x_{21}, x_{30}, x_{31} ; R \rangle$
7	6	$\langle x_8, x_{14}, x_{18}, x_{21}, x_{29}, x_{30}, x_{31} ; R \rangle$
6	8	$\langle x_8, x_9, x_{14}, x_{18}, x_{21}, x_{28}, x_{29}, x_{30}, x_{31} ; R \rangle$
5	9	$\langle x_8, x_9, x_{10}, x_{14}, x_{18}, x_{21}, x_{28}, x_{29}, x_{30}, x_{31} ; R \rangle$
4	10	$\langle x_2, x_8, x_9, x_{10}, x_{14}, x_{18}, x_{21}, x_{28}, x_{29}, x_{30}, x_{31} ; R \rangle$
3	11	$\langle x_8, x_9, x_{10}, x_{14}, x_{18}, x_{20}, x_{21}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32} ; R \rangle$
2	12	$\langle x_8, x_9, x_{10}, x_{12}, x_{14}, x_{18}, x_{20}, x_{21}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32} ; R \rangle$
1	14	$\langle x_2, x_4, x_8, x_9, x_{10}, x_{12}, x_{14}, x_{18}, x_{20}, x_{21}, x_{28}, x_{29}, x_{30}, x_{31}, x_{32} ; R \rangle$

Here we see yet another good correlation. From our set of variables, 15 appear in similar amounts in both game types, all of which appear in one primitive from each set. Variable x_{21} is the most common, appearing 139 of the common hubs. Here, the set $\langle x_8, x_{21}, x_{30}, x_{31} \rangle$ is particularly strong. The significant variables and their values are shown in table 4.21.

Table 4.21 Common Offensive Variables and Values across Football Leagues

Variable	Meaning	Relation	Value
x_2	Number of frames analysed	>	4576.1
x_4	Total number of passes	>	176.3
x_8	Percentage of player turns spent closer to the home goal than the ball	>	74.5
x_9	Percentage of player turns spent further from the away goal than the ball	>	57.1
x_{10}	Percentage of player turns spent inside the triangle described by the ball and the home goal posts	>	18.3
x_{12}	Standard deviation in home player areas (%)	>	9.2
x_{14}	Percentage of successful home passes	>	65.3
x_{18}	Number of home passes	>	110.8
x_{20}	Number of shots at opponent goal	>	4.8
x_{21}	Percentage of shots becoming goals	>	39.1
x_{28}	Players closer to ball than closest opponent	<	0.6
x_{29}	Players closer to ball than second closest opponent	<	1.3
x_{30}	Players closer to ball than third closest opponent	<	2.1
x_{31}	Players closer to ball than fourth closest opponent	<	2.9
x_{32}	Players closer to ball than fifth closest opponent	<	3.6

These results show that the better teams in both types of robot football spend longer with the ball in the opponents half and, whilst there, they pass the ball more, pass more successfully, shoot more, and convert more shots into goals.

Having found that there are common hubs relating to similar concepts in both types of robot football, we can suggest that they may also apply to similar concepts in other robot football games, as they contribute to two generic play concepts.

4.11 Computational Efficiency

Performing the star-hub analysis is a computationally intensive process. In section 4.3.4 we computed the hubs formed by pairs and triples of variables. The set of 85 variables generated 3570 hubs of dimension 1 and 98770 hubs of dimension 2. In general, the number of subsets of N variables is of the order 2^N, which is a high level of computational complexity.

For a subset of size q, figure 4.19 indicates the number of possible combinations that can be obtained from a set of N variables. The set of combinations in the central region of the graph are intractable. The entire set of subsets can only be computed in the tail regions of the graph as $q \to 0$ or $q \to N$. However, these regions contain some of the most useful subsets.

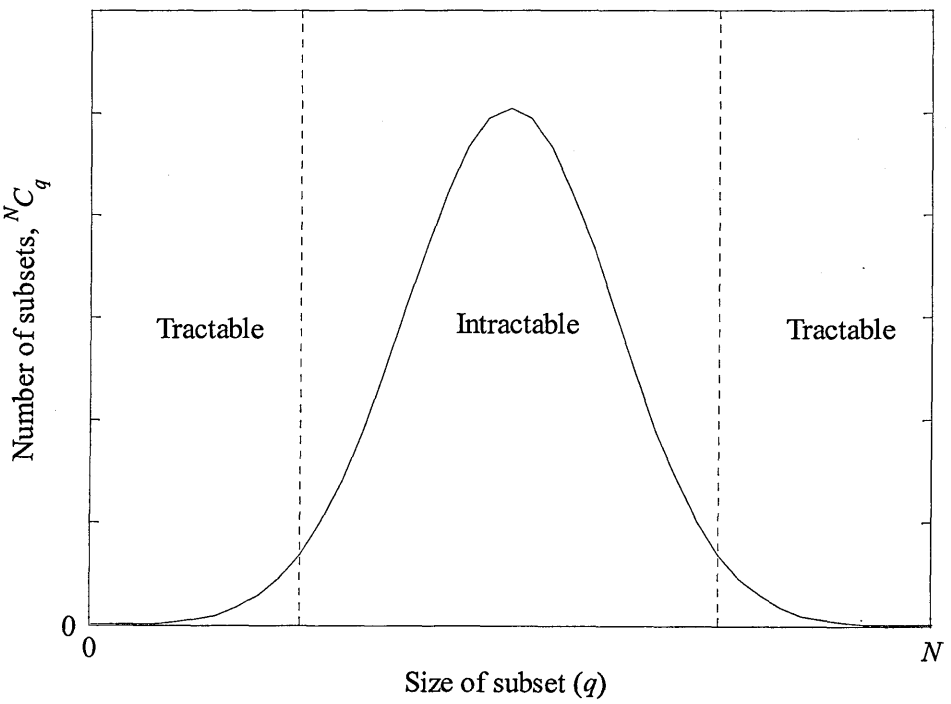


Figure 4.19 Tractability of Subsets

As shown in section 4.3.4, the ability of a hub to classify a set of data, in terms of the ration of successful to unsuccessful classifications, increases with the number of variables incorporated into the hub. However, the increase reduces as more variables are introduced into the hub. For the problems introduced in this thesis, a small number of variables have been sufficient to achieve a 100% successful classification rate. Furthermore, as the number of variables included in the classification increases, so too does the number of unclassifiable samples, providing a counter argument to calculating much larger hubs.

To enable the computation of the more important hubs from the entire range of subsets, we have implemented a number of heuristics as described below:

In our algorithms, we do not search for every possible subset. Rather, we focus on only finding the hubs corresponding to the intersections of a small number of simplices. For the majority of experiments run in this chapter, this was sufficient to enable the analysis. For example, in section 4.7 we generated just 9240 hubs out of a possible 2^{100} from 25 primitives of 100 variables.

For larger data sets, the abstraction of hubs did provide some difficulty. In these situations, we can reduce N by omitting variables with a lesser significance from the analysis. The choice of variables to exclude could be based on the measures of specificity and broadness introduced in section 4.3.4, or on constraining the size and frequency of desired hubs.

Further reduction can be achieved by using a subset of the available primitives as training data. By reducing the number of primitives analysed, the number of simplices, stars, and intersections are also reduced, whilst still being likely to capture

the common hubs. This is the case when using a small set of primitives to generate hubs to classify further primitives.

The use of comparing averages to identify desirable variables has also provided a considerable reduction in computation from the methods used in (Iravani, 2005a). As stated in section 4.3.5, a single variable as identified in this work was previously represented by multiple binary variables. The example given shows a 5.5 times reduction in the number of variables. We have further reduced the value of N by using the comparison of averages to identify only the most desirable variables for inclusion in our hub search.

The analysis outlined in this chapter has been performed off line as a separate learning process. By generating hubs in this manner, the computation time required is not a major issue. The benefit is that by abstracting the hubs off line we can identify tasks and plans prior to robots undertaking a task. This removes much of the decision making and planning at run time, to the relatively simple selection and reconstruction of hubs from a predefined list. This is demonstrated in the following chapter, where a team of real robots is controlled, in real time, in the dynamic football environment, using the hubs abstracted here.

4.12 Summary

Robot football has a complex multidimensional and multilevel structure. To analyse such a system and abstract the sub tasks required to complete the objective requires a new type of architecture. In this chapter we have introduced such an architecture, based the theory of concept generation. From a set of variables, which could possibly be used to describe the system, our architecture can identify the relations

which describe higher level concepts. These relations can be turned into hypotheses for classifying further data, or representatives for reconstructing the emergent properties of concepts. We have demonstrated the processes at various levels within a multilevel structure, and shown how the relations map between similar concepts, and across levels in the structure. It is important to note that although the method is used to abstract relationships from recorded data, we are not simply trying to recreate existing rule based systems, but identify the more subtle emergent properties.

The architecture was introduced in section 4.1. Each segment of the architecture was described, along with an explanation of how it might be applied to a generic problem. The process attempts to find relations of variables commonly found in desirable primitives to describe a concept. For a given concept, a number of primitives, examples of the concept, are identified in recorded data. These are described in terms of a set of chosen variables. After classifying primitives into desirable and undesirable sets, we can identify which variables commonly occur in the more desirable instances of our concept. Finally, we find the relations of variables, or hubs, which occur in the desirable primitives.

We propose a new method for classification of variables, based on comparison of averages. This method differentiated variables according to the desirability of the primitive, based on average values. In previous experiments, variables had been ordered into binary sets depending on whether they fulfilled some arbitrarily chosen specification. Our method has the advantage of generating its own, more significant, boundary values for specification, and allows for continuous variables to be analysed without loosing the information stored in them by the grouping into binary sets.

To run our abstraction architecture, we first need to identify possible concepts. We have shown how robot football has a complex multilevel structure, consisting of concepts which we have grouped into sets representing ideas of strategies, plays, tactics and skills. We have generated a number of concepts which could be used to describe the game, and shown how they might be connected into a multilevel structure. The multidimensional structure has been shown by considering 66 variables, which are used to represent the concepts at each level in the multilevel structure. Using these two sets of structures as a framework, we set out to gradually build up a representation of the game of robot football.

We have shown how sets of variables combine to describe the various concepts in our multilevel structure. We have repeated the technique to build strategy, play, and tactic concepts. Previous work has been confined to the analysis of concepts on a single level in the structure, whereas here we investigate its application to multiple levels, showing how the same techniques and sets of variables can be equally relevant at each level. The definition of a desirable primitive for defining these concepts has also been investigated. We show, for the first time, how the meaning of the concept is affected by changing the duration over which the primitive is measured, or by changing the definition of success used to identify the primitives. We show this by measuring primitives over whole matches, and specific events, and by measuring different criteria for success.

We have also indicated how primitives can be classed to give concepts different meanings, whether they relate to general or winning structures. We have shown how some variables appear as constants across all strategies, or plays, and that these are fundamental indications of how the game is played. Other variables,

which appear to a greater or lesser extent in the desirable concepts, show how to play the game well. More importantly, we show how specific sets of these structures occur together, and are reliant on each other.

To verify our methods, we have measured the statistical likelihood of the measured hubs occurring randomly. A major concern was that given the number of variables and primitives being considered, it was possible that the hubs being recorded were occurring by chance. Although there are many hubs which could be classed as chance occurrences, our analysis suggests there are also many which are highly unlikely to occur by chance. These hubs represent the largest and most frequent common structures between primitives, and give the strongest representatives for our concepts.

Another new theory was generated in this chapter, and supported by experimental results. This related to the idea that given the relation of concepts between levels in the structure, there must also be a relation between structures of variables existing at each level. We hypothesised that by combining hubs on one level using logical operators, we could generate the hubs occurring at a level above, or below. Using our set of 8 play concepts, we showed how these relations could be represented diagrammatically using Venn diagrams, and suggested possible logical relations between trios of concepts spanning multiple levels. Our experimental results showed a strong correlation to back up our theory.

Finally, we have shown how common sets can also be found between different types of robot football. From our understanding of hierarchies, we hypothesised that there is a 'form' of robot football encompassing the football variants, and that there may be higher forms of 'football', 'team games' etc. We generated data pertaining

to Mirosot robot football using robots constructed for the purpose. By repeating our analysis on this data, we compared structures found occurring in the play concepts common to both Mirosot and the RoboCup Simulation League. We found that certain structures appear in both sets, and can state with some confidence that there are links between the two game types, and that these might represent play concepts in a wider variety of robot football types. These structures are evident, despite the two game types being different in many ways.

A limiting factor in this analysis is the selection of variables, and concepts, and identification of the multilevel structure. In the example presented here, these decisions have been made based on our knowledge of the game of football, and structures which we perceive to be relevant and useful. It is desirable that this whole procedure be automated, so that it can be used more efficiently as a learning tool for teaching robots to work together in such complex environments. However, this will require the system to identify its own variables, concepts, hierarchies and primitives, which may be difficult.

This chapter has detailed a method for generating a multilevel structural representation of the game of robot football from a set of measured variables. We have introduced a new method in concept generation using compared averages to classify variables according to their desirability, and used this to generate concepts relating to tactics, play, and strategies. Furthermore, we have shown how primitives can be classified in different ways to generate alternative concepts and have demonstrated how the generated hubs are statistically significant. Finally, we have revealed how variables and their relations link concepts on different levels, and between different varieties of football. The hubs generated in this chapter identify

structures of variables which together create the emergent properties evident in the related concepts.

Chapter 5

Strategy Generation and Performance Evaluation on Real Robots

So far, in this thesis, we have introduced robot football as a complex multilevel and multidimensional system. In chapter 3 we investigated the properties of this system, and showed how spatial structures were integral to the game. In chapter 4 we examined the importance of these structures in more detail, showing how sets of particular structures were more prevalent in winning teams. Furthermore, we showed how these sets formed concepts, which themselves formed a multilevel structure of tactics, plays, and strategies, which could be used to describe the game of robot football. In this last chapter of experiments we will carry forward all these ideas and, using the results from the previous chapters, attempt to develop a controller capable of instructing our robots how to react in the complex football environment.

We generated a possible multilevel strategy structure in section 4.2 comprising of concepts defined by relations of variables. For each concept, we generated a number of sets of variables, which could be used as representatives for the concept. Each set was calculated experimentally, and is an abstraction of the concept from real world data. If we can design a controller to recreate these structures over their valid periods, and implement it on our robots, then we propose that a football playing strategy will emerge.

A strategy formed in this way will be a new development in robot controllers, consisting of experimentally abstracted objectives, and limited in its use of weakly

defined tasks. Being a multilevel and multidimensional structure, it should also be better suited to complex tasks. Furthermore, since many of the structures represented by the variables are functions of opponent positions, the emerging strategy should adapt to rival strategies, and be relevant against all opponents.

5.1 Control Architecture

Figure 5.1 depicts our control architecture. It describes the process of converting an abstracted strategy, found using the analysis in chapter 4, into a controller for a set of robots. We input a structure of representatives we desire to reconstruct, a list of the variables used to describe these representatives, and data pertaining to the current state of the system. The controller identifies the representative which most closely matches the current state of the system, and produces a list of actions required to reproduce the structures described by that representative. By reproducing these structures, we recreate the interactions which form the emergent system behaviour.

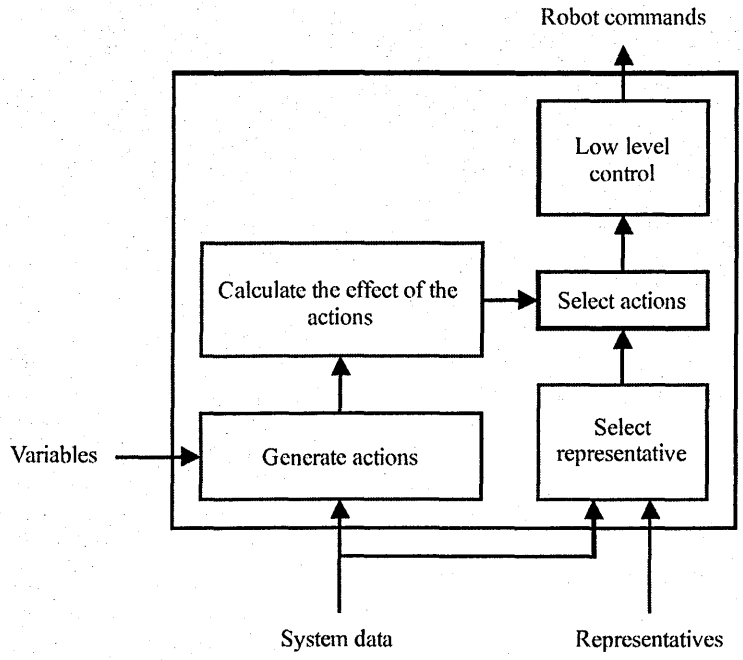


Figure 5.1 Block Diagram of the Proposed Control Architecture

Representatives, derived using our abstraction architecture, and data, recorded through available sensors, are fed into the control architecture. Based on these inputs the controller selects the representative which most closely matches the state of the system (recall that each representative contains a description of the relationship it describes, which includes information on the state of the system at the time the representative is valid). If a multilevel structure of representatives is defined, then the each representative will point to a number of lower level representatives. In this way, the controller can identify all the low level structures required to recreate the highest level representative.

The individual variables within the representative describe structures which interact to form the governing concept. By recreating these structures, we recreate the concept. The next phase of the control, therefore, is to turn these variables into actions. Each variable inputted into the abstraction architecture contained a description of a structure to measure. In this section we use that description to define the action to be performed.

If there are more actions than robots, then action sets must be generated, so that a single robot can recreate a number of the required variables. The controller searches for possible action groupings and outputs them a list. Each grouping is assigned a set of values, which identify the effect of the action set on reconstructing the variables.

The controller searches through these groupings to find the set of action groups that will most closely reconstruct the variables in the representative. This set of actions is then sent to the robots to undertake. The low level control decides which robots perform which tasks, and handles the undertaking of each task.

In the following example we search for actions corresponding to all the variables listed in the representatives. The efficiency of the controller could be increased by only searching for actions relating to variables in the active representative.

5.2 Strategy Generation

The control architecture requires a set of representatives to recreate. For these experiments, we are interested in forming a strategy to run on real robots. This section describes how we form a strategy of representatives, based on the hubs abstracted in chapter 4.

We shall be using our Mirosot system as the main test bed, but will also use the Simurosot simulator as a comparison, to see how our controllers operate in both real and simulated environments. The performance of our Mirosot system has been measured, and details of the experiments can be found in appendix A. From the results we know that our control over the system is limited. Therefore, we will focus on recreating only the higher level concepts relating to the organisation of robots during different plays. We will avoid the lower tactical and skills levels, since these require a greater level of control over the robots, which our lower level algorithms cannot currently supply.

The hubs relating to the Mirosot matches are smaller and less frequent than those for the RoboCup matches, indicating a lesser degree of separation between the good and bad strategies. This may be due the greater ball control available in the RoboCup Simulation League, or the fact that all our Mirosot matches have been played on the same set of robots, and are limited by the low level control. However,

as we have demonstrated, there are links between the two types of robot football. We will therefore use the more abundant and descriptive RoboCup hubs to provide representatives to generate our abstracted strategies. Mirosot strategies, as we stated in chapter 2, are usually based on the two play concepts: *IN HOME* and *IN AWAY*, therefore we will use RoboCup representatives of these plays to form our strategies.

The limitations of the Mirosot robots prevent us from reliably using ball controlling behaviours, and so we shall focus on developing a purely formational strategy. We do not see this as impeding our research, as the primary aim is to show that it is possible to recreate concepts using representatives, rather than to play football. If we improve the low level control, and measure the concepts relating to ball manipulation, then we will be in a position to generate a more complete strategy. The main issue is whether these concepts can be converted into control algorithms. We can show this by attempting to recreate the measured play hubs under similar conditions in a football match. If we can sufficiently reproduce the structures representing the variables in these hubs, then we have achieved our task.

Of all the variables we measured in section 4.5, some are easily controllable, whereas others are purely observable. For instance, we can measure, but not control, the percentage of shots which become goals. Ignoring variables which are purely observable, we select five of the most significant hubs from the *IN HOME* and *IN AWAY* play concepts measured in section 4.5. We add to these the common variables measured for each concept, then rescale all the variables from 11-a-side to 5-a-side values, as done previously in section 4.10. These form our play representatives, which we will use to generate a Mirosot strategy, and are shown in table 5.1. The meanings of these variables, and their values, are given in table 5.2.

Note that we are using the inverse of the x_9 variable used in section 4.5, as this is an easier structure to construct.

Table 5.1 Play Representatives Selected for Strategy Generation

Strategy number	*IN HOME*		*IN AWAY*	
	Representative	Rectangle number	Representative	Rectangle number
1	$\langle x_8, x_9, x_{28}, x_{32}, x_{37}, x_{38} ; R \rangle$	30	$\langle x_{28}, x_{29}, x_{30}, x_{31}, x_{37} ; R \rangle$	30
2	$\langle x_8, x_9, x_{10}, x_{32}, x_{37}, x_{38} ; R \rangle$	30	$\langle x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{37} ; R \rangle$	30
3	$\langle x_8, x_9, x_{10}, x_{32}, x_{37}, x_{38}, x_{41} ; R \rangle$	28	$\langle x_8, x_9, x_{28}, x_{29}, x_{30}, x_{31}, x_{37} ; R \rangle$	35
4	$\langle x_8, x_9, x_{28}, x_{30}, x_{32}, x_{37}, x_{38} ; R \rangle$	28	$\langle x_8, x_9, x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{37} ; R \rangle$	32
5	$\langle x_8, x_9, x_{10}, x_{28}, x_{32}, x_{37}, x_{38} ; R \rangle$	28	$\langle x_{28}, x_{29}, x_{30}, x_{31}, x_{32}, x_{37}, x_{41} ; R \rangle$	28

Table 5.2 Variable Descriptions and Values for Strategy Generation

Variable	Description	*IN HOME*		*IN AWAY*	
		Relation	Value	Relation	Value
x_8	Percentage of player turns spent closer to the home goal than the ball	>	63.33	>	63.03
x_9	Percentage of player turns spent closer to the away goal than the ball	<	42.50	<	42.99
x_{10}	Percentage of player turns spent inside the triangle formed by the ball and home goal posts	>	13.69	-	-
x_{28}	Players closer to ball than opponent 1	<	0.34 0.89	<	1.01
x_{29}	Players closer to ball than opponent 2	-	-	<	1.41
x_{30}	Players closer to ball than opponent 3	<	3.59	<	2.20
x_{31}	Players closer to ball than opponent 4	-	-	<	3.01
x_{32}	Players closer to ball than opponent 5	=	4.99	<	3.87
x_{37}	Players closer to home goal than opponent 5	=	5	=	4.99
x_{38}	Players closer to away goal than opponent 1	=	0	-	-
x_{41}	Players closer to away goal than opponent 4	<	2.40	>	2.39

Due to the effects of scaling, variable 28 has two possible values when used in the *IN HOME* play. This is because three variables in the 11-a-side RoboCup data correspond to just one variable in the 5-a-side Mirosot data. In most cases, the three variables appear together and the lowest, or highest, value of the three can be used as appropriate. However, in the *IN HOME* representatives above, they appear separately, and with significantly different values. The lower value of 0.33 is used in *IN HOME* representatives 1 and 5, whereas the value of 0.87 is used in representative number 4.

We generate five strategies, each consisting of a pair of play representatives as shown in table 5.1. Each strategy will contain a simple switch based on the ball position to select one of the two representatives to be active at any time. The robots will then be controlled to reproduce the structures in the active representative.

5.3 Controller Implementation

Having identified a structure of plays, and selected our representatives, we now examine the implementation of our control architecture.

Recall, in chapter 4, we introduced our variables and placed emphasis on selecting parameters which could be represented mathematically. Although this helps to simplify the analysis, the main reason was to facilitate their use in a robot controller. We stated that human footballers may use complex descriptive or intuitions to understand what is happening on the pitch, but that these are of little use to robots. We purposely selected variables which could be easily converted into robot instructions so that the abstracted description of football formed in this thesis could be directly used to control a robot team.

The primitives used to describe the concepts were measured at every frame, which corresponds to the inherent sampling time of the robot football system. This is also the maximum rate at which we can provide meaningful control signals to our robots. Although it is possible to respond to a series of past data, or predicted future states, we will focus on reacting to the state of the game at the present instant. At every frame we will endeavour to recreate the desired representative, by passing target locations to each robot.

The variables we will be using in our controller all represent the number of robots in a particular segment of pitch. Since our primitives are averaged over a period of time, we need to convert the values stored within them to an integer number of robots in each segment at each frame. We will, therefore, round each variable to the nearest integer value. The modified target values for each variable per frame are shown in table 5.3.

Table 5.3 Modified Variable Values for Controller Generation

Variable	*IN HOME*		*IN AWAY*	
	Relation	Value	Relation	Value
x_8	>	3	>	3
x_9	<	2	<	2
x_{10}	>	1	-	-
x_{28}	<	0	<	1
		1		
x_{29}	-	-	<	1
x_{30}	<	4	<	2
x_{31}	-	-	<	3
x_{32}	=	5	<	4
x_{37}	=	5	=	5
x_{38}	=	0	-	-
x_{41}	<	2	>	2

Our controller needs to calculate a set of positions which satisfy the criteria of the relevant representative. To do this, we begin by dividing the pitch up into regions which correspond to the variables given in the representative. These regions will intersect to form segments where more than one variable is affected.

Each of these segments is assigned a binary array equal in length to the number of variables; with each element of the array valued '1' if placing a robot in the segment will affect the corresponding variable. Once all the segments have been valued in this way, we then search for the set which best reconstructs the representative. Each robot is then sent commands to move it into one of the selected segments. This process is repeated every frame.

Using this approach provides the additional benefit that it allows us to define regions, such as the goal area, penalty area, and centre circle. These are often subject to game rules, and may restrict the number of players that can be present within them. By segmenting the pitch in this way, we can add these limitations caused by the rules directly into our controller.

5.3.1 Controller Example

Consider a game played using strategy 3 given in table 5.1. At the moment of interest, the ball is in the home end of the pitch. The corresponding representative $\langle x_8, x_9, x_{10}, x_{32}, x_{37}, x_{38}, x_{41}; R \rangle$ is selected for implementation by the controller. Listing 5.1 shows the pseudo code relating to this operation.

```
Select representative
1:  input data
2:  input representatives
3:  create an empty store for the most valid primitive
4:  for each primitive
5:       extract validity criteria from primitive
6:       measure the validity against the current system state
7:       if primitive is more valid that that stored
8:            store primitive and validity measure
9:       end
10: end
11: output most valid primitive
```

Listing 5.1 Pseudo Code for Representative Selection

Each strategy representative contains two play representatives, of which one is always valid. In our implementation, this simplifies the operation for measuring which representative is most valid to the current system state. The representative for *IN HOME* is selected if the ball is in the home half of the pitch, and the representative for *IN AWAY* is selected if the ball is in the opponents half of the pitch.

To recreate the representatives, the robots must perform actions which affect the variables they describe. Our method is to generate a list of actions, each of which influences the variables in some way, then select those actions which most closely satisfy the representative.

The variables in our representatives are all spatial structures, and represent players being present in specific areas on the football pitch. The actions to be generated can be thought of as target positions for the robots to move to. The pseudo code relating to the action generation procedure is given in listing 5.2.

```
Generate actions
1:  input data
2:  input variables
3:  define important features
4:  add features into the list of variables
5:  point features = call generate intersections function on the
      list of variables
6:  sectors = perform Delaunay triangulation on point features
7:  find centres of sectors
8:  remove centres outside of perimeter
9:  output centres

Generate intersections
10: input variables
11: create list of empty points
12: for each variable
13:      for every other variable
14:           create an empty list of intersections
15:           if the two variables are circles
16:                call the circle intersection function
17:                add the result into the list of intersections
18:           else if the two variables are polygons
19:                add all vertices into the point list
20:                for each pair of possibly intersecting sides
21:                     call the line intersection function
22:                     add the result to the list of intersections
23:                end
24:           else if the variables are a polygon and a circle
25:                add the polygon vertices into the point list
26:                for each polygon side
27:                     call the circle-line intersection function
28:                     add the result to the list of intersections
29:                end
30:           end
31:           remove any repeated intersections from the
                intersection list
32:           add remaining intersections into the point list
33:      end
34: end
35: return point list

Circle intersection function
36: calculates and returns the intersection(s) of two circles

Line intersection function
37: calculates the and returns the intersection of two lines

Circle-line intersection function
38: calculates and returns the intersection(s) of a line and a
```

Listing 5.2 Pseudo Code for Action Generation

The code generates a model of the pitch in terms of the structures described by the variables. On line 1 the current positions of the robots and ball are inputted, on line 2 we input geometric representations of the variables, and on line 3 we define

any other important features; we specify the perimeter of the pitch. Line 5 passes these structures to the function listed on lines 9-35. This extracts the vertices of the structures, and finds the points where they intersect. The result is a list of points which describe the vertices of important segments of pitch. Figure 5.2 shows this process diagrammatically. Here the structures are represented by solid black lines, and each segment is identified by a letter.

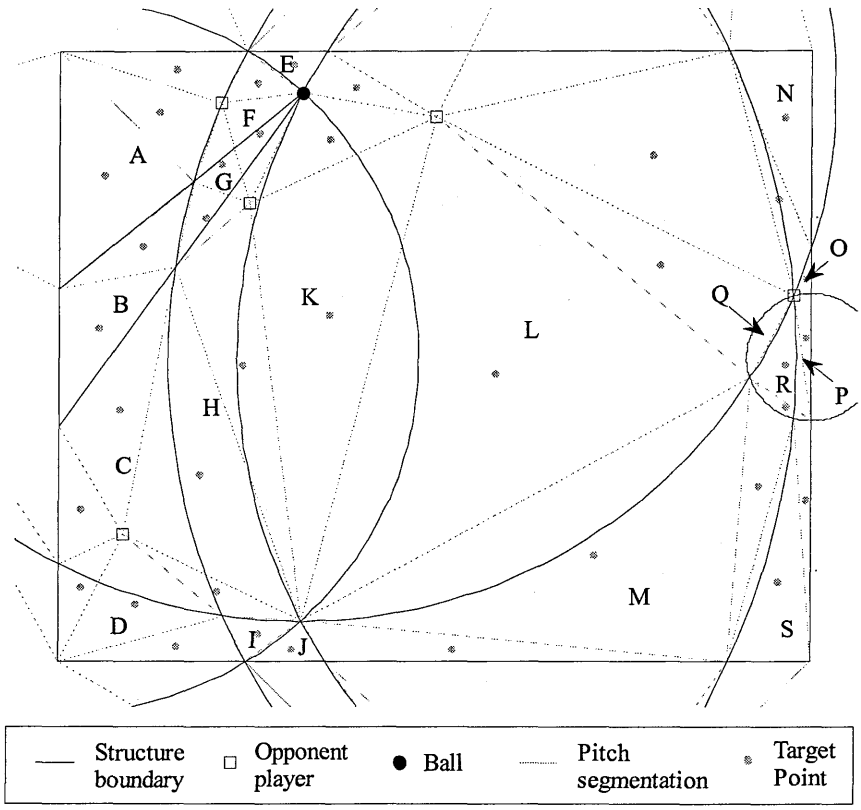


Figure 5.2 Pitch Segmentation Diagram

Having identified these segments, we generate target positions relating to each. On line 6 we use a built-in Delaunay triangulation of the vertices to further segment the pitch into triangular regions. This is illustrated by the grey lines in figure 5.2. The centre of each triangle is recorded as possible robot target point, provided it lies within the pitch perimeter.

Although the Delaunay diagram creates more segments than necessary, it is a simple and effective method of dividing the pitch into sections. It also allows us to generate more than one target point within each major segment, allowing us to place more than one robot in each area if necessary. In this example there are 37 Delaunay triangles representing 19 interesting segments of pitch, as divided up by the structural variables.

The next process is to calculate what effect each target point will have on recreating the representative. In the case of our example, placing a robot at each target point at each frame will simply add one to the value of each of the affected variables. The pseudo code for this function is given in listing 5.3.

```
Calculate the effect of each action
1:  input actions
2:  input variables
3:  create a zero valued effect matrix with actions as rows and
      variables as columns
4:  for each action
5:       for each variable
6:            evaluate the effect the action has on the variable
7:            store the result in the corresponding cell of the
                effect matrix
8:       end
9:  end
10: output effect matrix
```

Listing 5.3 Pseudo Code for Action Effect Calculation

The actions inputted on line 1 are the target points. On line 6 we call a function to evaluate the effect of placing a robot at a target point. This function, which is not listed, simply measures whether that point is within the area defined by the corresponding variable.

For the diagram given in figure 5.2, the effect matrix corresponding to the labelled segments is shown in table 5.4. A '1' indicates that placing a robot in that segment will increase the value of the associated variable. To simplify presentation,

segments rather than targets are listed; every target within a particular segment will give the same effect.

Table 5.4 Mapping of Variables to Named Pitch Segments

Segment	Variables						
	x_8	x_9	x_{10}	x_{32}	x_{37}	x_{38}	x_{41}
A	1	0	0	1	1	0	0
B	1	0	1	1	1	0	0
C	1	0	0	1	1	0	0
D	1	0	0	0	1	0	0
E	0	0	0	1	1	0	1
F	1	0	0	1	1	0	1
G	1	0	1	1	1	0	1
H	1	0	0	1	1	0	1
I	1	0	0	0	1	0	1
J	0	0	0	0	1	0	1
K	1	1	0	1	1	0	1
L	0	1	0	1	1	0	1
M	0	1	0	0	1	0	1
N	0	1	0	1	0	0	1
O	0	1	0	0	0	0	1
P	0	1	0	0	0	1	1
Q	0	1	0	1	1	1	1
R	0	1	0	0	1	1	1
S	0	1	0	0	0	0	1

Next a search is conducted to find the set of target points which best matches the representative. We search each target to find the one that will bring the primitive for the frame closest to the representative. The process is then repeated for each robot.

In testing, we have been able to generate formations which are further from the representative, maximising variables which are desired to be greater than the average, and minimising those desired to be less than the average. However, these formations were often undesirable, with robots clustering in single segments. This led to our conclusion in section 4.3.2 that there is a range over which variables are desirable. To counter this, we aim to build formations which recreate values close to those stated in the representative, whilst still coinciding with the relation to the average. Listing 5.4 gives the pseudo code for the process.

```
Action selection
1:  input action effect matrix
2:  input representative
3:  create an empty list of actions
4:  create a zero valued temporary array equal in length to the
      representative to hold the values of the recreated variables
5:  for each robot
6:       calculate the difference between each variable in the
           temporary representative array and the true
           representative
7:       create a list of variables that are over the maximum
           values specified by the representative
8:       create a list of variables that are under the minimum
           values specified by the representative
9:       for each action in the effect matrix
10:           calculate if variables in the action will add to the
                  overvalued and undervalued variables
11:           if the action will affect fewer of the overvalued
                variables and more of the undervalued variables
                than the current best choice
12:                store the action
13:                store the overvalued variables
14:                store the undervalued variables
15:           else if the action will affect fewer of the
                overvalued variables and the same number of
                undervalued variables than the current best choice
16:                store the action
17:                store the overvalued variables
18:                store the undervalued variables
19:           else if the action will affect the same number of the
                overvalued variables and more of the undervalued
                variables than the current best choice
20:                store the action
21:                store the overvalued variables
22:                store the undervalued variables
23:           end
24:      end
25:      add best action to the action list
26:      remove the action from the effect matrix
27:      add the action effect to the temporary representative
           array
28: end
28: output actions
```

Listing 5.4 Pseudo Code for Action Selection

Variables in the representative are declared to be less than or greater than a specified value. This code attempts to generate values close to, but on the desired side, of the specified value. On line 4, we declare a zero valued temporary array. This is used to store the values of the variables generated by the selected actions. In effect, this holds the value of the reconstructed representative. The code then loops

to find the best action for each available robot. This begins by finding the difference between the original and reconstructed representative on lines 6-8. The code then loops through each available action to asses its suitability on lines 9-24.

Desirability is measured in terms of how many variables are over or under valued. We consider a variable that is required to be less than a given amount to be overvalued once it exceeds that value. Similarly, a variable that is required to be more than a given amount is considered undervalued until it exceeds that value. An action is chosen to replace any existing action for that robot if it fulfils one of the following criteria: 1) it reduces the value of the overvalued variables, but adds value to the undervalued variables 2) it will not affect the value of undervalued variables, but reduces the value of the overvalued variables 3) it will not affect the value of overvalued variables, but adds value to the undervalued variables. These are listed in order of importance, with criteria (1) overriding criteria (2) and (3). This code only examines variables which do not yet meet the specification given by the representative. The effect of the action on the remaining variables is not considered.

Once all available actions have been tested, the most desirable is added to the action list on line 25, and removed from the list of actions available to the remaining robots on line 26. The effect of this action is added to the reconstructed representative on line 27, for use in selecting actions for subsequent robots.

A more rigorous search method could be employed to improve the selection of target points. However, the time available to perform these calculations is limited, and we find that this technique is adequate for our needs.

Using this method, the five targets which best allow us to satisfy the representative are chosen. The segments corresponding to the targets chosen in this

example are shown in table 5.5. The total value given is the value of the reconstructed representative. The desired value for each variable is given at the head of each column.

Table 5.5 Selection of Segments for Target Generation

Segment	Variables						
	$x_8 > 3$	$x_9 < 2$	$x_{10} > 1$	$x_{32} = 5$	$x_{37} = 5$	$x_{38} = 0$	$x_{41} < 2$
G	1	0	1	1	1	0	1
K	1	1	0	1	1	0	1
C	1	0	0	1	1	0	0
A	1	0	0	1	1	0	0
B	1	0	1	1	1	0	0
Total	5	1	2	5	5	0	2

Finally, the low level control interprets the actions into robot commands. In our experiments it implements a positional controller to send each robot to one of the selected target points. Since all our robots are identical, we have implemented an algorithm to match robots to target positions which minimises the overall movement of the team.

5.4 Performance Evaluation

Our experiment consists of three parts. Firstly, we test the ability of our controller to construct representatives, by using it to generate targets in response to 1000 randomly selected frames of recorded Mirosot data. We then measure the average values of the variables for each play, and check whether they conform to the original

representative in each case. This demonstrates whether the controller is capable of generating acceptable targets.

The second experiment demonstrates the ability of our software to control a team of Simurosot robots. For each of our generated strategies, we run one match against a traditional role-based team. At each frame our controller selects appropriate target points, and drives the simulated robots toward those points. Again, we analyse 1000 frames and check whether the positions of our robots in those frames conforms to the appropriate representative. This experiment investigates the affect of game dynamics on the ability of robots to reach target points.

The third experiment is very similar to the second; however, instead of using the Simurosot simulator, we run the tests on our Mirosot robots. This experiment investigates the application of our controller to real world conditions.

The results of these experiments are shown in table 5.6, table 5.7, and table 5.8. For each experiment, table 5.6 shows the percentage of variables, and full frames, which satisfy the representative. In table 5.7 and table 5.8, we have analysed the data from each experiment, and measured the primitives as done in section 4.5. We match each primitive with the corresponding representative to asses how well we have reproduced the corresponding structures. Variables which fail to meet the specification of the representative are shown in bold.

Table 5.6 Results for Reproduced Representatives

	Representative	Correctly reproduced frames (%)			Correctly reproduced variables (%)		
		Controller	Simurosot	Mirosot	Controller	Simurosot	Mirosot
IN HOME	1	70.9	28.4	22.8	94.4	75.6	80.2
	2	87.0	21.3	20.7	97.1	75.7	73.2
	3	75.2	28.7	12.8	95.8	77.7	73.3
	4	87.8	41.7	20.2	97.8	83.7	81.5
	5	83.4	19.8	7.7	97.1	74.5	70.6
IN AWAY	1	99.0	80.2	71.4	99.8	94.1	89.2
	2	100	92.9	41.8	100	98.4	76.6
	3	100	96.4	71.3	100	99.3	93.4
	4	100	83.6	84.7	100	97.3	94.9
	5	100	41.4	23.6	100	89.4	79.8

Table 5.7 Results for Defensive Play Primitives

			Variables								
			$x_8 > 3$	$x_9 < 2$	$x_{10} > 1$	$x_{28} < 0, 1$	$x_{30} < 4$	$x_{32} = 5$	$x_{37} = 5$	$x_{38} = 0$	$x_{41} < 2$
IN HOME primitives	Play 1	Controller	3.05	1.70	-	**0.03**	-	5	5	0	-
		Simurosot	**2.99**	**2.99**	-	**0.70**	-	**4.93**	**4.99**	0	-
		Mirosot	3.70	**3.18**	-	**0.58**	-	**4.98**	**4.99**	0	-
	Play 2	Controller	3.29	1.47	**0.88**	-	-	5	5	0	-
		Simurosot	3.32	**3.09**	**0.52**	-	-	**4.98**	5	0	-
		Mirosot	**2.98**	**3.18**	**0.38**	-	-	**4.96**	5	0	-
	Play 3	Controller	3.44	1.45	**0.97**	-	-	5	5	0	0.94
		Simurosot	3.54	**2.72**	**0.64**	-	-	**4.94**	5	0	1.62
		Mirosot	3.79	**3.38**	**0.33**	-	-	5	5	0	1.74
	Play 4	Controller	3.28	1.65	-	0.37	2.40	5	5	0	-
		Simurosot	3.19	**2.56**	-	0.65	3.02	**4.97**	5	0	-
		Mirosot	3.08	**3.43**	-	0.62	2.90	**4.99**	5	0	-
	Play 5	Controller	3.11	1.50	**0.94**	**0.08**	-	5	5	0	-
		Simurosot	**2.95**	**2.93**	**0.52**	**0.70**	-	**4.99**	5	0	-
		Mirosot	3.00	**3.44**	**0.26**	**0.51**	-	**4.98**	5	0	-

Table 5.8 Results for Offensive Play Primitives

IN AWAY primitives			Variables								
			$x_8 > 3$	$x_9 < 2$	$x_{28} < 1$	$x_{29} < 1$	$x_{30} < 2$	$x_{31} < 3$	$x_{32} < 4$	$x_{37} = 5$	$x_{41} > 2$
	Play 1	Controller	-	-	0.09	0.21	0.39	0.81	-	5	-
		Simurosot	-	-	0.12	0.25	0.45	0.87	-	5	-
		Mirosot	-	-	0.34	0.76	1.09	1.83	-	**4.97**	-
	Play 2	Controller	-	-	0	0.09	0.41	0.92	1.79	**4.99**	-
		Simurosot	-	-	0.04	0.30	0.51	1.08	2.02	5	-
		Mirosot	-	-	0.35	0.95	1.78	2.64	3.55	5	-
	Play 3	Controller	4.68	0.78	0.08	0.11	0.25	0.63	-	5	-
		Simurosot	4.94	0.52	0.02	0.09	0.38	0.80	-	5	-
		Mirosot	4.30	0.71	0.10	0.37	0.80	2.16	-	5	-
	Play 4	Controller	4.57	0.50	0.12	0.23	0.41	0.85	1.39	5	-
		Simurosot	4.82	0.34	0.17	0.34	0.50	1.10	1.90	5	-
		Mirosot	4.82	0.58	0.14	0.32	0.64	1.04	2.10	5	-
	Play 5	Controller	-	-	0.07	0.27	0.67	1.29	2.16	5	2.18
		Simurosot	-	-	0.01	0.09	0.39	1.19	2.63	**4.98**	**1.54**
		Mirosot	-	-	0.14	0.48	1.47	2.21	3.00	5	**1.32**

From table 5.6, we can see that our method of control has a high success ratio for reproducing variable structures across all experiments. In each test we reproduce at least 70% of the variables correctly. There is a much lower chance of reproducing the full representative at each frame, with structures in as few as 7.7% of frames meeting the specification. However, given the high proportion of variables correctly replicated, it is evident that it will only be one or two variables which prevent the whole representative being achieved in each case.

We can also see that some representatives are easier to reproduce than others. In particular, the representatives for the *IN AWAY* are generated with a much higher success rate than those for the *IN HOME* plays. It may be that *IN AWAY* plays are less strictly specified in general, or that we have not measured some critical structures.

The performance of the controller for generating targets is very good. As shown in table 5.6, it correctly reproduces the representative in at least 70% of frames, with 100% success achieved in a large proportion of cases. This indicates that we can use the techniques covered in this thesis to measure and reproduce structures relating to complex environments. By generating concepts, measuring primitives, constructing representatives, and reproducing these using the type of control described in this chapter, we can generate instructions to control robots to perform in the ways we have measured to be desirable. It has been noticed, however, that dividing the pitch into sections using a Delaunay diagram does not always give a target point in every segment. It also limits the number of possible targets in each section. Generally, as can be seen from the results, the targets are satisfactory, but we could improve this further by using a more accurate method of generating target options.

Although the controller itself is successful in generating targets, there is a loss in performance when applying that control to the robots. The results for the Simurosot experiment, shown in table 5.6, indicate a drop of 0-20% in the rate of reproducing variables, and 7-51% in the rate of reproducing the full representative. This difference arises from the movement characteristics, in particular the time taken for a robot to move between two target points. If a robot has to move to a new,

distant, target point, it will take a finite time to cross the region in between. During the majority of this time, the robot will be in segments which do not meet the specification of the representative, hence causing the response of the system to deteriorate. The larger, and more frequent, the required change in robot position, the worse the response of the system will become. A related factor which is evident in the simulator is the possibility of collisions, which prevent robots reaching their targets. Ways of improving these results could include improving the motion control of the robot to give a faster, more accurate response, implementing better obstacle avoidance, using prediction to estimate target points in advance, and attempting to reduce large target changes in the control by taking into account current robot positions.

The performance of the system decreases again when we implement the control on our Mirosot robots. We measure a drop of 5-24% in the reproduction of variables, and 15-63% in the reproduction of representatives, when compared to the targets generated by the controller. The reasons for this are similar to those given above for the deterioration in performance of the Simurosot system. However, they are even more evident in the Mirosot system due to the introduction of real world factors, including noise in the vision system and radio communications, and more disruptive collisions. Similar issues cause the motion control of the Mirosot robots to be worse than that of the Simurosot robots, producing a longer transition between target points.

As we have shown, the failure to reproduce the full representative is most frequently due to misrepresenting a single variable. In table 5.7 we can see the most unachievable of those variables is x_{10}, which is under valued in every primitive.

This relates to the number of robots in the triangle with vertices corresponding to the home goal posts and the ball. This is a small area and, using the Delaunay diagram to dissect the pitch, does not always contain a target point, let alone more than one. By altering our target generation algorithm to give more target points in the smaller segments, we may be able to improve the chances of reproducing such variables. Furthermore, we can see in table 5.8 that variables x_{32} and x_{37} are also occasionally inadequate. Our definition in this experiment requires all common variables to be reconstructed equal to the value given in the representative. Recall, however, when we defined common variables back in section 4.3.2, that we allowed a 5% variance between the measured value and the average. If we apply the same criteria here, then all the measurements for these variables fall within 5% of the target value, and can be marked as successful. When taking into account these allowances, the primitive measured for *IN AWAY* play is achieved in all but play 5. Even then, the primitive for play 5 is only out by one variable.

In our final piece of analysis, we relate the results back to our initial investigation into the spatial qualities of robot football. Recall we stated that the area controlled by a team of robots was linked to the position of the ball, and that attacking teams occupied more space than defending teams. This was the initial foundation on which this work was based. In figure 5.3 we show a typical section of a Mirosot match played using the controller developed in this chapter. Once again, it can be seen that there is a relationship between area controlled and ball position. Although we have not explicitly programmed this, it has emerged from the interaction of the variables in the concepts we have defined. Despite having not produced a fully functional football strategy, these results show that we have

managed to reproduce the spatial structures of football on a strategic level using our new approach of task abstraction and concept generation.

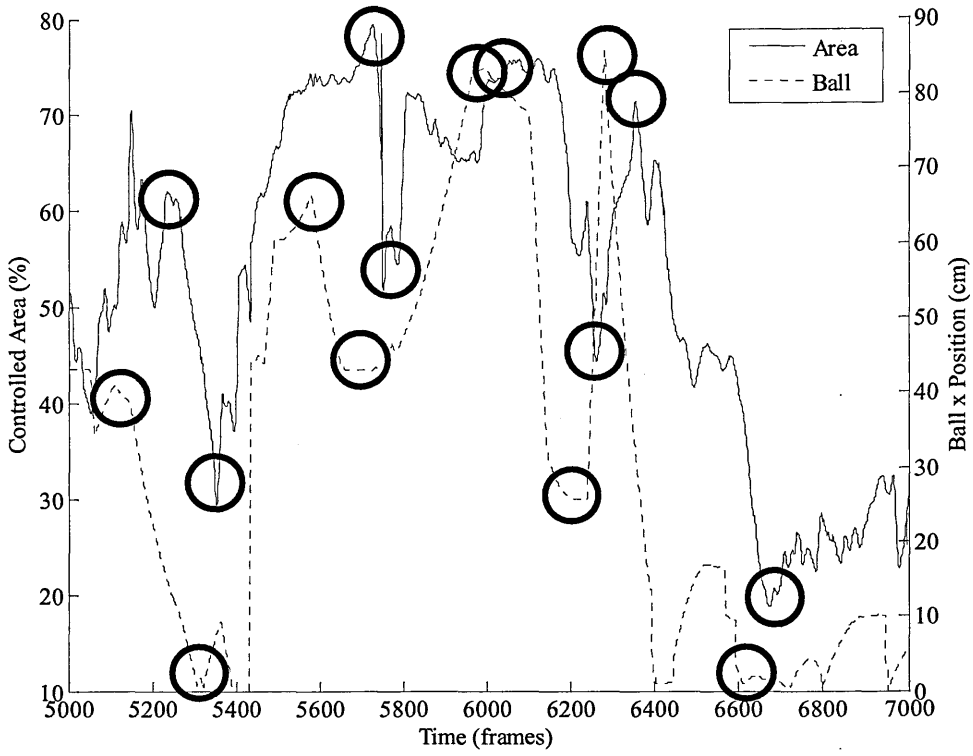


Figure 5.3 Comparison of Ball Position and Controlled Team Area

5.5 An Architecture for Analysis and Control

We have now demonstrated an architecture for both analysis and control of complex systems. The block diagrams in figure 4.1 and figure 5.1 have been combined below in figure 5.4 to show how the complete architecture functions.

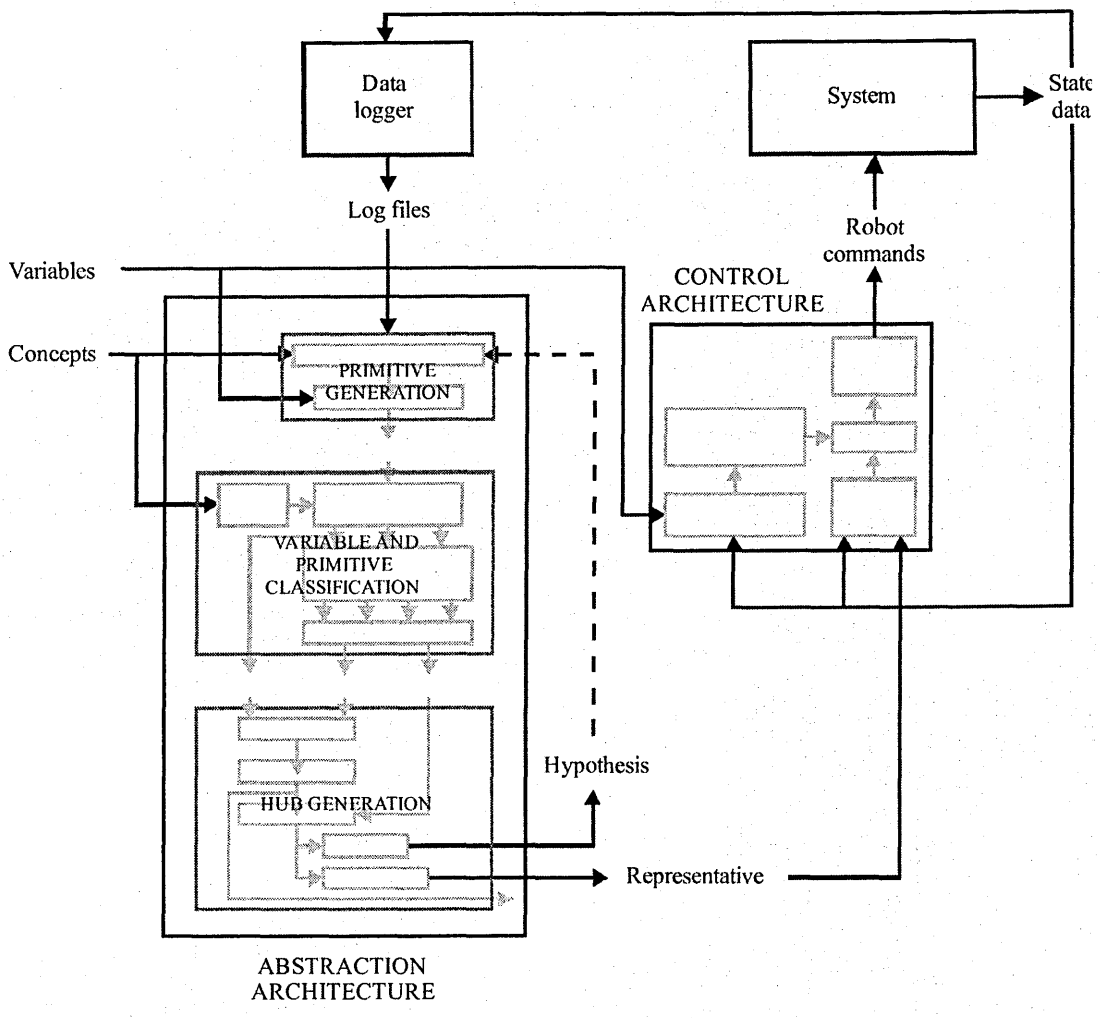


Figure 5.4 An Architecture for Analysis and Control

In this diagram, the robots' are modelled as being part of the system. Low level feedback, actuating, and sensing processes are all contained within the system block. The output from the system block is the data recorded at that instant by the available sensors. The inputs to the block are the set of low level commands for the robots to perform.

The data logger accumulates state data over time, and stores it in a log file. The log is then used as training data from which the abstraction architecture can generate hypotheses and representatives. In the work undertaken in this thesis, the logs have been used to perform learning off line. Whenever new data relating to a

football match has become available, it has been added to the log, and used to update the abstraction.

Although not attempted in this thesis, it may be possible to perform some of the abstraction on line. In this case, data accumulated during the match could be used on the fly to update the analysis, and tailor the abstracted tasks to the opposition. However, due to the processing time required by the algorithms implemented here, this is not currently feasible.

The abstraction architecture generates a list of representatives and hypotheses, from the log file, relating to the variables and concepts selected by the user. The representatives are then passed to the controller for implementation, whereas the hypotheses are used as a method for classification. The dotted line in figure 5.4 indicates how the hypotheses can be fed back into the analysis to classify future primitives. This would be used in situations where primitives could not be easily classified: A small set of training data, and hand selected primitives are used to generate a hypothesis, which is then used as the classification criteria for all future primitives.

Given the representatives, variables, and state data, the control architecture then attempts to create robot commands which will result in the recreation of the representative. These commands are sent to the robots to perform.

5.5.1 Implementation Flexibility

We have implemented our architecture on a centralised, deliberative system. However, we suggest that, with minor modification, it would be applicable to both distributed and reactive systems.

In our implementation, the architecture runs as a centralised process, creating representatives based on all available data, and generating control commands in a coordinated fashion. Consider the alternative, in which the architecture is implemented on a single robot. Data is only available from the robot's own sensors, of from communication with other robots. Consequently, concepts are chosen which only relate to the individual robot, and the derived representatives will describe how that robot should attempt to accomplish the tasks using only the data available to it. Furthermore, the analysis architecture can be used separately for concepts which require two or more robots to interact. In this case, data available to each robot is included as a variable, and the representative will highlight what data needs to be communicated between the two.

The control architecture generated in section 5.1 is deliberative. Given a representative, it models the possible actions, creates plans for the possible action sets, and then selects the best option. However, a reactive architecture could be implemented in its place. In this situation, the abstraction architecture becomes a tool for learning and defining behaviours. If sensor and actuator data is fed into the abstraction process as the learning data, and the concepts are the desired low level behaviours, then the representatives give the appropriate sensor-actuator mappings. Higher level behaviour concepts can then be generated as mappings on lower level behaviours in the usual way.

5.6 Summary

In this chapter we have introduced an architecture for generating emergent controllers by reconstructing representatives. Based on the results from our earlier

experiments investigating structures in robot football, we generate an abstracted strategy composed of high level positional plays. These plays are described in terms of low level structures, but, through their interaction, appropriate formational plays emerge.

The chapter began by introducing a control architecture for reconstructing representatives created using our abstraction architecture. These representatives describe relations on sets of variables, which describe how to recreate given concepts. From a set of representatives, the controller selects the most appropriate to recreate based on the state of the system. It generates actions to recreate each variable individually, then combines these to satisfy the relation described by the representative. Once a satisfactory set of actions has been identified, these are turned into commands and sent to the robots for execution.

In the following section we created the strategies to implement. This was achieved by selecting maximal hubs measured in the earlier chapters of this thesis to use as representatives. Although there are a number of issues preventing us from creating a complete Mirosot strategy at this point, we do succeed in developing a simple 2-play strategy, which focuses on the high level coordination of robot movement. Using our ideas on multilevel and multidimensional structure in complex systems, we combine representatives taken from the RoboCup data to build five distinct, simple, 2-level strategies.

To implement these strategies on our robotic platform, we generate an instance of our control architecture. Running once for every frame acquired by the camera, the controller uses the position of the ball as a switch to select between play representatives from within a strategy representative. It creates a model of the pitch,

consisting of the variable structures in the representative. These structures dissect the pitch into segments, which each describe a relation between the variable. Each segment is then converted into a number of target points, which can be used to recreate the representative. The controller selects the five targets which together most closely generate the representative, and drives the robots to the chosen positions.

We test the controller's ability to generate targets which match the representatives, and its performance in both Simurosot and Mirosot matches. The results show that although the controller does generate suitable sets of target points, the relatively slow response of the robots in moving to these positions causes the Simurosot and Mirosot teams to under perform. We show that it is only a small fraction of the representative that is missing in these cases, and suggest that using enhanced robot motion controllers, and predictive algorithms will substantially improve the performance. Despite these failures, which are evident during individual frames, the primitive measured over the duration of the play can match the representative. Furthermore, we show how the emergent spatial ownership of the pitch correlates with the initial investigation carried out in chapter 3. This is the objective of the work, and shows that we can abstract ideas from a complex system, such as robot football, and by carefully reproducing them, create a controller enabling robots to perform in that environment.

Finally, we laid out the analysis and control architectures in a single block diagram, showing the relevant interfaces and feedback loops. Although having implemented the architectures on a centralised, deliberative system, we described how they could be modified to work on both distributed and reactive systems.

Chapter 6

Conclusions

In this thesis we have demonstrated how task abstraction using concept generation can be used to analyse and control a team of robots in the complex football environment.

We began, in chapter 1, by introducing our research problem. In chapter 2, we reviewed some of the main multirobot architectures, and showed that none of them are applicable to problems where the task is not well defined. We further showed that robot football possesses all the qualities of our problem, and that current competition strategies are designed based on poorly defined tasks. Methods using multilevel mathematics were introduced in section 2.4 as our proposed approach to solving the problem.

In chapter 3, we showed that robot football involves problems which cannot be solved using the game tree search algorithms traditionally associated with AI game problems. We showed that useful spatial configurations exist, and can be identified, which can be used to represent the game. These configurations, and the corresponding areas controlled by each team, are dependant on the state of play, with some areas becoming more valuable at different times. We showed that a team in which players work together to control space is more successful than a group of greedy individuals, which are again more successful than a randomly moving group of players.

In chapter 4 we introduced an architecture for task abstraction, and showed that sets of good spatial relationships can be extracted to represent structural concepts, through examination of winning and loosing football teams. Sets of spatial representations common to winning and losing teams indicate general structures required for playing the game, whereas sets distinct to winning teams indicate structures which enable a team to play well. These structures form a multilevel representation of the game, with variables propagating between levels; logical operators relate hubs on one level to hubs on another. We showed how representations of different types of football were linked, and inferred that concepts are mathematical representations of epistemological forms.

Our hypothesis throughout this work has been that controllers can be designed, based on representations of measured concepts, and used to control real robots. This was proven in chapter 5, in which we described a control architecture for reconstructing representatives. We developed 5 strategies based on measurements taken in chapter 4, and used these to develop 5 controllers. Our experiments showed that these controllers performed satisfactorily when generating actions to reconstruct representatives in response to randomly selected frames of robot football data. When implemented as a strategy in a in a game situation, the response deteriorated due to the dynamics and low level control of the robots.

6.1 Answers to the Research Questions

In chapter 1 we posed the question "How can we control a team of robots to perform a weakly-defined cooperative task in a complex, dynamic, unpredictable and competitive environment?" Throughout this thesis we have strived to answer this

question by tackling, in turn, a number of subsidiary questions. These are listed below, along with the answers and reference to supporting evidence supplied by this thesis.

1. **Can we identify and construct useful representations of complex, dynamic, unpredictable and competitive environments in ways which facilitate the use of robots?**

   We identified robot football as a suitable system possessing the required qualities in section 2.3. In sections 3.2, 3.4.1 and 3.4.2 we showed a relation between the areas controlled by robots during a football match and the state of play. These areas are directly linked to the spatial structures formed by the players, which were further investigated in chapter 4. All of these spatial representations can be described mathematically with relative ease, and so are ideal for describing a robotic task

2. **Can we extract useful information on how to control a team of robots by recognising the occurrence of key structures in different teams operating in similar situations?**

   In chapter 4 we searched for the occurrence of our spatial representations in data obtained from the RoboCup Simulation League. Our analysis showed that certain structures appeared, on average, more, or less, frequently in the winning teams. These structures, we concluded, were important in defining whether a team strategy would be successful or unsuccessful in terms of goals scored. Other structures were found to occur in similar amounts in both winning and losing teams. These were indicative of how to play the game in general.

3. **Can we use the same techniques to extract information about tasks in the environment at varying levels of representation?**

   We proposed, in section 2.3, that a robot football strategy has a multilevel structure. In section 4.2, we extended this proposal to show how we could represent it in terms of a multilevel structure of plays, tactics and skills. Having demonstrated our method of task abstraction to generate concepts at the strategic level in section 4.4, we moved on to show how the same technique could also be used to abstract information about plays in sections 4.5 and 4.6, and tactical events in section 4.7.

4. **Can we identify relations between interacting levels of representation?**

   Using Venn diagrams relating to a carefully chosen set of play concepts, we hypothesised, in section 4.9, that there must be a relation between the variables prevalent in associated concepts, and that these relations would cross the boundaries between levels in the strategy structure. We supported this claim in section 4.9.2 by providing experimental results which showed that sets of variables could be found, which propagated through the multilevel structure. Moreover, we showed how hubs relating to concepts at a higher level in the structure could be generated by combining hubs from separate concepts at a lower level, using an OR-aggregation.

5. **Can we build our own representation of a set of tasks of varying complexity by combining information from different levels of analysis?**

In chapter 4 we generated representatives for a range of tactics, plays, and overall strategies. In section 4.2 we showed how the related concepts would fit together to form a multilevel strategy. Although not a complete representation of a football strategy, this structure of concepts did described many aspects of the game. If all the necessary concepts can be identified, the connecting structure understood, and suitable representatives found, then a complete system representation can be built incorporating many tasks at many levels.

6. **By combining information in this way, can we create an emergent strategy for controlling robots in the environment?**

   Our ideas on structure presented in section 4.2 showed how strategies, which were considered as our highest level concept, could be seen as composed of plays, which exist at a lower level in the multilevel structure. From the results taken in chapter 4, we selected ten of the largest representatives for two complimentary plays. In section 5.2 we combined these representatives following the structure proposed in section 4.2 to generate five distinct strategies. The variables contained within these representatives related to spatial structures, which on their own do not hold any information on how to play football. The resulting strategies, therefore, emerged from the interaction of these variables.

7. **Can a team of robots, built upon these principles, function effectively in the given environment?**

   In section 5.4 we tested the ability of a controller to recreate the structures, in the abstracted strategies, at the appropriate stages in the game. We

showed that our controller reproduces the structures on 71%-100% of occasions, when responding to static sensor data. The results are inferior when observed in simulated (20%-96%) and real robots (8%-85%) due to the dynamics and lack of prediction. However, these failures are only evident in one or two variables, with the remainder of the representative being correctly recreated. It is significant to note that when we investigated the motion of the robots during these games, that the relationship between ball position and area controlled was very similar to that identified in our analysis of RoboCup matches conducted in section 3.4.2. Although not a complete football playing strategy, our methods have produced appropriate spatial configurations at the strategic level.

8. **Can these ideas be combined to form an architecture whereby a robotic system can learn to perform in a given environment?**

   In section 4.1 we described an architecture for abstracting tasks from a complex system. This was used throughout chapter 4 to find hubs relating to the various concepts. In section 5.1 we described an architecture for control based on reproducing representatives of abstracted concepts. The two architectures were combined in section 5.5 to describe a single architecture through which robots can learn to perform in a given environment. Using this architecture we have supervised a team of robots to abstract and learn their own set of structures for playing football. The resulting strategy emerges from the interactions of these structures.

Returning to our original question, we now have sufficient evidence to show that our method for generating a control strategy by concept generation satisfies the

problem. We showed, in section 2.3, that robot football is a complex, dynamic, unpredictable and competitive environment, and that methods for coordinating teams have been limited by a deficiency in the definition of the task. Although further work needs to be undertaken to demonstrate a complete football controller using this technique, the evidence given in this thesis supports the idea that the methods described are capable of satisfying the need. Furthermore, this work represents a completely new approach to multirobot control.

6.2 Contributions to Knowledge

During the course of this work we have made a number of innovations:

1. We have conducted a series of studies into the importance of spatial structures in competitive team games. The Space-Time Possession Game has been designed as an extension to the existing set of Voronoi Games, and presents a more advanced challenge. It provides a clearer AI challenge for studying spatial competition and cooperation (section 3.2.1), and has been used to highlight the benefits of cooperative, over non-cooperative, behaviour (section 3.2.2). We have also introduced new theories regarding the importance of geometric structures and spatial control in the game of robot football (section 3.2). These have been supported by experimental results (sections 3.4.1 and 3.4.2).

2. A new approach to concept generation has been introduced, using global and local averages to classify variables (section 4.3.2). This has been used to analyse both RoboCup (sections 4.4 - 4.7) and Mirosot (section 4.10.1) robot football. We have demonstrated that the results from the analysis

enable combinations to be found that have considerably increased probability of occurring in desirable primitives (section 4.3.4), and have extended previous investigations into passing events (Iravani, 2005a) to analyse additional spatial configurations (section 4.7). The use of concept generation has been demonstrated at multiple levels within a multilevel structure. We have used the technique to generate concepts relating to strategies (section 4.4), plays (sections 4.5 and 4.6), and tactics (section 4.7). We have also shown how different properties can be used to classify primitives in these cases (sections 4.5 and 4.6).

3. Using experimental evidence, we have shown how sets of variables from two separate concepts at one level in the multilevel structure combine to form hubs in a single concept at a higher level (section 4.9.2). This property is closely linked to the idea of emergence. We have also shown that relationships exist between similar concepts in related complex systems. This was shown by comparing play hubs from Mirosot and RoboCup matches (section 4.10.2). From this we have been able to hypothesise that concepts relate to theoretical forms, and that there are higher level concepts which contain both types of robot football.

4. This thesis has also shown how representatives extracted from simulation data can be used to form a functional robot controller (chapter 5). Representatives corresponding to the concepts measured in chapter 4 were combined in a multilevel structure, effectively recreating the decomposed strategy. By moving robots to recreate the structures in these representations, a formational strategy emerged. This controller has been

demonstrated, on both simulated and real robots, and its performance evaluated (section 5.4).

The contributions identified in points 1 and 2 have been published by the author in (Law, 2005; Law & Johnson, 2004, 2006, 2008).

6.3 Further Work

The research described by this thesis generated many more questions and highlighted areas which require further development to show the full utility of the method. The list below describes some of these areas, and the work required, as well as indicating some possible avenues for future research.

Work to support the theory:

1. Demonstrate the method is capable of producing a full solution by generating controllers for simpler problems.
2. Verify the relations (AND & OR) found to exist between the variables constituting concepts at different levels of the structure, by analysing other systems or groups of concepts.
3. Attempt to reproduce a known robot football strategy. Choose variables and concepts to match those programmed into a traditional strategy, and asses the difference in response.

Modifications to the algorithms:

4. Automate the procedure for choosing and creating variables, concepts, and multilevel structures. These are currently done by hand, and are likely to

be hindering performance. Consequently, automate the whole process so it can be run as an unsupervised learning process.

5. Investigate further heuristics for use in the hub finding algorithms.
6. Add the ability to investigate temporal structures.
7. Add boundary limits to the method of variable classification by average.

Additional work to enhance the results:

8. Analyse more primitives relating to strategies and plays.
9. Improve the reliability and low level motion control of the robots.
10. Incorporate short term prediction to estimate the position of the ball and players in advance. This will improve the desired positioning of the robots using the current control architecture.
11. Create and test a RoboCup simulation strategy using the RoboCup representatives.
12. Analyse pitch space using weighted Voronoi diagrams to account for differences in player and ball speeds.
13. Add further levels of abstraction into the robot football controller.
14. Investigate the effect of using representatives generated from undesirable primitives.
15. Examine other winning conditions in the Space-Time Possession Game.

Additional proving to demonstrate the effectiveness of the method:

16. Test the architectures ability to cope with classes of problems of specific interest in the field of multirobot systems. These may include scheduling, task allocation and tightly coupled task control.

17. Implement the approach on a reactive system.

18. Implement the approach on a distributed system.

19. Further demonstrate the scalability of the approach.

These last two points are currently of great interest in the community. Our approach is suitable for implementation on a distributed system. In this instance, variables would be limited to the sensors or communications available to each robot. This would reduce the number of possible variables, and simplify the search for maximal hubs. Regarding the scalability of the architecture, we have already shown how this approach works with both 11 and 5 robots. We can perform the scaling before or after generating representatives. If we do it beforehand, for a known number of robots, then it is only necessary to change the number of variables used in the analysis. Problems may occur if the number of variables becomes too high and overly affects the complexity of the hub search. If we perform the scaling after generating representatives, then we will lose information by scaling down, or create redundancy by scaling up. In our robot football example, each of our Mirosot robots had to take on the role of two RoboCup agents. Obviously they could not perform both roles completely, so some functionality was lost. Conversely, if we used Mirosot representatives in the RoboCup simulator, there would be two agents performing each role.

6.4 Closing Statement

In this thesis we have investigated a new approach to generating emergent multirobot controllers using ideas taken from complexity science. Our focus has been on abstracting multidimensional task information at multiple levels to construct a representation of the mission, which can be converted into a controller for a multirobot team. This is an alternative to the typical multirobot control architectures currently being researched and offers an approach to the problem of task decomposition.

During the course of this work we have formalised an architecture for multirobot task abstraction and control based on these principles. It is appropriate for problems which are complex, weakly-defined, multilevel, dynamic, competitive, unpredictable, and which display emergent properties.

Appendix A

Benchmarking the Performance of Real Robots

The performance of a robotic system depends on an abundance of factors, from the environment it is operating in, through the mechanical and electrical design of the robot itself, to the controller guiding its movements. The same robot will operate differently on rough terrain to on a smooth, hard laboratory floor. Whether a robot has legs, wheels, or tracks will influence its ability to move in these environments, as will its weight and size. Different sensors will cause the robot to view the environment in different ways, and change its approach to tackling each obstacle or task. The type of movement control programmed onto the robot, its decision making software, whether behavioural or deliberative, even the order of loops within the software will affect the way it behaves. This list is not exhaustive, and even if each factor only makes a small impact on the robot, the combination can have a drastic influence on how a robot operates. An example of this is the demonstrated by the motion of a small mobile robot in seemingly constant conditions (Nehmzow & Walker, 2003).

The ability of a robot to perform the same actions time and time again is its repeatability. Some robots, for example those used in car manufacture, have a high repeatability. They have precision encoders, and few joints, which enables them to be controlled very accurately time and again. Other robots, like those used in our experiments have a low repeatability. The system is highly sensitive to the

conditions, and the feedback control loop is not sufficient to reproduce exact movements time and again.

Despite our robot football system being very similar to others used around the world, we cannot expect it to perform in the same way, or for our experiments to be exactly reproduced. In order to enable a comparison, and to distinguish effects of our proposed controller from those of the platform, we must quantify the performance of the robots in a meaningful and repeatable way.

A set of tests were introduced by Johnson (1997) and Johnson et al.(1998) designed to benchmark the performance of a robot football system such as ours. They represent the most common tasks required of the robots, and are designed to enable direct comparison between similar robot football systems. These tests, and some of our own design, are used to benchmark the performance of our system prior to running our experiments.

A.1 Static Vision Calibration

Measurement and sensing are both provided for by the vision system. Using the overhead camera and associated software, we can identify the position of the ball and all robots on the pitch. In the case of the home team (that being controlled by the strategy of interest), we can also identify their orientation. These positions and orientations are used as inputs to the control software, but are also recorded for later analysis and generation of results. For these purposes, it is required that the vision system has sufficient accuracy, and that this can be measured.

Although the vision system is kept as simple as possible, with unique colours and patterns used to identify each robot and the ball, problems in accuracy still arise due to spherical aberration and parallax error.

Since the camera is located over the centre of the pitch, it views robots around the perimeter from a slight angle. Because the identification patch for the robot is positioned on its top side, the vision system will assume the robot is in a different position as shown in figure A.1. This is the parallax error. The vision system overcomes this using an algorithm to calculate a robot's actual position based on its height and distance from the centre of the pitch.

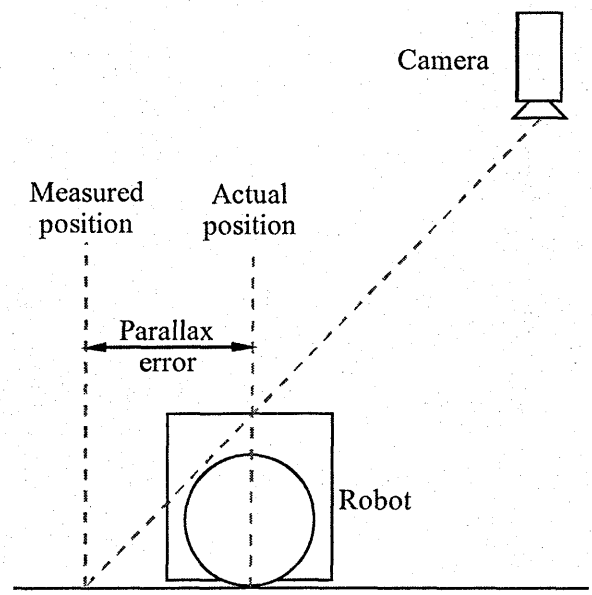


Figure A.1 Parallax Error

The height of the robot, combined with its distance from the centre of the image, results in an error in apparent position.

Spherical aberration is caused by the curvature of the lens distorting the image, making it slightly circular. In effect giving the image of our rectangular pitch curved edges, and forcing us to use a slightly modified coordinate system. Our software

includes an algorithm to measure the amount of spherical aberration, and perform a coordinate transform, enabling us to record positions on a standard Cartesian grid.

Before running any robots, a calibration sequence is always performed. This includes setting up the aforementioned algorithms, as well as calculating and compensating for the cameras orientation, which may not be directly over the centre of the pitch. We are now ready to measure the static accuracy of the vision system.

A 40 cm grid is drawn out on the pitch and robots placed at each intersection by hand. The vision system is then used to measure the positions of the robots. Measurements are recorded over 100 frames (3.3 seconds at 30 frames per second) and averages taken to compensate for fluctuations due to noise. The average positions are then compared to the grid on the pitch. Figure A.2 shows the average measured positions and orientations.

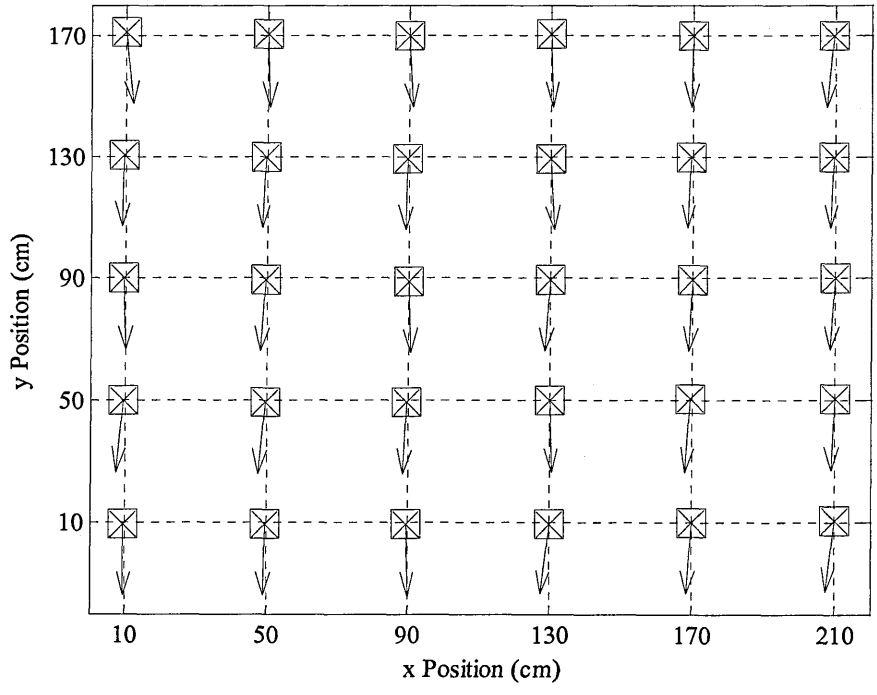


Figure A.2 Static Vision Calibration Results

Measured positions and orientations of 30 robots.

The errors evident in figure A.2 are negligible, particularly when compared to the errors measured before compensating for the effects of spherical aberration and parallax. Further analysis of the errors shows no discernable pattern, and it is considered that no further calibration is necessary. Table A.1 gives values for the most significant errors measured over all points.

Table A.1 Positional Errors Measured in the Vision System

	x Direction (mm)	y Direction (mm)	Orientation (radians)
Absolute maximum individual errors from 100 frames	9.000	9.001	0.157
Absolute maximum errors averaged by position over 100 frames	7.800	8.471	0.117
Average errors for all positions over 100 frames	-2.699	-0.556	-0.028

During the calibration, the camera was recorded capturing 3.77 mm per pixel. Given the results above, this indicates average accuracy to within 1 pixel, and worst case accuracy to within 3 pixels, which is satisfactory. It should be noted that the centre of the robot can be measured to within the size of a pixel since we are calculating the geometric centre of the set of pixels relating to the identification patch. The small errors remaining after calibration may be due to inaccuracies in placing the robots or identification patches, quantisation of the image, lighting fluctuations, or defects in the camera lens.

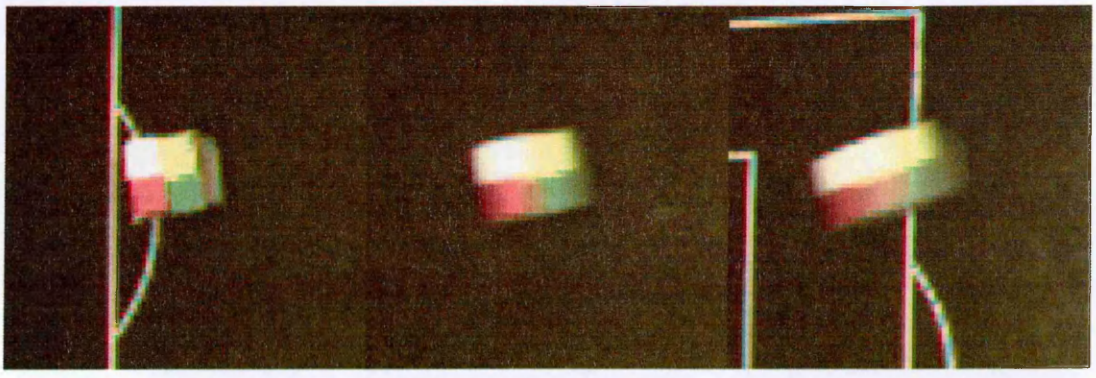
A.2 Dynamic Vision Calibration

During testing it was noticed that robots would occasionally be recorded as behaving differently to the way they did on the pitch. On some occasions the recorded

position would not change for significant periods. On others it would fluctuate between a number of distinct positions. This was due to the vision system losing track of the identification patch in the image, and usually occurred when the robot was travelling with a high velocity. This section examines the effect of movement on the accuracy of the measurements, and describes further calibration procedures which were instigated to overcome this.

Our camera has a variable shutter speed to control the amount of light reaching the CCD sensor. If the shutter is open for a significant length of time and there is movement in the scene, then the image will blur. This creates one of two effects when considering our robot footballers.

In the first instance, the robot moves through the image a short distance. Colours on the identification patch are still detectable, but the patch layout is distorted (figure A.3(b)). If the patch can still be recognised, the vision system identifies the robot as being somewhere under the centre of its image. This is an approximation of the robot's location, and is still useful, though inaccurate.



(a) Static: The position can be measured accurately

(b) Slow moving: The patch can be identified, though the shape is distorted and the measurement will not be accurate

(c) Fast moving: The colours are too faded to be identified correctly, and the position cannot be measured

Figure A.3 Effect of Motion on Patch Identification

In the second instance the robot travels a further distance. Light reflected from the identification patch is now spread over a much larger area, and so colours appear faint (figure A.3(c)). The colours may appear so faint that the vision system can no longer recognise them correctly, and the robot's position is lost. If this happens, the vision system will search for another feature resembling the robot, or failing this, presume the robot is at its last known location.

The solution to these problems is to suitably increase the shutter speed. The downside is that the faster the shutter speed, the lower the amount of incident light falling on the CCD, and the dimmer the colours will appear. Once again, this can lead to a situation where identification patches cannot be distinguished. Better lighting can be used to illuminate the pitch, though this is not an option available in competition matches. The solution requires finding a shutter speed which gives a suitable balance of image brightness and capture duration.

There is no definitive shutter speed which will overcome these effects on all systems as cameras, illumination, and identification patches vary across robot football teams. We use an experimental iterative approach to find a suitable value for our setup.

We begin by calibrating the identification colours in the vision system under a low level of illumination. When we are confident that we can reliably distinguish the colours, we increase the illumination to standard levels and increase the shutter speed to bring the colours back into their calibrated regions. Next, a robot is driven at maximum velocity down the pitch and an image is recorded. If the robot is still sufficiently distorted, then the robots' maximum velocities must be limited to

prevent them becoming lost in the image. If the robot is not distorted, then the shutter speed can be incrementally reduced to improve the colour definition.

The results of our calibration allowed us to track a robot moving at full velocity (as described in the following section) by reducing the exposure time from 1/30 s to 1/50 s under an illumination of 1721 lux. Now we have sufficiently calibrated the vision system we are ready to begin testing of the robots themselves.

A.3 Velocity Testing

The first test of robot performance is a straightforward measurement of its maximum velocity in a straight line. Our robots are controlled by communicating a value for the speed of each wheel, in pulses, between -255 and 255. At this stage, the relation between this value and the actual velocity is unknown. In the experiment, the robot is programmed to move in a straight line down the pitch, with the maximum velocity increased for each run; the controller onboard the robot handles its acceleration. The velocity of the robot is subsequently measured by differentiating its position over the period at which it is no longer accelerating. Figure A.4 shows the results from this experiment.

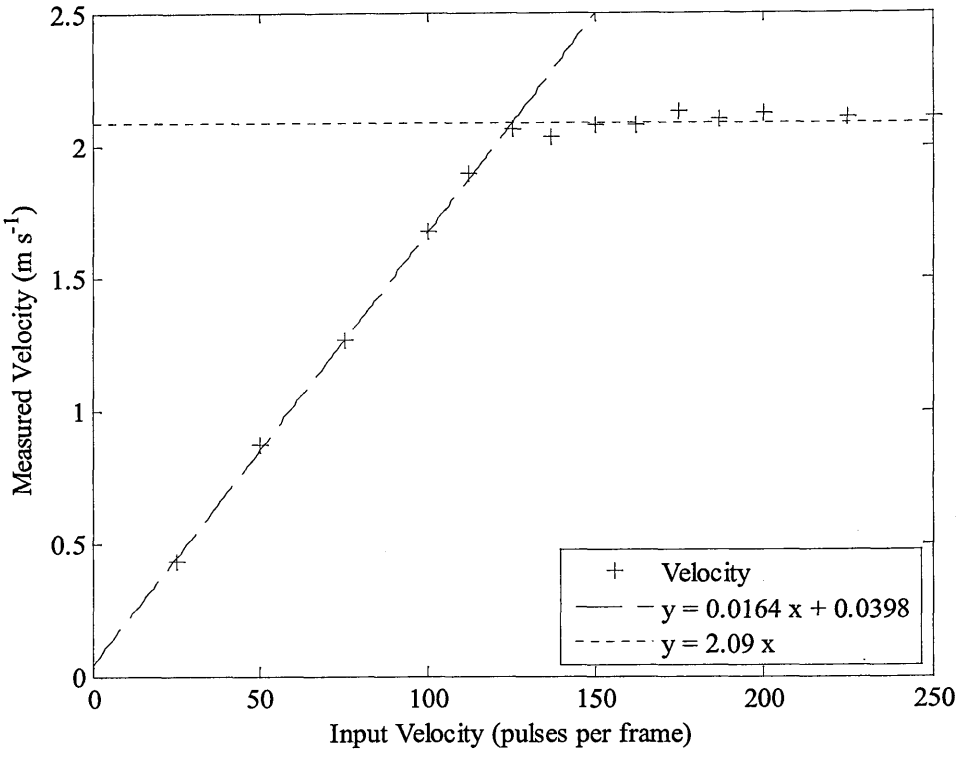


Figure A.4 Velocity Profile of a Mirosot Robot

The results show that our robot has a maximum velocity of just over 2 m s^{-1}. At lower speeds there is a linear relation between the input value and output velocity. We calculate the relationship to be approximately,

$$output\ velovity\ (m\ s^{-1}) = 0.0161 \times input\ (pulses) + 0.0392$$

These results are consistent with other robots of this type (Lepetic, Klancar, Skrjanc, Matko, & Potocnik, 2003). It is worth noting that the Faulhaber motors used in our robots are rated at an unloaded maximum of 8200 rpm. With a wheel diameter of 5.2 cm and a gear ratio of 8:1, the theoretical maximum velocity is 2.79 m s^{-1}. The difference may be due to the load, friction in the drive mechanism, wheel slip, or poor processor management. It has been suggested that this final issue may be a critical issue on our robots. Due to the way in which the onboard processor

handles the encoder feedback, it is possible for the processor to become overloaded at higher velocities, and therefore fail to handle commands in real time.

A.4 Acceleration Testing

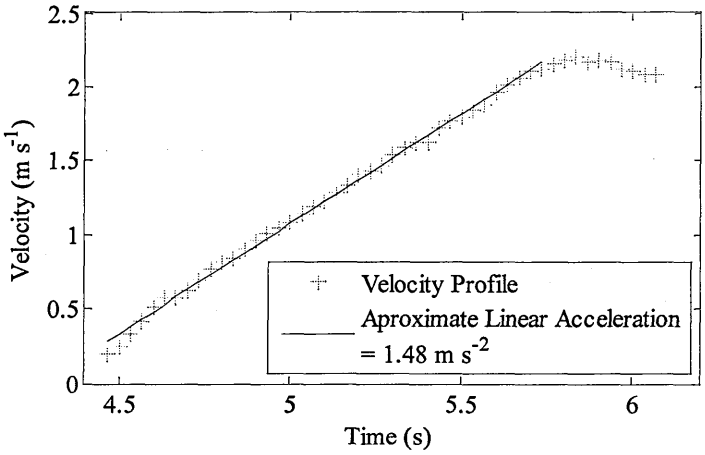
The second performance test is based on experiments undertaken by Lepetic et al. (2003), and aims to find the acceleration limits of the robots. This test is performed in two halves: first the tangential acceleration is determined for motion in a straight line, followed by the centripetal acceleration for motion in a circle. The results from these two experiments combine to give us an estimate of the robot's complete acceleration profile.

A.4.1 Tangential Acceleration

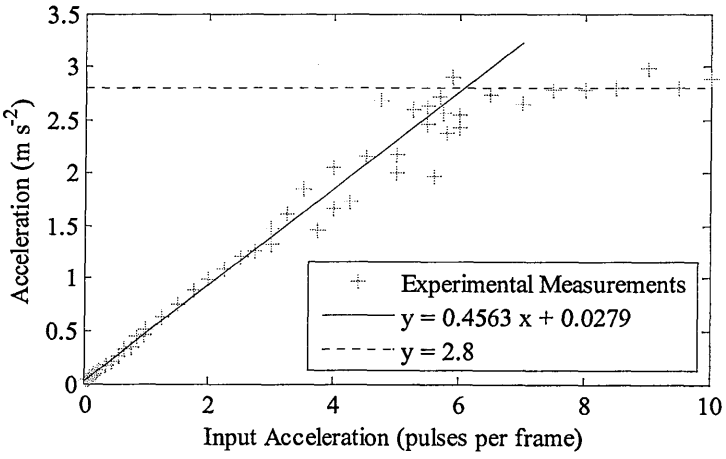
We begin by measuring a robot's ability to accelerate along a straight path, and determining the linear relationship between input and output accelerations. It may be that maximum performance requires non-linear acceleration, but to simplify matters we shall focus only on linear acceleration.

The experiment is set up as follows: A robot is placed at a starting position and programmed to accelerate at a predefined rate along a straight line. The actual acceleration is measured through the vision system. We repeat the test, each time increasing the acceleration, and record the resulting profile. To measure the acceleration, we take the second differential of the robot's position. The differentiation process exacerbates any position errors, and care is required in the analysis.

Figure A.5 shows the results of this experiment. Figure A.5(a) shows the linear acceleration of a robot in response to an increase in velocity of 3 pulses per frame, whilst figure A.5(b) shows the acceleration response of the robot over a range of inputs. We can see that the profile flattens out at around 2.8 m s^{-2} giving us the maximum acceleration of the robot, although the linear relationship begins to deteriorate at around 1.4 m s^{-2}. The limits on these values may be due to an acceleration restriction set in the on board software, limitations in processing speed, or physical aspects of the robots drive train.



(a) Acceleration in response to an increase of 3 pulses per frame



(b) Acceleration profile over a range of inputs

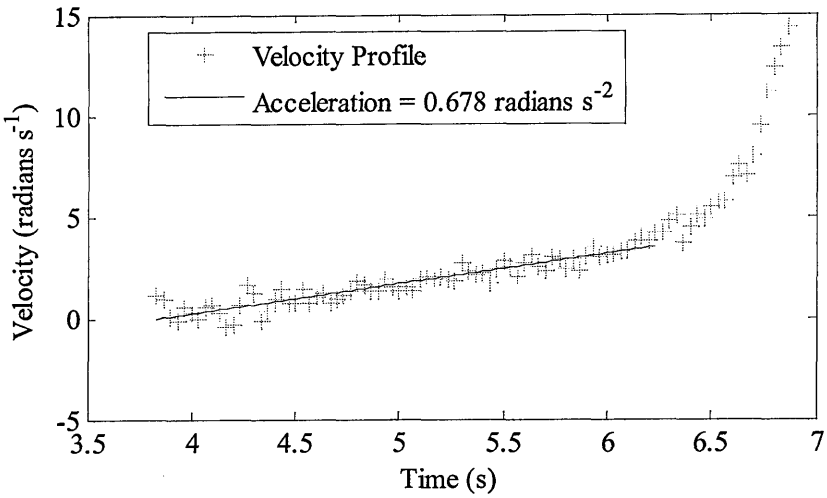
Figure A.5 Tangential Acceleration Characteristics of a Mirosot Robot

A.4.2 Centripetal Acceleration

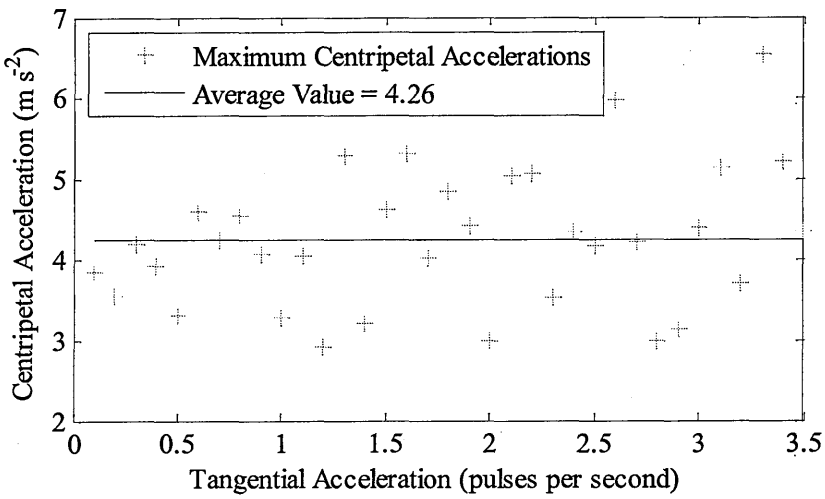
We now focus on the ability of the robot to accelerate in a circle. Again, we are looking to determine the maximum acceleration that the robot is capable of achieving.

In this experiment the robot is driven around a circle with a radius of 0.37 m. During each test the robot is programmed to perform a constant tangential acceleration. Unlike in the previous experiment where both wheels were driven with the same velocity, in this test the wheels are driven with a preset ratio between left and right wheel velocities, to obtain the circular trajectory. By gradually increasing the tangential velocity, we can find the point at which the robot drifts from the desired trajectory. This is the point at which the centripetal acceleration is at its maximum for controllable robot performance.

Figure A.6 shows the results of this experiment. Figure A.6(a) shows the velocity profile of a robot in response to an increase in tangential velocity of 1 pulse per frame, whilst figure A.6(b) shows the maximum centripetal acceleration of the robot over a range of tangential acceleration inputs. As we increase the rate of tangential acceleration, the number of measurements we can take decreases, making our linear interpolation less accurate. This accounts for the larger spread of measurements toward the right of the graph.



(a) Acceleration in response to a velocity increase of 1 pulse per frame



(b) Maximum acceleration measured over a range of inputs

Figure A.6 Centripetal Acceleration Characteristics of a Mirosot Robot

Our experiments conclude that the maximum centripetal acceleration our robots are capable of is approximately 4.26 m s^{-2}. Above this, the robot's acceleration becomes non-linear, and increases rapidly.

We would expect that as the tangential velocity increased the forces acting on the robot would cause it to slip, and the radius of the circle would increase, thus limiting the centripetal acceleration. However, in these experiments we observed the

opposite. As the centripetal velocity approached 4.26 m s^{-1} the robot began to spiral into the centre of the circle, as shown by the motion path plotted in figure A.7.

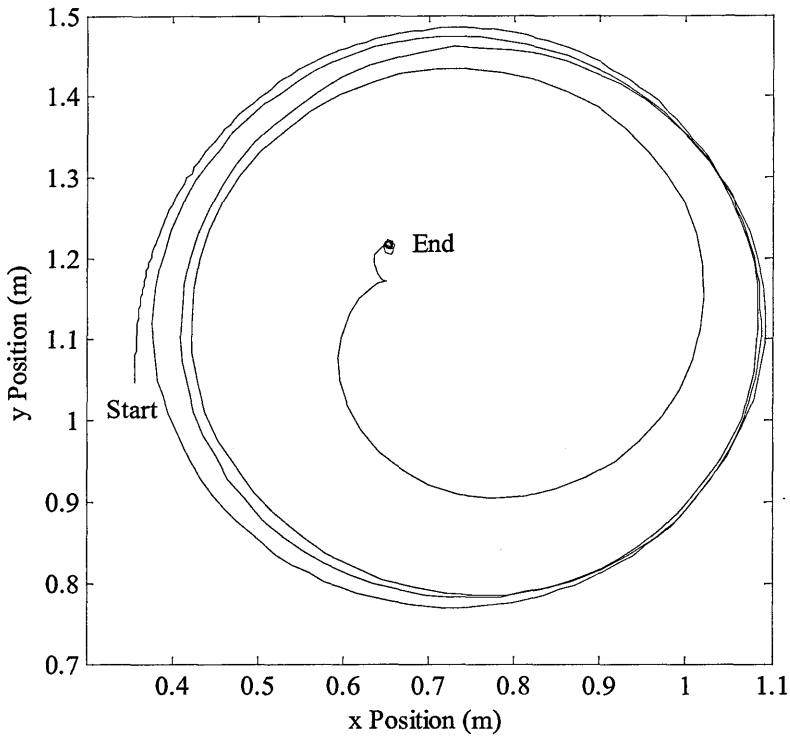


Figure A.7 Motion Trajectory under Centripetal Acceleration

Further investigation indicated this effect may be due to the movement of the centre of effort of the robot. Our robot has a square footprint with one driving wheel on two opposite sides as shown in figure A.8. To provide some stabilisation and ground clearance, the robot also has two casters, one on each of the remaining sides. The casters and wheels are set so that during normal operation there will be three points of contact with the playing surface.

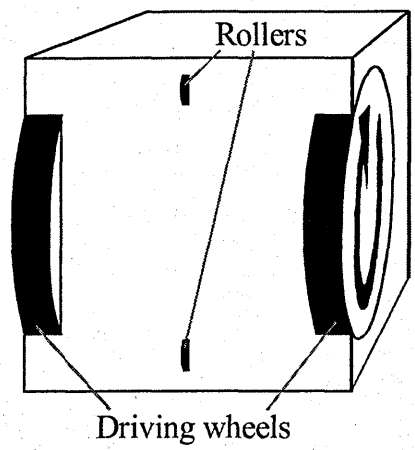
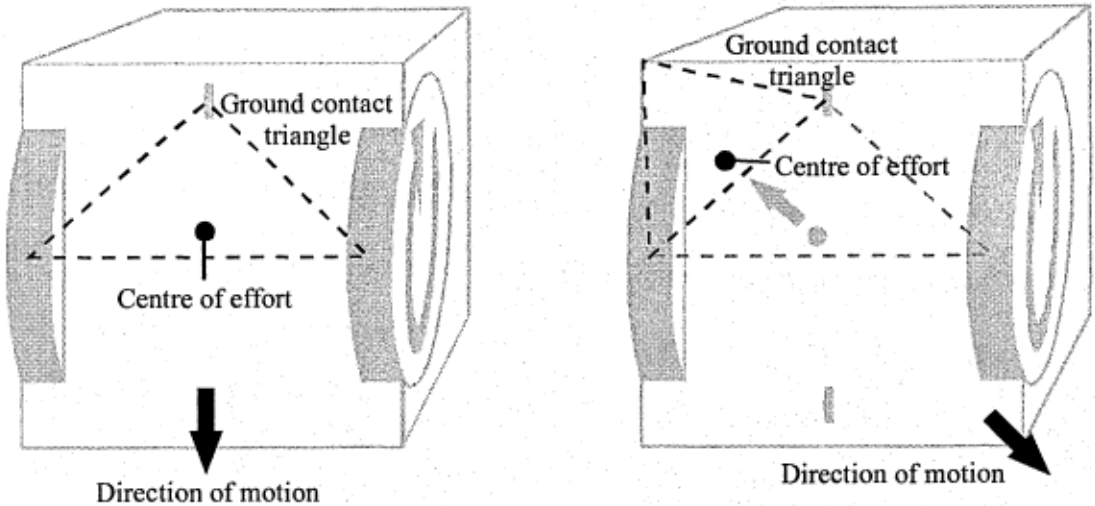


Figure A.8 Mirosot Robot Wheelbase

If the robot is stationary, the centre of effort will be somewhere above the axel, and the robot should theoretically balance on the two driving wheels. This is an unstable position, and so the robot will usually tip so that one of its castors is also in contact with the floor. If the robot moves in a straight line perpendicular to its axel, the centre of effort will move behind the axel as shown in figure A.9(a). When this happens, the robot will tip back and rest on its rear castor, and will have three points of contact with the floor as shown. If the robot is turning, centripetal forces come into play, and the centre of effort is moved laterally as well as transversely. Whilst it remains within the triangle described by the driving wheels and castor the robot will continue to function normally. However, if the centripetal forces increase, there is a chance that the centre of effort will move outside of this triangle as shown in figure A.9(b). If this happens, the inside driving wheel will no longer form part of the base triangle. Instead the robot will sit on the triangle formed by the external driving wheel, rear castor, and the corner of the chassis. Since only one driving wheel is now in contact the ground, the robot will begin to spin.



(a) Under slow linear motion the robot is stable, resting on both driving wheels

(b) Under fast angular motion the robot can lose contact between its inside wheel and the ground

Figure A.9 Effect of Motion on Centre of Effort

A.4.3 Acceleration Constraints

We have now determined the practical velocity and acceleration limits of our robots. We presume that our robots are capable to perform the same motions in both forward and reverse orientations. To remain under control we must ensure that all proposed movement remains within these constraints. Following the work by Lepetic et al. (2003) we can estimate an acceleration characteristic as shown in figure A.10. Our limits are close to those measured by Lepetic et al. for their Mirosot system.

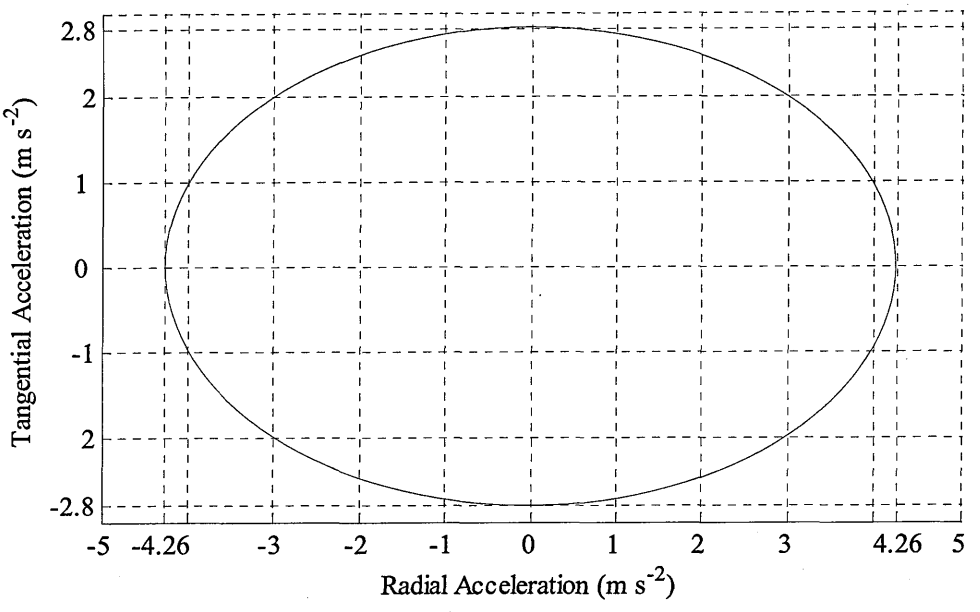


Figure A.10 Acceleration Limits of a Mirosot Robot

Acceleration and velocity limits are imposed within our strategy software to prevent forcing a robot to make an uncontrollable move.

A.5 Motion Accuracy

Now we have determined some of the limiting characteristics of the robot, we can move on to investigating its higher level movement and tactical abilities. This section describes the first benchmark suggested by Johnson et al. (1998), and examines the capacity to move between two points.

We have inherited a number of movement and shooting algorithms from our colleagues at Warwick and Plymouth. Rather than designing our own controllers from scratch, we will test the existing algorithms and select the best performing ones for use in our generated strategies. The algorithms described as GoTo functions convey the robot from a starting position 'A' to a target position 'B'. Some, but not all, have the capacity to control the angle or velocity, or both, of the robot as it

approaches the target point. We have little information on these algorithms, but do know they are mainly P, PD, or PID type controllers, varying the angle and linear velocity of the robot with respect to the target point. Some teams use trajectory follower functions for guiding their robots, which produce good results, although they are more difficult to design, and have higher computational requirements (Klancar, Matko, & Blazic, 2005).

This experiment will test the ability of each function to control the movement between a designated start and end point, 150 cm apart. We will measure the path of the robot and the duration of its movement. To enable a direct comparison between the 7 available GoTo functions, we will allow the robot to finish its movement at any angle, with zero velocity. To allow for small errors, we will assume an acceptable end position is anywhere within 2.5 cm of the designated target point. The robot is deemed to have finished its movement at the time it enters the target area, provided it does not subsequently leave the area. The robot will begin each run at 90° to the target point.

Figure A.11 shows representative routes taken by the robot for each GoTo algorithm. The duration for each of these paths is shown in table A.2.

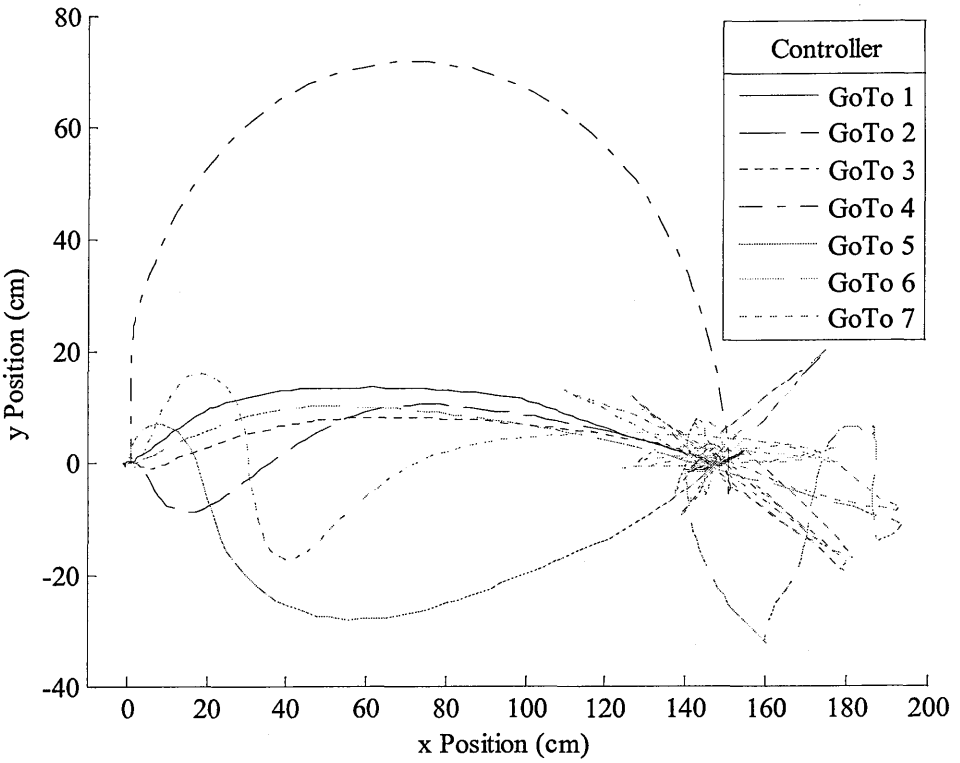


Figure A.11 GoTo Algorithm Trajectory Results

Typical paths taken by 7 different motion controllers inherited from the Universities of Plymouth and Warwick. Repeated tests of each algorithm show similar responses.

Table A.2 Measured GoTo Algorithm Durations

Algorithm	Time taken to reach destination (s)
GoTo 1	2.97
GoTo 2	3.33
GoTo 3	4.43
GoTo 4	4.17
GoTo 5	4.90
GoTo 6	7.50
GoTo 7	22.63

Of the 7 algorithms, only 1-3 give suitable trajectories. The other paths overshoot the target point, fail to settle, or take exaggerated routes. This is mainly due to the poorly calibrated gain parameters in those algorithms. For our experiments we desire fast, precise movement. From the results shown above, we select GoTo algorithm 1 for its trajectory, speed and accuracy.

Now we have selected our GoTo algorithm of choice, we need to measure its performance over a range of distances. We do this by repeating the above procedure 50 times for each of 10 distances, and recording the time taken to reach the target circle. The results are shown in figure A.12.

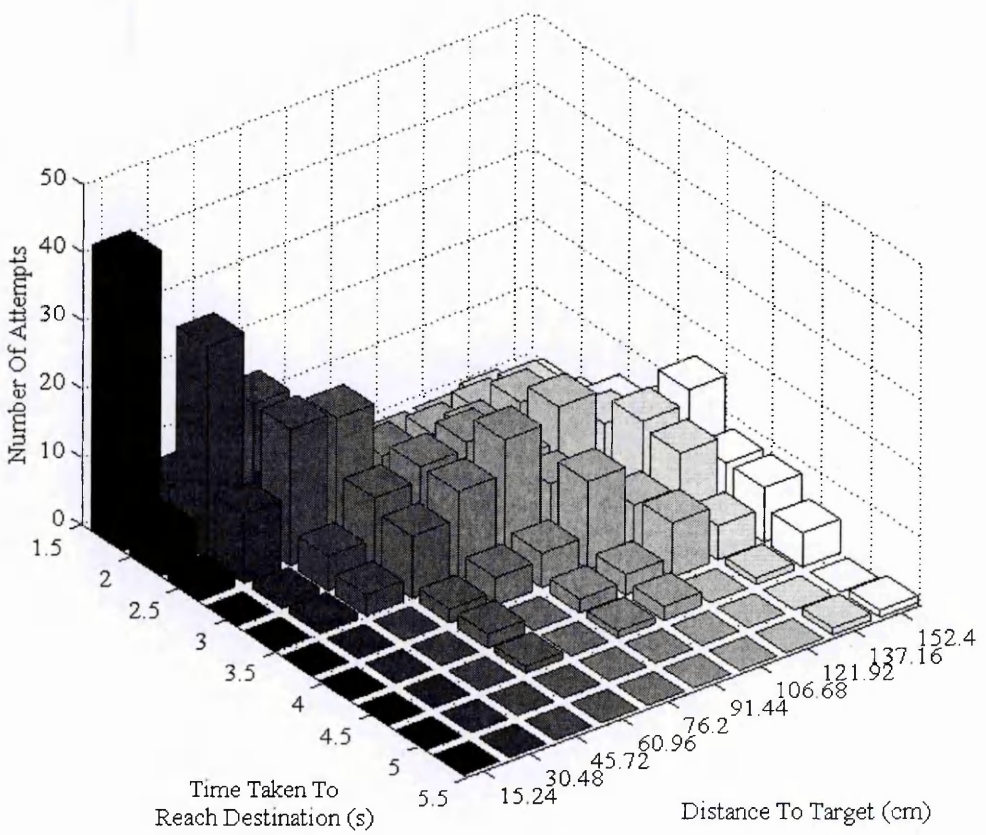
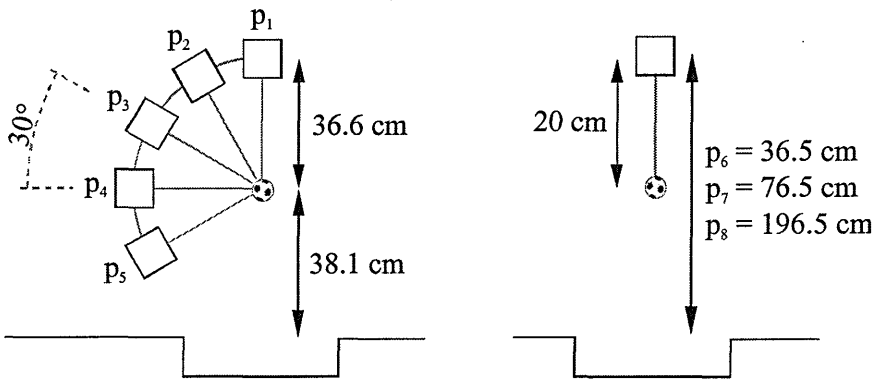


Figure A.12 GoTo Algorithm Speed Performance

We can now estimate how long it will take for a robot to move between two points on an unobstructed path, from stationary.

A.6 Striking a Static Ball

The second benchmark is a measure of how often a robot can strike a stationary ball through a target point. This will be required for a penalty situation, free balls, goal kicks and kick-offs. The experiment is divided into two halves, and is set up as shown in figure A.13. In the first part of the experiment the ball is placed on the penalty spot, and the robot is placed at 30° intervals facing the ball at a distance of 36.6 cm as shown. In the second part of the experiment the robot and ball are placed in line with the centre of the goal, with the ball placed at distances of 16.5 cm, 56.5 cm and 176.5 cm from the goal. In each instance the robot is placed 20 cm behind the ball. Both experiments are repeated, with the target at the centre of the goal.



(a) Varying strike angle (b) Varying strike distance

Figure A.13 Experimental Setup for Striking a Static Ball

From our catalogue of inherited functions we only have one controller appropriate for this task. Using this algorithm we run the experiments described above, and measure the point at which the ball crosses the front of the goal mouth, or the point at which it hits the side wall if it misses. The distribution of shots taken

from each of the 8 starting positions is given in figure A.14. Table A.3 gives the number of shots from each position, the number of shots successfully entering the goal, and the value as a percentage. Failed attempts indicate either the ball not reaching the goal line, or the robot missing the ball completely.

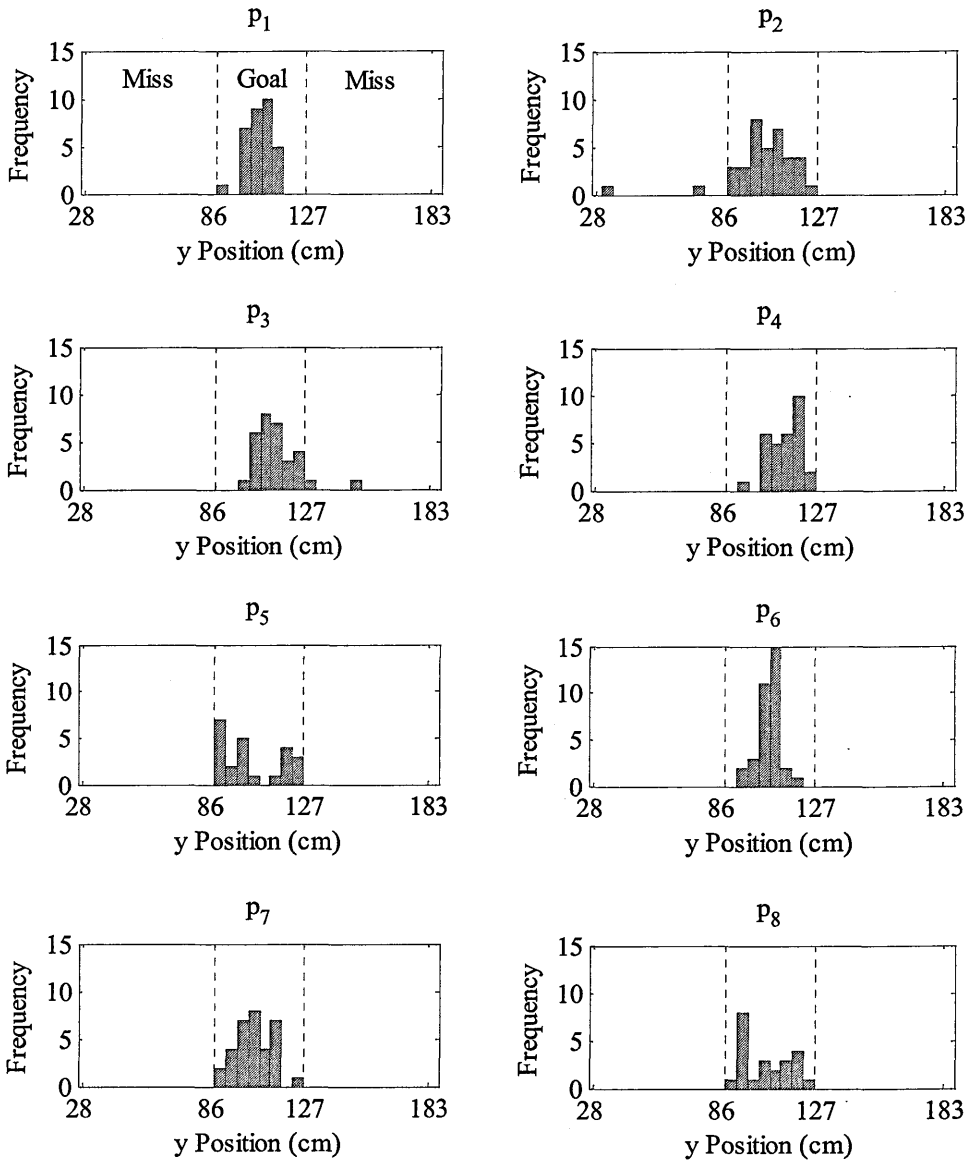


Figure A.14 Results for Striking a Static Ball

Measurements corresponding to a y position between 86 cm and 127 cm indicate a goal scored.

Table A.3 Results for Stationary Striker Performance

Starting position	Number of shots	Goals scored	Goals scored (%)
p_1	35	32	91
p_2	49	35	71
p_3	42	29	69
p_4	52	30	58
p_5	58	23	40
p_6	34	34	100
p_7	35	33	94
p_8	37	23	62

The results indicate there is a better chance of success if the robot and ball are in line with the target, and if there is only a short distance to cover. If the robot has to travel through a large angle, or over a large distance, the chances of success are much smaller. This is as we would expect. However, looking at the distribution of shots given in figure A.14, we can see that almost all are on target. It would appear that if the robot successfully approaches the ball, then there is a high chance of success. Most misses occur as a result of the robot failing to strike the ball altogether.

A.7 Striking a Moving Ball

The third benchmark investigates the ability to strike a moving ball toward a target point. This is important in shooting and passing situations. From the three striking algorithms available to us, only one seems to perform with any success. This will be used in the following test, which is set up as follows:

A robot is placed at a distance of 72 cm facing the target point, which lies at the centre of the goal mouth, as shown in figure A.15. The ball is rolled in front of the robot, which attempts to strike the ball at the target. We measure the motion of the ball and robot and calculate two sets of results. The first is the point at which the ball crosses the goal mouth, and is shown as a distribution in figure A.16. The second is a measure of the ball's velocity before being struck toward the target. This is split into velocity distributions for ball hits and misses, and is shown in figure A.17.

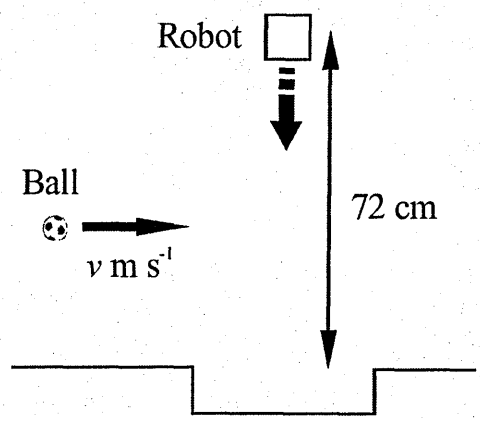


Figure A.15 Experimental Setup for Striking a Moving Ball

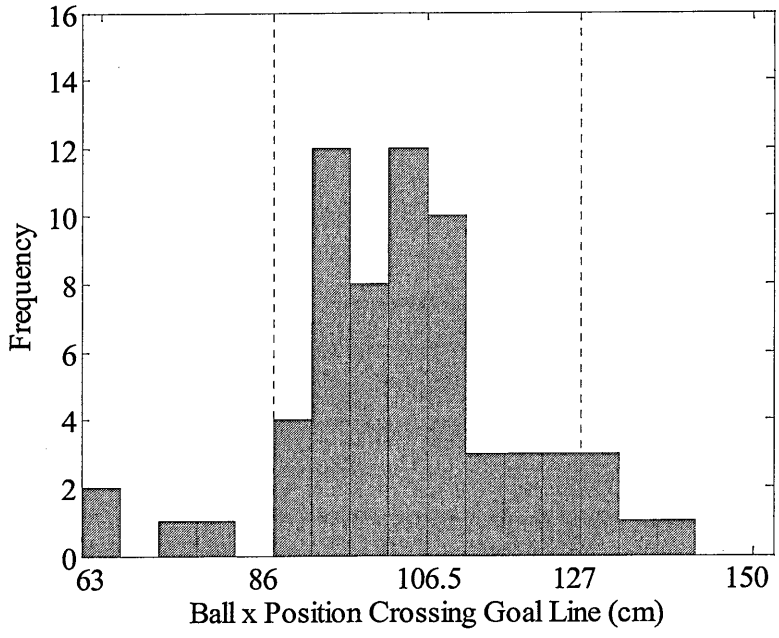


Figure A.16 Results for Striking a Moving Ball I

The measured position of the ball as it crosses the ball line, on the 64 occasions when the robot successfully struck the ball.

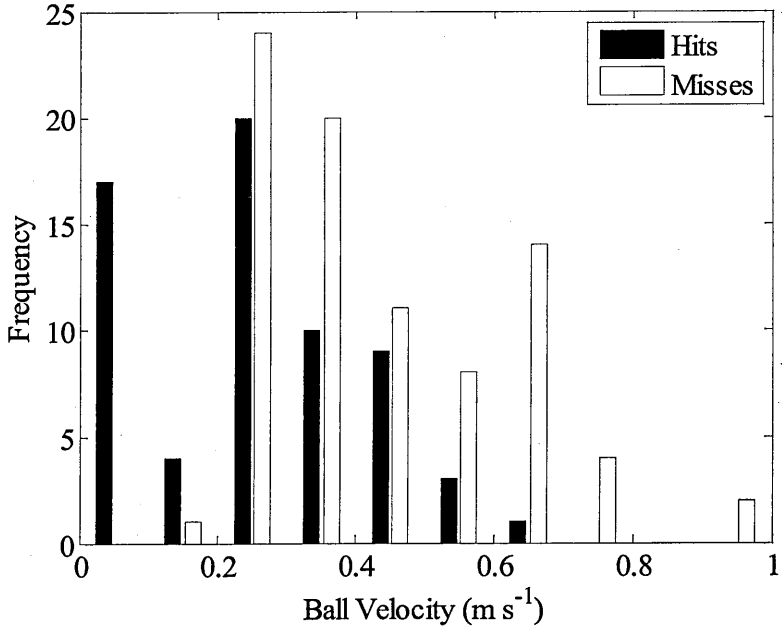


Figure A.17 Results for Striking a Moving Ball II

Shows the velocity of the ball at the impact point for all 149 attempts.

We conducted 149 tests. The robot hit the ball on 64 of these, striking it into the goal on 55. We can see from figure a.17 that for ball velocities over 0.2 m s^{-1} the robot is more likely to miss than hit the ball. Below this velocity the robot is more successful in its task. However, the results for velocities under 0.1 m s^{-1} are misleading, since they represent attempts where the robot stopped the ball by blocking its path, before taking its shot. This occurred when the robot reached the intercept point before the ball.

Our robot controller contains a filter to predict the motion of the ball and allow it to intercept it. However, the filter coefficients are only valid for a small range of ball velocities. This limits the robots performance, and enables it to only correctly intercept the ball on a limited number of occasions.

At the ball velocities tested above, the robot only has a success rate of 43% for hitting the ball, and only 37% for hitting the ball in a cone of 0.55 radians. Consider a robot attempting to pass the ball to a waiting robot. If the passer is required to pass the ball within a cone of 0.55 radians to a receiver, who is lined up in front of the goal, then the chance of successfully scoring by this tactic is only 13.7%, as two successive hits are required. Given that the accuracy of the direction of hit may need to be narrower, the chance of much faster ball speeds, and interference of other players, the likelihood of being able to perform one pass, let alone a string of passes, is extremely small.

A.8 Ball Passing in a Triangle

The final benchmark suggested by Johnson et al. (1998) is that of continually passing a ball in a triangle between three robots. This measures the ability of the

robots to perform controlled passing. Given our robots poor success rate in simply striking a moving ball, we have omitted this experiment from our tests.

A.9 Summary

We have collated a set of benchmark tests by which we can measure and compare robot football systems. We have described the methods, useful measurements, and the results for our particular system. In terms of our the experiments conducted in this thesis, the information is necessary for differentiating between problems occurring due to the existing low level robot control, and problems in our new strategic controllers.

The significance of this work to the wider field of robotics is in its use as a comparative set of measurements. Using these tests we can evaluate the performance of any robot football team at a number of levels, not just at the overlying goal-scoring level focused on in competitions. By using these methods, we can investigate what makes teams different in terms of performance.

Until recently the factor which seemed to best define a teams overall performance was the ability of its vision system. We can now measure this above the level of frame rates and resolution, which typically dominate these discussions, and instead focus on the accuracy of the camera combined with the software and lighting. By using the methods described here to compare teams in more detail, we can identify the best aspects of different teams. These can then be collated to construct the best overall system.

References

Ahn, H.-K., Cheng, S.-W., Cheong, O., Golin, M., & Oostrum, R. (2001). Competitive facility location along a highway. In Proc. *7th Annual International Computing and Combinatorics Conference*, Guilin, China, pp.237-246.

Alami, R. (2005). Multi-robot cooperation: architectures and paradigms. In Proc. *Journées Nationales de la Recherche en Robotique JNRR '05*, Guidel, France.

Alami, R., & Botelho, S. C. (2002). Plan-based multi-robot cooperation. *Lecture Notes In Computer Science; Revised Papers from the International Seminar on Advances in Plan-Based Control of Robotic Agents*, vol 2466, pp.1-20.

Alami, R., Fleury, S., Herrb, M., Ingrand, F., & Robert, F. (1998). Multi-robot cooperation in the MARTHA project. *IEEE Robotics & Automation Magazine*, vol 5(1), pp.36-47.

Alvaro, C., Freedman, H. G., & Gonzalo, M. (2006). How Spiritual Machine works. In Proc. *FIRA RoboWorld Congress*, Dortmund, Germany, pp.175-178.

Arkin, R. (1987). Motor schema based navigation for a mobile robot: an approach to programming by behavior. In Proc. *IEEE International Conference on Robotics and Automation*, pp.264-271.

Arkin, R. C. (1990). Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and Autonomous Systems*, vol 6, pp.105-122.

Arkin, R. C., Endo, Y., Lee, B., MacKenzie, D., & Martinson, E. (2003). *Multistrategy learning methods for multirobot systems*. Mobile Robot Laboratory, College of Computing, Georgia Tech, Atlanta, GA. Unpublished.

Asama, H., Matsumoto, A., & Ishida, Y. (1989). Design of an autonomous and distributed robot system: ACTRESS. In Proc. *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Tsukuba, Japan, pp.283-290.

Asama, H., Ozaki, K., Ishida, Y., Habib, M. K., Matsumoto, A., & Endo, I. (1991). Negotiation between multiple mobile robots and an environment manager. In Proc. *Fifth International Conference on Advanced Robotics ICAR '91*, Pisa, Italy, pp.533-538.

Asama, H., Ozaki, K., Ishida, Y., Yokota, K., Matsumoto, A., Kaetsu, H., et al. (1994). Collaborative team organization using communication in a decentralized robotic system. In Proc. *IEEE/RSJ/GI International Conference on Intelligent Robots and Systems IROS '94*, Munich, Germany, pp.816-823.

Asama, H., Ozaki, K., Itakura, H., Matsumoto, A., Ishida, Y., & Endo, I. (1991). Collision avoidance among multiple mobile robots based on rules and communication. In Proc. *IEEE/RSJ International Workshop on Intelligent Robots and Systems IROS '91*, Osaka, Japan, pp.1215-1220.

Atkin, R. H. (1974). *Mathematical Structures in Human Affairs*. London: Heinemann Educational Books.

Atkin, R. H., Hartston, W. R., & Witten, I. H. (1976). Fred CHAMP, positional-chess analyst. *International Journal of Man-Machine Studies*, vol 8, pp.517-529.

Aun, L. H., Lin, H. L., Quiang, C. Z., & Seng, C. F. (2005). ARICC 3rd generation robots for Mirosot. In Proc. *FIRA RoboWorld Congress*, Singapore.

Balch, T. (1999). Reward and diversity in multirobot foraging. In Proc. *16th International Joint Conference on Artificial Intelligence IJCAI '99 Workshop: Learning About, From and With other Agents*, Stockholm, Sweden.

Barfoot, T. D., & D'Eleuterio, G. M. T. (2001). Multiagent coordination by stochastic cellular automata. In Proc. *Seventeenth International Joint Conference on Artificial Intelligence*, Seattle, Washington.

Barnes, D. P., Ghanea-Hercock, R. A., Aylett, R. S., & Coddington, A. (1997). Many hands make light work? An investigation into behaviourally controlled co-operant autonomous mobile robots. In Proc. *1st International Conference on Autonomous Agents*, Marina del Rey, pp.413-420.

Behring, C., Bracho, M., Castro, M., & Moreno, J. A. (2000). An algorithm for robot path planning with cellular automata. In Proc. *Fourth International Conference on Cellular Autómata for Research and Industry ACRI 2000 in Theoretical and Practical Issues on Cellular Automata*, pp.11-19.

Botelho, S. C., & Alami, R. (1999). M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In Proc. *IEEE International Conference on Robotics and Automation ICRA '99*, Detroit, Michigan, pp.1234-1239.

Botelho, S. C., & Alami, R. (2000). Robots that cooperatively enhance their plans. In Proc. *5th International Symposium on Distributed Autonomous Robotic Systems DARS*, Knoxville, Tennessee, pp.55-68.

Bouzy, B., & Cazenave, T. (2001). Computer Go: an AI-oriented survey. *Artificial Intelligence Journal*, vol 123, pp.39-103.

Bouzy, B., & Chaslot, G. (2006). Monte-carlo Go reinforcement learning experiments. In Proc. *IEEE 2006 Symposium on Computational Intelligence in Games*, Reno, USA, pp.187-194.

Bowling, M., Browning, B., Chang, A., & Veloso, M. (2004). Plays as team plans for coordination and adaptation. *RoboCup 2003: Robot Soccer World Cup VII, LNCS*, vol 3020, pp.686-693.

Bowling, M., Browning, B., & Veloso, M. (2004). Plays as effective multiagent plans enabling opponent-adaptive play selection. In Proc. *International Conference on Automated Planning and Scheduling ICAPS '04*, Whistler, Canada, pp.376-383.

Bracho, M., Castro, M., & Moreno, J. A. (2001). A robotic architecture for RoboCup. In Proc. *Conferencia de la Asociación Española para la Inteligencia Artificial*, CAEPIA-TTIA.

Braitenberg, V. (1984). *Vehicles : Experiments in Synthetic Psychology*. Cambridge, Mass.: MIT Press.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, vol 2(1), pp.14-23.

Brooks, R. A. (1989). A robot that walks; emergent behaviors from a carefully evolved network. In Proc. *IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, pp.292-296.

Brooks, R. A. (1990). Elephants don't play chess. *Robotics and Autonomous Systems 6*, pp.3-15.

Brooks, R. A., Maes, P., Mataric, M. J., & More, G. (1990). Lunar base construction robots. In Proc. *IEEE International Workshop on Intelligent Robots and Systems IROS '90*, Tsuchiura, Japan, pp.389-392.

Búason, G., & Ziemke, T. (2003). Competitive co-evolution of predator and prey sensory-motor systems *Applications of Evolutionary Computing, LNCS*, vol 2611, pp.605-615.

Buckland, R., & Johnson, J. (1999). The Arthur C. Clarke mission: self-organising imaging robot explorers in the oceans of Europa. In Proc. *50th International Astronomical Congress*, Amsterdam, Netherlands.

Buro, M. (1993). *Methods for the evaluation of game positions using examples.* Ph.D. thesis, University of Paderborn, Germany.

Caloud, P., Choi, W., Latombe, J.-C., Le Pape, C., & Yim, M. (1990). Indoor automation with many mobile robots. In Proc. *IEEE International Workshop on Intelligent Robots and Systems IROS '90*, pp.67-72.

Campbell, M., Hoane, A. J., & Hsu, F.-h. (2002). Deep Blue. *Artificial Intelligence*, vol 134(1-2), pp.57-83.

Cao, Y. U., Fukunaga, A. S., & Kahng, A. (1997). Cooperative mobile robotics: antecedents and directions. *Autonomous Robots*, vol 4(1), pp.7-27.

Chaimowicz, L., Sugar, T., Kumar, V., & Campos, M. F. M. (2001). An architecture for tightly coupled multi-robot cooperation. In Proc. *IEEE International Conference on Robotics and Automation ICRA '01*, pp.2992-2997 vol.2993.

Cheong, O., Har-Peled, S., Linial, N., & Matousek, J. (2002). The one-round Voronoi game. In Proc. *18th Annual ACM Symposium on Computational Geometry*, Barcelona, Spain, pp.97-101.

Churchill, J., Cant, R., & Al-Dabass, D. (2001). A new computational approach to the game of Go. In Proc. *Second International Conference on Intelligent Games and Simulation*, London, pp.81-86.

de la Rosa, J. L., Oller, A., Vehi, J., & Puyol, J. (1996). Soccer team based on agent-oriented programming. In Proc. *IEEE Micro-Robot World Cup Soccer Tournament MIROSOT'96*, Korea.

Dias, M. B. (2004). *TraderBots: a new paradigm for robust and efficient multirobot coordination in dynamic environments.* Ph.D. dissertation, The Robotics Institute, Carnegie Mellon Univ., Pittsburgh, Pennsylvania.

Dias, M. B., Zinck, M. B., Zlot, R. M., & Stentz, A. (2004). Robust multirobot coordination in dynamic environments. In Proc. *IEEE International Conference on Robotics and Automation ICRA '04*, Barcelona, Spain, pp.3435-3442.

Dias, M. B., Zlot, R. M., Zinck, M. B., Gonzalez, J. P., & Stentz, A. (2004). *A versatile implementation of the TraderBots approach for multirobot coordination.*

Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania. Unpublished.

Emery, R., Sikorski, K., & Balch, T. (2002). Protocols for collaboration, coordination and dynamic role assignment in a robot team. In Proc. *IEEE International Conference on Robotics and Automation ICRA '02*, Washington DC, pp.3008-3015.

Fassi, H., Scarpettini, F., & Santos, J. (2003). Development of the UBASOT simulation team. In Proc. *FIRA Robot World Congress*, Vienna, Austria.

Fekete, S. P., & Meijer, H. (2003). The one-round Voronoi game replayed. In Proc. *8th International Workshop on Algorithms and Data Structures*, pp.150-161.

FIRA. (2002). *FIRA Middle League MiroSot game rules.* [Internet]. Available at: http://www.fira.net/soccer/mirosot/MiroSot.pdf [accessed, November 2002].

Floreano, D., & Nolfi, S. (1997). God Save the red Queen! Competition in co-evolutionary robotics. In Proc. *Second Annual Conference on Genetic Programming*, San Francisco, CA, pp.398-406.

Flynn, A. M. (1987). Gnat robots (and how they will change robotics). In Proc. *IEEE Micro Robots and Teleoperators Workshop: An investigation of micromechanical structures, actuators and sensors*, Hyannis, MA.

Garnier, S., Jost, C., Jeanson, R., Gautrais, J., Asadpour, M., Caprari, G., et al. (2005). Collective decision-making by a group of cockroach-like robots. In Proc. *IEEE Swarm Intelligence Symposium SIS '05*, pp.233-240.

Gasser, R. (1996). Solving nine men's morris. *Games of No Chance, MSRI Publications*, vol 29, pp.101-113.

Gat, E. (1992). Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In Proc. *Tenth National Conference on Artificial Intelligence AAAI*, San Jose, California, pp.809-815.

Gerkey, B. P., & Mataric, M. J. (2001). Principled communication for dynamic multi-robot task allocation. *Experimental Robotics VII, LNCIS*, vol 271, pp.353-362.

Gerkey, B. P., & Mataric, M. J. (2002). Sold!: auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, vol 18(5), pp.758-768.

Gerkey, B. P., & Mataric, M. J. (2003). Multi-robot task allocation: analyzing the complexity and optimality of key architectures. In Proc. *IEEE International Conference on Robotics and Automation ICRA '03*, pp.3862-3868.

Gerkey, B. P., & Mataric, M. J. (2004). On role allocation in RoboCup. *RoboCup 2003: Robot Soccer World Cup VII, LNCS*, vol 3020, pp.43-53.

Goldstein, J. (1999). Emergence as a construct: history and issues. *Emergence*, vol 1(1), pp.49-72.

Gravot, F., & Alami, R. (2001). An extension of the plan-merging paradigm for multi-robot coordination. In Proc. *IEEE International Conference on Robotics and Automation ICRA '01*, Seoul, Korea, pp.2929-2934.

Hackwood, S., & Wang, J. (1988). The engineering of cellular robotic systems. In Proc. *IEEE International Symposium on Intelligent Control*, Arlington, VA, pp.70-75.

Han, K.-H., Lee, K.-H., Moon, C.-K., Lee, H.-B., & Kim, J.-H. (2002). Robot soccer system of SOTY 5 for Middle League MiroSot. In Proc. *FIRA Robot World Congress*, Seoul, Korea.

Hewitt, C., Bishop, P., & Steiger, R. (1973). A universal modular actor formalism for Artificial Intelligence. In Proc. *IJCAI*, Palo Alto, California, pp.235-245.

Hsu, F. H., Anantharaman, T., Campbell, M., & Nowatzyk, A. (1990). A grandmaster chess machine. *Scientific American*, vol 263(4), pp.44-50.

Hu, H., Kostiadis, K., & Liu, Z. (1999). Coordination and learning in a team of mobile robots. In Proc. *IASTED Robotics and Automation Conference*, Santa Barbara, CA.

Huntsberger, T., Pirjanian, P., & Schenker, P. S. (2001). Robotic outposts as precursors to a manned Mars habitat. In Proc. *Space Technology and Applications International Forum*, Albuquerque, New Mexico, pp.46-51.

Huntsberger, T., Pirjanian, P., Trebi-Ollennu, A., Nayar, H. D., Aghazarian, H., Ganino, A. J., et al. (2003). CAMPOUT: a control architecture for tightly coupled coordination of multirobot systems for planetary surface exploration. *IEEE Transactions on Systems, Man & Cybernetics: Part A*, vol 33(5), pp.550-559.

Iravani, P. (2005a). *An architecture for multilevel learning and robotic control based on concept generation.* PhD. thesis, Department of Design and Innovation, The Open University, Milton Keynes, UK.

Iravani, P. (2005b). Discovering relevant sensor data by Q-analysis. In Proc. *RoboCup 2005 : Robot Soccer World Cup IX*, Osaka, Japan, pp.81-92.

Iravani, P., & Johnson, J. H. (2005). The emergence of a visual communication language in robotic football. In Proc. *Towards Autonomous Robotic Systems TAROS*, London.

Johnson, J. (1983). Hierarchical set definition by Q-analysis. Part I: the hierarchical backcloth. *International Journal of Man-Machine Studies*, vol 18, pp.337-359.

Johnson, J. (2000). Visual communication in swarms of intelligent robot agents. In Proc. *Fifth International Symposium of Artificial Life and Robotics*, Oita, Japan.

Johnson, J. (2006). Hypernetworks for reconstructing the dynamics of multilevel systems. In Proc. *European Conference on Complex Systems*, Oxford, UK.

Johnson, J., de la Rosa, J. L., & Kim, J. H. (1998). Benchmark tests in the science of robot football. In Proc. *Mirosot 98*, Paris.

Johnson, J. H. (1997). Robot football: new frontiers in control and complexity theory. In Proc. *International Conference on Systems Engineering ICSE97*, Coventry, UK.

Johnson, J. H., & Price, B. A. (2003). Complexity science and representation in robot soccer. In Proc. *RoboCup 2003: Robot Soccer World Cup VII*, Padua, Italy, pp.67-76.

Johnson, J. H., & Sugisaka, M. (2000). Complexity science for the design of swarm robot control systems. In Proc. *IEEE Conference on Industrial Electronics & Control IECON'00.*

Kai, J., Ping, L., & Beni, G. (1994). Stability of synchronized distributed control of discrete swarm structures. In Proc. *IEEE International Conference on Robotics and Automation ICRA '94*, San Diego, CA, pp.1033-1038.

Kalra, N., & Stentz, A. (2003). A market approach to tightly-coupled multi-robot coordination: first results. In Proc. *ARL Collaborative Technologies Alliance Symposium.*

Kim, J.-H., Kim, K.-C., Kim, D.-C., Kim, Y.-J., & Vadakkepat, P. (1998). Path planning and role selection mechanism for soccer robots. In Proc. *IEEE Int. conference on Robotics and Automation ICRA'98*, Leuven, Belgium, pp.3216-3221.

Kim, S. (2004). Voronoi analysis of a soccer game. *Nonlinear Analysis: Modelling and Control*, vol 9(3), pp.233-240.

Kitano, H., & Asada, M. (1998). RoboCup humanoid challenge: that's one small step for a robot, one giant leap for mankind. In Proc. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Victoria, BC, Canada, pp.419-424.

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., & Matsubara, H. (1997). RoboCup: a challenge problem for AI. *AI Magazine*, vol 18(1), pp.73-85.

Klancar, G., & Matko, D. (2005). Strategy control of mobile agents in soccer game. In Proc. *FIRA RoboWorld Congress*, Singapore.

Klancar, G., Matko, D., & Blazic, S. (2005). Mobile robot control on a reference path. In Proc. *13th Mediterranean Conference on Control and Automation*, Limassol, Cyprus, pp.1343-1348.

Kobrin, A. I., & Sinyavsky, O. Y. (2006). Research opportunities of management by movement models of the mobile robot-football player with the help of neural net algorithms. In Proc. *FIRA RoboWorld Congress 2006*, Dortmund, Germany, pp.74-78.

Kube, C. R., & Bonabeau, E. (2000). Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, vol 30(1-2), pp.85-101.

Kube, C. R., & Zhang, H. (1992). Collective robotic intelligence. In Proc. *Second International Conference on Simulation of Adaptive Behavior*, pp.460-468.

Law, J. (2005). Analysis of multi-robot cooperation using Voronoi diagrams. In Proc. *The 3rd International RCL / VNIItransmash Workshop on Planetary Rovers, Space Robotics and Earth-Based Robots*, St. Petersburg, Russia.

Law, J., & Johnson, J. (2004). Discrete dynamics and the space time possession game. In Proc. *FIRA Robot World Congress*, Busan, Korea.

Law, J., & Johnson, J. (2006). The Voronoi Game in robot coordination. In Proc. *FIRA RoboWorld Congress*, Dortmund, Germany, pp.57-62.

Law, J., & Johnson, J. (2008). Multilevel hypernetworks in the design of complex multirobot control systems. In Proc. *IEEE International Symposium on Industrial Electronics ISIE '08*, Cambridge, UK, pp.902-907.

Le Pape, C. (1990). A combination of centralized and distributed methods for multi-agent planning and scheduling. In Proc. *IEEE International Conference on Robotics and Automation*, Cincinnati, OH, pp.488-493.

Lee, D., Hwang, K., Kim, D., Chung, C., & Kuc, T. (2005). A dual camera based vision processing system of ICRO (KINGGO) for the Large League Mirosot. In Proc. *FIRA RoboWorld Congress*, Singapore.

Lemaire, T., Alami, R., & Lacroix, S. (2004). A distributed tasks allocation scheme in multi-UAV context. In Proc. *IEEE International Conference on Robotics and Automation ICRA '04*, pp.3622-3627 Vol.3624.

Lepetic, M., Klancar, G., Skrjanc, I., Matko, D., & Potocnik, B. (2003). Time optimal path planning considering acceleration limits. *Robotics and Autonomous Systems*, vol 45(3-4), pp.199-210.

Liu, L., Wang, L., Zhiqiang, Z., & Zengqi, S. (2004). A learning market based layered multi-robot architecture. In Proc. *IEEE International Conference on Robotics and Automation ICRA '04*, New Orleans, Louisiana, pp.3417-3422.

Luke, S., Hohn, C., Farris, J., Jackson, G., & Hendler, J. A. (1998). Co-evolving soccer softbot team coordination with genetic programming. In Proc. *RoboCup-97: Robot Soccer World Cup I*, Nagoya, Japan, pp.398-411.

Mackworth, A. K. (1993). On seeing robots. *Computer Vision: Systems, Theory, and Applications*, pp.1-13.

Maigret, P. (1991). Experiments in reactive planning and control with mobile robots. In Proc. *IEEE International Symposium on Intelligent Control*, Arlington, VA, pp.306-311.

Mataric, M. J., Nilsson, M., & Simsarin, K. T. (1995). Cooperative multi-robot box-pushing. In Proc. *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS '95*, Pittsburgh, PA, pp.556-561.

McMillen, C., & Veloso, M. (2006). Distributed, play-based role assignment for robot teams in dynamic environments. In Proc. *DARS 2006*, Minneapolis, MN.

Merriam-Webster Online Dictionary. (2007). *concept.* [Online]. Available at: http://www.merriam-webster.com [accessed September 11, 2007].

Messom, C. H. (1998). Robot soccer:- sensing, planning, strategy and control, a distributed real time intelligent system approach. In Proc. *The Third International Symposium on Artificial Life and Robotics AROB III '98*, pp.422-426.

Messom, C. H., Sen Gupta, G., & Sng, H. L. (2001). Distributed real-time image processing for a dual camera system. In Proc. *IEEE International Conference on Computational Robotics and Autonomous Systems CIRAS '01*, Singapore, pp.53-59.

Mitchell, T. M. (2006). *The discipline of machine learning* (Technical report No. CMU-ML-06-108). Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA.

Müller, M. (2002). Computer Go. *Artificial Intelligence*, vol 134(1-2), pp.145-179.

Nehmzow, U., & Walker, K. (2003). The behaviour of a mobile robot is chaotic. *AISB*, vol 1(4), pp.373-388.

Nitschke, G. (2006). Emergent cooperation in RoboCup: a review. *RoboCup 2005, LNAI*, vol 4020, pp.512-520.

Novak, G. (2004). Roby-go, a prototype for several MiroSOT soccer playing robots. In Proc. *Second IEEE International Conference on Computational Cybernetics, ICCC '04*, pp.207-212.

Okabe, A., Boots, B., Sugihara, K., & Chiu, S. N. (2000). *Spatial Tessellations : Concepts and Applications of Voronoi Diagrams* (2nd ed.). New York: Wiley.

Østergaard, E. H., & Lund, H. H. (2003). Co-evolving complex robot behavior. In Proc. *The 5th International Conference on Evolvable Systems: From Biology to Hardware, ICES'03*, Trondheim, Norway, pp.308-319.

Parker, C., & Hong, Z. (2002). Robot collective construction by blind bulldozing. In Proc. *IEEE International Conference on Systems, Man and Cybernetics*, Hammamet, Tunisia, pp.59-63.

Parker, L. E. (1994). ALLIANCE: an architecture for fault tolerant, cooperative control of heterogeneous mobile robots. In Proc. *IEEE/RSJ/GI International Conference on Intelligent Robots and Systems IROS '94*, Munich, Germany, pp.776-783.

Parker, L. E. (1997). L-ALLIANCE: task-oriented multi-robot learning in behavior-based systems. *Advanced Robotics, Special Issue on Selected Papers from IROS '96*, vol 11(4), pp.305-322.

Parker, L. E. (1998). ALLIANCE: an architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, vol 14(2), pp.220-240.

Parker, L. E. (2002). Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous Robots*, vol 12(3), pp.231-255.

Parker, L. E. (2003). Current research in multi-robot systems. *Journal of Artificial Life and Robotics*, vol 7, pp.1-5.

Pirjanian, P., Leger, C., Mumm, E., Kennedy, B., Garrett, M., Aghazarian, H., et al. (2002). Distributed control for a modular, reconfigurable cliff robot. In Proc. *IEEE International Conference on Robotics and Automation ICRA '02*, Washington, DC, pp.4083-4088.

Reynolds, C. W. (1987). Flocks, herds, and schools: a distributed behavioral model, in computer graphics. *SIGGRAPH Computer Graphics*, vol 21(4), pp.25-34.

Robinson, P., Wolf, J. C., Law, J. A., Oliver, J. D., Young, K. W., & Harewood-Gill, D. A. (2004). Recent Mirosot developments in the UK. In Proc. *FIRA Robot World Congress*, Busan, Korea.

Ross, W. D. (1951). *Plato's Theory of Ideas*. Oxford: Clarendon Press.

Rus, D., Donald, B., & Jennings, J. (1995). Moving furniture with teams of autonomous robots. In Proc. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Pittsburgh, PA, pp.235-242.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, vol 3(3), pp.210-229.

Schaeffer, J., Culberson, J., Treloar, N., Knight, B., Lu, P., & Szafron, D. (1992). A world championship caliber checkers program. *Artificial Intelligence*, vol 53(2-3), pp.273-290.

Schenker, P. S., Pirjanian, P., Balaram, J., Ali, K. S., Trebi-Ollennu, A., Huntsberger, T. L., et al. (2000). Reconfigurable robots for all-terrain exploration. In Proc. *SPIE Sensor Fusion and Decentralized Control in Robotic Systems III*, Boston, pp.454-468.

Simmons, R., Apfelbaum, D., Fox, D., Goldman, R. P., Haigh, K. Z., Musliner, D. J., et al. (2000). Coordinated deployment of multiple, heterogeneous robots. In Proc. *Conference on Intelligent Robots and Systems (IROS)*, Takamatsu, Japan.

Smith, R. G. (1980). The contract net protocol: high-level communication and control in a distributed problem solver. *Transactions on Computers*, vol C-29(12), pp.1104-1113.

Sng, H. L., Sen Gupta, G., & Messom, C. H. (2002). Strategy for collaboration in robot soccer. In Proc. *IEEE DELTA*, New Zealand, pp.347-351.

Solc, F., & Honzik, B. (2002). Modelling and control of a soccer robot. In Proc. *7th International Workshop on Advanced Motion Control*, pp.506-509.

Sorbello, R., Chella, A., & Arkin, R. C. (2004). Metaphor of politics: a mechanism of coalition formation. In Proc. *Forming and Maintaining Coalitions and Teams in Adaptive Multiagent Systems AAAI '04 Workshop*, San Jose, California.

Stentz, A., & Dias, M. B. (1999). *A free market architecture for coordinating multiple robots* (Technical report No. CMU-RI-TR-99-42). Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Stentz, A., Dias, M. B., Zlot, R. M., & Kalra, N. (2004). Market-based approaches for coordination of multi-robot teams at different granularities of interaction. In Proc. *ANS 10th International Conference on Robotics and Remote Systems for Hazardous Environments*.

Stilwell, D. J., & Bay, J. S. (1993). Toward the development of a material transport system using swarms of ant-like robots. In Proc. *IEEE International Conference on Robotics and Automation*, Atlanta, GA, pp.766-771.

Stone, P., & Veloso, M. (1998a). A layered approach to learning client behaviors in the RoboCup soccer server. *Applied Artificial Intelligence*, vol 12, pp.165-188.

Stone, P., & Veloso, M. (1998b). Towards collaborative and adversarial learning: a case study in robotic soccer. *International Journal of Human-Computer Studies*, vol 48(1), pp.83-104.

Stone, P., & Veloso, M. (2000). Multiagent systems: a survey from a machine learning perspective. *Autonomous Robots*, vol 8(3), pp.345-383.

Sugar, T., & Kumar, V. (1999). Multiple cooperating mobile manipulators. In Proc. *IEEE International Conference on Robotics and Automation ICRA '99*, Detroit, MI, pp.1538-1543.

Sugar, T. G., & Kumar, V. (2002). Control of cooperating mobile manipulators. *IEEE Transactions on Robotics and Automation*, vol 18(1), pp.94-103.

Sugawara, K., Sano, M., Yoshihara, I., Abe, K., & Watanabe, T. (1999). Foraging behaviour of multi-robot system and emergence of swarm intelligence. In Proc. *IEEE International Conference on Systems, Man, and Cybernetics SMC '99*, Tokyo, pp.257-262.

Thangavelauthma, J., Barfoot, T. D., & D'Eleuterio, G. M. T. (2003). Coevolving communication and cooperation for lattice formation tasks. In Proc. *7th European Conference on Artificial Life ECAL*, Dortmund, Germany.

Veloso, M., Bowling, M., Achim, S., Han, K., & Stone, P. (1999). The CMUnited-98 champion small robot team. *RoboCup-98: Robot Soccer World Cup II*, pp.77-92.

Vidal, R., Shakernia, O., Kim, H. J., Shim, D. H., & Sastry, S. (2002). Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *IEEE Transactions on Robotics and Automation*, vol 18(5), pp.662-669.

Voronoi, G. (1907). Nouvelles applications des paramètres continus à la théorie des formes quadratiques, deuxième memoire, recherche sur les parallelloèdres primitifs. *Journal für die Reine und Angewandte Mathematik*, vol 133, pp.198-287.

Walter, G. W. (1953). *The Living Brain*. London: Duckworth.

Wang, Q., Yao, J., Wang, J.-g., & Luo, K. (2005). Shooting action control of soccer robot based on genetic-fuzzy algorithm. In Proc. *FIRA RoboWorld Congress*, Singapore.

Wawerla, J., Sukhatme, G. S., & Mataric, M. J. (2002). Collective construction with multiple robots. In Proc. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, pp.2696-2701.

Weiss, N., & Hildebrand, L. (2004). An exemplary robot soccer vision system. In Proc. *CLAWAR/EURON Workshop on Robots in Entertainment, Leisure and Hobby*, Vienna, Austria.

Yu, W., Shi, H., Lim, Y. S., Huang, L. L., Chen, Y. P., & Zhou, Z. (2003). Robust posture measurement and identification of soccer robots. *Measurement Science and Technology*, vol 14, pp.1640-1647.

Zlot, R. M., & Stentz, A. (2003). Market-based multirobot coordination using task abstraction. In Proc. *International Conference on Field and Service Robotics*.