Decision Tree Learning for Intelligent Mobile Robot Navigation

by

G. H. Shah Hamzei

A Doctoral Thesis

Submitted in partial fulfilment of the requirements

for the award of

Doctor of Philosophy

of

Loughborough University

Department of Electronic and Electrical Engineering

October 1998

Synopsis

I think,
therefore I am.

R. Descartes
The French Mathematician, Philosopher and Psychologist

The replication of human intelligence, learning and reasoning by means of computer algorithms is termed Artificial Intelligence (AI) and the interaction of such algorithms with the physical world can be achieved using robotics. The work described in this thesis investigates the applications of concept learning (an approach which takes its inspiration from biological motivations and from survival instincts in particular) to robot control and path planning. The methodology of concept learning has been applied using learning decision trees (DTs) which induce domain knowledge from a finite set of training vectors which in turn describe systematically a physical entity and are used to train a robot to learn new concepts and to adapt its behaviour.

To achieve behaviour learning, this work introduces the novel approach of hierarchical learning and knowledge decomposition to the frame of the reactive robot architecture. Following the analogy with survival instincts, the robot is first taught how to survive in very simple and homogeneous environments, namely a world without any disturbances or any kind of "hostility". Once this simple behaviour, named a *primitive*, has been

established, the robot is trained to adapt new knowledge to cope with increasingly complex environments by adding further worlds to its existing knowledge. The repertoire of the robot behaviours in the form of symbolic knowledge is retained in a hierarchy of clustered decision trees (DTs) accommodating a number of primitives. To classify robot perceptions, control rules are synthesised using symbolic knowledge derived from searching the hierarchy of DTs.

A second novel concept is introduced, namely that of multi-dimensional fuzzy associative memories (MDFAMs). These are clustered fuzzy decision trees (FDTs) which are trained locally and accommodate specific perceptual knowledge. Fuzzy logic is incorporated to deal with inherent noise in sensory data and to merge conflicting behaviours of the DTs.

In this thesis, the feasibility of the developed techniques is illustrated in the robot applications, their benefits and drawbacks are discussed.

Keywords: Robotics, Intelligent Navigation, Decision Tree Learning, Fuzzy Decision Trees, Fuzzy ITI, Multi Dimensional Fuzzy Associative Memory (MDFAM), Intelligent Control

Acknowledgements

What we observe
is not nature itself,
but nature exposed to our method of questioning.

Werner Heisenberg
Physics and Philosophy

As far as certain habits and traditions are concerned, I am a fairly unorthodox person. One such tradition in the academic domain is this part of my PhD thesis that I am authoring now. Technically expressed, I always used to consider this part of others' PhD theses as very predictable in terms of its contents and hence, somewhat "redundant".

However, now, after spending a number of years in carrying out this research, I have at least one good reason to be orthodox and follow the path of tradition, as far as this part is concerned, namely to have a monologue with some closest people to me throughout the accomplishment of this work.

I very much wish to express my gratitude and appreciation to my mentor Dr. David Mulvaney, for his generous, critical and constructive supervision and friendship without which I would not be able to be at the stage where I am now. I would also like to address my gratefulness to Dr. Ian Sillitoe of The University of Borås, Sweden, as my first supervisor with whom I embarked on this undertaking. My special thanks and LOVE go to

my parents and my sisters, Roya, Kathy and Zari who were my greatest source of inspiration and encouragement to educational excellence. My sincere thanks and Dankeschön goes out to Saide, Babak and Mehri who were a great fun and supported me genuinely throughout my work.

Of great acknowledgement and thankfulness, is the support of all my friends, academic and non-academic, who in one way or another inspired and entertained me scientifically and mentally. Mentioning their very names would be a great pleasure, but beyond the limited space considered in this thesis. Therefore, I define, in the following, a "crisp" set, F, that incorporates all my friends that I met in Loughborough to thank them all collectively, without ordering, numbering, or ranking.

$$F_{crisp}$$

All my friends in Loughborough,
whom I met, I laughed with,
and I had compassion with.

Finally, I need to thank my University for providing financial support for this work.

Dedication

I dedicate this work,

to my loving grandmother,

Mrs. Soghra Hamzei,

and to my loving aunt,

Mrs. Zahra Hamzei,

for their unconditional LOVE and incentives. This work is a small token of my immense LOVE and gratitude to them both.

Es möge, daß ihre Namen verewigt bleiben.

Kawe

Publications

The Following is the list of author's publications (in their chronological order) which have been produced during the time this research was carried out.

1. G.H. Shah Hamzei and D.J. Mulvaney, "Behaviour-driven decision tree switching to identify and resolve system instability in reactive robot control, *Fifth International Workshop on Advanced Robotics and Intelligent Machines*, March 1997, Salford, Manchester, UK.

2. G.H. Shah-Hamzei and D.J. Mulvaney, "System instability and oscillation resolution in reactive robotics using DT-based approach to learning", *International Conference on Artificial Intelligence and Soft Computing*, Banff, Canada, July 27 - August 1, 1997.

3. G.H. Shah-Hamzei, D.J. Mulvaney and I.P.W. Sillitoe, "Multi-layer hierarchical rule learning in reactive robot control using incremental decision trees", *The International Journal of Intelligent and Robotics Systems*, Kluwer Academic Publishers (In Press).

4. G.H. Shah-Hamzei, D.J. Mulvaney and I.P. Sillitoe," Batch-mode decision tree learning applied to intelligent reactive robot control", *IEEE Sixth International Conference on Emerging Technology and Factory Automation*, Los Angeles, September 9-12, 1997.

---

5. G.H. Shah-Hamzei, D.J. Mulvaney and P. Sillitoe, "Becoming incrementally reactive: on-line learning of an evolving decision tree array for robot navigation", *Robotica, International Journal of Information, Education and research in Robotics and Artificial Intelligence* (In Press).

6. G.H. Shah-Hamzei and D.J. Mulvaney, "Self-organising fuzzy decision trees for robot navigation: an On-line learning approach", *IEEE International Conference on Systems, Man and Cybernetics SMC'98*, October 1998, San Diego, CA, USA.

7. G.H. Shah-Hamzei and D.J. Mulvaney, "On-line learning of fuzzy decision trees for global path planning", *The International Journal of Intelligent Real-Time Automation, Engineering Applications of Artificial Intelligence, An Affiliated Journal of IFAC*, Elsevier Science Ltd. (In Press).

8. G.H. Shah-Hamzei and D.J. Mulvaney, "Fuzzy ITI for automatic generation of control rules", *The Fifth UK Fuzzy Systems, Recent Advances in and Applications of Fuzzy Systems*, 26-27 May, 1998, The University of Sheffield, UK.

Contents

Abbreviations

AGV: Automatically Guided Vehicle

AI: Artificial Intelligence

ALife: Artificial Life

ANN: Artificial Neural Networks

ART: Adaptive Resonance Theory

BR: Back Right

BL: Back Left

CL: Close

COG: Centre of Gravity method for defuzzification

DT: Decision Tree

E: East

F: Front

FAM: Fuzzy Associative Memory

FDT: Fuzzy Decision Tree

FL: Fuzzy Logic, Front Left

FLC: Fuzzy Logic Control

FR: Front Right, Front

FSM: Finite State Machine

GA: Genetic Algorithm

GMP: Generalised Modus Ponens

GMT: Generalised Modus Tollens

goal_rel_loc: Goal location relative to the robot

GoalRelLoc: Goal location relative to the robot

GR: Gain Ratio

GRL: Goal Location Relative to the robot

IFSA: International Fuzzy Systems Association

KB: Knowledge Base

KS: Kolmogrov-Smirnoff

L: Left

LB: Left Big

LE; Left

MDFAM: Multi Dimensional Fuzzy Associative Memory

MDFDT: Multi Dimensional Fuzzy Decision Tree

MISO: Multi Input Single Output

ML: Machine Learning

MOMO: Multi Input Multi Output

N: North

NE: North East

NN: Neural Networks

NW: North West

R: Right

RAM: Random Access Memory

RB: Right Big

RI: Right

RIA: Robot Institute of America

RL: Reinforcement Learning

ROM: Read Only Memory

SAMUEL: Strategy Acquisition Method Using Empirical Learning

SC: Soft Computing, Slightly Close

SE: South East

SF: Slightly Far

SL: Slightly Left

SR: Slightly Right

SW: South West

T: Temperature

TFW: Turn Forward

TLB: turn Left Big

TLE; Turn Left

TRI: Turn Right

TRB: Turn Right Big

TSL: Turn Slightly Left

TSR: Turn Slightly Right

U: Universe of discourse

TD: Temporal Difference

T-conorms: Triangular Conorms

T-norms: Triangular Norms

VC: Very Close

VF: Very Far

W: West

WallLoc: Wall Location relative to the robot

2D: Two-Dimensional

3D: Three-Dimensional

Symbols

E_i: Entropy related to the i-th branch of a decision tree

w_i: Weight related to the i-th branch of a decision tree; perception classified as world i

p_i: Probability of the i-th event or the i-th branch of a decision tree

log_2: Logarithm to the basis of 2

$F_A(x)$: The cumulative distributive function of a class A

S_i: the i-th proximity sensor of the robot

D_i: The distance associated with S_i

$\underline{h}$: The robot heading vector

$\underline{\tilde{h}}$: The robot heading unit vector

$\underline{g}$: The goal vector

$\underline{\tilde{g}}$: The goal unit vector

θ: The angle between the robot heading vector and the line connecting the robot to the goal (divergence angle)

α: The robot absolute heading angle

(X_R, Y_R): The Cartesian co-ordinates of the robot

(X_G, Y_G): The Cartesian co-ordinates of the goal

E_n: Euclidean distance between the robot and the goal at time step n

f_i: Feature i used in the training examples

f_{ij}: The j-th value of feature i

c_i: The i-th class or label used to categorise a training example

P_n : Perceptual state vector at time step n

μ_A : Membership function associated with fuzzy set A

$\varnothing$: An empty set

$I(a,b)$: Implication operation on fuzzy sets a and b

$T(x)$: Term set of variable x

$\Re_i$: The i-th rule of a given rule set $\Re$

Chapter 1

Introduction

To be, or not to be: that is the question:
whether 'tis nobler in the mind to suffer
the slings and arrows of outrageous fortune,
or to take arms against a sea of troubles,
and, by opposing, end them.

William Shakespeare
Hamlet, Act III, Scene I, 1602

1.1 Background

How human beings are able to learn, represent and reason about the physical world has, for at least many thousands of years, been subjected to exploration by scientists and philosophers. In recent years, this has led to the emergence of new scientific areas such as evolutionary computing, neural networks and fuzzy logic. These techniques have been introduced to solve complex and challenging industrial problems of the modern age by mimicking human behaviour in learning, reasoning and adapting to new environments.

Artificial Intelligence (AI) algorithms have been designed and implemented that are able to perform logical inferences on a domain of given knowledge. Once applied to a certain domain, these are able to demonstrate improved performance on repetition of the same task or even "unseen" situations (from the same population) due to their generalisation capabilities. They are said to be able to "learn". Such learning algorithms have been adopted in both physical and social sciences and applied to applications such as banking, management and engineering. There is currently extensive activity in exploiting concepts from the field of AI to conceive, design and realise intelligent control systems [1]. These systems incorporate the new scientific areas individually, or combine these into hybrid systems to exploit complementary effects.

One branch of intelligent control systems research is robotics, which, in recent decades has proved to be a suitable test-bed for AI techniques that incorporated autonomous learning and reasoning. These so-called *autonomous robots* are mechanical systems which are able to perceive their environment using a variety of sensing devices, process this information, make an appropriate decision, and act on their environment. Although today's autonomous robots and the achievements in this area are still some way from the totally autonomous robots which are the ultimate goal of researchers and engineers, they provide significant and well-developed background knowledge for the development of future artificially intelligent "beings".

1.2 Aim, motivations and objectives

The problem of designing intelligent robots operating in uncertain environments with the minimum of supervision and with the capability to interact with humans is a greatly pursued but challenging task. The present work is aimed at making a novel contribution to the body of the existing knowledge regarding intelligent and learning robots by designing and implementing of an intelligent control strategy for an autonomous mobile robot.

In view of the drawbacks of traditional approaches to robotics (to be discussed in Section 2.4), there is a clear need to develop new methods that can more effectively lead to the next generation of intelligent robots. One way this could be pursued follows from the observation of biological organisms such as ants and bees, who exhibit primitive

perceptual reflexes or *behaviours* for their survival. Such a methodology was first set out by Brooks [2,3], and was called *behaviour-based learning*, in which intelligence is organised [2] and distributed in a hierarchy.

The work presented in this thesis takes its inspirations from biological motivations, from survival instincts inherent in biological systems, and is based upon the general principles of human processes in remembering and forgetting events to build up experience and hence intelligence. This means that the objective learning system should macroscopically parallel learning processes in humans and their survival mechanism. The system should exhibit the following characteristics.

- Learn concepts and behaviours by being taught (training) as well as exploration (self-supervision).
- Learn from past experiences in an incremental fashion similar to humans.
- Make predictions (decision-making) in unseen scenarios, based on what was learned previously, in order to survive in unstructured and dynamically changing environments.
- Perform a globally tuned learning rather than use local intelligence or map-making.
- Synthesise automatically adaptive hypotheses in the form of intelligible control rules.
- Easy to implement and be computationally low-cost.
- Cope with noise and imprecision of input information to deliver appropriate predictive estimates (control actions).
- Show smooth responses to sudden environmental changes due to differing behaviours.

To develop an intelligent control architecture, this work formulates the methodology of *Hierarchical Learning and Knowledge Decomposition* in the frame of a reactive robot architecture. Intelligence is decomposed into a number of simple behaviours in a layered hierarchy. As there is little direct relevance which has been done before, the current work takes a somewhat exploratory approach to the design of a suitable intelligent control systems. To meet the aim and considering the learning technique which has led to the motivation of the current work, the objectives can now be stated as follows:

- To develop an intelligent control technique based on hierarchical learning and knowledge decomposition.
- To test the feasibility of the technique and qualitatively assess its performance in a simplified robot environment.
- To extend the application of the technique to a realistic mobile robot environment.
- To compare the performance of the technique with existing approaches and suggest refinements.
- To introduce fuzzy logic to provide smoother transitions between the levels in the hierarchy.

1.3 Structure of the thesis

This thesis is the documentation of the research carried out to design and implement an intelligent control strategy for an autonomous robot. The thesis has been divided into nine chapters including the current chapter. In the following, the structure of the thesis and the arrangement of its chapters in the order they appear in the thesis along with the issue of discussion in each chapter are briefly described. To allow the reader to select their own path through the thesis, Figure 1.1 provides a graphical "road-map" of the thesis showing its division between theory and background, application and evaluation.

Chapter 2 first provides a concise introduction to the most commonly used learning techniques, and provides further relevant references for the interested reader. It then presents an up-to-date review of past work and research carried out in the design of intelligent robots and systems, discussing their efficiency, drawbacks and their relevance to the current work.

Chapter 3 presents a taxonomical view of machine learning techniques with emphasis on learning from examples, which is a paradigm for supervised learning. It then introduces and discusses thoroughly the concept of decision tree learning, one of the most common representations of inductively drawn knowledge. The related terminology, semantics and the underlying principles of decision trees and their construction are discussed and explained. References to external literature are also provided if the reader requires details or derivations. A simple worked numerical example is used to demonstrate the

fundamentals of feature partitioning in growing a decision tree for classification. The various types of decision trees are introduced, their modes of operation are discussed, and some recent developments and implementations of learning from examples (decision tree and rule set generation) are introduced.

Chapters 4 and 5 discuss the development of the hierarchical learning technique based on decision trees. Chapter 4 implements and tests the algorithm in an off-line mode and makes a number of simplifying assumptions about the robot and its environment to make a qualitative assessment on the feasibility and applicability of the approach. Chapter 5 concentrates on developing and testing the learning approach in an on-line and incremental fashion to build an array of adaptive hypotheses. It uses realistic assumptions about both the robot and the environment.

Chapter 6 gives an introduction to the history as well as the theory and foundations of fuzzy sets and fuzzy logic with an emphasis on its application to control engineering and intelligent and multi-strategy systems. This chapter delivers the necessary mathematical background for the understanding of an application chapter, namely chapter 7. This chapter introduces a novel approach to the design of a hybrid learning system, namely fuzzy decision trees, for the automatic generation of linguistically formulated control rules. This chapter also introduces the concept of multi-dimensional fuzzy associative memories which are able to encode as well as fuse multi-variable inputs into lower dimensions.

Results and a comparison of the learning techniques introduced in this thesis, are presented in chapter 8. This chapter also reviews and discusses the contribution of this work, and identifies areas of application and research directions along which future work may be conducted.

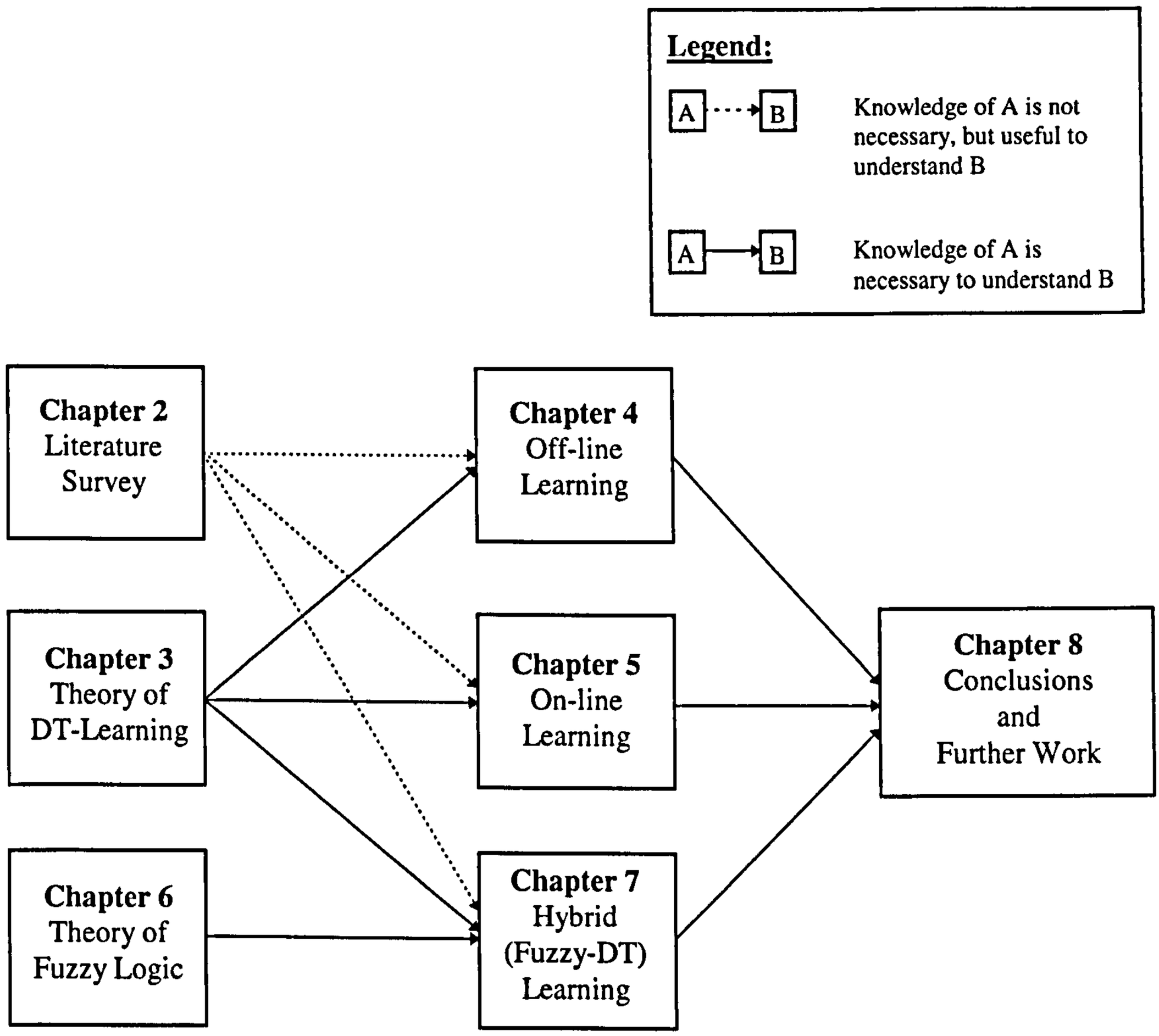


Figure 1.1 The general structure of the thesis and the relationship between chapters.

References

[1] Irwin, "Computing and Control: Back to the Future", *IEE Computer and Control Engineering Journal*, Vol. 9, No. 1, February 1998, pp. 39-45.

[2] R.A. Brooks, "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1, March 1986, pp. 14-23.; also MIT AI Memo 864, September 1985.

[3] R.A. Brooks, "New Approaches to Robotics", *Science*, Vol. 253, September 1991, pp.1227-1232.

Chapter 2

Literature Survey:

Learning and Intelligent Robots

That which is apprehended by intelligence and reason
is always in the same state, but
that which is conceived by opinion with the
help of sensation and without reason, is
always a process of becoming and
perishing and never really is.

Timaeus, in the "Dialogues of Plato"

The objective of this literature survey is to provide an up-to-date review of approaches to robot learning techniques, compare their performance, and draw conclusions on their drawbacks, efficiency and applicability to industrial problems, especially in the area of mobile robotics and control.

This survey should provide a broad basis in understanding and efficient deployment of learning techniques towards the development of intelligent robots. It discusses learning, provides an epistemological view of intelligence in humans and artificial agents, and introduces the robot architectures, described in the literature. Particular emphasis is placed on the review of past and recent work pertaining to robot learning.

2.1 Introduction

Complex computer software and both fast and highly parallel hardware have been designed and developed to meet the emerging needs of high performance, robust, reliable and automated environments. This need is, in particular, perceived in working areas where the presence of human beings is either expensive or hazardous. Mobile robots have received great attention in recent years due to their potential to fulfil these demands, as their presence is becoming increasingly common in the manufacturing industries.

Mobile robotics has been employed in such diverse fields as: plant control stations, underwater research and remote disposal of nuclear waste. If mobile robots are to be integrated into human environments, they must be able to closely parallel human learning, by being able to learn from experience. They must also be able to communicate and be safe [1]. Machine learning techniques can be utilised to satisfy these challenging demands, and overcome the inherent limitations of static behaviours resulting from purely hand-coded programs. In principle, machine learning techniques can allow a robot to adapt successfully its behaviour in response to changing circumstances without the intervention of a human programmer [2]. Next chapter discusses in more detail the paradigm of learning from examples which underlies the current work.

2.2 Epistemology of learning and intelligence

Although, an appropriate definition of intelligence has been a subject of much controversy [3], sciences such as neuro-anatomy, neuro-physiology, neuro-pharmacology, psycho-physics, anthropology and behavioural psychology have revealed much about the mechanisms and function of intelligence. Biological paradigms borrowed from humans and animals such as learning, adaptation, self-repair, social behaviour, evolution and cognition have been the subjects of extensive research, including that associated with building

intelligent machines such as intelligent robots. These should not only be able to exhibit, at least in part, the innovation and the degree of achievement of humans in creating intelligent "beings", but also they should assist humans in performing a variety of tasks. Imparting intelligence to man-made beings or artificial agents is a challenging task for scientists and engineers. Engineers may be interested in developing an analytical model of a physical system in order to learn how to manipulate the system efficiently. However, if this is impossible due to the intrinsic complexities of the physical system or due to the lack of available data, they may resort to human cognition, common sense and expertise to resolve the problem. This is, for example, the founding issue of the well-known fuzzy theory and fuzzy logic, introduced by Zadeh [4].

Intelligence and learning are complementary terms, in the sense that a system (biological or artificial) is called *intelligent* if it is able to learn, and a *learning* system is assumed to exhibit intelligence. However, these terms find different definitions and interpretations in different disciplines. For psychologists, intelligence might be defined as a behavioural strategy that gives each individual a means for maximising the likelihood of propagating its own genes [3]. From an engineering point of view, Albus in [3] defines intelligence as "the ability of a system to act appropriately in an uncertain environment, where appropriate action is that which increases the probability of success, and success is the achievement of behavioural sub-goals that support the system's ultimate goal. Both the criteria of success and the system's ultimate goal are external to the intelligent system. For an intelligent machine system, the goal and success criteria are typically defined by the designers, programmers, etc.. For intelligent biological creatures, the ultimate goal is gene propagation, and success criteria are defined by the process of natural selection".

Albus is of the opinion that "intelligence can be observed to grow and evolve ... In natural systems, intelligence grows over the life time of an individual, through maturation and learning...". He also maintains the opinion that "learning is not required to be intelligent, only to become more intelligent as a result of experience.... It is, however, assumed that *many* creatures can exhibit intelligent behaviour using instinct, without having learned anything". He leaves to be inferred that intelligence is instinctive, on the other hand, he believes that intelligence is subject to experience. The author finds these statements of

Albus in [3] rather paradox and somewhat confusing in trying to present a definition of intelligence. The author argues that intelligence (both in biological and artificial systems) is the consequence of learning, and learning is experience, and experience is *History*. The author also believes that if intelligence were instinctive, *all* creatures would have to be intelligent and not "*many* creatures", as stated by Albus, because *all* creatures enter history with a set of instincts or as stated by K. Kautsky in [5] "a set of inborn drives", namely self-preservation drive, sex drive and a social drive.

The definition of learning introduced by Simon in [6], is the closest definition in the context of this work, namely robot learning. His definition is as follows: "learning is any change in a system that allows it to perform better the second time on repetition of the same task or on another task drawn from the same population". The system showing this ability is called an intelligent system. This definition comes close to the author's interpretation of intelligence as, "intelligence is the capacity of exhibiting altered behaviour because of experience, in favour of an ultimate goal, where the ultimate goal can be, for example, gene propagation (biological systems) or optimising a cost function (artificial or machine systems)."

As a contribution towards the ultimate target of introducing a coherent theory of intelligence encompassing the biological and machine instantiations, Albus in [3] introduces and elaborates a theoretical model of intelligence with the aim of combining all separate but related areas of knowledge and expertise into a unified framework. This should eventually help engineers, in particular, build intelligent systems.

2.3 Robot technology

Technical aspiration, combined with economic rationale, including mass production, cost and time efficiency, have together brought the concept of automation and automatic control, with humans playing an ever decreasing role in the physical manipulation of production systems. Although the idea of total automation is not new and dates back to the post war era, it initiated a new academic and engineering discipline, namely *robotics*.

2.3.1 What is a robot?

This term was first used during 1920's and 1930's, following the appearance of a play by the Czech author Karel Capek, called *RUR* (Rossum's Universal Robots) [7]. In the play, small, artificial and anthropomorphic creatures strictly obeyed the instructions of their master. These creatures were called *robot*. This word is derived from the Czech word *robota*, meaning "forced labour".

There exists, however, various definitions, partially complementarily, in the literature for the term robot. For example, the definition supplied by the Robot Institute of America (RIA) is as follows [7]:

> "A programmable and multifunctional manipulator, devised for the transport of materials, parts, tools or specialised systems, with varied and programmed movements, with the aim of carrying out varied tasks".

Though, not perfect, the definition above reflects some of the capabilities of today's robots. These definitions are all subject to changes, developments and advances in the field of robotics.

2.3.2 Why is there a need of learning in robotics?

At their introduction, the versatility and flexibility of the robotic systems markedly distinguished them from automated machines. It was soon realised that a wide range of physical tasks could be entrusted to robots, especially, the repetitive ones. It was also noticed that robots have the potential of replacing a large number of existing machines, and also human operators to an extent.

The majority of early robots, especially the industrial robots, were task-specifically programmed machines, yet capable of performing a variety of tasks (versatility). They were limited by the fact that they were not able to react to changes in their environments. To achieve the aspiration of total automation, a new era in the development of robotic systems began with the aim of achieving flexible automation by building mechanical systems that could execute anthropomorphic functions and mimic the behaviour of the biological systems (such as humans or animals). The aim was to build *intelligent robots*.

The development of intelligent robots arose from the following considerations:

- The range of application and ability of programmed robots was limited and these could be significantly enhanced using sensing devices.
- Although many tasks performed by humans seemed to be repetitive in nature, and hence directly replaceable by programmed robots, they required constant adjustments due to slight changes in the working environment.

In contrast to programmed robots whose operation is limited to structured environments, intelligent robots should be capable of reacting (in favour of some goal) to unpredictable changes in unstructured environments. Consequently, intelligent robots need to exhibit decision-making capabilities aimed at mimicking the process of decision-making in biological systems. An intelligent robot is expected to demonstrate the following characteristics [8]:

- *Perceive* the environment
- *Reason* about the perceived information
- *Make decisions* based on this perceptions, and
- *Act* according to some strategy at a very high level.

The question of which aspects of the robot system should be physically designed, which should be left to the robot to learn, and to what extent the above features can be realised, has been the issue of extensive research and development over the past decades. This has lead to the introduction and development of a number of diverse approaches in the development of robot learning techniques and control architectures which are reviewed in the following sections.

2.4 Basic robot architectures

The control architecture of an autonomous robot determines the way it perceives its environment, the way it reasons and how it acts in reaching an ultimate goal. The spectrum of the basic robot architecture includes the following strategies:

- *Planner-based*
- *Purely reactive*
- *Behaviour-based*

The first two strategies provide the architectural extremes [9], while the behaviour-based approach falls somewhere in between. In the following subsections, these basic robot architectures are outlined and their characteristic constraints are discussed.

2.4.1 Planner-based architectures

This type of control strategy constructs and maintains a model of the world as a centralised representation. Path planning is performed by using the information contained in the model in order to generate the most appropriate sequence of actions, i.e. the plan. There exists various approaches to plan generation and map making of the environment. One approach, used in [10], is the generation of visibility graphs in which the vertices of the visible obstacles are scanned. An algorithm chooses then the one with the highest priority determined at the design stage. These points are generated iteratively and the robot is navigated along the line connecting these points. A somewhat similar approach to map-making of the robot environment has been taken in [11] in which fuzzy logic is used to reason about the location and the distance of perceived obstacles.

A further method for building topological maps of the environment while navigating the robot, is the potential field approach [12,13]. This is based on a virtual force field which is composed of individual repulsive force vectors issued by proximity sensors and a constant attracting force which guides the robot towards the target. Approaches based on the potential field do not always guarantee a solution to navigation, and it is likely that the robot will fall into oscillatory motions around the local minima.

Since most practical systems rely on ultrasonic sensors for distance measurements, they are susceptible to errors due to uncertainty and the noisy characteristic of the sensors. An alternative method in this category, which also reduces the noise error, is the grid representation [14,15,16] of obstacles in the environment which is derived from sensory information. In this method, a so-called certainty value is calculated (based on the repeated sensor measurements) and allocated to each grid cell according to the corresponding

distance sensor. These values are continuously updated and used for navigation. A limitation of grid-based methods is the rapid growth of information that needs to be stored to provide such a representation as the number of cells discovered by the robot increases and this limits the use of this technique in practical applications [14].

Path planning systems are often criticised as being slow and of being unable to cope with and survive in dynamic scenarios [17]. This is demonstrated by the need to re-plan a task needed if an area which was free at planning time is subsequently blocked as the robot is attempting its navigation. Typically, the robot will fail to reach the desired goal. However, due to their structural architecture, planning systems can be efficiently employed in structured and human built environments with predefined and almost stationary layouts, such as hospitals and offices.

2.4.2 Reactive architectures

Reactive systems [18,19] embed the control strategy of the agent into an array of simple perception-action stimuli. They maintain no world model and relate directly the sensor information to the appropriate responses. These action-stimuli pairs can be encoded in the form of a set of reactive rules, a table, a whole set of potential field functions, or a set of weights stored in a connectionist network.

Purely reactive strategies have proved effective for a variety of problems that can be specified at the design stage [20]. However, such strategies are inflexible at run time due to their inability to acquire information dynamically. Although analytically difficult to prove, it is commonly believed that pure reactive systems are less powerful than behaviour-based approaches [21]. Strictly speaking, this is a task specific observation, meaning that if a robot is expected to achieve certain goals at run time which are not specified at the design stage, then purely reactive systems would, in general, not be able to perform these tasks satisfactorily. For example, dynamically changing or unknown robot environments are more elegantly tackled using behaviour-based approaches. Reactive systems, however, offer greater efficiency in computation time as less information needs to be stored and processed.

2.4.3 Behaviour-based architectures

The foundations of behaviour-based robot control were biologically inspired and first set out by Brooks in [22,23]. Brooks [23] concentrated on *organising intelligence* in such a way that it is reactive to those dynamic aspects of the environment that a mobile robot experiences. Intelligence should also be able to generate robust behaviour in the presence of uncertain sensor information, an unpredictable environment and a changing world [23]. Brooks argued for the decomposition of tasks that a robot is expected to perform rather than the traditional functional decomposition of a task. The development of *subsumption* architecture [22] was the consequence of this methodology. In this approach, control layers were implemented as networks of message-passing finite state machines [23] which operate asynchronously. The behavioural competence of the robot can be improved by adding further behaviour-specific networks to those already existing. Brooks called this process *layering* in that the output of a higher-level layer can inhibit (subsume) that of its predecessor, hence the term subsumption. His argument in favour of the subsumption architecture was two fold.

- Previous reasoning approaches were slow and they were unable to adapt to changing and dynamic environments.
- Many actions of agents are quite separable; coherent intelligence can merge from independent sub-components interacting in the world.

Brooks implemented this approach on real robots that could explore the environment, build maps, navigate, walk [24] and learn how to co-ordinate internal conflicting behaviours [25]. According to Brooks, behaviour-based systems facilitate the ability of robots to find out about their particular world by themselves. This appears to be a suitable substratum for behaviour learning robots and has been demonstrated on the mobile robots "Don Group" by Mataric [20].

A shortcoming inherent to basic behaviour-based systems is the problem of conflicting behaviours when multiple behaviours compete to generate a control action. Brooks solved this problem in his early implementations with a pre-prioritised scheme to handle conflicts which was a behaviour arbitration. More elegant and efficient approaches for merging

behaviours rather than arbitration have been introduced in recent years (to be discussed in sections 2.5.5) using multi-strategy systems.

2.5 Previous work and approaches to intelligent robotics and control

Robotics as an engineering discipline encompasses industrial and mobile robots and has been a suitable test bed for the application of automatic control techniques, especially applied artificial intelligence (AI). The aspiration of replicating human intelligence as closely as possible in mechanical systems has inspired researchers to create *autonomous* robots. Autonomy in this context, means that, in an unguided fashion, a robot is able to perceive its environment, has the capacity of reasoning about it by making appropriate decisions in favour of some goal(s), and eventually to act on its environment.

A robot can perceive, in general, its environment by three types of information sources:

- Numerical data from measuring sensors (vision, actuators, wheel encoders, etc.)
- Heuristics in the form of linguistic data from expert human operators (speed is high, temperature is medium, pressure is low)
- Reactions of the robot to its environmental characteristics (behaviours)

With the aim of providing autonomy, a variety of algorithms and techniques have been developed which are based on one of the above sources, some combination of them, and also the extent of built-in information. The amount of meta-knowledge and degree of autonomy of a robot determine the type of learning approach, such as *on-line* and *off-line*. In the following, the most common learning techniques applied to robotics are discussed, and conclusions are drawn on their efficiency and shortcomings.

2.5.1 Reinforcement learning (RL)

Learning from interaction is the underlying principle in almost all theories of learning and intelligence. RL is a computational approach to learning from interaction and is much more focused on goal-directed learning from interaction than are other approaches to machine learning [26]. RL is rooted in the idea: "what policy to take, in order to maximise a reward signal". The learner must find out which actions result in the highest reward by simply trying them. This implies that an action may affect not only the immediate reward, but also

the next situation, and this in turn, all subsequent rewards. This gives rise to two characteristics, namely the trial-and-error search and delayed rewards, which are specific to RL.

An important difference between RL and most machine learning techniques (such as neural networks, and decision trees) is that the latter are *supervised*. Supervised learning involves learning from examples which are correct, and representative, and which are provided either by an external teacher or supervisor or by the system itself (self-supervised systems). However, RL is able to achieve learning as a result of interactions or of typical problem scenarios, where it is impractical to learn from examples, or where these are inaccessible. RL is in its philosophy similar to that of *learning from observations and discovery*, which is a type of machine learning typically with the highest degree of inference. This method is addressed in greater detail in section 3.1.

Basic elements of RL

An RL system is in fact an interface between an agent and its environment. RL can be identified by 3 basic elements (and an optional fourth) as shown below.

- *A policy*
- *A reward function*
- *A value function (also called a utility function)*
- *A model of the environment* (this is an optional feature [26])

A *policy* specifies the behaviour of the agent at a given time by, for example, mapping the perceived state to an action. In certain circumstances, the policy can be a simple lookup table, or computationally as expensive as a search space.

A *reward function* provides the short term goal of an RL system, in the sense that it determines, in terms of some aspect of the system performance, what actions are good and what are bad in an immediate sense. The reward function is required to map state-action pairs on a scalar which is called *reward*, and which indicates the desirability of that state. The ultimate goal of an RL system is to maximise the total reward it receives over a

significant period of operation and if the reward is too small this is used as a basis to change the policy.

The *value function* (also called the *utility function*) is used to determine the quality of a large sequence of operations in many states. Its value obtained in a state *n* can be formulated as the total reward an agent scores following initiation in the state *n*. For instance, a state may have a low immediate reward, but still score a high value because later states yield high rewards. The main concern of RL in decision processes is the magnitude of the values and to maximise the number of states that generate high values rather than high rewards. Rewards are obtained directly from the immediate environment, whereas values must be estimated from the sequence of available rewards that an agent experiences. Hence, efficiently estimating value functions (or learning the utility function) of an RL system is one of the most important components of such systems. The vast majority of RL methods have been structured to predict values or utility functions. They may employ search methods such as genetic algorithms, or function approximators such as neural networks for their prediction. Alternative methods to approximate values have been introduced by a number of researchers including *temporal difference* (TD) learning developed by Sutton [27]. TD is an incremental learning procedure and is driven by the error obtained from the differences between the temporally successive predictions for the value function. Another optimisation technique to learn the utility function is Q-learning [28,29]. Q-learning is fundamentally a hybrid approach combining both policy and value function optimisation. The reader is referred to [30,31,32] for a concise introduction, whereas [27,28] provide a rigorous and formal treatment of Q-learning.

In contrast to early RL systems, which were based only on trial and error, a number of researchers now incorporate planning into their RL systems. In such systems, a *model* is needed which simulates the behaviour of the environment and makes decisions on the executions of some actions by considering possible future situations. For alternative concise introductions to RL, the reader is referred to [30,33,34], and [26] deals with RL in a rigorous manner.

Applications of RL to robotics

RL has found applications in a wide variety of learning tasks such as manufacturing systems, game playing and robotics [31]. Similar to many other learning approaches, RL has been applied to a variety of robotic tasks involving learning through interaction with the environment.

Mahadevan and Connell [35] applied RL incorporating Q-learning to a mobile robot in an attempt to train it to push large boxes around its environment. In spite of immense uncertainty in the outcome of the actions, the robot's performance was similar to that achieved by an expert-programmed algorithm.

Millan and Torras describe an RL-based control architecture, named TESEO [36], which incorporates a connectionist method for optimising the policy of the RL system. This means that the algorithm learns to perform those actions that maximise the total reinforcement during a target-seeking task. TESEO was implemented on a mobile robot to perform target-seeking navigation in the office environment and the robot was able to learn the shortest and safest trajectory after 10 training epochs. The TESEO architecture was not intended to be learned from scratch: some basic reflexes [36] were built in the system to be accessed when the neural network failed to generalise from the available knowledge, and also to facilitate fast learning. These reflexes were particularly used in the first epoch of the training where the robot was approaching a dead-end, and they were needed in order to navigate the robot away. Further examples of RL applications to robotics can be found in [37,38].

The drawbacks of RL are that it may not perform appropriately if the environment is constantly changing [34], its convergence is often slow in comparison with other learning algorithms and most implementations lack incremental improvement. However, RL has shown relatively good results when applied to non-manipulative tasks with a very large state-space, such as game playing. In TD-gammon [30], a backgammon playing program which incorporates the TD-learning algorithm in combination with a multi-layer neural network, two agents played each other for more than a million times. Only at the end of each game did the agents receive a reward and the mappings stored in the weights of the

neural net. The resulting program could play at world championship level [30]. However, perhaps, due to the unstructured representation of the learned knowledge which was in the form of real numbers stored in the connectionist network, the developers were unable to understand or explain the program's performance.

A comprehensive narrative history of RL development from its early days is provided by Richard Sutton in [26].

2.5.2 Neural learning

Another class of learning algorithms, is that provided by neural networks [39]. A neural network (NN) is an information processing system mimicking the behaviour of the human brain in a mathematical model. NNs exhibit fast processing speeds (due to their massive parallelism) and are able to learn a concept from a set of training examples. They have been applied in a wide range of areas, including function approximation [36], pattern recognition [40], optimisation problems [41] and, relevant to the current work, in intelligent robot control and learning [42,43]. For further detailed information on NNs, the reader is referred to [39,44,45,46].

Neural networks are able to implement a number of learning algorithms, such as back-propagation [39,47] and reinforcement learning [42]. In the ALVINN system [47], a multi-layer NN is trained using back-propagation learning algorithm to map digital images on appropriate steering angles for a mobile robot. The learning process is off-line, since the training patterns are collected while the vehicle is driven by a human operator. The trained system aims to follow a road mimicking the human reactions. ALVINN was successfully implemented on a real robot to carry out this mission. However, back-propagation sets limitations on the system flexibility (especially reactive systems), as successful learning is not always guaranteed [45], training is time consuming, learning is off-line and may become trapped in local minima (in case of multi-valued outputs). Research has been carried out [48,49,50] with the aim of reducing the convergence time of the back-propagation algorithm and of finding the global minimum of a neural system by coding variables and attempting to determine the required number of neurons needed in its layers, and, in particular, for hidden layers.

In [42], a neural network is used to learn a set of reactive rules that model two basic locomotion reflexes of a mobile robot, namely target-seeking and obstacle-avoidance. The learning process is on-line and is performed from scratch. The control architecture is divided into two modules both to partition the perceptual space (classification) and to associate the control actions and these are implemented as two separate NNs. The former is a single layer NN and implements a reinforcement learning rule, and the latter is a fuzzy adaptive-resonance-theory NN [51]. The learning NN has three neurons (one for each steering command) which can be independently excited by the fuzzy-ART NN. The action associated with the *most excited* neuron of the second NN is chosen to drive the robot. The corresponding weights are reinforced or punished, if the performed action is feasible or if it carries the risk of a collision, respectively. This technique aimed to demonstrate the efficiency of learning by trial-and-error and from scratch. It was implemented in a simulated robot and showed an improved steady state efficiency [42] at the end of the third epoch and this is attributed to the characteristic of the ART-NN which preserves the already existing knowledge. It was reported, however, that this system failed to perform in the presence of local minima where the goal is hidden behind long walls, and the system falls into oscillatory movements.

2.5.3 Evolutionary learning

Evolutionary learning is based on genetic algorithms (GAs), which were first introduced by John Holland [52] in 1970s. GAs are a class of adaptive search techniques based on the mechanics of natural selection, natural genetics and evolutionary principles such as inheritance and mutation. GAs have proven to be a powerful tool within the area of machine learning [53], allowing computers to evolve solutions to problems, using function optimisation, search and learning.

The basic operation of a GA is conceptually simple and can be summarised as follows:

- maintain a population of trial solutions to a problem (chromosomes)
- select the better solutions for recombination with each other, and
- use their offspring to replace poorer solutions.

GAs are, in general, able to find good solutions to a wide class of application problems in reasonable amounts of time. However, the time required to find adequate solutions tends to increase when the dimensionality of the problem rises. To remedy this, a great deal of research has been carried out into increasing the speed of GAs by introducing parallel architectures [54].

Evolutionary algorithms have found application in intelligent robots, including areas such as collision avoidance [55], and, in particular, in automatic behaviour learning [56,57]. One of the well-known research paradigms in evolutionary learning robotics is the *artificial life (ALife)* paradigm, and perhaps one of the most successful applications is the evolutionary learning system SAMUEL (Strategy Acquisition Method Using Empirical Learning) [58].

The ALife paradigm has been inspired by a number of issues in robotics.

- Emergence of and learning complex behaviours, especially in multi-agent systems where individual agents compete under a variety of situations for resources and struggle for survival.
- A more realistic alternative to the model-based robot design rooted in traditional AI. This usually fails to reflect the complexities, noise, errors that arise in real sensors and actuation operating in real world conditions [57]. Moreover, AI-based reasoning usually assumes the error free translation of continuous signals to symbols.

These issues and shortcomings have inspired researchers such as Mataric [59] and Brooks [60] to argue for the development of adaptive robots that evolve behaviours without using a pre-specified model of their environments. Current research themes in evolutionary learning robots include behaviour evolution, organisation and the study of multi robot systems for identifying the emergence of complex behaviours.

Behaviour learning in SAMUEL is an evolutionary process in which candidate solutions to a problem (e.g. obstacle avoidance) are situation-action reactive rules. Rule representation in SAMUEL is designed to promote the inclusion of heuristics into initial populations for complex robotics tasks where random initial rule populations are unlikely to perform well. SAMUEL has been used for behaviour learning to control an autonomous underwater robot

[55], missile evasion [58], and other simulated tasks. For more details and information on SAMUEL, and also applications of GAs in robotics, the reader is referred to [58,61,62,63, 64,65].

The main strength of an evolutionary algorithm is in rapidly finding the most promising regions to investigate the complex search space. This ability is largely due to the implicitly parallel search that it performs on a population of candidate solutions. However, these methods are less efficient at fine-tuning candidate solutions, and hybrid systems have been developed which provide efficient local optimisation methods to improve the final solutions found by evolutionary systems [57].

2.5.4 Inductive learning

Inductive learning is the process of acquiring knowledge (new facts) or to discover patterns in collections of observations (existing facts) by drawing inductive inferences. This is, in general, concerned with the generation of hypotheses and their validation [66]. Historically, inductive learning is an area of contention between philosophers and logicians, and this is due to an observation made by the Scottish philosopher David Hume in the 18th century. He observed that inductively gained assertions are hypotheses, and these can potentially have an infinite number of consequences, while only a finite number of these can be validated. Discussions regarding the validation of a hypothesis are often found in philosophical debates and are generally of lesser importance in the context of inductive learning in engineering applications. In common with most empirical studies, the current work assumes that the generated hypotheses are assessed by human experts, or are tested by known methods.

Studying and modelling inductive learning is one of the central topics of machine learning. The knowledge representation in inductive learning can be in the form of decision trees [67] (see chapter 3), a set of production rules [68], or a set of real-valued numbers (weights) stored in a connectionist network. The two most widely used forms of inductive learning technique are:

- Learning from observations and discovery
- Learning from examples [69]

These two techniques are further discussed in chapter 3. Inductive learning as a learning methodology is relevant to a wide range of applications such as automatic knowledge acquisition for expert systems in PLANT/DS for the diagnosis of soybean disease [70], diagnostic systems for fault detection [71] and various experimental sciences such as biology, chemistry, psychology, medicine and genetics [72], where traditional mathematical and statistical techniques, such as regression analysis or factor analysis, are not sufficiently powerful [66]. More important to the current work are the applications of inductive learning to robotics [73], automatic behaviour learning [74,75], industrial process control [76] and power system security [77].

Although the learning mechanism in neural networks is of an inductive nature, decision tree and rule learning approaches are the most common and, perhaps, the most established representations of inductively learned knowledge. Inductive learning has proven to be a powerful tool for the automatic learning of domain knowledge. An example is the PLANT/DS expert system in which diagnostic rules were formulated in two ways, namely by formalising experts' diagnostic knowledge and by induction from examples. These sets of rules were then tested on a few hundred disease cases. Michalski and Chilauski reported in [70] that the inductively derived knowledge (rules) outperformed those derived from experts.

Michalski formulates in [66] a general paradigm for inductive inference. The major forms of inductive learning, namely learning from examples and learning from observations, are discussed in detail in chapters 3, 4, 9, 11 of [78]. Chapter 10 introduces the system BACON.4 which is an application of learning from observation and is a layered inductive learning system which is able to discover empirical laws, whose heuristics are general mechanisms and are applicable to diverse range of domains.

2.5.5 Hybrid learning techniques

The learning techniques discussed above generally perform well in their application to a specific problem, but their performance is generally poor in their transference to other application areas and under a different set of constraints. It is becoming increasingly evident that it is advantageous to employ multi-strategy hybrid systems in the conception

and design of intelligent systems such as intelligent autonomous robots. Soft computing (SC) techniques may present a promising way forward for intelligent systems that need to be able to perceive under conditions of noise and uncertainty, and which need to make decisions and act in dynamically changing environments. SC aims at accommodation among the imprecision of the real world, its guiding principle being to:

> "*exploit the tolerance for imprecision, uncertainty and partial truth to achieve tractability, robustness and low cost solutions* [79]".

The principle constituents of SC are: fuzzy logic [80,81], neural networks and probabilistic reasoning, which includes genetic algorithms. In the following, the most common hybrid techniques which have been applied to intelligent robotics are outlined, and conclusions are drawn on their degree of applicability and efficiency.

Neuro-fuzzy learning

Neuro-fuzzy techniques are one of the most intensely researched multi-strategy systems. In this type of hybrid learning system, fuzzy logic (FL) is mainly concerned with imprecision and approximate reasoning, and NNs deal with learning and curve fitting (function approximation). This implies that the overall system should be able to learn to approximate functions under conditions of noise and uncertainty, in which NNs influence the learning parameters of the fuzzy system. Nauk and Kruse [82] define neuro-fuzzy systems as: "the development of heuristic learning strategies derived from the domain of neural network theory to support the development of fuzzy systems".

In single strategy systems, FL has been applied to a variety of process control and robotic tasks in order to either incorporate expertise of human operators formulated in linguistic rules, or exploit the qualitative nature of fuzzy theory for approximate reasoning. Chapter 6 introduces FL and its main areas of application.

In most robotic applications involving navigation, neuro-fuzzy algorithms have two distinct tasks:

- NNs are used to learn certain patterns describing behaviours [83];
- FL is used to deal with uncertainty, blend conflicting behaviours and assure smooth trajectories [84,85].

Using such an approach, the neuro-fuzzy system would be able to automate the process of generating fuzzy rules. A novel approach was taken by Li in [83], who employed a cascaded system with a NN in series with a fuzzy system. The numeric values of 15 sonar sensors as well as the heading angle of the robot were supplied to the NN. The output of the NN is a reference direction of motion which is supplied to the fuzzy systems along with the 15 sensor readings. According to [83], the real direction of motion is calculated by either system (FL and NN) in such a way that the erroneous output of one system is compensated by the other. However, the details of this mechanism have not been reported. The only navigational improvement of this configuration, compared to the earlier work, appears to be its ability to prevent the robot turning into shallow U-shaped obstacles, and instead to navigate past the obstacles. How the robot behaves should the dead-end become greater in depth, is not documented. Although, the overall performance of the system in demonstrated navigation tasks seems to be satisfactory, the process of learning the fuzzy rules appeared to be hand-crafted.

Fuzzy-genetic learning

The inherent parallelism of GAs in search and optimisation can be used to automate a number of tasks in fuzzy-based systems. Since the design and shaping of fuzzy membership functions is often an *ad hoc* and problem specific process, the GA's capability to performing parallel search can be used to tune a set of membership functions. In [86], GAs are used to tune the membership functions of 33 fuzzy rules which were manually configured prior to the training of a simulated robot for navigation purposes. The population sizes used in the experiment ranged from 100 to 1000 individuals (chromosomes) and, after tuning the membership functions, the results were transferred to a real robot to perform wall following missions.

An alternative method of implementing the neural training of fuzzy rules is to synthesise fuzzy rules by using GAs [87,88]. The underlying aim of this operation is to overcome the

common weaknesses of neural computing stated in [87], namely the absence of an analytical *recipe* to determine the network configuration, and the tendency of the process to become trapped in local optima during the learning process. GAs offer a procedural mechanism of learning design and perform a global search which results in a population of solutions to a specific problem, although these may not necessarily include an optimum solution, since GAs do not perform a local search in order to find the "best" solution.

The evolutionary fuzzy system used in [89], employs fuzzy reasoning to assess the fitness of evolving solution populations in a linguistic manner. An example of one of the rules used is: IF move slowly THEN fitness low, or IF obstacle hit THEN fitness zero. The system is used to train a robot to follow long walls without divergence or collision and the linguistic character of FL systems is used to mimic human behaviour in assessing the quality of the automatically tuned fitness function of the learning system.

Fuzzy decision tree learning

Operationally, fuzzy decision tree learning systems are identical to neuro-fuzzy learning systems, in that decision trees are incorporated to generate automatically fuzzy control rules from a finite set of training examples. Rule synthesis is performed in either an off-line or an on-line manner (incremental mode). In the former, fuzzy rules are extracted from a batch of data sample, and are applied to the process to be controlled to classify unseen scenarios, whereas in the latter, more than one hypothesis is set up and updated successively as new data increments arrive.

Fuzzy decision trees (FDTs) have been investigated theoretically by a number of researchers [90,91], and have also been applied to a variety of problems, in particular in robot learning and intelligent navigation [92,93,94]. In [92], Shibata *et al* developed a fuzzy ID3 system to automate the process of motion planning for industrial robots. The model they used was based on the situation-action scenarios which occur when a human operator, while cutting 3-D work pieces, sets a path with many points each specifying a rotational angle for the mounted tool. Each training example consisted of nine input variables and a single output and the generated FDT incorporated three features from a total of nine in its classification of 297 training examples. Shibata *et al* also formulated an

evaluation function for a GA using these three features as the criteria of a skilled operator and the GA was used to optimise the path planning process. The FDT was applied to a robot to whose gripper a plasma torch for cutting 3-D objects was connected. They report that the algorithm showed effective results, and also could save labour and time for supervising other industrial robots.

Hsu *et al* [93] generated 21 fuzzy rules extracted from a set of 300 training examples, in which each example consisted of 16 ultrasonic sensor readings and a corresponding direction output and were collected using a skilled human expert instructing the mobile robot to follow walls on its right side. Using ID3 [95] as the learning algorithm, these rules synthesised a fuzzy controller of the mobile robot Nomad 200 on wall-following missions, and the resulting controller was able to navigate the robot along walls in a terrain similar to the training but in which the layout was slightly modified. The quality of the trajectory produced by the fuzzy controller was claimed to be at least as good as that achieved by the human expert. However, the process of rule extraction was off-line, as a skilled operator was used and is not fully automated, since the induced DT is built on crisp data and the rules generated by the crisp DT are manually fuzzified prior to navigation.

The current work [94], acquires fuzzy data (rather than crisp data) and this process is fully automated and is accomplished algorithmically. Rule synthesis is performed by searching the space of FDTs and in an on-line fashion, see chapter 7. A particularly important attribute of FDTs is the intelligibility of control rules due to the symbolic nature of the induced knowledge. The symbolic nature of the approach is complementary to that of the linguistic mechanism of the fuzzy reasoning approach.

Neuro-fuzzy genetic learning

Once a set of input/output relationships describing a certain control system or behaviours is available, a multi-strategy learning system such as the evolutionary neuro-fuzzy system in [96] is able to extract optimised fuzzy rules. This system is capable of the automatic construction of fuzzy membership functions (by the incorporated neural network) and the tuning of them, resulting in an improved behaviour of the GAs. Surmann *et al* have applied this technique in [96] to data samples from gas furnace [97] control data consisting of 296

input/output training examples which are measured every nine seconds. The input is the gas flow rate into the furnace, and the output is the concentration of CO_2 in the exhaust gas. Surmann *et al* report that their genetically optimised neuro-fuzzy system outperforms single-strategy fuzzy-based systems using the same training data.

2.6 Summary

This chapter has presented an overview of recent contributions to the building of intelligent and autonomous robots. This has included both the early approaches to intelligent robots as well recent intelligent agent which integrates multi-strategy learning techniques borrowed from a variety of soft computing algorithms. The trend evident in the pattern of recent techniques for autonomous robots demonstrates that there is an increasing interest in robotic architectures which impose constraints taken from biological paradigms in their interaction with the environments, and these include: perception, action, adaptation, learning and social behaviour (in multi-robot systems).

In recent research more information is being left for robots to learn than is being designed into their initial architecture. Modern learning techniques also try to capture the environmental changes (information) in numeric, cognitive or behavioural form, or as some combination of these, and this is in contrast to early approaches which used only numeric information. This highlights the current emphasis being placed on multi-strategy systems in intelligent robotics.

In the context of this literature survey, the work reported in chapters 4, 5 and 7 relating to intelligent robotics takes its inspiration from the survival instincts of biological systems. Within this field, as far as the author is aware, no system has yet been reported which fully automates the process of acquiring fuzzy data. This important contribution of the current work is reported in chapter 7. The following three chapters of this thesis concentrate on the development and testing of a DT-based algorithm for the automatic behaviour learning of an autonomous robot.

References

[1] M. Kaiser, V. Klingspor, J.R. Milan and M. Accame, "Using Machine Learning Techniques in Real-World Mobile Robots", *IEEE Expert*, April 1995, pp. 37-45.

[2] M. Dorigo an M. Colombetti, "Robot Shaping, An Experiment in Behaviour Engineering", The MIT Press, ISBN: 0-262-04162-2, 1998.

[3] J.S. Albus, "Outline for a Theory of Intelligence", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 21, No. 3, May/June 1991, pp. 473-509.

[4] L.A. Zadeh, "Fuzzy Sets", *Information and Control*, Vol. 8, New York: Academic Press, 1965, pp. 338-352, Fuzzy Sets and Applications: Selected Papers by L.A. Zadeh, R.R. Yager, S. Ovchinnikov, R.M. Tong and H.T. Nguyen (Eds.), pp. 29-44, ISBN 0-471-85710-6.

[5] K. Kautsky, "The Materialist Conception of History", J.H. Kautsky (Ed.), Translation of: Die Materialistische Geschichtsauffaβung, Yale University Press, 1998, ISBN: 0-300-04168-3.

[6] H.A. Simon, "Why Should Machines Learn?", *Machine Learning, An Artificial Intelligence Approach*, R.S. Michalski, J.C. Carbonell and T.M. Mitchell (Eds.), 1983, ISBN: 0-938382-05-4.

[7] P. Coiffet and M. Chirouze, "An Introduction to Robot Technology", Translated by Meg Tombs from "Elements de Robotique", Hermes Publishing (France), ISBN: 0-85038-637-3, 1983.

[8] J. Angeles, "Fundamentals of Robotic Mechanical Systems, Theory, Methods and Algorithms", Springer-Verlag, 1997, ISBN: 0-387-94540-7.

[9] M.J. Mataric, "Behaviour-based Control: Main Properties and Implications", *Proceedings of IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Systems*, Nice, France, May 1992, pp. 46-54.

[10] S.V.R. Nageswara and S.S. Iyengar, "Autonomous Robot Navigation in Unknown Terrains: Incidental Learning and Environmental Exploration", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 20, N0.6, November/December 1990, pp. 1443-1449.

[11] B. Beaufrer and S. Zeghloul, "Navigation Method for a Mobile Robot using a Fuzzy Based Method: Simulation and Experimental Aspects", *International Journal of Robotics and Automation*, Vol. 10, Issue 3, 1995, pp.106-113.

[12] J. Borenstein and Y. Koren, "Real-time Obstacle avoidance for Fast Mobile Robots", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 19, 1989, pp.1179-1187.

[13] C.W. Warren, "Multiple Robot Path Coordination using Artificial Potential Fields", *Proceedings of IEEE Conference on Robotics and Automation*, Cincinnati. Ohio, 1990, pp.500-505.

[14] Z.Q. Ma and Z.R. Yuan, "Real-time Navigation and Obstacle Avoidance on Grids Method for Fast Mobile Robots", *Engineering Applications of Artificial Intelligence*, Vol. 8, No. 1, 1995, pp. 91-95.

[15] A. Elfes, "Sonar-based Real World Mapping and Navigation", *IEEE Journal of Robotics and Automation*, Vol. 3. No. 3, 1987.

[16] M. Mataric, "Integration of Representation into Goal-driven, Behaviour-based Robots", *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 3, June 1992, pp. 304-312.

[17] C. C. Neves and J.O. Gray, "Comparative Analysis of Three Architectures using a Generalised Framework", *Proceedings of the Fifth International Workshop on Advanced Robotics and Intelligent Machines*, March 1997, Manchester, UK.

[18] A. Dubrawski and J.L. Crowley, "Self-supervised Neural Systems for Reactive Navigation", *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 3, 1994, pp. 2076-81.

[19] P. Reignier, "Fuzzy Logic Techniques for Mobile robot Obstacle Avoidance", *Journal of Robotics and Autonomous Systems*, Vol. 12, 1994, pp. 143-153.

[20] M.J. Mataric, "Behaviour-based Control: Examples from Navigation, Learning and Group Behaviour", *Journal of Experimental and Theoretical Artificial Intelligence, special issue on Software Architectures for Physical Agents*, Vol. 9, Nos. 2-3, Hexmoor, Horswill and Kortenkamp (Eds.), 1997, pp. 323-336.

[21] M.J. Mataric, "Behaviour-Based Control: Main Properties and Implications", *Proceedings of IEEE International Conference on Robotics and Automation*,

Workshop on Architectures for Intelligent Control Systems, Nice, France, May 1992, pp. 46-52.

[22] R.A. Brooks, "A Robust Layered Control System for a Mobile Robot", *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1, March 1986, pp. 14-23.; also MIT AI Memo 864, September 1985.

[23] R.A. Brooks, "New Approaches to Robotics", *Science*, Vol. 253, September 1991, pp.1227-1232.

[24] R.A. Brooks, "A Robot that Walks; Emergent Behaviours form a Carefully Evolved Network", *Neural Computation*, Vol. 1 No. 2, 1989, pp. 253-262, Also in *IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, May 1989, pp. 292-296.

[25] P. Maes and R.A. Brooks, "Learning to Coordinate Behaviours", *AAAI*, Boston, MA, August 1990, pp. 796--802.

[26] R.S. Sutton and A.G. Barto, "Reinforcement Learning: An Introduction", *MIT Press*, A Bradford Book, Cambridge, MA, 1998.

[27] R.S. Sutton, "Learning to Predict by the Methods of Temporal Differences", *Machine Learning*, 3 (1988), pp. 9-44.

[28] C.J.C.H. Watkins, "Learning from Delayed Rewards", *Doctoral Thesis*, Cambridge University, Cambridge, England, 1989.

[29] C.J.C.H. Watkins and P. Dayan, "Technical Note: Q-Learning", *Machine Learning*, 8 (1992), pp. 279-292.

[30] S.T. Hagen and B. Kröse, "A Short Introduction into Reinforcement Learning", *Proceedings of the Seventh Belgian-Dutch Conference on Machine Learning*, BENELEARN'97, 1997, pp. 7-12.

[31] L.P. Kaelbling, M.L. Littman and A.W. Moore, "Reinforcement Learning: A Survey", *Journal of Artificial Intelligence Research*, 4 (1996), pp. 237-285.

[32] M.E. Harmon, "Reinforcement Learning: A Tutorial", Available at: http://www-anw.cs.umass.edu/~mharmon/rltutorial/frames.html.

[33] B. Kröse, "Learning from Delayed Rewards", *Special Issue of Robotics and Autonomous Systems*, 15 (1995), pp. 233-235.

[34] S. Singh, P. Norvig and D. Cohn, "How to Make Software Agents to Do Right Things: An Introduction to RL", http://envy.cs.umass.edu/People/singh/RLMasses/RL.html.

[35] S. Mahadevan and J. Connell, "Automatic Programming of Behaviour-based Robots using Reinforcement Learning", *Proceedings of the Ninth National Conference on Artificial Intelligence*, Anaheim, CA, 1991.

[36] J.R. Milan and C. Torras, "Efficient Reinforcement Learning of Navigation Strategies in an Autonomous Robot", *Proceedings of IEEE Conference on Intelligent Robots and Systems*, 1994, pp. 15-22.

[37] A.H. Fagg, D. Lotspeich an G.A. Bekey, "A Reinforcement Learning Approach to Reactive Control Policy Design for Autonomous Design", *Proceedings of IEEE International Conference on Robotics and Automation*, San Diego, CA, 1994, pp. 39-44.

[38] H.R. Beom and H.S. Cho, "A Sensor-based Navigation for a Mobile Robot using Fuzzy Logic and Reinforcement Learning", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 25, No. 3, March 1995, pp.464-477.

[39] R.P. Lippmann, "An Introduction to Computing with Neural Nets", *The IEEE Acoustics, Speech and Signal Processing Magazine*, 4 (2), April 1987, pp. 4-22.

[40] H.C. Lui and S.L. Lau, "Character Recognition on Grey Level Image using Neural Nets", *Proceedings of IJCNN'91*, Vol. 1, Singapore, 1991, pp. 325-330.

[41] G. Fahner, "An Algorithm Structured Neural Net for the Shortest Path Problem", *Proceedings of IJCNN'91*, Vol. 1, 1991, pp. 153-158.

[42] A. Dubrawski and J.L. Crowley, "Self-supervised Neural System for Reactive Navigation", *IEEE International Conference on Robotics and Automation*, Vol. 3, 1994, pp. 2076-2081.

[43] A. Dubrawski and J.L. Crowley, "Learning Locomotion Reflexes: A Self-supervised Learning Systems for a Mobile Robot", *Journal of Robotics and Autonomous Systems*, 12 (1994), pp. 133-142.

[44] C.M. Bishop, "Neural Networks and Their Applications", *Review of Scientific Instruments*, 65 (6), June 1994, American Institute of Physics, pp. 1803-1832.

[45] T. Fokuda and T. Shibata, "Theory and Applications of Neural Networks for Industrial Control Systems", *IEEE Transactions on Industrial Electronics*, Vol. 39, No. 6, December 1992, pp. 472-489.

[46] D.R. Hush and B.G. Horne, "Progress in Supervised Neural Networks", *IEEE Signal Processing Magazine*, January 1993, pp. 8-39.

[47] D.A. Pomerleau, "Efficient Training of an Artificial Neural Network for Autonomous Navigation", *Journal of Neural Computation*, Vol. 3, 1991.

[48] A.R. Jacobs, "Increased Rates of Convergence Through learning Rate Adaptation", *IEEE Transactions on Neural Networks*, Vol. 1, No. ?, 1988, pp.295-307.

[49] N. Baba, "A New Approach for Finding the Global Minimum of Error Function of Neural Networks", *IEEE Transactions on Neural Networks*, Vol. 2, No. ?, 1989, pp. 367-373.

[50] T. Fukuda, T. Shibata, M. Tokita and T. Mitsuoka, "Neural Network Application for Robotic Motion Control: Adaptation and Learning", *Proceedings of International Joint Conference on Nural Ntworks'90*, San Diego, CA, 1990, pp.447-451.

[51] G.A. Carpenter, S. Grossberg and D. Rosen, "Fuzzy ART: Fast Stable Learning of Analogue Patterns By an Adaptive Resonance System", Neural networks, Vol. 3, 1991.

[52] J.H. Holland, "Adaptation in Natural and Artificial Systems",, Univresity of Michigan Press, 1975.

[53] U. Rost and P. Öchtering, "Knowledge-based Genetic Learning", *Proceedings of the Sixth Scandinavian Conference on Artificial Intelligence (SCAI-97)*, Helsinki, Finnland, 18-20 August, 1997.

[54] E. Cantu-Paz, "A Survey of Parallel Genetic Algorithms", *Technical Report*, No. 95007, Illinois GA Laboratory IlliGAL.

[55] A.C. Schultz, "Using a Genetic Algorithm to Learn Strategies for Collision Avoidance and Local Navigation", *Proceedings of the Seventh International Symposium on Unmanned Untethered Submersible Technology*, University of New Hampshire, Marine Systems Engineering Laboratory, 1991, pp. 213-225.

[56] A.C. Schultz, "Learning Robot Behaviours using Genetic Algorithms", *Proceedings of The International Symposium on Robotics and Manufacturing*, August 14-18, 1994.

[57] J. Grefenstette, "Evolutionary Algorithms in Robotics", *Proceedings of The Fifth International Symposium on Robotics and Manufacturing ISRAM'94*, 1994.

[58] J.J. Grefenstette, C.L. Ramsey and A.C. Schultz, "Learning Sequential Decision Rules using Simulation Models and Competition", *Machine Learning 5(4)*, 1990, pp. 355-381.

[59] M.J. Mataric, "Coordination and Learning in Multi-Robot Systems", *IEEE Intelligent Systems*, Mar/Apr 1998, pp. 6-8.

[60] R. Brooks, "Artificial Life and Real Robots", *Towards a Practice of Autonomous Systems: European Conference on Artificial Life*, Paris, France, MIT Press, December 1991, pp. 3-10.

[61] J.J. Grefenstette, "Lamarkian Learning in Multi-agent Environments", *Proceedings of The Fourth International Conference on Genetic Algorithms*, San Diego, CA, 1991, pp. 303-310.

[62] J.J. Grefenstette and H.C. Cobb, "User's Guide for SAMUEL, Version 4.0", 1994, *Naval Research Lab Report*, Washington, DC.

[63] J.J. Grefenstette, "Evolutionary Algorithms in Rbotoics", *Robotics and Manufacturing: Recent Trends in Research, Education, and Applications*, Vol 5, (Eds. Mohammad Jamshidi, Charles Nguyen), *Proceedings Of the First World Automation Congress (WAC '94) and Fifth International Symposium on Robotics and Manufacturing (ISRAM '94)*, August 14-17, 1994, p127-132.

[64] A.C. Schultz and J.J. Grefenstette, "Using a Genetic Algorithm to Learn Behaviours for Autonomous Vehicles", *Proceedings of American Institute of Aeronautics and Astronautics Guidance, Navigation and Control Conference*, Hilton Head, SC, AIAA, 1992, pp.739-749.

[65] M. Dorigo and U. Schneff, "Genetics-based Machine Learning and Behaviour-based Robotics", *IEEE Transactions on System, Man and Cybernetics, SMC-23*, 1, 1993.

[66] R.S. Michalski, "A Theory and Methodology of Inductive Learning", *Machine Learning, An Artificial Intelligence Approach*, R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), Tioga Publishing Company, 1983.

[67] J.R. Quinlan, "Decision Trees and Decisionmaking", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 20, No. 2, March/April 1990, pp. 339-346.

[68] J. Cendrowska, "PRISM: An Algorithm for Producing Modular Rules", *International Journal of Man-Machine Studies*, Vol. 27, 1987, pp. 394-370.

[69] J.G. Carbonell, R.S. Michalski and T.M. Mitchell, "An Overview of Machine Learning", *Machine Learning, An Artificial Intelligence Approach*, R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), Tioga Publishing Company, 1983.

[70] R.S. Michalski and R.L. Chilauski, "Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for the Soybean Disease Diagnosis", *Policy Analysis and Information Systems (Special Issue on Knowledge Acquisition and Induction)*, Vol. 4, No. 2, June 1980, pp. 125-160.

[71] Y. Nakakuki, Y. Koseki and M. Tanaka, "Inductive Learning in Probabilistic Domain", *Machine Learning*, Vol. ?, Year ?, pp. 809-814.

[72] S. Salzberg, "Locating Protein Coding Regions in Human DNA using a Decision Tree Algorithm", *Journal of Computational Biology*, 2(3), 1995, pp. 473-485.

[73] I. Sillitoe and T. Elomaa, "Learning Decision Trees For Mapping The Local Environment in Mobile Robot Navigation", *Proceedings MLC-COLT Workshop on Robot Learning,* July 1994, New Brunswick, N.J, pp. 119-125.

[74] T. Zrimec and P. Mowforth, "Learning By an Autonomous Agent in The Pushing Domain", *Robotics and Autonomous Systems*, 0921-8830/91, 1991, Elsevier Science Publishers B.V.

[75] G.H. Shah Hamzei, D.J. Mulvaney and I.P.W. Sillitoe, "Batch-Mode Decision Tree Learning Applied to Intelligent Reactive Robot Control", *Sixth IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'97)*, September 9-12, 1997, Los Angeles, USA.

[76] B. Evans and D. Fisher, "Overcoming Process Delays with Decision Tree Induction", *IEEE Expert*, February 1994, pp.60-66.

[77] L. Wehenkel, "Machine Learning Approaches to Power-Systems Security Assessment", *IEEE Expert*, September/October 1997, pp. 60-72.

[78] R.S. Michalski, J.G. Carbonell and T.M. Mitchell, "Machine Learning, An Artificial Intelligence Approach, R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), Tioga Publishing Company, 1983.

[79] L.A. Zadeh, "Foreword", *Third European Congress on Intelligent Techniques and Soft Computing EUFIT'95*, Aachen Germany, August 28-31, 1995.

[80] L.A. Zadeh, "Fuzzy Sets", *Information and Control*, Vol. 8, New York: Academic Press, 1965, pp. 338-352, Fuzzy Sets and Applications: Selected Papers by L.A. Zadeh, (Edit.) R.R. Yager, S. Ovchinnikov, R.M. Tong and H.T. Nguyen, pp. 29-44, ISBN 0-471-85710-6.

[81] L.A. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Processes", *IEEE Transactions on Systems, Man and Cybernetics*, SMC-3 (1973), pp.28-44, Fuzzy Sets and Applications: Selected Papers by L.A. Zadeh, (Edit.) R.R. Yager, S. Ovchinnikov, R.M. Tong and H.T. Nguyen, pp. 105-146, ISBN 0-471-85710-6.

[82] D. Nauk and R. Kruse, "What are Neuro-Fuzzy Classifiers?", *Proceedings of Seventh International Fuzzy Systems Association World Congress IFSA'97*, Vol. IV, 1997, pp. 228-233.

[83] W. Li, "Neuro-Fuzzy Systems for Intelligent Robot Navigation and Control Under Uncertainty", *Proceedings of IEEE*, 1995, pp. 1747-1754.

[84] P. Reignier, "Fuzzy Logic Techniques for Mobile Robot Obstacle Avoidance", *Robotics and Autonomous Systems*, Vol. 12, 1994, pp. 143-153.

[85] A. Saffiotti, E.H. Ruspini and K. Konolige, "Blending reactivity and Goal-Directedness in a Fuzzy Controller", *Proceedings of the Second IEEE Conference on Fuzzy Systems*, San Francisco, CA, March 1993, pp. 134-139.

[86] R. Braunstigl, J. Mojika and J.P. Uribe, "A Wall Following Robot with a Fuzzy Logic Controller Optimised with a Genetic Algorithm", *FUZZ-1EEE/IFES'95 Fuzzy Robot Competition*, Yokohama, March 1995, pp. 77-82.

[87] Y, Tuan and H. Zhuang, "A Genetic Algorithm for Generating Fuzzy Classification Rules", *Fuzzy Sets and Systems*, 84 (1996), pp. 1-19.

[88] E. Dadios, "Non-conventional Control of the Flexible Pole-Cart Balancing Problem", *PhD Thesis*, Department of Manufacturing Engineering, Loughborough University, Loughborough, UK, 1996.

[89] R. Braunstigl and A. Ollero, "Evaluating the Wall Following Behaviour of a Mobile Robot with Fuzzy Logic", *Proceedings of The International Workshop on Artificial Intelligence in Real Time Control*, Bled, Slovenia, November 1995, pp. 89-93.

[90] C.Z. Janikow, "Fuzzy Decision Trees: Issues and Methods", *IEEE Transactions on Systems, Man, and Cybernetics*, 1997.

[91] L.S. Sison and E.K.P. Chong, "Fuzzy Modelling by Induction and Pruning of Decision Trees", *The International Symposium on Intelligent Control*, August, 1994, Columbus, Ohio, USA, pp. 166-171.

[92] T. Shibata, T.Abe, K. Tanie and M. Nose, "Motion Planning of a Redundant Manipulator-Criteria of Skilled Operators by Fuzzy-ID3 and GNDH and Optimisation by GA", *The IEEE Interdnational Conference*, 1995, pp. 99-102.

[93] S.H. Hsu, J.Y. Hsu and I.J. Chiang, "Automatic Generation of Fuzzy Control Rules by Machine Leaning Methods", *The IEEE International Conference on Robotics and Automation*, Nagoya, Japan, 1995, Vol. 1, pp. 287-292.

[94] G.H. Shah-Hamzei and D.J. Mulvaney, "Self-organising Fuzzy Decision Trees for Robot Navigation: an On-line Learning Approach", *IEEE International Conference on Systems, Man and Cybernetics SMC'98*, October 1998, San Diego, CA, USA, (Forthcoming).

[95] J.R. Quinlan, "Learning Efficient Classification Procedures and Their Application to Chess and Games", *Machine Learning, An Artificial Intelligence Approach*, R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), Tioga Publishing Company, ISBN: 0-935382-05-4, pp. 463-482.

[96] H. Surmann, A. Kanstein and K. Gosar, "Self-organising and Genetic Algorithms for an Automatic Design of Fuzzy Control and Decision Systems", *Proceedings of the 1st European Congress on Fuzzy and Intelligent Technologies*, EUFIT'93, Aachen 7-10 September, 1993, pp. 1097-1104.

[97] G.E.P. Box and G.M. Jenkins, "Time Series Analysis", *Forecasting and Control*, San Diego, CA, Holden Day, 1976.

Chapter 3

Decision Tree Learning:

Theoretical Issues and Background

If I hear, I forget
If I see, I remember
If I do, I understand

A Proverb

Quoted by S. Yalamanchili in "VHDL Starter's Guide"

This chapter addresses the underlying principles and learning strategies of decision trees (DTs) which are presented as a classification model. An in-depth explanation of the formalisms and associated algorithms is avoided, since this is beyond the scope of the current work. A general view of machine learning techniques, with emphasis on inductive

learning techniques, is presented. Basic terms and expressions of inductive knowledge acquisition, especially DT learning, are defined, explained and illustrated, and the interested reader is directed to the related literature and theoretical foundations, where necessary.

3.1 A taxonomy of machine learning techniques

This section provides a taxonomic consideration of machine learning (ML) in terms of the underlying learning strategy, the available data and the source of information. Particular attention is paid to the inductive learning approaches, whose application is the issue of chapters 4, 5 and 7. Machine learning approaches can be categorised into the following groups.

- *Rote learning* The learning task is performed either by being programmed or by memorising given facts and data [1]. Examples of this approach are procedural computer programming or the use of data bases.
- *Learning from instructions* The source of knowledge is a teacher or other organised sources such as a text book. The learner's task is to transform the knowledge to an internally usable representation and the amount of inference from the learner is very limited. Most formal education methods where the advice and instructions of the teacher are accepted and applied, are examples of this learning method.
- *Learning by analogy* In this approach, new knowledge is derived by transforming or augmenting the existing knowledge known to the learner. For example, a person who is just learning how to play tennis, but has a good command of table-tennis may be able to transfer their table-tennis skills to learn the new task. The amount of inference needed is more than that in the first two groups.
- *Learning from examples* This method is a special case of inductive learning. The learner is presented with a finite number of examples (training vectors) and is expected to induce a general concept. The amount of inference performed by the learner is significantly greater than in the first two approaches, and also more than in analogous learning, as there exists no background knowledge from which the new concept can be learned. As far as the underlying learning mechanism is concerned, this type of inductive learning is central to learning and adaptation in the author's

current work in its application to the robotics domain. The formalisms of learning and knowledge representation in this approach are further discussed in the application chapters of this work, namely chapters 4, 5 and 7 and the more specific details and the related aspects of learning from examples are thoroughly discussed in [2].

- *Learning from observations and discovery* This is a general form of inductive learning, and is also called *unsupervised learning* [2]. The amount of inference that needs to be performed by the learner is more than that in any of the approaches so far discussed. The learner is not provided with a set of examples of a particular concept, nor does it have access to a mechanism of labelling the internally generated instances. An important sub-set of learning from observations is *active experimentation*, where the learner perturbs its environment to observe its result. For example, this can be guided by some theoretical constraints or some general criteria of interest.

Learning from observations as a result of environmental experimentation, as a general concept, is the issue of the learning algorithms addressed in chapters 5 and 7 of the current work. However, these algorithms fall in the category of learning from examples, as far as the learner (the induction mechanism) is concerned. Chapters 4, 9, 10 and 11 of [2] provide a rigorous treatment of this type of learning.

3.2 Introduction to DTs

From the point of view of data abstraction, a DT is a data structure, usually organised in a top-down manner, incorporating a finite number of nodes (blocks 1 to 6 of Figure 3.1) connected together by straight lines termed *branches*. The nodes of a DT are called *decision nodes* (blocks 1 and 3) and *terminal nodes* (shaded blocks). A decision node accommodates a feature and is a node in which the outcome of testing this feature results in one or more feature values. A terminal node, as the name suggests, has no "children" nodes and is commonly known as a *leaf*. This is a feature value rather than a feature and represents the *class* or the category which labels a certain pattern initiated in the root of the DT (block 1).

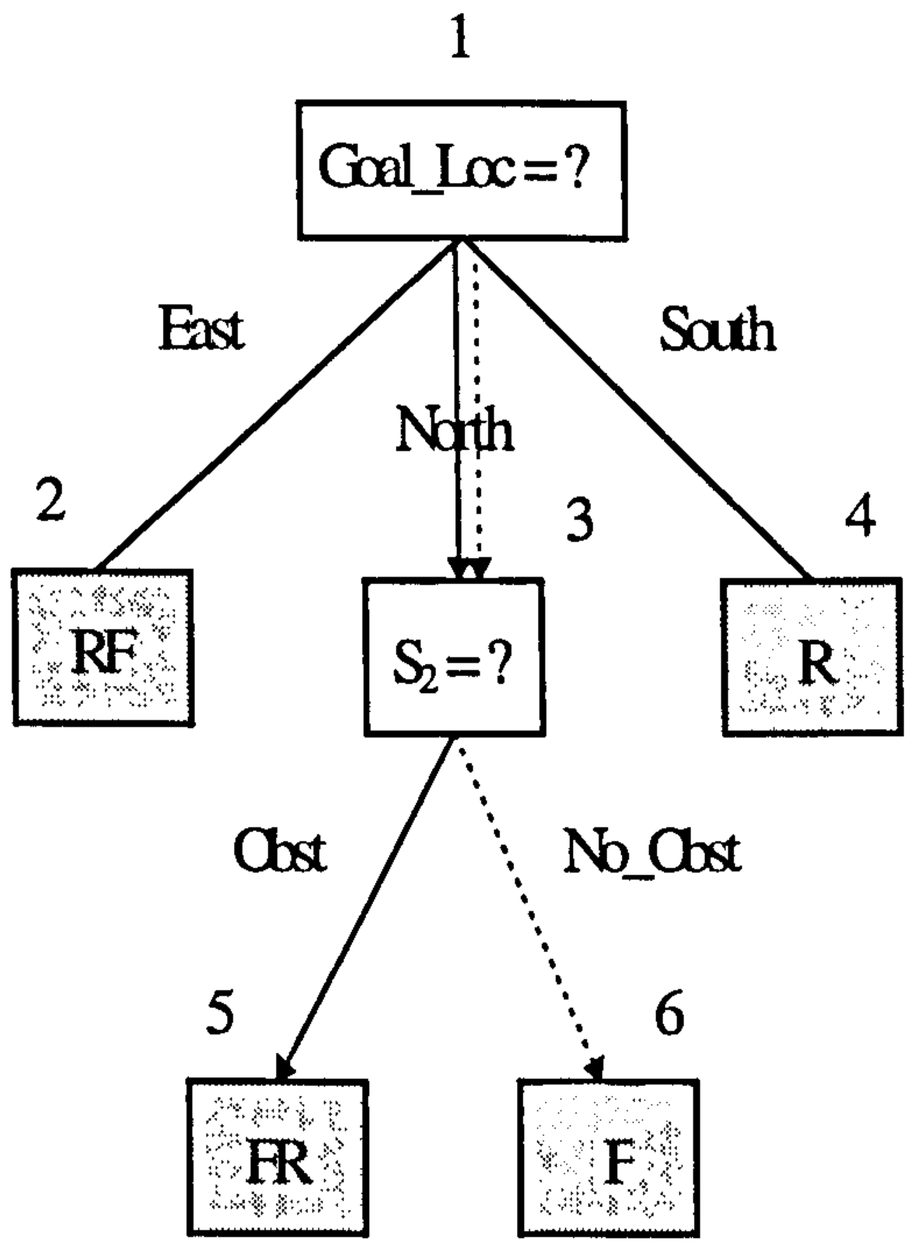


Figure 3.1 A typical decision tree organised in a top-down structure

In Figure 3.1, Goal_Loc and S_2 are both features (attributes), and their outgoing branches are the sub-set of the possible values that these features can assume. The terminal nodes, however, are the labels or classes that categorise a certain pattern starting in the root of the tree (block 1) and terminating in that class. For further illustration, consider the two directed paths (broken and solid lines). Searching the DT along those paths would generate two rules, as shown in Table 3.1.

$\Re_1$: **IF** Goal_Loc = North $\wedge$ S_2 = Obst **THEN** FR.	(Path 1, 3, 5)
$\Re_2$: **IF** Goal_Loc = North $\wedge$ S_2 = No_Obst **THEN** F.	(Path 1, 3, 6)

Table 3.1 A sub-set of the possible rules generated by searching the space of the DT

3.3 The underlying principles of DT construction

DTs are one way of representing inductive knowledge. The foundations of DT learning are rooted in concept learning systems. Such systems attempt to learn or to model a certain

concept by a set of examples. The learned concept can then be modelled either in the graphical form (as in Figure 3.1) or as a set of rules, as partially shown in Table 3.1.

Two different implementations of concept learning systems are the algorithms ID3 [3] and AQ [4]. The structural difference between these two implementations is that ID3 generates DTs of the type shown in Figure 3.1, whereas AQ produces a set of production rules more suitable for expert system applications. These two algorithms have been used as the basis for the development of many other learning techniques, for instance ID3 is the ancestor of ASSISTANT-86 [5], C4 [6], C4.5 [7], ID4 [8], ID5 [9], ID5R [10] and all the versions of ITI, namely ITI-2.5 [11] and ITI-2.8 [12]. The AQ algorithm is the origin of AQ11 [13], AQR [14], AQ15 [15] and CN2 [14]. The descendants of ID3 and AQ have been designed to cope with larger data sets, missing values, noise and improved feature selection, are able to produce smaller DTs or rule spaces and can perform incrementality while keeping the core learning algorithms intact.

The skeleton of the mechanism for DT construction is that of "divide and conquer". This formalism was first implemented in ID3. In the following, this is briefly discussed in a rather descriptive manner. However, for further details the reader is referred to [3,7,16].

Consider a set of training examples T and a set of classes $C = \{C_1, C_2, ..., C_n\}$. The DT generation starts at a node containing all the training examples. ID3 checks if the set T can be classified by a single class. If such is the case, the DT is limited to the one node and the algorithm stops. If the examples are of mixed classes, branches need to be grown on that node which will test an attribute of the examples, and classify them into groups corresponding to the values of that attribute [16]. An attribute is considered as a good test to use when most examples with the same attribute value also have the same class label.

The efficiency of attributes in ID3 is measured by a heuristic function called entropy E (the lower the entropy, the more efficient the classification), defined as follows:

$$E = \sum_i w_i E_i \tag{3.1}$$

where w_i is the weight of the i^{th} branch defined as the number of training examples in branch i divided by the total number of examples at the parent node, and E_i is the entropy of the i^{th} branch given by:

$$E_i = -\sum_j p_j \log_2 p_j \tag{3.2}$$

p_j is the probability of occurrence the j^{th} class in this branch, as estimated from the training data. The attribute yielding the lowest entropy is placed on the node which is expanded, and branches are attached to that node which correspond to the different values of that attribute.

Each individual leaf in the new DT is examined. If all the examples at a leaf have the same class, this node is complete and the class is assigned to that node. If the examples are of mixed classes, the DT must be expanded at this node. To accomplish this, the same procedure is repeated again, as follows:

- Test each individual feature
- Pick the best (the lowest entropy)
- Expand the node by adding branches corresponding to the number of feature values

To clarify the above algorithm, we consider a small data set adapted from [16] which contains eight training vectors of the form shown in Table 3.2. To demonstrate how the algorithm works, consider the placement of the feature *Age* at the root of a virtual DT. To evaluate the entropy E and using the data in Table 3.2, it can be observed how the examples are classified into groups at this node.

Age = Old: 1 example is of Lion and 2 are of Not lion

Age = Young: 2 examples are of Lion and 3 are of Not lion

The values of E_i and E are calculated as follows:

$E_{old} = -(1/3 * \log_2 1/3 + 2/3 * \log_2 2/3) = 0.918$

$E_{young} = -(2/5 * \log_2 2/5 + 3/5 * \log_2 3/5) = 0.971$

$E = 3/8 * E_{old} + 5/8 * E_{young}$

$E = 0.951$

Attributes (Features)			Class
Furry	Age	Size	
Yes	Old	Large	Lion
No	Young	Large	Not lion
Yes	Young	Medium	Lion
Yes	Old	Small	Not lion
Yes	Young	Small	Not lion
Yes	Young	Large	Lion
No	Young	Small	Not lion
No	Old	Large	Not lion

Table 3.2 A small set of training vectors for illustrating DT induction (adopted from [16])

Applying the same procedure, the entropies for *Furry* and *Size* would be 0.607 and 0.500, respectively. It is evident that the feature *Size* scores the lowest entropy value, hence it is chosen for splitting the DT at the root. The feature values *Small* and *Medium* classify the corresponding examples into distinct classes Not Lion and Lion, respectively. However, the training examples containing *Large* are of mixed classes and hence the DT must be expanded at this node. According to the previous calculations, *Furry* scores the second highest entropy value, and is chosen to expand the DT at this node. This attribute is able to classify the remaining examples into unique classes, and the algorithm stops. This is shown in the DT generated by the ID3 algorithm (Figure 3.2), and contrasted with the rule set which is produced by the AQ algorithm (Table 3.3).

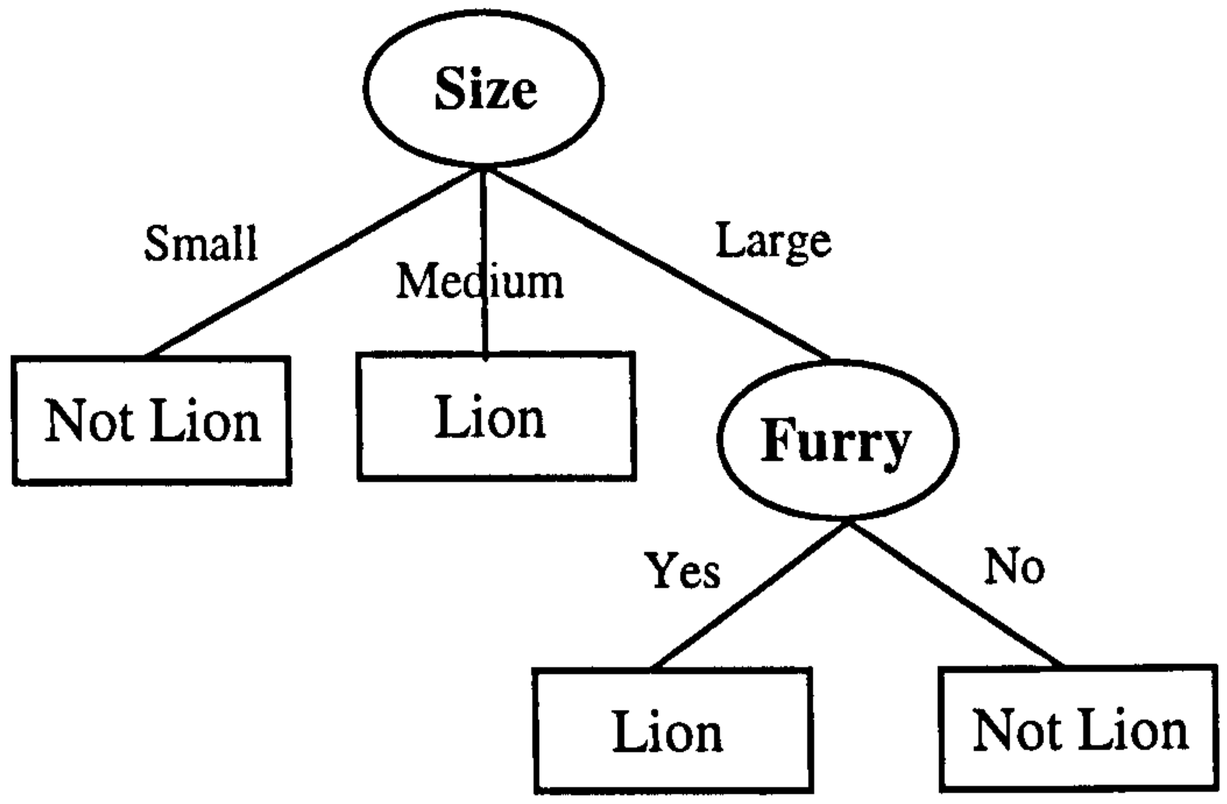


Figure 3.2 A multi-split DT generated by ID3 [3] using the data set of Table 3.2

Criteria for efficient feature selection are discussed in [7,9,10,12,17,18,19]. For example, the algorithm FOCUS-2, introduced in [17], is reported to improve greatly the performance of ID3 if the training data is pre-processed (*prior* to supplying to ID3) using FOCUS-2 to filter out the irrelevant features.

$\Re_1$: **IF** Furry = Yes $\wedge$ Size = Large **THEN** Class = Lion
$\Re_2$: **IF** Size = Medium **THEN** Class = Lion
$\Re_3$: **IF** Furry = No **THEN** Class = Not Lion
$\Re_4$: **IF** Size = Small **THEN** Class = Not Lion

Table 3.3 A set of rules induced by AQ [4] using the data set of Table 3.2

3.4 Types of DTs

As far as the tree structure is concerned, DTs can generally be divided into two categories:

- Multi-split DTs
- Binary DTs.

As shown in Figures 3.1 and 3.2, multi-split DTs have the characteristic of being able to generate multi-dimensional outcomes as the result of testing a feature in a decision node. Binary DTs, however, always evaluate the result of testing a condition into a fixed set of logical values, namely *yes* and *no*. The left branch emerging from any decision node corresponds to *yes* and the right branch to *no*, as depicted in Figure 3.3.

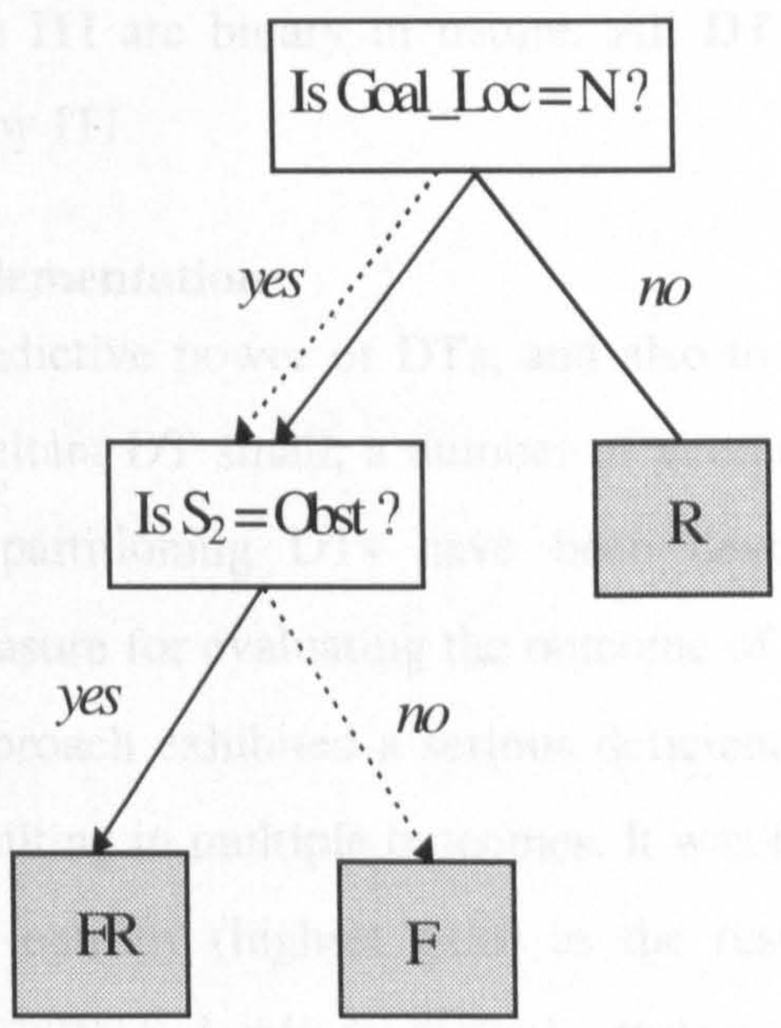


Figure 3.3 An example of a binary DT (e.g. ITI [11])

To illustrate what information is contained in binary DTs, consider the two distinct paths (solid and broken lines) starting at the root of the DT shown in Figure 3.3 and ending at the two terminal nodes. Traversing the DT along these paths would result in the following rules:

$\Re_1$: **IF** Goal_Loc = North $\wedge$ S_2 = Obst **THEN** FR. (Solid Lines)
$\Re_2$: **IF** Goal_Loc = North $\wedge$ S_2 = $\neg$ Obst **THEN** F. (Broken Lines)

Table 3.4 A set of rules produced after traversing the binary DT of Figure 3.3

An important difference in the formulation of the two sets of rules (Table 3.1 and Table 3.4) can be seen when testing the state of S_2 in the second rule in each set. Although *No_Obst* and ¬ *Obst* may initially appear to be logically identical, *No_Obst* is a unique feature value from the set of values that feature S_2 can take, whereas ¬ *Obst* specifically excludes the feature value *Obst*, thereby implying that S_2 can be in any state other than *Obst*. Techniques such as ID3 and C4.5 implement the multi-split approach, whereas the tree networks generated by ITI are binary in nature. All DT networks produced in the current work are generated by ITI.

3.4.1 Alternative DT implementations

In order to enhance the predictive power of DTs, and also to meet other criteria such as keeping the size of the resultant DT small, a number of alternative measures of assessing the results of tests and partitioning DTs have been developed. For example, ID3 implements the entropy measure for evaluating the outcome of a decision test, but Quinlan [7] considered that this approach exhibited a serious deficiency, in that ID3 has a strong bias towards these tests resulting in multiple outcomes. It was found that these tests tended to yield the smallest total entropy (highest gain) as the result of multiple partitioning (according to expression (3.2)) and this is a weak strategy as far as predictiveness is concerned. To remedy this problem, the inherent bias in gain was replaced by a normalised gain which implements the *gain ratio* (GR) [7]. This measures the information relevant to classification, whereas the entropy method represents an information gain by dividing the DT into *n* sub trees (see [7] for more details). Quinlan reports in [7] that the GR criterion is robust and typically gives a consistently better choice of test than the gain criterion.

The early implementations of ID3 could only handle symbolic knowledge, and were consequently limited in the applications which could be tackled. Different application domains and constraints such as noise tolerance, dealing with numerical feature values and missing data have posed heavy demands on the quality of DT learning and construction. C4.5, a successor to the ID3 algorithm, is able to cope with the problem of numerical feature values and missing data (although not in an optimised sense). C4.5 allowed the application of DTs to a wider range of subject areas.

Resting on the same core mechanism of DT construction, the early versions of ITI such as ITI 2.5 [11] also implement the GR for testing attributes. However, an alternative mechanism, namely the modified Kolmogrov-Smirnoff (KS) distance, has been adopted in ITI-2.8 and is able to generate smaller trees and reduce the expected number of tests in each decision node without incurring a significant change in the classification accuracy [12]. For example, consider a two-class case, A and B, with a continuous variable x. The KS method attempts to find an optimal cut point α that partitions the values of x into two distinct blocks. Given the cumulative distributions functions associated with each class, namely $F_A(x)$ and $F_B(x)$, an optimal cutpoint α is the one that maximises $|F_A(\alpha) - F_B(\alpha)|$. This maximum value is the KS distance for that specific variable. In the majority of cases, neither $F_A(\alpha)$ nor $F_B(\alpha)$ is analytically available, in which case they are approximated by counting how many instances from each class fall into each block of the partition [12]. For further details on KS distance, the reader is referred to [12,20].

Utgoff has incorporated the KS distance feature selection metric in ITI-2.8 and has demonstrated in [12] that the size of DTs generated are significantly smaller than those produced using the GR as the partition metric. Similarly, the expected number of tests in induced DTs is significantly fewer than that produced using GR. The number of expected tests plays an important role in DT induction, as it determines how many tests should be performed on average to determine a classification [12]. Both approaches, however, typically achieve the same classification accuracy.

3.4.2 Modes of operation

The paradigms of learning from examples can be implemented to learn the new concept in one of two modes:

- Batch-mode, one-trial mode or off-line mode
- Incremental or on-line mode

In batch-mode learning, all training examples are presented to the system at once, and the learner forms one hypothesis about the concept to be learned. In incremental learning, however, training examples are supplied singly and in succession, the learner builds one or more hypotheses consistent with the training data and the new hypotheses refine the

previous ones, thereby accommodating the new knowledge. Incremental mode learning more closely parallels its human counterpart, in that the learner is able to use partially-learned concepts. In the currently available DT learning systems (both commercial and non-commercial), ID3 and C4.5 implement the batch-mode approach, whereas all versions of ITI are capable of both batch-mode and incremental learning.

The author argues that the characteristics found in incremental learning paradigms such as ITI are appropriate to the current work and to on-line learning systems, in general. This argument is justified in the discussions and analyses presented in chapters 4, 5 and 7 and is supported by the experimental results presented in these chapters.

3.5 Recent advances and developments

DT learning paradigms have undergone extensive research and development since the introduction of ID3 in early 1980s. These efforts and investigations have been in response to the needs of the industrial, economic and business sectors, and include the following.

- Optimisation of tree size
- Improving classification accuracy
- Achieving optimum feature selection
- Noise tolerance and coping with noisy data
- Coping with incomplete and partial data
- Incrementality and successive learning
- Generating production rules
- Handling numerical values as well as symbolic knowledge

It is apparent that one can not integrate all the above features in any one implementation, as the dominance of one characteristic usually reduces the effect of one or more of the others. In general, any implementation will involve making a trade-off between the individual characteristics of the learning process.

Recently, there have been three systems of significance developed from the original ID3. These are ITI-3.8, See5/C5.0 and Cubist and those are now briefly discussed and their capabilities outlined.

ITI-3.8 is the most recent offspring of the ITI family. This algorithm remains similar to the version applied in the current work, namely ITI-2.8, but the improvements achieved provide a more compact and faster learning process. The ITI family is well known for its incremental nature and knowledge induction and is also capable of producing, on average, smaller DTs while maintaining the same classification accuracy compared to techniques such as C4.5. The ITI family is freeware (for research purposes), non-commercial data mining tools.

The latest DT development from the author of ID3 is See5 (for Windows 95/NT) and C5.0 (for UNIX). This tool is capable of constructing production rules which replicate their counterpart DT produced on the same set of data. The emphasis of this implementation has been placed on rule induction rather than DT construction, but this is also available for historical reasons [21]. A sample of rule induction by See5 which has been adopted from [21] is shown below. The data set used has been obtained from bank credit card applicants.

Rule 1: (cover 8)
age <= 41
home telephone = given
current occupation = sales
time with bank <= 3
monthly housing expense > 110
savings account balance <= 8
-> class reject [0.900]

Rule 2: (cover 6)
current occupation = office staff
current job status = private sector
liability reference = f
monthly housing expense > 110
savings account balance <= 228
-> class reject [0.875]

Above, *cover* indicates how many training examples from the data set match this rule, and the figure in square brackets (e.g., [0.900]) is an approximation of the classification accuracy of this rule which is in this case 90%.

See5/C5.0 and its "sister system" (as it is called by Quinlan [22]) *Cubist* are similar in that both can handle either numerical or symbolic knowledge. However, See5/C5.0 produces *classification models*, whereas Cubist generates *numerical models* of the concept to be learned. This means that classes (the labels in See5/C5.0) are symbolic, whereas they are numeric in Cubist. Unlike See5/C5.0 which produces both DTs and rule sets, Cubist is designed only for the generation of the production rules. In statistical terminology, the former is called classification-based learning, whereas the latter is called regression-based

learning, hence the term regression DTs is often used. Cubist associates each rule with a multivariate numerical model and when a new example matches the antecedents of a rule, the corresponding multivariate model is used to predict a numerical value.

In the following, a small set of rules induced by Cubist is shown (adopted from [22]) to demonstrate the way the inductive knowledge is represented by Cubist.

Rule 1: [40 cases, mean 11.6, range 3 to 23, est err 3.2]

IF

SDAFBTMP <= 65
INVHT <= 2270
DAGPG > -10

THEN

Max O3 = -14.11 + 0.029 HMDTY + 0.047 SDAFBTMP + 0.013 DAGPG +0.345 INVTMP

Rule 2: [132 cases, mean 19.0, range 4 to 38, est err 4.2]

IF

SDAFBTMP > 65

THEN

Max O3 = -27.4 + 0.185 HMDTY + 0.034 SDAFBTMP + 0.433 INVTMP

Above, the information given in the square brackets is statistical data (these are not vital for the understanding of the rule). The features shown in the rules are abbreviations for "Temperature, Sandberg AFB", "Inversion base height, LAX", "Pressure gradient LAX/Daggett" and "Humidity", respectively. These rules are a sub-set of rules generated by Cubist based on 330 training examples [22] indicating the maximum ozone level (Max O_3) in LA as a function of atmospheric information (the parameters mentioned above). Cubist produces a model described by a total of five rules that is able to predict the ozone level in the air by providing the daily atmospheric information. The model shows that Cubist performs a non-linear numerical mapping in contrast to See5/C5.0 which generate a symbolic label. See5/C5.0 and Cubist are both commercial data mining tools.

3.6 Applications of DTs

As an inductive learning paradigm, DT learning systems offer a number of attractive features such as high prediction accuracy (comparable with that of connectionist methods) [23], comprehensibility (unlike connectionist methods) and ease of use which render them

practical for a wide range of subject areas. They have been applied to a number of application areas, in one of the following two manners.

- In an off-line manner to construct the model of a general concept for prediction of unseen cases.
- Integrated in multi-strategy systems for learning, adaptation and automation.

Some examples of applications that fall into the first category include: medical diagnosis [24], aviation [25], texture classification [26], cosmic rays identification [27], protein coding location in human DNA [23] and robot perception classification [28]. DTs are also used in embedded control, adaptive systems and multi-strategy systems to enhance learning and automation such as: pattern classification [29], motion planning of a robot manipulator [30], fuzzy rule induction [31] and fuzzy DTs for the automatic generation of control rules in the current work [32].

3.7 Summary

This chapter has provided an introduction to machine learning techniques with an emphasis on learning from examples. The learning systems based on this paradigm are able to produce rule sets or DTs (which are employed extensively in this thesis) to conform to specific applications.

This theoretical chapter conveys the necessary information and background which are central to the understanding of the application chapters of this thesis, namely chapters 4, 5 and 7. Due to the improved performance of ITI-2.8 (compared with that of ITI-2.5, C4.5 and ID3) and its availability at the time of the development of this work, it is used in its two modes of operation, namely off-line and on-line learning, and serves as the core of the learning algorithms proposed in chapters 4, 5 and 7 for the synthesis of control rules. The most significant attribute in the current work which favours the use of ITI-2.8 compared to other DTs, is its ability to be embedded in multi-strategy systems for on-line learning and this is the issue of chapter 7. Chapters 4 and 5 apply ITI-2.8 to an off-line and supervised learning mode, and specifically chapter 4 assesses the feasibility of the proposed learning algorithm in qualitative terms, and chapter 5 discusses its application in realistic environments. In chapter 7, ITI-2.8 is integrated into a self-supervised hybrid system which

is able to learn incrementally fuzzy control rules which are in turn used to navigate a robot in unseen and unstructured environments.

References

[1] Carbonell, R.S. Michalski and T.M. Mitchell, "An Overview of Machine Learning", R.S. Michalski, J.G. Carbonell and T.M. Mitchell, *Machine Learning, An Artificial Intelligence Approach (Eds.)*, Tioga Publishing Company, pp. 3-23, 1983.

[2] R.S. Michalski, J.G. Carbonell and T.M. Mitchell, "Machine Learning, An Artificial Intelligence Approach" (Eds.), Tioga Publishing Company, 1983.

[3] J.R. Quinlan, "Learning Efficient Classification Procedures and Their Application to Chess and Games", *Machine Learning, An Artificial Intelligence Approach*, R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), Tioga Publishing Company, ISBN: 0-935382-05-4, pp. 463-482.

[4] R.S. Michalski, "On the Quasi-minimal Solution of the General Covering Problem", *Proceedings of the Fifth International Symposium on Information Processing (FCIP 69)*, Vol. A3 (Switching Circuits), Bled, Yugoslavia, pp. 125-128, 1969.

[5] B. Cestnik, I. Kononenko and I. Bratko, "Assistant 86: A Knowledge Elicitation Tool for Sophisticated Users", *Progress in Machine Learning (Proceedings of the second European Working Session on Learning*, I. Bratko and N. Lavrac (Eds.), Sigma Wilmslow, UK, pp.31-45, 1987.

[6] J.R. Quinlan, P.J. Compton, K.A. Horn and L. Lazarus, "Inductice Knowledge Acquisition: A Case Study", *In Applications of Expert Systems*, Edison-Wesley Wokingham, UK, pp. 157-173, 1987.

[7] J.R. Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers, 1993.

[8] J.C. Schlimmer and D. Fisher, "A Case Study of Incremental Concept Induction", *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA: Morgan Kaufmann, pp.496-501, 1986.

[9] P.E. Utgoff, "ID5: An Incremental ID3", *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI: Morgan Kaufmann, pp.107-120, 1988.

[10] P.E. Utgoff, "Incremental Induction of Decision Trees", *Machine Learning*, Vol. 4, Kluwer Academic Publishers, pp. 161-186, 1989.

[11] P.E. Utgoff, N.C. Berkman and J.A. Clouse, "Decision Tree Induction Based on Efficient Tree Restructuring", *Machine Learning*, Vol. 29, 1997, pp. 5-44 (P.E. Utgoff, Technical Report 95-18, March 17, 1995).

[12] Utgoff and J.A. Clouse, "A Kolmogroph-Smirnoff Metric for Decision Tree Induction", *Technical Report 96-3*, January 10, 1996.

[13] R.S. Michalski and J. Larson, "Incremental Generation of VL1 Hypotheses: The Underlying methodology and the Description of Programms ESEL and AQ11", UIUCDCS-R 78-867, Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, 1978.

[14] P. Clark and T. Niblett, "The CN2 Induction Algorithm", *Machine Learning Journal*, Kluwer Academic Publishers, 3 (4), pp. 261-283, 1989.

[15] R. Michalski, I. Mozetic, J. Hong and N. Lavarac, "The Multi-purpose Incremental Learning System AQ15 and its testing Application to Three medical Domains", *AAAI-86*, Morgan Kaufmann, California, pp.1041-1045, 1986.

[16] P. Clark, "Machine Learning: Techniques and Recent Developments", *Artificial Intelligence, Concepts and Applications in the Engineering (Ed.)*, Chapman and Hall, pp. 65-93.

[17] H. Almualim and T.G. Dietterich, "Efficient Algorithms for Identifying Relevant Features", *Proceedings of the Ninth Canadian Conference on Artificial Intelligence*, Vancouver, BC: Morgan Kaufmann, May 11-15, pp. 38-45, 1992.

[18] U.M. Fayyad and K.B. Irani, "The Attribute Selection Problem in Decision Tree Generation", *The Tenth Annual Conference on Artificial Intelligence*, San Jose, pp. 104-110, 1992.

[19] U.M. Fayyad, "Branching on Attribute Values in Decision Tree Generation", *The National Conference on Artificial Intelligence*, Vol. 1, pp. 601-606, 1994.

[20] J.H. Friedman, "A Recursive Partitioning Decision Rule for Nonparametric Classification", *IEEE Transactions on Computers*, April 1977, Vol. C-26, pp. 404-408.

[21] J.R. Quinlan, "Data Mining Tools See5 and C5.0", Available at: http://www.rulequest.com/see5-info.html.

[22] J.R. Quinlan, "Data Mining with Cubist", Available at: http://www.rulequest.com/cubist-info.html.

[23] S. Salzberg, "Locating Protein Coding Regions in Human DNA using a Decision Tree Algorithm", *Journal of Computational Biology*, 2:3, 1995, pp. 473-485.

[24] I. Kononenko, "Inductive and Baysian Learning in Medical Diagnosis", *Applied Artificial Intelligence*: vol. 7, 1993, Taylor and Francis, pp. 317-337.

[25] C. Sammut, S. Hurst, D. Kedzier and D. Michie (Eds.), "Learning to Fly", *Proceedings of the Ninth Machine Learning Conference,* Morgan Kaufmann, 1992, pp. 385-393.

[26] I. Sillitoe and T. Elomaa, "Learning Decision Trees For Mapping The Local Environment in Mobile Robot Navigation", *Proceedings MLC-COLT Workshop on Robot Learning,* July 1994, New Brunswick, N.J, pp. 119-125.

[27] S. Salzberg, R. Chander, H. Ford, S.K. Murthy and R.L. White, "Decision Tree for Automated Identification of Cosmic Ray Hits in Hubble Space Telescope Images", *Publications of Astronomical Society of the Pacific 107*, May 1995.

[28] G.H. Shah-Hamzei, D.J. Mulvaney and I.P. Sillitoe," Batch-Mode Decision Tree Learning Applied to Intelligent Reactive Robot Control", *IEEE Sixth International Conference on Emerging Technology and Factory Automation*, Los Angeles, September 9-12, 1997, pp. 416-420.

[29] J. Bala, J. Huang and H. Vafaie, "Hybrid Learning Using Genetic Algorithms and Decision Trees for Pattern Classification", *IJCAI Conference*, Montreal, August, 1995.

[30] T. Shibata, T. Abe, K. Tanie and M. Nose, "Motion Planning of a Redundant Manipulator-criteria of Skilled Operators by Fuzzy-ID3 and GMDH and Optimisation by GA", 0-7803-2461-7/95, IEEE.

[31] L.O. Hall and P. Lande, "Generating Fuzzy Rules from Data", *Fifth IEEE International Conference on Fuzzy Systems*, 1996, New Orleans, Vol. 3, pp. 1757-1762.

[32] G.H. Shah-Hamzei and D.J. Mulvaney, "Self-organising Fuzzy Decision Trees for Robot Navigation: An On-line Learning Approach", *IEEE International Conference on Systems, Man and Cybernetics SMC'98*, October 11-14, 1998, San Diego, CA, USA.

Chapter 4

Off-line Learning of a DT Hierarchy Applied to Robot Control:

Simplified Environments

Keep it simple,
as simple as possible,
but no simpler.

Albert Einstein
Quoted by B. Stroustrup in The C++ Programming Language

This chapter presents the application of decision trees (DTs) to behaviour learning in robotic environments. In particular, it discusses the methodology of domain knowledge decomposition into an array of decision trees to grow a hierarchy of individual and homogeneous DTs.

To verify the feasibility of DT applications to robot path planning in the frame of hierarchical learning, this chapter considers the application of this methodology to simplified environments, in which only discrete movements and turning angles are considered and the robot is assumed to be a point object. The next chapter is concerned with the implementation of the same learning concept, but in realistic environments and in an on-line mode.

4.1 Introduction

Robotics has been a popular test bed in recent years for a wide range of learning algorithms. Robot motion planning and navigation, especially reactive-based approaches, have received particular attention, witnessing the application of symbolic techniques [1,2,3,4], connectionist methods [5,6] and fuzzy logic [1,7,8,9,10,11].

Purely reactive strategies [7] implement control laws from a collection of perception-action stimuli to navigate a robot. Reignier utilises a fuzzy data base [7] to implement a system which is not capable of learning, but in which the domain knowledge has been accommodated beforehand in the database in the form of a set of fuzzy rules which cover the entire perception space. Behaviour-based approaches [12,13] tend to be more distributed in nature, incorporating a repertoire of parallel executing behaviours performed by individual units. Although, analytically difficult to prove, it is commonly believed that pure reactive systems are less powerful than behaviour-based approaches [14]. Strictly speaking, this is a task-specific observation, meaning that if a robot is expected to achieve certain goals at run time which are not specified at the design stage, then, generally, purely reactive systems would not be able to perform these tasks satisfactorily. For example, dynamically changing or unknown robot environments are more elegantly tackled using behaviour-based approaches. Reactive systems, however, offer greater efficiency in computation time as less information needs to be stored and processed.

Decision trees [15,16,17,18] have been successfully employed in a number of areas of robotics, particularly in classifiers designed to make decisions based on a set of training examples of symbolic data or numerical values [19,20]. They are used in [3] to classify the contours of local environment after training using data collected from echoes of an

ultrasonic sensor array, in [1] to predict the motion of a robot manipulator and in [2,21] for behaviour learning.

This chapter presents a DT-based hybrid architecture combining the features of low computational complexity (reactive systems) with flexibility at run time to cope with dynamic and unknown environments (behaviour-based systems). As the system is self-learning, no initial knowledge of the environment is provided to the robot, and there is no map making or planning, since planned navigation of a mobile robot is either susceptible to failure due to unpredictable environmental changes or requires the assumption that the surrounding world is stationary [8]. The overall control is distributed over a hierarchy of decision tree networks. The hierarchy is globally trained rather being tuned to a certain perceived world and is able to cope with dynamically changing environments.

The robot explores and is trained in a series of homogenous, yet increasingly complex environments in such a way that the robot sensory perception is decomposed at the training stage into a hierarchy of progressively complex worlds. The worlds' complexity depends on the immediate obstacle configuration rather than the total *number* of disjoint obstacles in the environment. Each generated world is then mapped on a unique layer represented by a DT which accommodates the perceptual situation-action knowledge encoded in rules. The robot builds up its knowledge base from scratch to a level involving a hierarchy of five individual layers. The simplest layer, in which the perception-action stimuli are highly goal oriented, represents the world with no obstacles; the most complex layer in which object avoidance behaviour is the dominant behaviour, is the one where the robot is trapped in a dead-end and all sensors detect objects within a specified distance from the robot.

The author chooses to use ITI-2.8 which is the latest version of the incremental decision trees induction (ITI) [16] as the fundamental building block of the learning system. This has the ability to generate tree networks in batch mode as well as in incremental mode.

This chapter presents the underlying idea of the hierarchical learning approach and describes how individual DTs are generated and are able to co-operate in the solution of complex navigatory tasks. Results of the experiments conducted using a simulated robot are presented.

4.2 The rationale of hierarchical learning design

A robust navigation system relies heavily on safe and simple motions. Safe and simple motions are generally local in nature [22]. That is, simple situation-action reactive rules are needed to deal with the intermediate robot environment to ensure collision free movements. On the other hand, global planning is needed to approach a remote target while avoiding obstacles.

Knowledge decomposition, which is introduced in this chapter, allows local tuning of the DTs in the hierarchy, as described in the previous section. That means, each individual DT in the hierarchy is local to a certain environmental perception. However, complex and global navigatory tasks can only be accomplished by introducing coherence and performing a sequence of consecutive elementary motions. The coherence in motions, on the other hand, is introduced into the algorithm by continuously sampling appropriate behaviour-based DTs for the time a navigation task is in process. This imparts global tuning to the conceptually *resultant* path. Another aspect of knowledge decomposition is the efficiency in managing complex knowledge and multi-dimensional input sensory data.

A somewhat similar approach to the decision tree method applied in the current work has been taken by Sammut *et al* [23] to control dynamic sub-systems of a Cessna aeroplane in an attempt to produce an artificial autopilot by cloning the behaviour of skilled human pilots. The constructed autopilot successfully managed to fly an entire flight mission.

4.3 Terminology and notation

A *world* is defined to be the instantaneous perception of the robot of its environment, based on the sensory stimuli. The robot is initially at a starting point S and is expected to reach the target G. The navigation process is the set containing the finite number of motions the robot carries out in moving from S to G. The input perception P is defined to be the set of five circumference sensors as $P = \{S_0, S_1, S_2, S_3, S_4\}$ in which each $S_i = \{0,1\}$. This is mapped on the symbolic set $S_i = \{n, y\}$, where n means no obstacle has been detected and y means that an obstacle has been detected. Any perceived world is an element of the set $W = \{w_0, w_1, w_2, w_3, w_4\}$, where w_0 is the simplest in which $\sum S_i = 0$ and w_4 the most

complex world (as far as the rule layers are concerned) in which $\sum S_i = 4$ with i = 0,1,...,4. w_4 is not represented as an independent rule layer in the hierarchy as the action space in w_4 is limited to a single class regardless of the goal location. This renders the generation of a corresponding tree unnecessary.

For example, considering the sensor pattern in Figure 4.1 and with the robot at state R_0 in Figure 4.2, the perceived world is w_0 and the returned value of all sensors is zero (no obstacles). At state R_1, however, w_1 is perceived, indicating that only one sensor from the sensor set detects an obstacle and the robot has four possible directions of motion.

4.3.1 Sensor configuration

Figure 4.1 demonstrates the configuration of range sensors scanning the proximity of objects in front of the robot. Adjacent sensors are separated by an angle of 45° relative to the robot centre where each sensor's angle of detection falls into a cone of $\pm 22.5^\circ$ about each sensor axis. In modelling the behaviour of the range sensors, the following assumptions are made:

- Each single movement of the robot covers one cell, i.e. from the centre of one cell to the centre of the adjacent cells, see Figure 4.2.
- Within a specified sampling time, the return value of each proximity sensor has a binary state which is set when an obstacle is detected.

Uncertainty in the values of sensor readings and the possibility of errors in observations and data increments, generally termed as noise, are discussed further in section 4.8. The nature and sources of error, the way it affects input training data and its impact on DT-structure are also addressed in section 4.8.

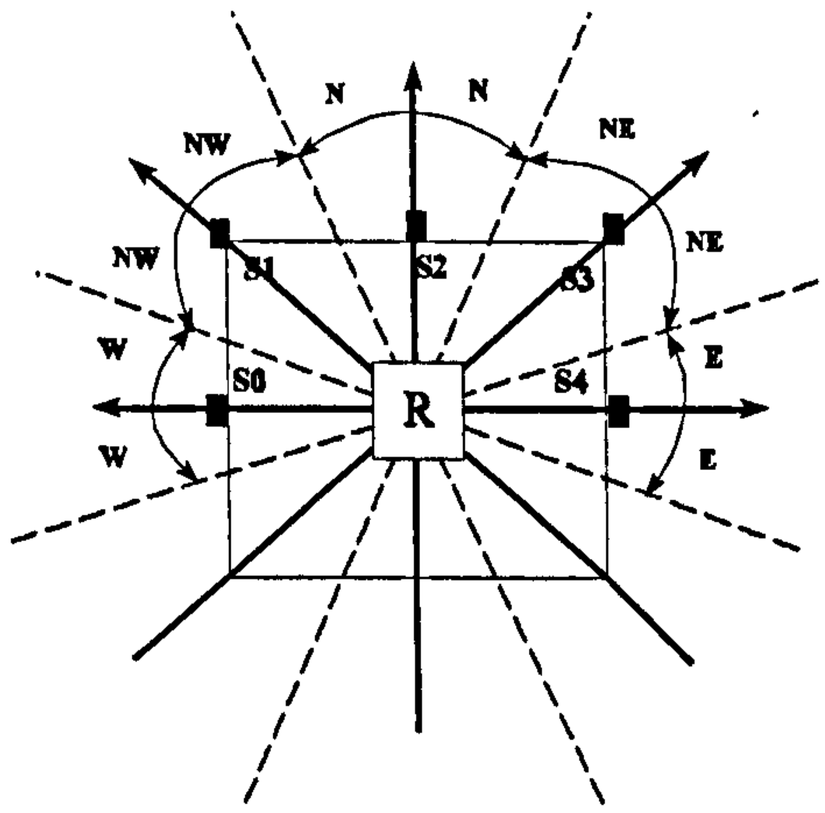


Figure 4.1 Robot shown with circumference sensors (S_0,S_1,S_2,S_3,S_4) and direction cones corresponding to compass points

4.3.2 Representation of the robot environment

The navigation terrain is represented as a 2D-grid, depicted in Figure 4.2. Objects may be individual and well scattered each occupying a single square on the grid or they may be combined to form extended objects covering a number of adjacent squares on the grid. The instantaneous heading vector of the robot is shown as an emphasised directional arrow in Figure 4.2 and sensor returns indicating obstacles and the absence of obstacles are shown by broken and unbroken arrows, respectively.

To clarify further the robot-world relationship, consider some examples of robot states. The sensory perception of the robot at state R3, heading -45°, is $P=\{n,y,y,y,n\}$ and this corresponds to w_3 as three of the sensors detect obstacles. Robot states such as R30, R31, R32 are further examples of w_3 in which the robot has two choices from which to select its next direction of motion. An assumption made is that the size of the robot is sufficiently small in comparison to that of the obstacles [8]. This implies that robot has sufficient space to manoeuvre between two objects which are diagonally adjacent, so for example, at R32 the robot is able to advance at an angle of either -135° or -45°.

If in a w_3 state such as R30, the target is exactly behind the wall, the resulting robot motion could be oscillatory in that it performs a repeating sequence of movements. This is

a problem inherent in minimum seeking behaviours [5,24]. This problem is resolved by the fact that the robot is trained to identify cases which may lead to oscillatory movements and this information is used to grow a so-called *oscillatory* decision tree [25]. Since this DT incorporates a wall-following behaviour, it is activated in circumstances when a possible oscillation is detected and remains active for classification of the robot perceptions as long as the oscillation remains. Figure 4.3 summarises the interrelationship between worlds in the hierarchy as their complexity varies.

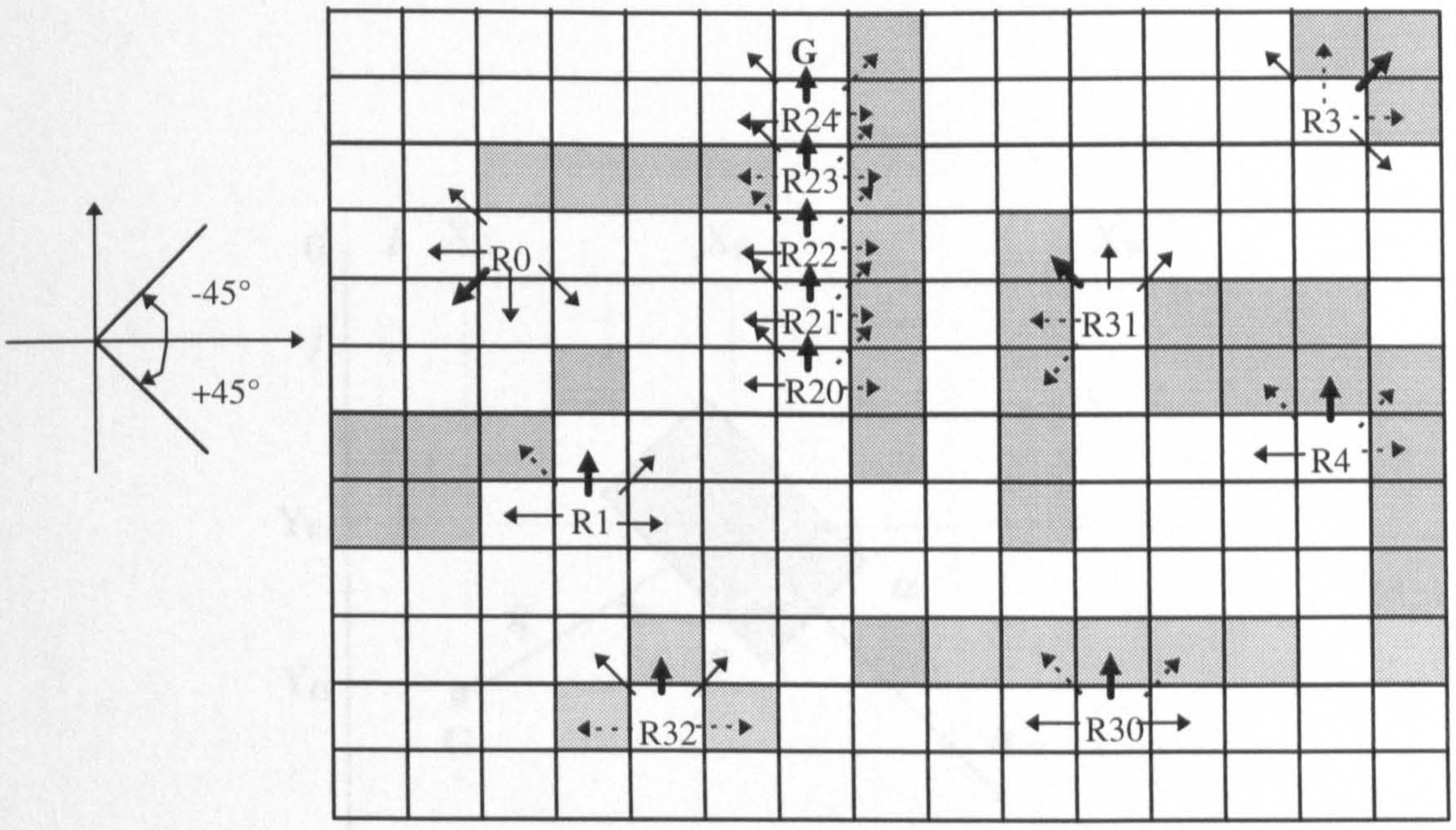


Figure 4.2 Grid representation of the environment and examples of the robot instantaneous perceptions

$$\left|\begin{array}{l} simpler \\ more\ specific\ rules \\ more\ goal-oriented \end{array}\right| \uparrow \left|\begin{array}{l} w_0 \\ w_1 \\ w_2 \\ w_3 \end{array}\right| \downarrow \left|\begin{array}{l} more\ complex \\ more\ generalised\ rules \\ more\ obstacle\ avoiding \end{array}\right|$$

Figure 4.3 Worlds' functionality and interrelationship in the hierarchy

4.4 Calculation of the robot divergence angle

The robot divergence angle θ is required for the following purposes:

- To generate an appropriate steering angle.
- To determine the goal location *relative* to the robot at any time rather than using absolute data. This significantly reduces the dimensionality of the feature vector used in DT generation.

The divergence angle is defined as the angle between the robot heading unit vector $\underline{h}$ and the goal unit vector $\underline{g}$, the vector joining the robot R and the goal G, as depicted in Figure 4.4

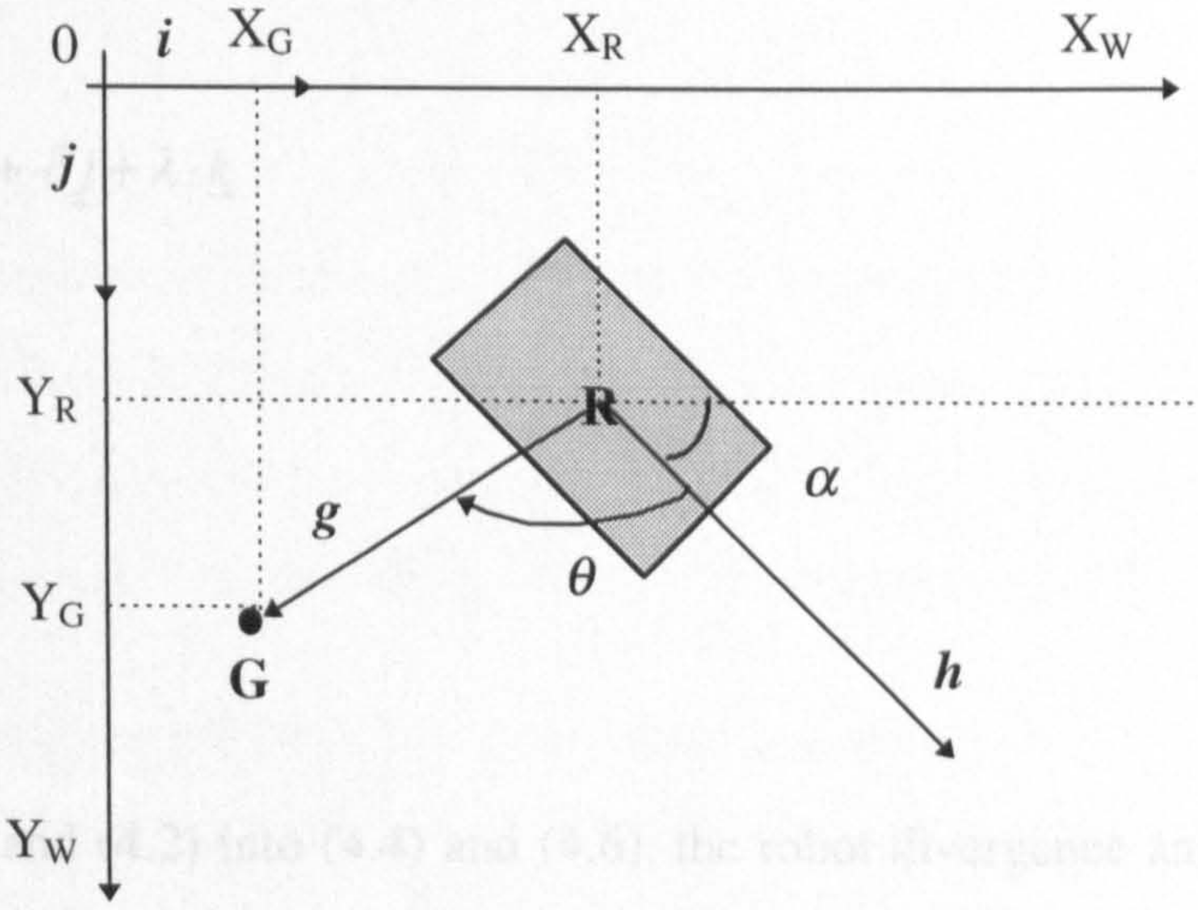


Figure 4.4 Instantaneous goal location relative to the robot at θ in azimuth in absolute world co-ordinates

In the following we define the corresponding unit vectors $\underline{\tilde{h}}$ and $\underline{\tilde{g}}$ to derive the divergence angle θ.

$$\underline{\tilde{h}} = \cos\alpha \cdot \underline{i} + \sin\alpha \cdot \underline{j} \tag{4.1}$$

$$\underset{\sim}{\tilde{g}} = \frac{1}{L}(X_G - X_R)\cdot \underset{\sim}{i} + \frac{1}{L}(Y_G - Y_R)\underset{\sim}{j} \tag{4.2}$$

$$L = \sqrt{(X_G - X_R)^2 + (Y_G - Y_R)^2} \tag{4.3}$$

We use the scalar product between $\underset{\sim}{h}$ and $\underset{\sim}{\tilde{g}}$ to calculate the modulus of θ and their vector product to work out the sign of θ as follows:

$$\cos\theta = \frac{\underset{\sim}{\tilde{h}}\cdot\underset{\sim}{\tilde{g}}}{\left|\underset{\sim}{\tilde{h}}\right|\left|\underset{\sim}{\tilde{g}}\right|} = \underset{\sim}{\tilde{h}}\cdot\underset{\sim}{\tilde{g}} \tag{4.4}$$

$$\begin{vmatrix} \underset{\sim}{i} & \underset{\sim}{j} & \underset{\sim}{k} \\ X_{\underset{\sim}{\tilde{h}}} & Y_{\underset{\sim}{\tilde{h}}} & 0 \\ X_{\underset{\sim}{\tilde{g}}} & Y_{\underset{\sim}{\tilde{g}}} & 0 \end{vmatrix} = \beta\cdot\underset{\sim}{i} + \cdot\delta\underset{\sim}{j} + \lambda\cdot\underset{\sim}{k} \tag{4.5}$$

where

$$\lambda = X_{\underset{\sim}{\tilde{h}}}Y_{\underset{\sim}{\tilde{g}}} - Y_{\underset{\sim}{\tilde{h}}}X_{\underset{\sim}{\tilde{g}}} \tag{4.6}$$

Substituting (4.1) and (4.2) into (4.4) and (4.6), the robot divergence angle in terms of its modulus and sign at any time is completely described by the following set of equations:

$$\theta = \cos^{-1}\left(\frac{\cos\alpha}{L}(X_G - X_R) + \frac{\sin\alpha}{L}(Y_G - Y_R)\right)\cdot\operatorname{sgn}\lambda \tag{4.7}$$

$$\lambda = \frac{\cos\alpha}{L}(Y_G - Y_R) - \frac{\sin\alpha}{L}(X_G - X_R) \tag{4.8}$$

In all the above equations, α and $R(X_R, Y_R)$ are the robot absolute heading and Cartesian co-ordinates respectively, and $G(X_G, Y_G)$ are the goal Cartesian co-ordinates.

4.5 Performance criterion for training set construction

Since the robot motion directions are generated randomly at the training stage, we define usefulness $U = f(C)$ as a measure of goodness of the performed actions. Following the transition from a previous state $n-1$ to a current state n, the cost associated with each motion C is defined as

$$C = \left(\frac{E_{n-1} - E_n}{d}\right)\mathrm{sgn}(\cos(\theta_n - \theta_{n-1})) \tag{4.9}$$

and the usefulness is defined as

$$U = \begin{Bmatrix} 1 & \text{if} & C \geq 0 \\ 0 & \text{if} & C < 0 \end{Bmatrix} \tag{4.10}$$

In (4.9), d is the total travelled distance, E is the Euclidean distance between the robot and the goal, and θ is the robot divergence angle. Only the states delivering a usefulness of unity are positively reinforced by being remembered, the remainder being forgotten. Remembered experiences are collected to form the training examples for DT generation.

4.6 Robot training and decision tree generation

In the training phase where the robot is set to exploratory mode, observations which lead to useful actions are reinforced by being remembered. An observation contains the state of all proximity sensors and the position of the robot relative to the target. A further step needs to be taken to conform each observation increment to the format suitable for DT induction.

To grow a decision tree network, ITI [15] requires a set of training vectors having the format $f_0, f_1, \ldots, f_i, \ldots, f_m, c_i$, in which f_i is a feature with $f_i \in F$ and $F = \{f_0, f_1, \ldots, f_i, \ldots, f_n\}$ is the space of features (input variables) each being defined on a unique space of feature values such as $f_i \doteq \{f_{0i}, f_{1i}, f_{2i}, \ldots, f_{mi}\}$. c_i is a class with $c_i \in C$

from the set of available classes $C = \{c_1, c_2, \ldots, c_i, \ldots, c_k\}$ in which each class has a finite number of discrete values.

The format of the training vectors are configured as a function of world intricacy in order to keep as small as possible the dimensionality of world-dependent trees. That means, in an environment such as w_0 (the first and the simplest layer in the hierarchy) with no obstacles, the perceptual state reduces to only a single feature, namely the relative position of the robot to the target. The robot divergence angle θ is used to supply the instantaneous relative location of the goal, requiring no absolute data to be present in the tree construction. This reduces significantly the size of the tree and tunes the tree globally, as far as the target-seeking behaviour is concerned. Since there is no obstacle in w_0, the robot position relative to the target is used as the only feature and a corresponding class is required in the form of: *goal_rel_loc*, c_i. The set of values that the feature (*goal_rel_loc*) and the class (c_i) can take are:

$$goal\_rel\_loc \equiv \{north, n\_east, east, s\_east, south, s\_west, west, n\_west\} \text{ and}$$

$$C = \{0,1,2,3,4\} \equiv \{left, left\_front, front, right\_front, right\}.$$

Consequently, depending on the location of the goal, each individual class c_i is mapped on an output reflex from the above set to drive the robot.

Having specified the format of the training entities, the algorithm generates an appropriate class to complete each training vector needed for tree construction. There is no domain expert intervention in the process of class generation; the algorithm evaluates each motion with the usefulness function and reinforces the positive ones. Positive experiences are remembered iteratively and collected to train the robot. Figure 4.5 (see page 84) shows the tree network to represent world w_0 of the tree hierarchy.

Tree induction in the remaining worlds is carried out in the same manner, except that the increase in the dimensionality of the feature vector is augmented by the sensory data to the

new format, namely $S_0, S_1, S_2, S_3, S_4, goal\_rel\_loc, c_i$. Figures 4.6, 4.7 and 4.8 (see pages 85 to 87) demonstrate DTs representing worlds w_1, w_2 and w_3.

When inconsistent training examples occur [15,16] and this will result in leaves in the DT representing more than one class, feature patterns can be classified into multiple classes. In terms of output reflexes, when the robot is in a certain state relative to the goal, more than one direction of motion is predicted leading to a conflict in the rules. To produce a single output reflex from those available, a random selection is made.

4.7 Decision trees for classification

Section 3.4 of the previous chapter introduced the notion of DTs, various types and their implementation, particularly in this work. In the following, a simple navigation task is decomposed into its world specific perceptions, and it is demonstrated how each perception is mapped on a unique DT to synthesise control rules.

4.7.1 An example of the rule layer switching

To clarify the fundamental principles on which the navigation is based and to demonstrate how different rule layers are sampled based on instantaneous perceptions, a simple example of a target-seeking task is considered.

With the robot located at position R20 in Figure 4.2, the aim is to reach the target G (shown top centre) while avoiding obstacles. The robot is assumed to have an initial heading angle of -90°. The goal location relative to the robot is initially calculated to be in the north, but will be updated in every state. Table 4.1 illustrates how different worlds are perceived as vector P changes state and how the robot acts on them. Using the state vector P and the state variable $goal\_rel\_loc$, the DTs corresponding to w_2 and w_3 are traversed in order to classify the perception patterns. These paths are shown as directed lines and numbered in the order they are searched in the DTs for the worlds w_2 and w_3 (Figure 4.7 and Figure 4.8 (see pages 86 and 87), respectively). The directions of motion are determined by the terminal nodes, their values consistently being "2" in this particular example. As described in section 4.6, this class corresponds to *front* as an output reflex that drives the robot forwards.

Firing Order	Robot State	Perception P	Goal_rel_loc	World State	Direction of Motion
1	R_{20}	n,n,n,y,y	north	w_2	forwards
2	R_{21}	n,n,n,y,y	north	w_2	forwards
3	R_{22}	n,y,n,y,y	north	w_3	forwards
4	R_{23}	y,n,n,y,y	north	w_3	forwards
5	R_{24}	n,n,n,y,y	north	w_2	goal state

Table 4.1 Decomposition of a sample navigation task in terms of state variables and output reflexes

Table 4.2 demonstrates how each individual control rule can be synthesised by switching between DTs using the same robot states as those used in Table 4.1. Each individual rule is the result of following a DT along the path leading to a terminal node.

Robot State	Control Rules
R_{20}	**IF** $(S_2 = \neg y) \wedge (S_1 = \neg y) \wedge$ (goal_rel_loc $= \neg$ s_east) $\wedge$ (goal_rel_loc $= \neg$ n_east) $\wedge$ (goal_rel_loc $= \neg$ n_west) $\wedge$ (goal_rel_loc $= \neg$ north) **THEN** C = 2
R_{21}	**IF** $(S_2 = \neg y) \wedge (S_1 = \neg y) \wedge$ (goal_rel_loc $= \neg$ s_east) $\wedge$ (goal_rel_loc $= \neg$ n_east) $\wedge$ (goal_rel_loc $= \neg$ n_west) $\wedge$ (goal_rel_loc $= \neg$ north) **THEN** C = 2
R_{22}	**IF** $(S_2 = \neg y) \wedge (S_1 = y) \wedge (S_4 = y)$ **THEN** C = 2
R_{23}	**IF** $(S_2 = \neg y) \wedge (S_1 = \neg y) \wedge$ (goal_rel_loc $= \neg$ n_east) **THEN** C = 2
R_{24}	**IF** $(S_2 = \neg y) \wedge (S_1 = \neg y) \wedge$ (goal_rel_loc $= \neg$ s_east) $\wedge$ (goal_rel_loc $= \neg$ n_east) $\wedge$ (goal_rel_loc $= \neg$ n_west) $\wedge$ (goal_rel_loc $= \neg$ north) **THEN** C = 2

Table 4.2 Control rules synthesised by searching different rule layers to describe a specific trajectory. Above, symbol $\neg$ is to be interpreted as logical NOT and C = 2 as "output reflex is *front*", *as* shown in section 4.6.

4.8 Noise and uncertainty

In the real-world, errors in the presentation of sensor values to the learning algorithm affect its performance. Such errors can arise because of mistakes in the recording feature values, incorrect classification of training vectors by the teacher [26,27] or erroneous sensory data. The first two types are both systematic errors, whereas the latter is categorised as random error or noise [28]. In the current work, the first two sources of error are excluded since the generation and classification of training examples are performed entirely automatically with no external intervention. Hence, the source of error is limited to the additive noise which can be attributed to uncertainty in sensor readings.

Under ideal circumstances, an obstacle's presence would always be correctly indicated by a sensor when it falls within its field of detection. However, in real world environments, due to sensor noise, there will be cases where sensor readings fall into a region where misclassification can occur. It is important that any learning process is able to continue to perform satisfactorily under these circumstances. In order to deal with such noisy data, a suitable noise model and a modification to the learning algorithm were developed and these are described below. As the introduction of noisy examples into the training of the DTs produces larger trees, methods for pruning trees are also considered.

4.8.1 Noise modelling

As discussed, behaviour learning is based on the observations of the robot from its physical environments. These observations which are suitably formatted for knowledge extraction correspond directly to the robot perceptions (sensor readings). These perceptions are derived using proximity sensor values and are used to deduce whether an obstacle is present. To determine whether an input value D_i associated with sensor S_i indicates an obstacle's presence, a suitable threshold function is used, Figure 4.9.

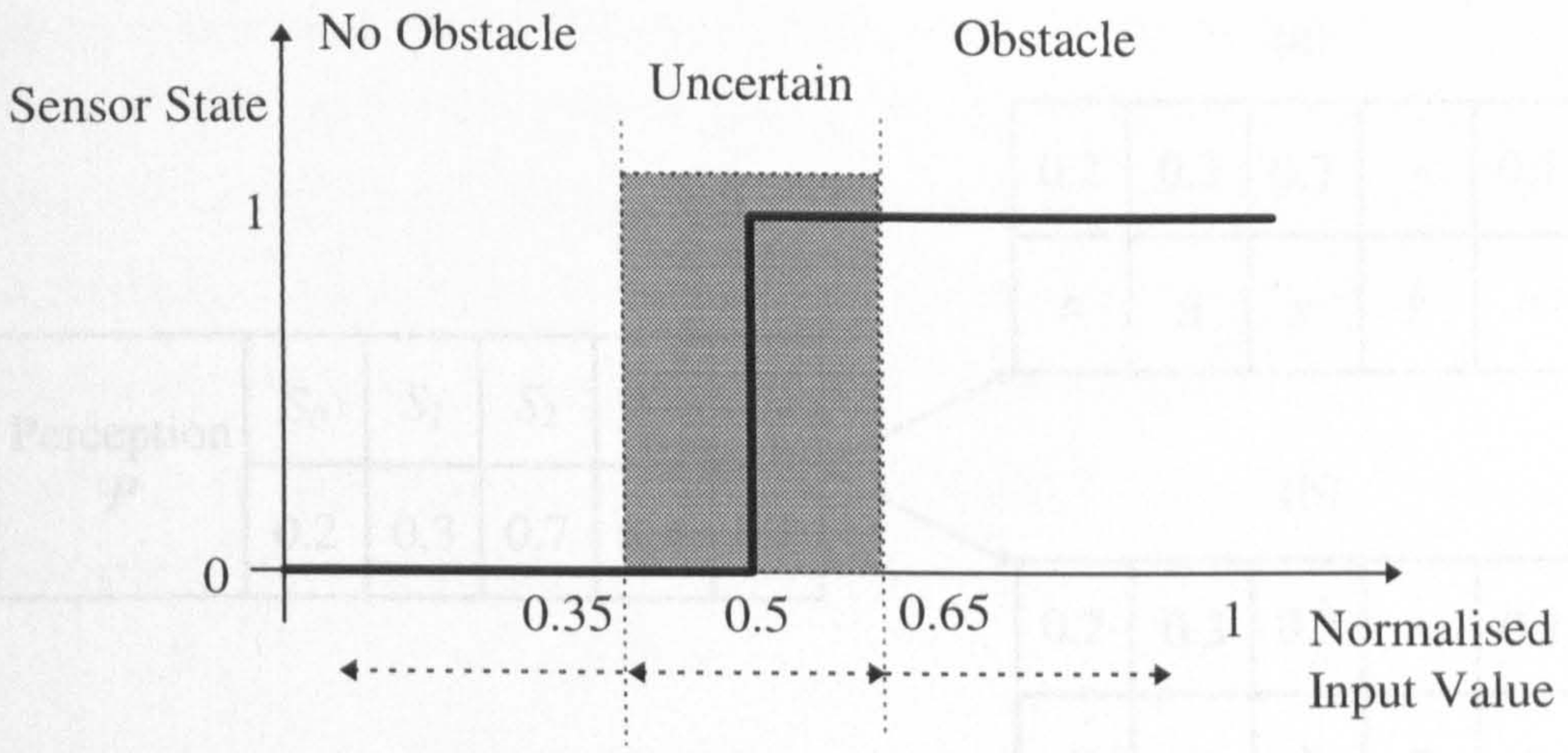


Figure 4.9 The threshold function used to classify sensor values into three categories: (a) no obstacle is detected if the input value falls below 0.35; (b) an obstacle is detected if the input value is above 0.65 or (c) it is uncertain whether an obstacle is present if the sensor value is within ±0.15 of the threshold.

From Figure 4.9, it is known that the presence of noise may cause an error in misclassification of those sensor readings close to the threshold. The extent of this area of potential misclassification is shown shaded. For the environment considered in the current work, a width of ±0.15 around the threshold was found to be suitable.

4.8.2 Modified learning algorithm

The training method previously described involves training a hierarchy of DTs ranging from w_0 with no obstacles, to w_4 where all sensors detect obstacles. To show how training can be carried out when sensor noise is present, consider the example shown in Figure 4.10 when training the DT for w_1, that is, it is known *a priori* that only one sensor should detect an obstacle.

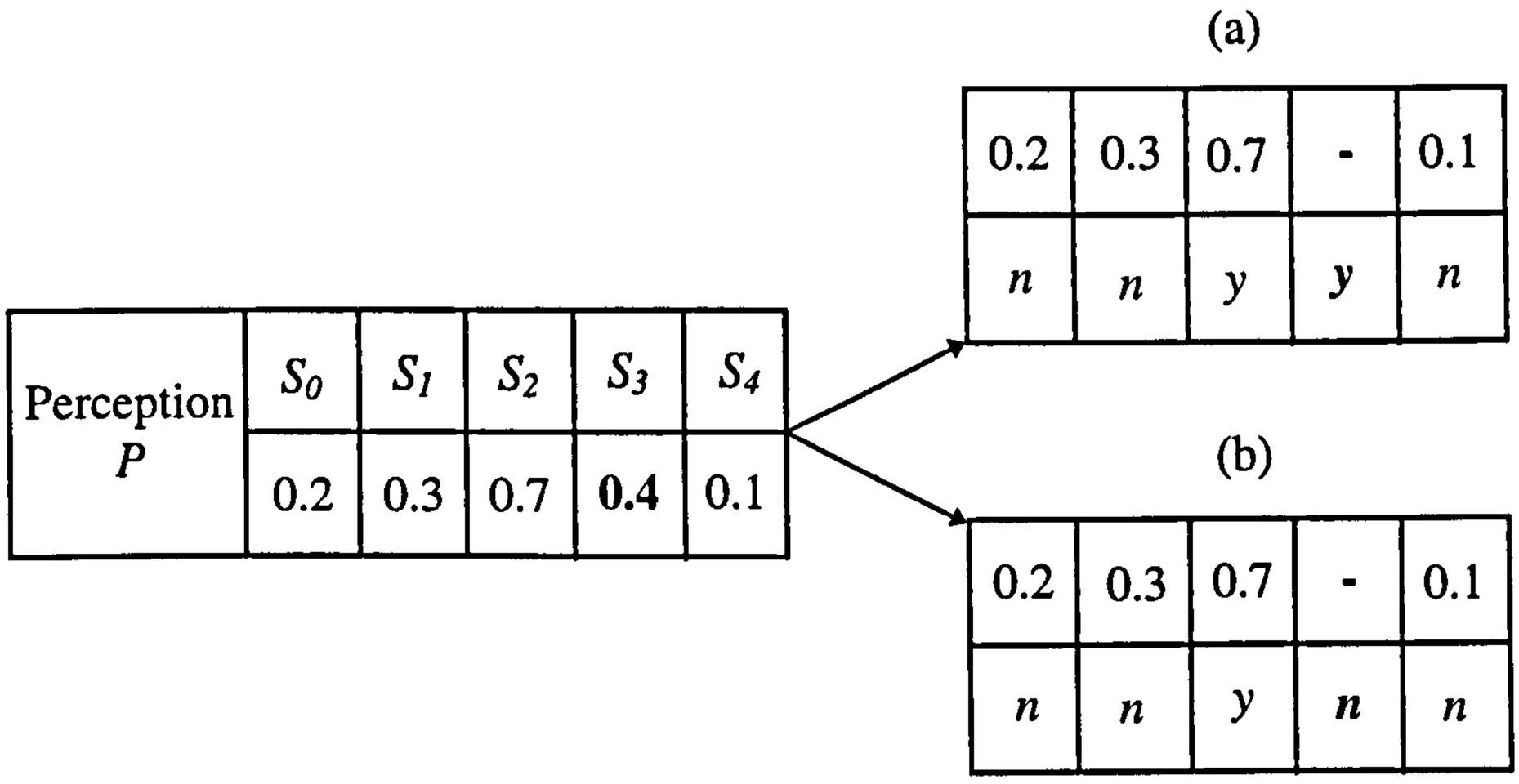


Figure 4.10 The mapping of robot perception P in w_1, where the input value for S_3 falls into the uncertain area shown in Figure 4.9. This gives rise to two possible symbolic interpretations of P for S_3, shown in (a) as "*y*" and in (b) as "*n*".

In order for the navigation system to operate successfully in the face of sensor noise, a perception where one or more sensor inputs fall into the uncertain area needs to be used as an example in the training of the DTs. In Figure 4.10, a single sensor input falls into the uncertain area and this is represented by allowing two possible symbolic interpretations. As training is taking place for w_1, the example must be used in the training of w_1, even though, strictly speaking, pattern (a) represents w_2, as two sensors detect obstacles. In the context of inductive learning, training examples such as pattern (a) are considered as counter-examples to the noise-free representative and homogeneous training examples describing w_1. Counter-examples can be introduced to the batch of uniform and representative examples to expand the search space so as to produce noise tolerant production rules [27]. The existence of counter-examples, however, leads to the specialisation of the training patterns which reduces the generalisation ability of DTs, and hence produces larger tree networks. This effect is shown in Figures 4.11 to 4.13 which are the noisy versions of Figures 4.6 to 4.8.

Once the DTs have been trained as described above, they can be used on real-world examples. Having described the mechanism of training under uncertainty, the strategy to classify imperfect data is considered using examples of real world perceptions.

4.8.3 An example of rule firing in the presence of uncertain data

The DTs in the hierarchy are able to cope with uncertain data existing in the robot perceptions when navigating in unseen environments. The algorithm is modified in such a way that mapping a perception on a world no longer depends on the total number of sensors that detect obstacles, but on the number of sensors whose readings fall outside the uncertain region of the threshold function. Three example of sensor values are shown shaded in Table 4.3 (a) as perceptions P_1, P_2 and P_3.

Perception	S_0	S_1	S_2	S_3	S_4
P_1	0.1	**0.5**	0.8	0.3	0.1
P_2	0.3	**0.5**	0.9	0.2	**0.6**
P_3	0.2	0.9	**0.4**	0.7	0.1

(a)

S_0	S_1	S_2	S_3	S_4	
n	***y***	*y*	*n*	*n*	w_1
n	***n***	*y*	*n*	***y***	w_1
n	*y*	***y***	*y*	*n*	w_2

(b)

Table 4.3 (a) Three examples of uncertain sensory data (in bold face) each interpreted into either a symbolic "*y*" or "*n*" using the threshold function. (b) The number of sensor values indicating obstacles with certainty (shaded) in each pattern gives rise to the world on which each pattern is mapped.

An instantaneous robot perception from its environment such as P_1 needs to be mapped on a unique rule layer to be further classified by searching the tree. In the presence of noise in sensor readings, there exists no direct mapping between the perceptions and the worlds. That is, perceptions such as P_1 are not mapped on w_2 (although two sensors detect obstacles), but on w_1 because the state value of S_1 in Table 4.3 (a) is uncertain and the classification is performed using only partial data. Since the hierarchy has been trained using counter-examples to the uniform representatives, it is able to cope with and to classify inhomogeneous patterns such as in Table 4.3 (b).

4.8.4 Decision tree post-pruning

To deal with the larger trees which result from the use of imperfect data examples in inductive learning systems, *rule-truncation* or decision tree post-pruning can be employed [18,27,28]. In this approach, after a DT has been grown to completion, components deemed unreliable are removed. These components are usually sub-trees which represent a weak correlation between classes and feature values; the classification task of these sub-trees are usually moved to terminal nodes. The resulting DTs are smaller in size, more generalised in character and better able to cope with imperfect data. For example, Figures 4.14 to 4.16 show the pruned versions of the DTs (demonstrated in Figures 4.11 to 4.13) obtained for the noisy sensor data.

Tree pruning is not appropriate when small sample sizes are involved as this would result in *overgeneralised* DTs with low classification accuracy. Conversely, a large number of training examples tends to produce *overfitted* DTs which are themselves large in size and which try to represent individual patterns. Consequently, achieving appropriately-sized DTs with acceptable classification accuracy is a trade-off between the two cases. Our experimental results demonstrate that a sample size of approximately 200 training vectors is appropriate if post-pruning is to be performed on the generated DTs.

4.9 Dynamic rule inhibition and DT augmentation

The learning algorithm of the intelligent system benefits from two mechanisms which refine the trees dynamically and these are discussed below.

Firstly, during the navigation process when the robot applies the learned knowledge to predict its future direction, rules are evaluated as they are fired. Since some rules, particularly those in multiple-class leaves, have been generated when the goal has been located in the overlapping area of two adjacent sensor cones (inconsistent training cases), they prove to be inefficient (by producing negative costs) when applied consecutively to a certain state. The algorithm identifies such rules and inhibits them and restructures the tree accordingly. The inhibited rules are not used thereafter and these are shown shaded in Figures 4.5 and 4.6 which show the DTs for worlds w_0 and w_1, respectively.

Secondly, the algorithm identifies rules which are overgeneralised in the process of tree induction. A rule such as this may drive the robot in certain circumstances towards obstacles and in others towards the target. In such conflicting cases and after the identification process, new training examples are automatically generated. These are integrated incrementally into the tree to augment it and to specialise such rules by re-inferring the tree. For example, the DT shown in Figure 4.17 is the augmented version of Figure 4.12, and Figure 4.18 being its truncated version.

4.10 Results and discussion

The learning mechanism presented in this thesis introduces an efficient approach to synthesising control rules by employing self-learning of domain knowledge from inception rather than under supervision. This has been achieved using DTs and performing symbolic learning without the intervention of human experts. This is similar in concept to the approach taken in [29] and contrasts with that taken in [11] where control rules are manually constructed *prior* to navigation. However, a major advantage of the approach presented in this work is that the generated rules are highly intelligible to human users and easy to follow as can be seen by examining the rules shown in Table 4.2.

Since the principal emphasis of this chapter is to demonstrate the feasibility of DT-based learning for high level decision making rather than low level control, the control algorithm is simulated with a simple character-based user interface. Consequently, the steering angles that can be externally resolved (in the user interface only) for the actuators are in 45° increments, making certain trajectories appear longer than expected, as will be shown in pictures to follow.

In all navigation examples, the robot is positioned at an arbitrary starting point S and is expected to reach the target G avoiding obstacles. Figure 4.19 (a) shows the first stage of learning in the environment without obstacles, namely w_0. The initial leg of the trajectory is rather more exploratory than target-seeking. The locations visited by the robot more than once are highlighted. This shows that the robot first moves away from the target, generating negative costs. The performance then improves and the robot heads towards the goal. After 10 training epochs, the robot demonstrates a significant improvement in finding

the target while keeping the same settings, namely the robot having the same location and starting angle, and also the same goal position. Figure 4.20 is a scenario where the robot is navigating in a sparsely obstacle-populated terrain. In such environments, the robot has access to the knowledge learned in w_0 and w_1. Figures 4.21 to 4.23 are examples of navigation scenarios in more complex surroundings, in which the DTs in the hierarchy have already been grown and are accessible to the robot. Table 4.4 shows the number of training vectors needed on average to build up each individual DT and the entire hierarchy.

	Worlds				
	w_0	w_1	w_2	w_3	w_4
Average Number of Runs	10	11	12	10	8
Number of Training Examples	58	112	95	60	68
Total number of training runs to build up the hierarchy (Mean)					51

Table 4.4 The approximate number of trials and the training examples needed to set up each individual DT in the hierarchy.

The simulation results are of rather more qualitative than quantitative significance for the evaluation of the learning approach. The significant improvement that the robot demonstrates during navigation is that it does not exhibit repetitive motions such as that shown in Figure 4.19 (a), and it is always able to find the target. In some instances, the generated trajectories appear longer than the shortest route available; this can be attributed in part to the simplification made in the resolution used for the turning angle, but it is well known that reactive path planners do not always produce the shortest trajectory [30].

A further limitation which is inherent to purely reactive systems is demonstrated in Figure 4.23. This occurs when the two available behaviours, namely reactivity and target-seeking behaviour are applied alternatively. This mostly happens when the robot follows a long

wall while heading the target and in such circumstances, absolute behaviour arbitration leads to the so-called zigzag trajectories for the time the robot shows wall-following behaviour.

As stated previously, the main objectives of this chapter are to demonstrate that the DT-based hierarchical learning approach is capable of generating safe and simple locomotion reflexes. To demonstrate this, a number of simplifying assumptions were made about the nature of the robot, actuation and the environment. These were implemented in order to keep the dimensionality of the problem domain at a manageable level, thereby allowing a qualitative judgement of the overall performance of the learning mechanism to be made. Therefore, no comprehensive comparison is made at this stage with previous work, as far as the performance of the learning algorithm is concerned. The experimental results prove the feasibility of the approach [21] and provide the incentive for the work presented in the next chapter where (a) the learning algorithm is modified to incorporate incremental and on-line learning, (b) realistic and higher resolution environments and robot configurations are considered.

4.11 Summary

This chapter has demonstrated a symbolic approach based on decision tree learning to the intelligent control of a mobile robot. The perceived world is decomposed into a hierarchy of simple, homogeneous worlds that a positively reinforced robot learning system experiences. At each level, the navigation data collected are applied to train and grow an ITI-based decision tree. Each world is mapped on a rule layer in which the learned knowledge is encoded and, depending on the complexity of the perceived world, rule layers are "switched on" to navigate the robot through an unknown and cluttered environment. The navigation algorithm behaves intelligently in that, following poor performance during the navigation process, learned rules in any layer can be dynamically inhibited or "switched off". The rule layers can also be dynamically augmented to specialise certain overgeneralised rules and this is achieved by the on-line restructuring of the DTs to integrate the new knowledge into existing trees.

The control concept, namely the knowledge decomposition, ensures safe and globally tuned navigation due to the emphasis on reactivity. Safety is achieved in performing local environmental mapping, whereas the global nature is provided by circular sampling of decision tree networks to generate a sequence of elementary motions.

The next chapter of this thesis demonstrates the application of the DT-based hierarchical learning to higher resolution and realistic environments by using simulated environments for the miniature robot Khepera [31,32].

References

[1] Shibata, T. Abe, K. Tanie, and M. Nose, "Motion Planning of a Redundant Manipulator-Criteria of Skilled Operators by Fuzzy-ID3 and GMDH and Optimisation by GA", *Proceedings of IEEE Conference*, 0-7803-2461-7/95, 1995.

[2] T. Zrimec and P. Mowforth, "Learning By an Autonomous Agent in The Pushing Domain", *Robotics and Autonomous Systems*, 0921-8830/91, 1991, Elsevier Science Publishers B.V.

[3] I. Sillitoe and T. Elomaa, "Learning Decision Trees For Mapping The Local Environment in Mobile Robot Navigation", *Proceedings MLC-COLT Workshop on Robot Learning,* July 1994, New Brunswick, N.J, pp. 119-125.

[4] M. Salganicoff and L.G. Kunin, "Active Exploration Based ID-3 Learning for Robot Grasping", *Proceeding MLC-COLT Workshop on Robot Learning,* July 10th 1994, New Brunswick, N. J.

[5] A. Dubrawski and J. Crowley, "Self-Supervised Neural System For Reactive Navigation", *Robotics and Automation IEEE International Conference,* 3 (1994), pp. 2076-81.

[6] J. Tani and N. Fukumura, "Learning Goal-Directed Sensory-Based Navigation of a Mobile Robot", *Neural Networks*, 7 (1994) 3. pp. 553-563, Elsevier Science Ltd.

[7] P. Reignier, "Fuzzy Logic Techniques For Mobile Robots Obstacle Avoidance", *Robotics and Autonomous Systems,* 12 (1994), pp. 143-153.

[8] C.J. Wu, "Fuzzy Robot Navigation In Unknown Environments", *IEEE International Workshop on Emerging Technology and Factory Automation,* August 11-14, 1992.

[9] R. Braunsting, J. Mujika and J.P. Uribe, "A Wall Following Robot With a Fuzzy Logic Controller Optimised by a Genetic Algorithm", *IEEE*, 0-7803-2461-7/95, 1995.

[10] R. Braunsting and A. Ollero, "Evaluating The Wall Following Behaviour of a Mobile Robot With Fuzzy Logic", *IFAC/IMACS International Workshop on Artificial Intelligence in Real Time Control,* Bled, Slovenia, Nov. 1995, pp. 89-93.

[11] H. Surmann, J. Huser and L. Peters, "A Fuzzy System For Indoor Mobile Robot Navigation", *IEEE*, 0-7803-2461-7/95, 1995.

[12] P. Maes and R.A. Brooks, "Learning to Co-ordinate Behaviours", *Proceedings of AAAI'91*, Boston, MA, pp. 796-802, 1991.

[13] M.A. Salichs, E.A. Puente, D. Gachet and J.R. Pimente, "Learning Behavioural Control by Reinforcement for an Autonomous Mobile Robot", *Proceeding of IECON'93*, Vol. 3, pp. 1436-1441, 1993.

[14] M.J. Mataric, "Behaviour-Based Control: Main Properties and Implications", *Proceedings of IEEE International Conference on Robotics and Automation*, *Workshop on Architectures for Intelligent Control Systems*, Nice, France, May 1992.

[15] P.E. Utgoff, N.C. Berkman and J.A. Clouse, "Decision Tree Induction Based on Efficient Tree Restructuring", *Machine Learning*, Vol. 29, 1997, pp. 5-44.

[16] P.E. Utgoff, "Incremental Induction of Decision Trees", *Machine Learning*, 4 (1989), Kluwer Academic Publishers.

[17] J.R. Quinlan, "Decision Trees and Decisionmaking", *IEEE Transactions on Systems, Man, And Cybernetics*; 20 (1990) 2, March/April 1990.

[18] J.R. Quinlan, "C4.5: Programming For Machine Learning"; Morgan Kaufmann Publishers, 1992.

[19] J.R. Quinlan, "Improved Use of Continuous Attributes in C4.5", *Journal of Artificial Intelligence Research,* 4 (1996), pp. 77-90.

[20] T. Elomaa, and J. Rousu, "Finding Optimal Multi-Splits for Numerical Attributes in Decision Tree Learning", *NeuroCOLT,* Technical Report Series, NC-TR-96-041, March 1996.

[21] G.H. Shah Hamzei, D.J. Mulvaney and I.P.W. Sillitoe, "Batch-Mode Decision Tree Learning Applied to Intelligent Reactive Robot Control", *Sixth IEEE International*

Conference on Emerging Technologies and Factory Automation (ETFA'97), September 9-12, 1997, Los Angeles, USA.

[22] M. Kaiser, V. Klingspor, J. Millan, and M. Accame, "Using Machine Learning Techniques in Real-World Mobile Robots", *Intelligent Robot Systems, IEEE Expert*, 0885-9000/95, 1995.

[23] C. Sammut, S. Hurst, D. Kedzier and D. Michie (Eds.), "Learning to Fly", *Proceedings of the Ninth Machine Learning Conference*, Morgan Kaufmann, 1992, pp. 385-393.

[24] G.H. Shah Hamzei and D.J. Mulvaney, "Behaviour-driven Decision Tree Switching to Identify and Resolve System Instability in Reactive Robot Control", *Proceedings of the Fifth International Workshop on Advanced Robotics and Intelligent Machines*, April, 1997, Manchester, UK.

[25] G.H. Shah Hamzei D.J. and Mulvaney, "Local Minima and Oscillation Resolution in Reactive Robotics Using Decision Tree Learning", *IASTED International Conference on Artificial Intelligence and Soft Computing*, July 27-August 1, 1997, Banff, Canada.

[26] R.J. Hickey, "Noise Modelling and Evaluating Learning from Examples", *Artificial Intelligence*, 82 (1996) 157-179.

[27] P. Clark and T. Niblett, "Induction in Noisy Domains", *Proceeding 2nd European Machine Learning Conference (EWSL - 87)*; 1987, pp. 11-30.

[28] P. Bradzil and P. Clark, "Learning from Imperfect Data", *Machine Learning, Meta-reasoning and Logics*, (Eds.) P.B. Bradzil and K. Konolige, 1990, Kluwer, pp. 207-232.

[29] A. Dubrawski and J. Crowley, "Learning Locomotion Reflexes: A Self-Supervised Neural System for a Mobile Robot", *Robotic and Autonomous Systems*, 12 (1994), pp. 133-142.

[30] D.T. Lawton, R.C. Arkin and J.M. Cameron, "Qualitative Spatial Understanding and Reactive Control for Autonomous Robots", *IEEE International Workshop on Intelligent Robots and Systems*, Vol. 2, pp. 709-714, 1990.

[31] O. Michel, Mage Team, I3s Laboratory, CNRS, University of Nice-Sophia Antipolis, France, Khepera Simulator was provided for the Official Khepera Contest at

Evolution Artificial Conference (Nimes, 1997), downloadable from: http://alto.unice.fr/~om/khep-contest.html.

[32] O. Michel and P. Collard, "Artificial Neurogenesis: An Application to Autonomous Robotics", *Proceedings of the 8th International Conference on Tools with Artificial Intelligence*, pp. 207-214, IEEE Computer Society Press, 1996.

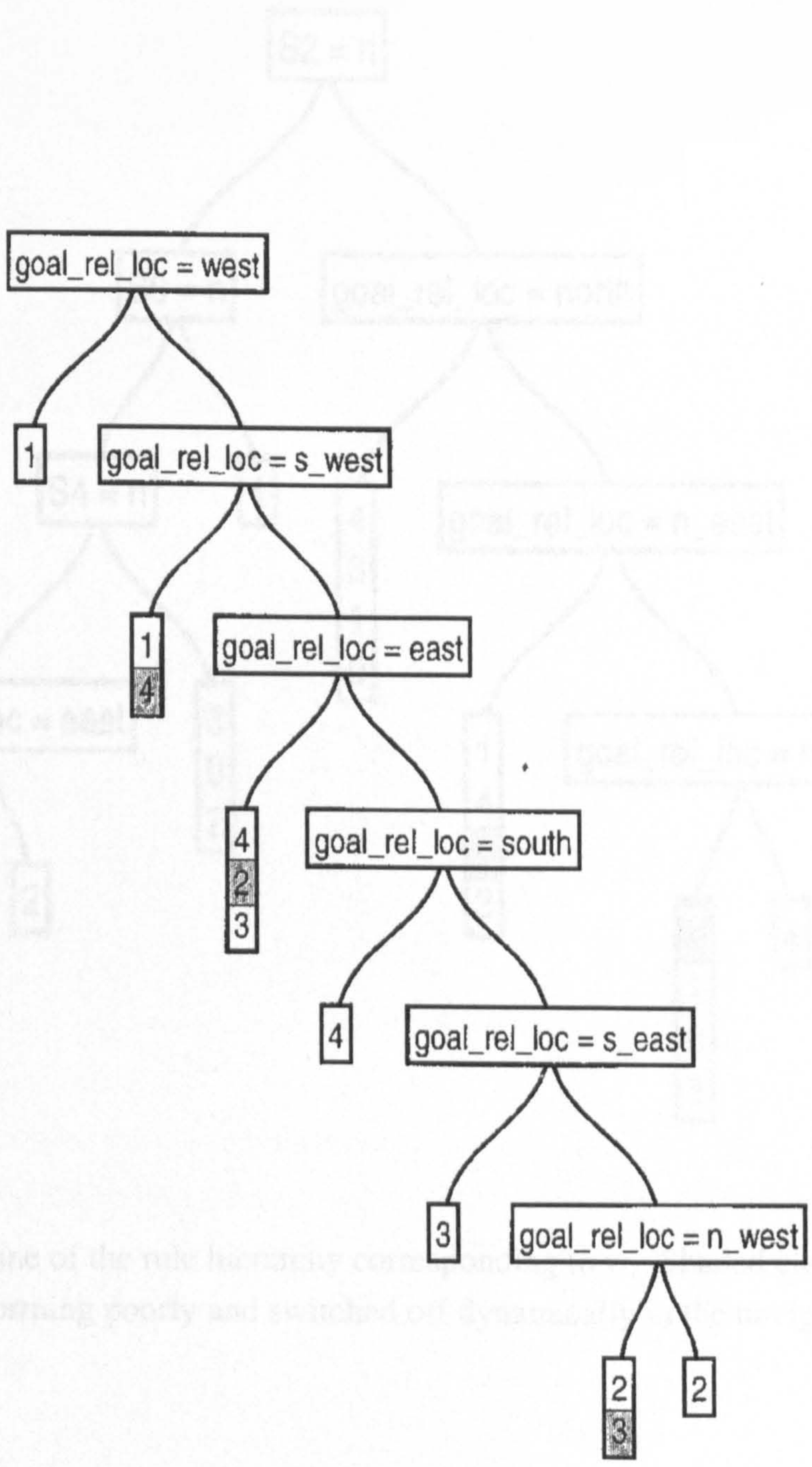


Figure 4.5 Layer zero of the rule hierarchy representing w_0. Classes which are shaded have delivered poor performance during navigation. These are completely inhibited.

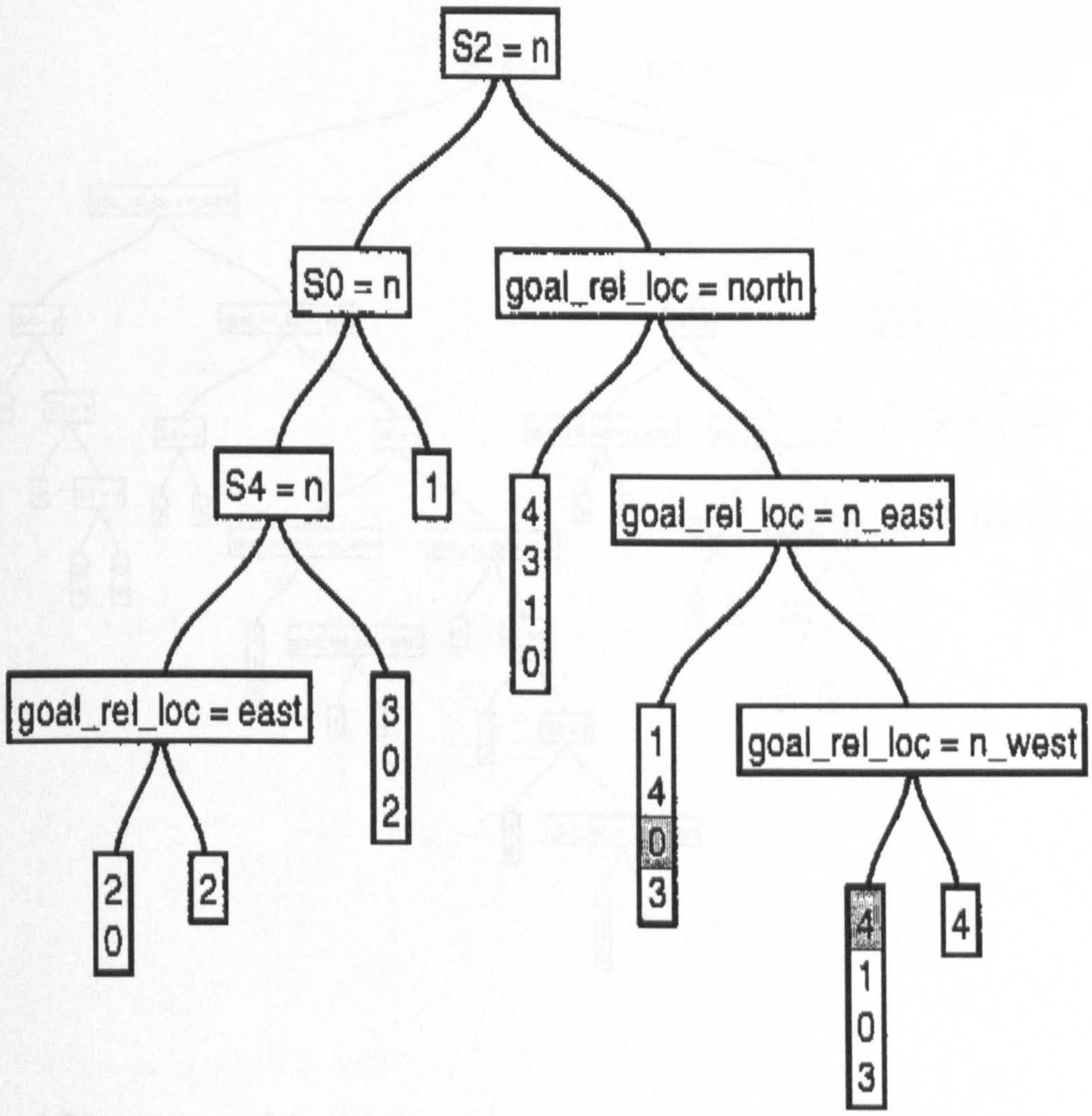


Figure 4.6 Layer one of the rule hierarchy corresponding to w_1. Shaded classes have been identified as performing poorly and switched off dynamically in the navigation process.

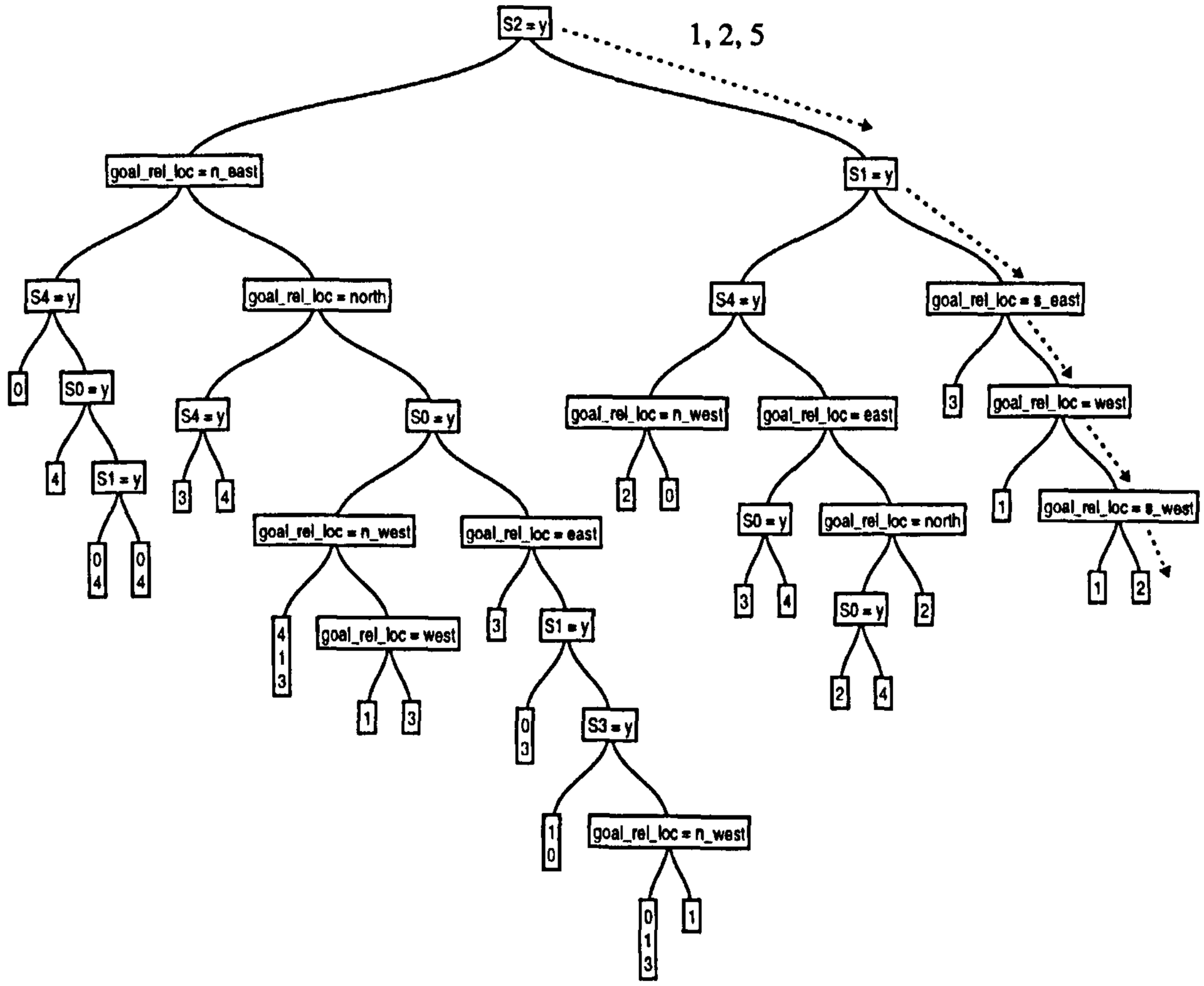


Figure 4.7 Layer-two of the rule layers to represent w_2. Directional lines demonstrate an example of traversing the tree to fire a rule given a certain robot state.

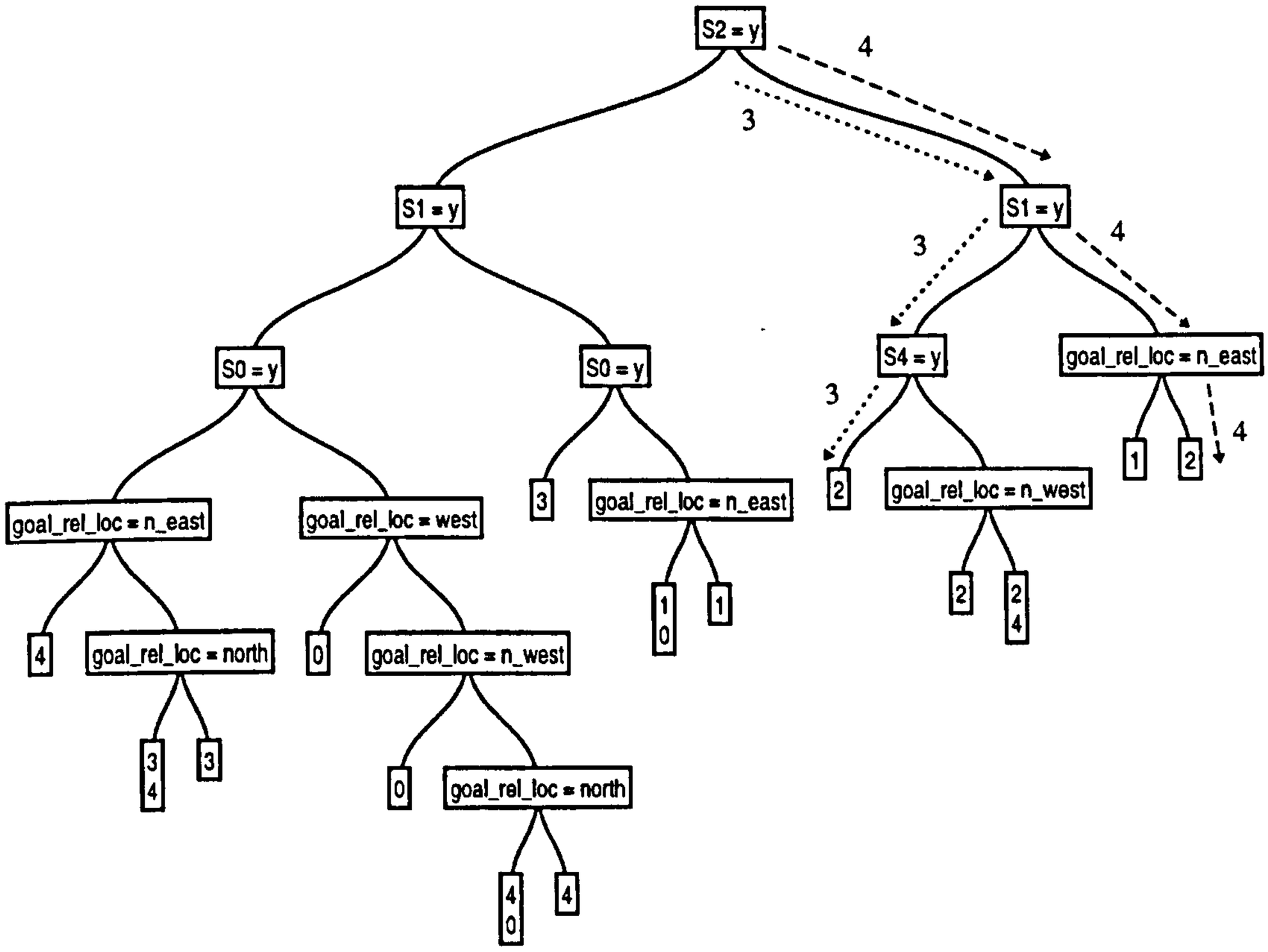


Figure 4.8 A typical much generalised layer-three of the rule hierarchy to represent w3. An example of how a rule is fired is shown.

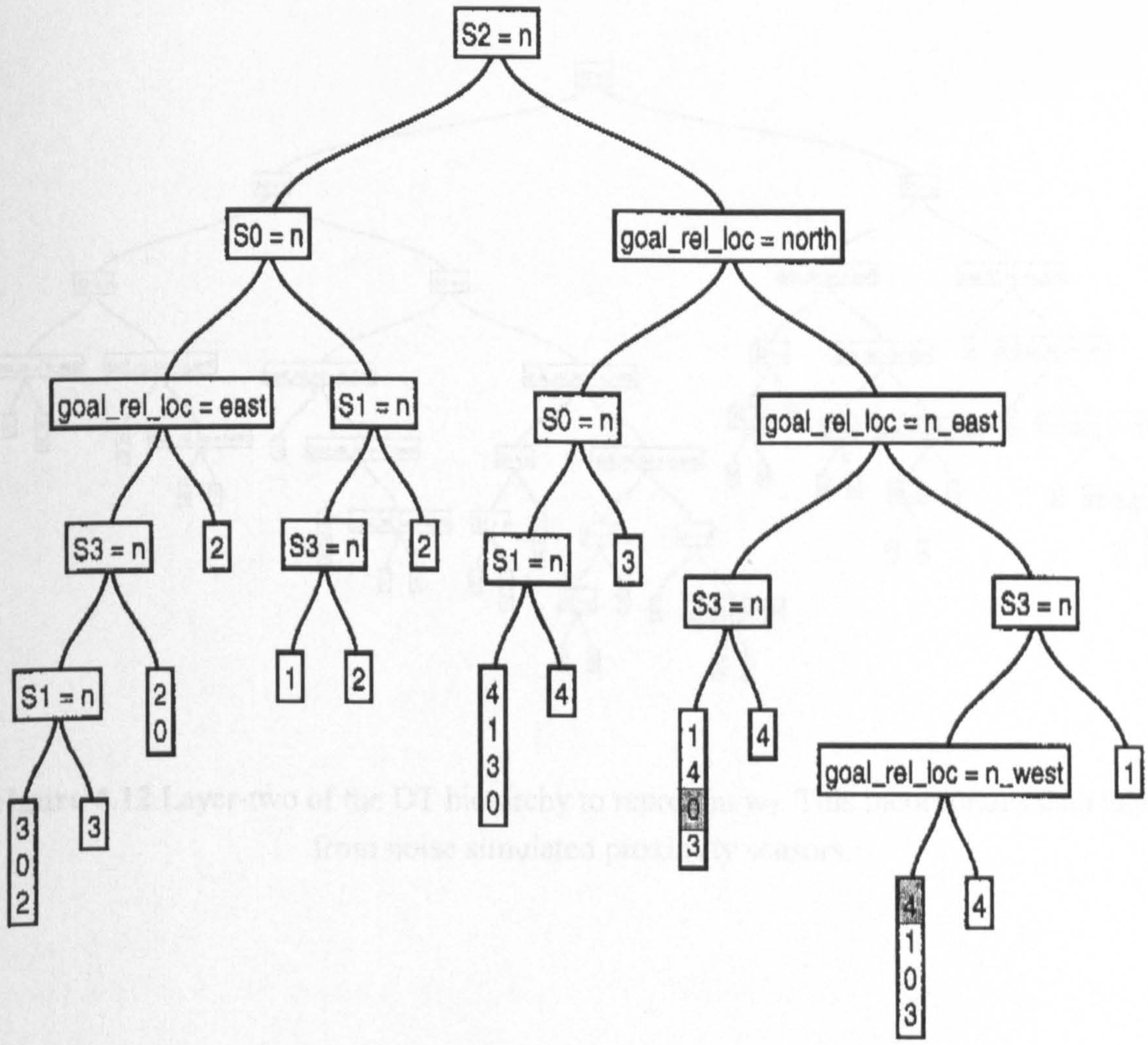


Figure 4.11 Layer-one of the hierarchy representing w_1. This DT has been grown on noisy sensory data, in contrast to Figure 4.6.

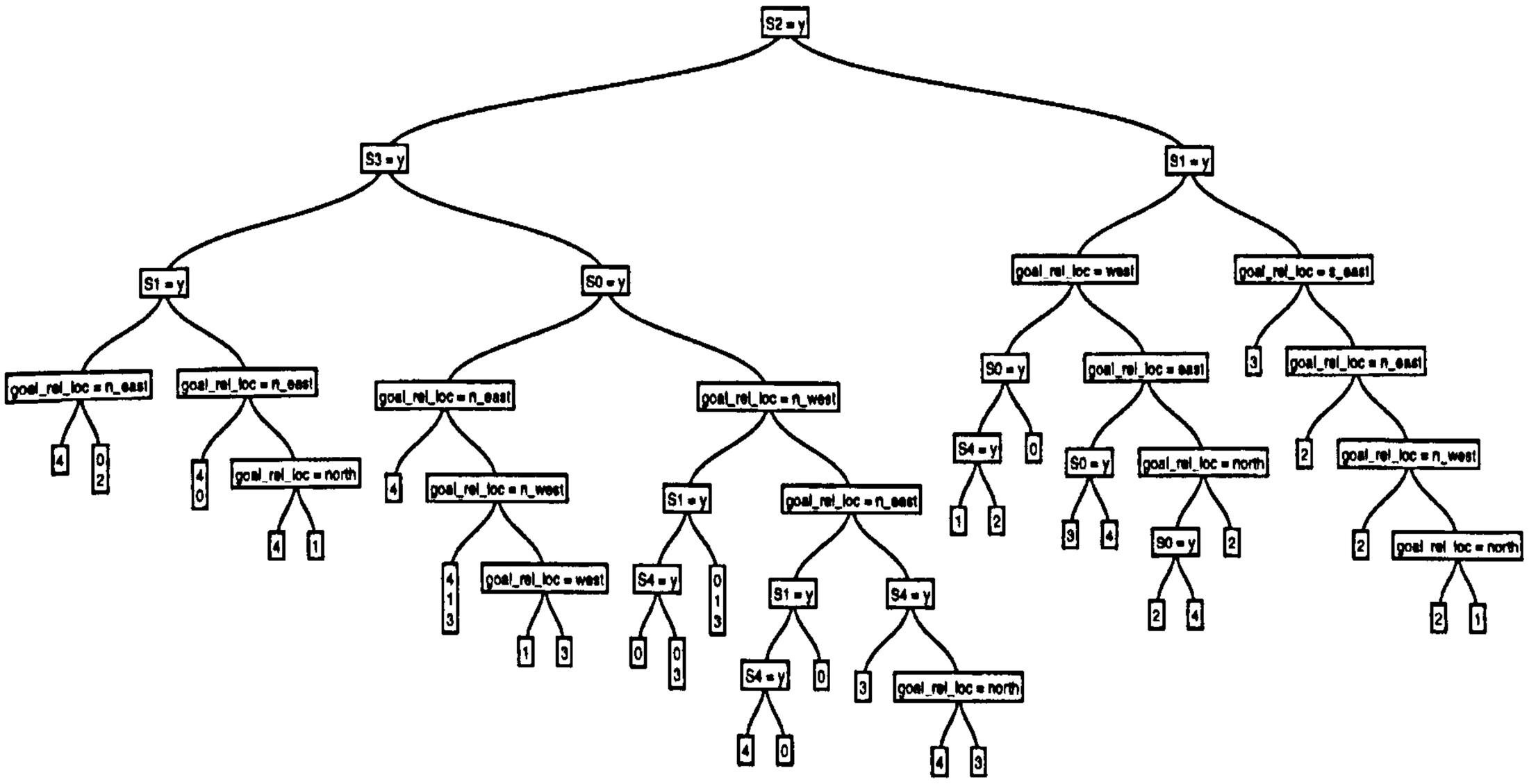


Figure 4.12 Layer-two of the DT hierarchy to represent w_2. This incorporates data taken from noise simulated proximity sensors.

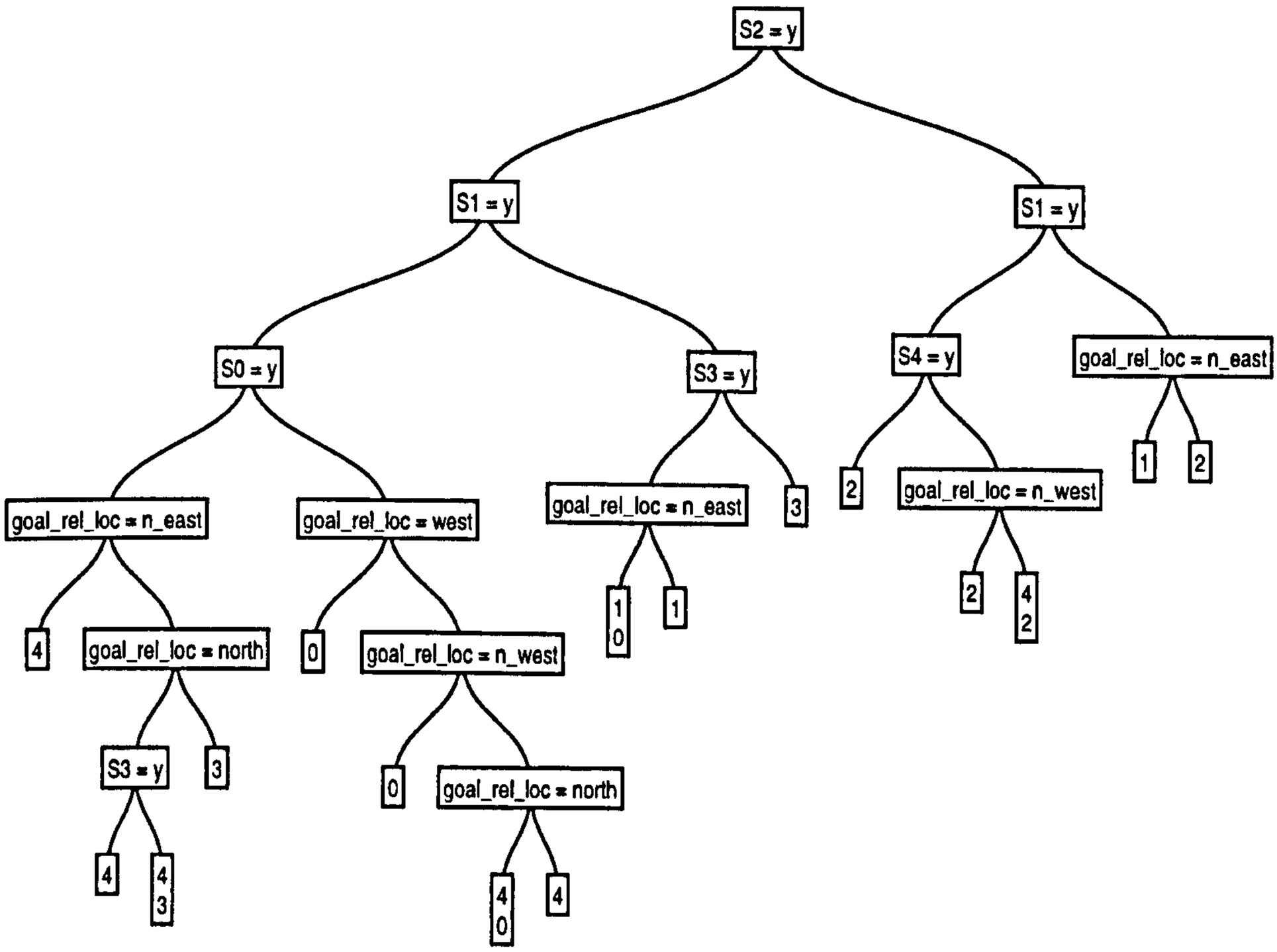


Figure 4.13 A DT network for classification of perception related to w_3. This is grown on noisy sensory data.

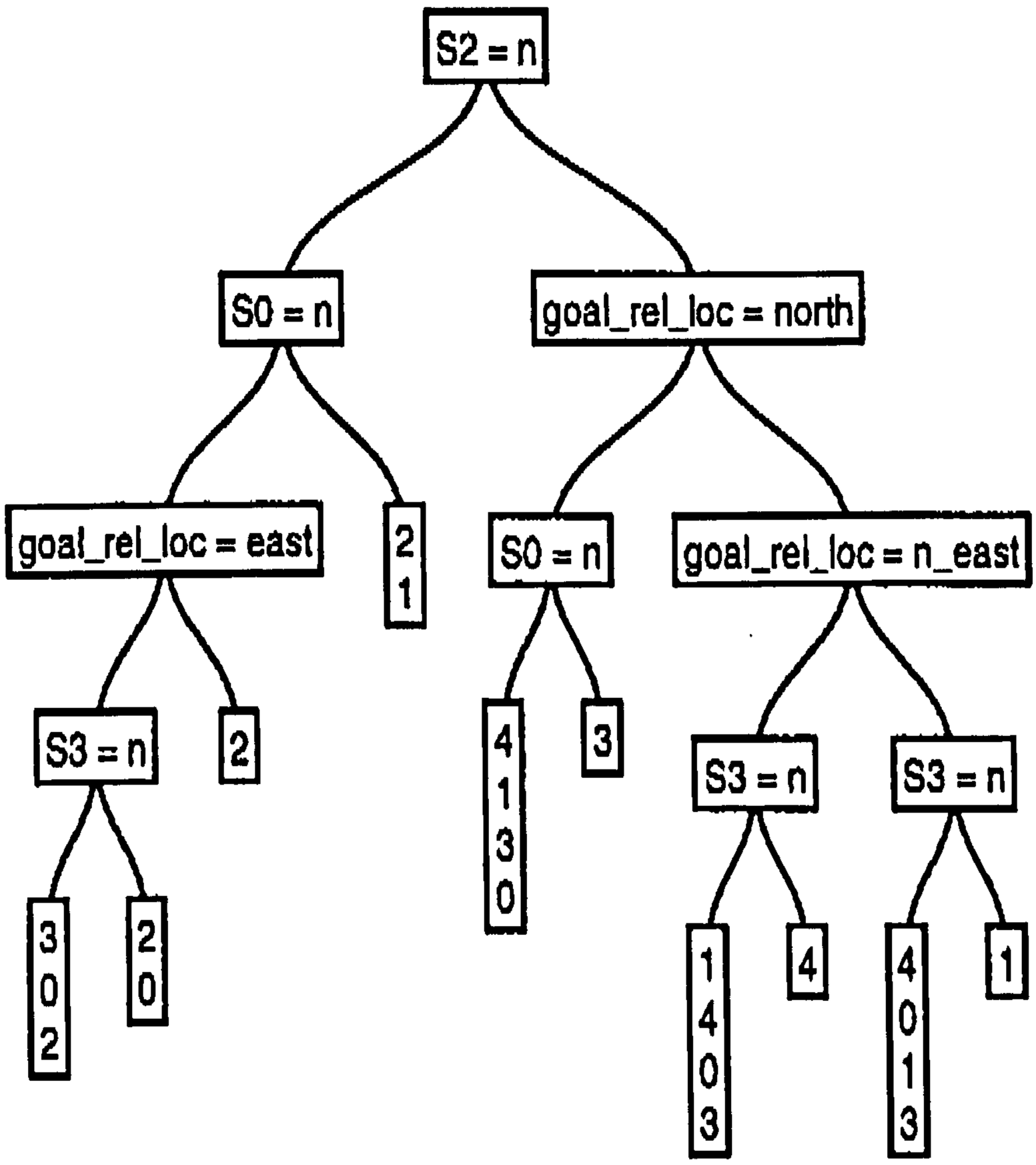


Figure 4.14 The post-pruned version of Figure 4.11. This DT represents w_1 incorporating noisy data.

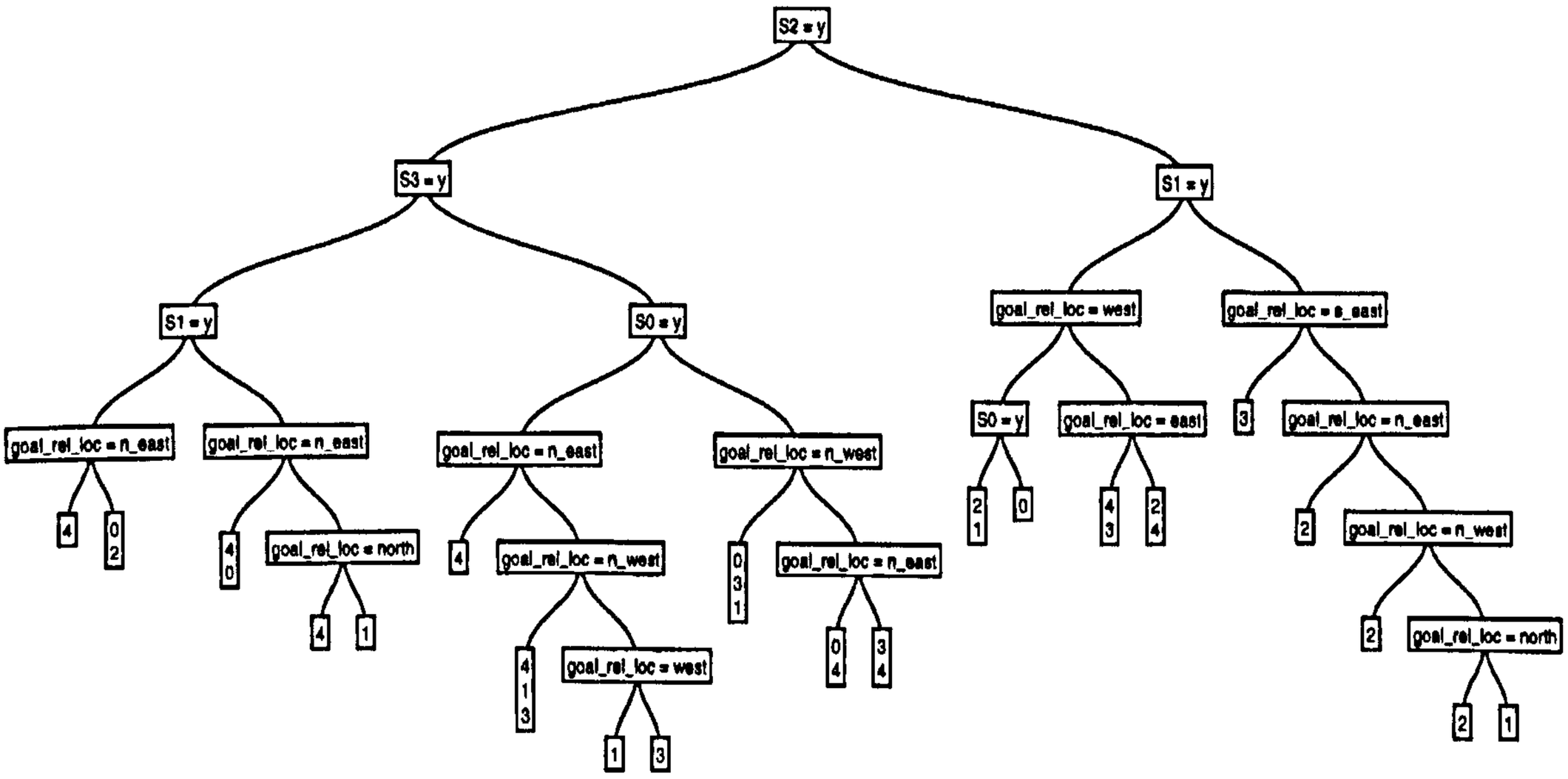


Figure 4.15 The DT representing layer-two of the hierarchy (shown in Figure 4.12) after post-pruning and replacing sub-trees with decision nodes.

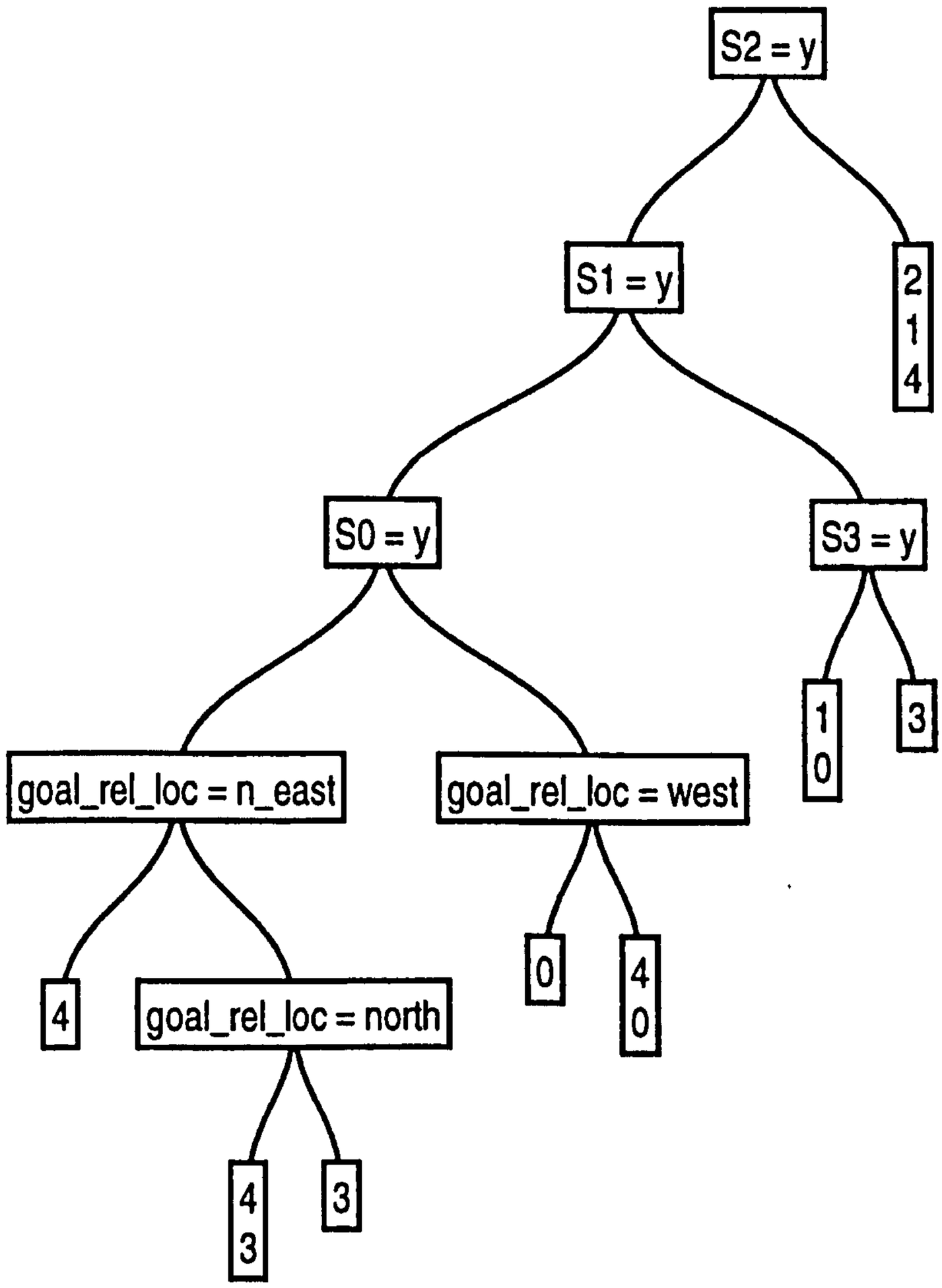


Figure 4.16 The post-pruned version of the DT shown in Figure 4.13. This represents layer-three of the hierarchy.

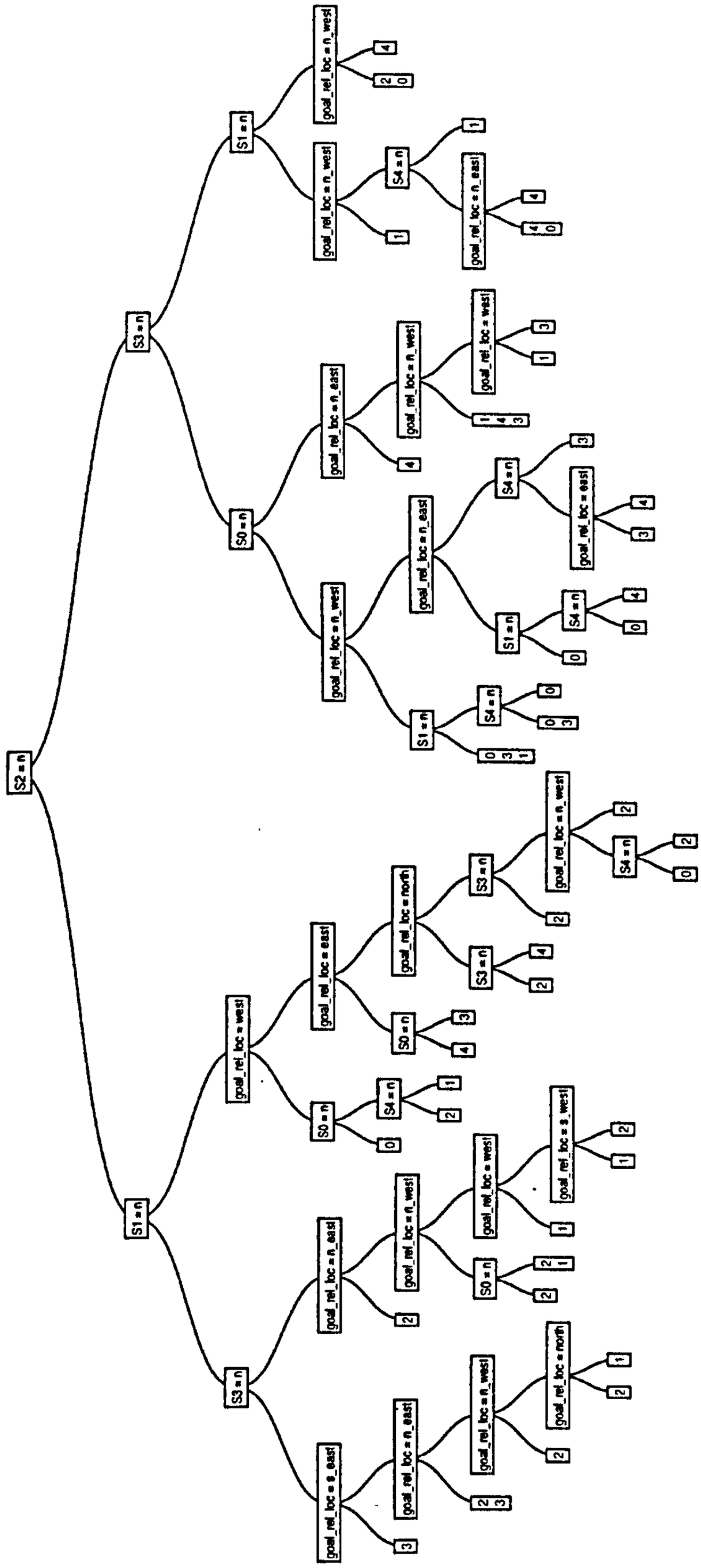


Figure 4.17 The augmented version of DT shown in Figure 4.12. This is activated to classify the robot perceptions representing w_2.

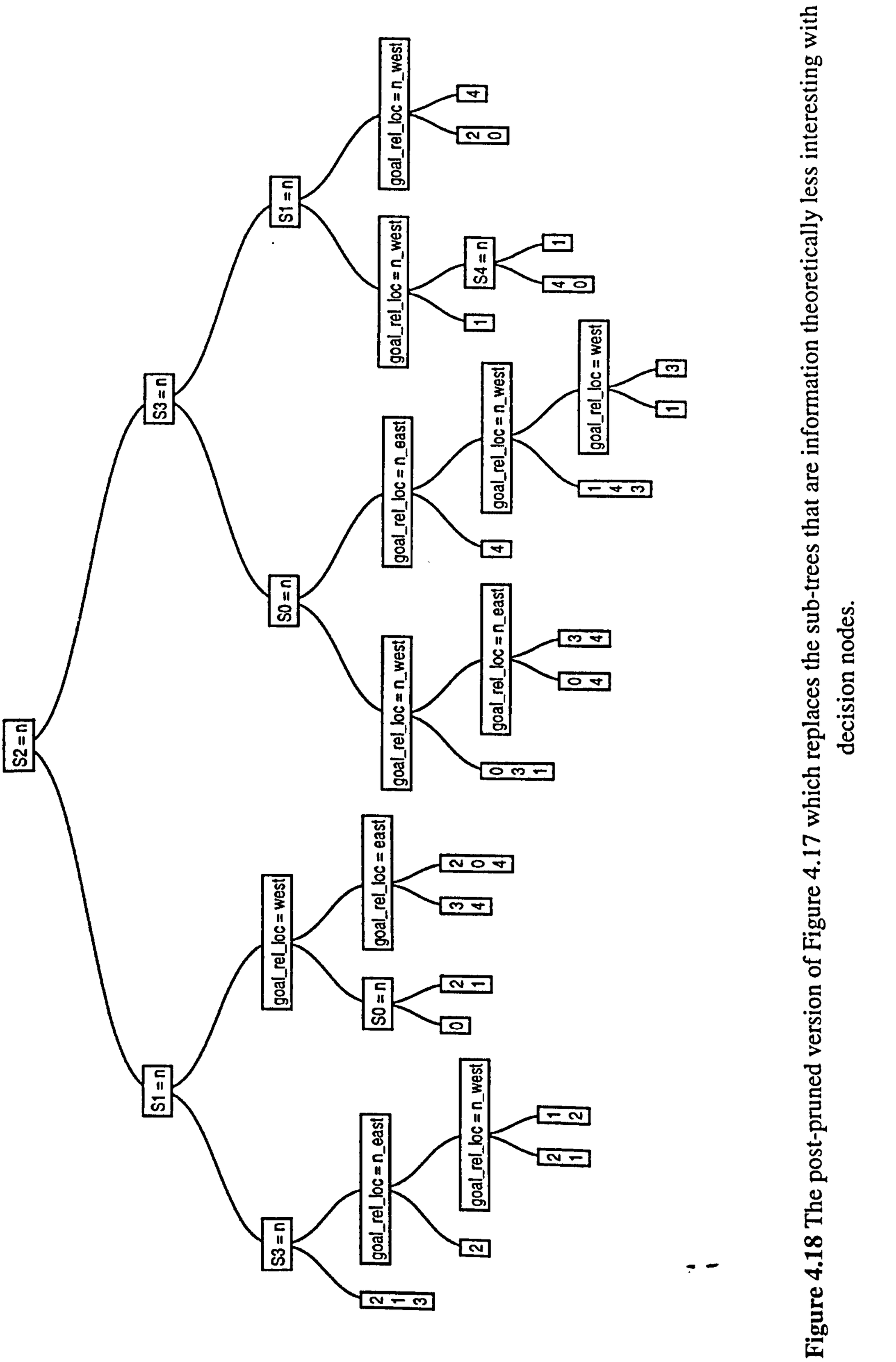
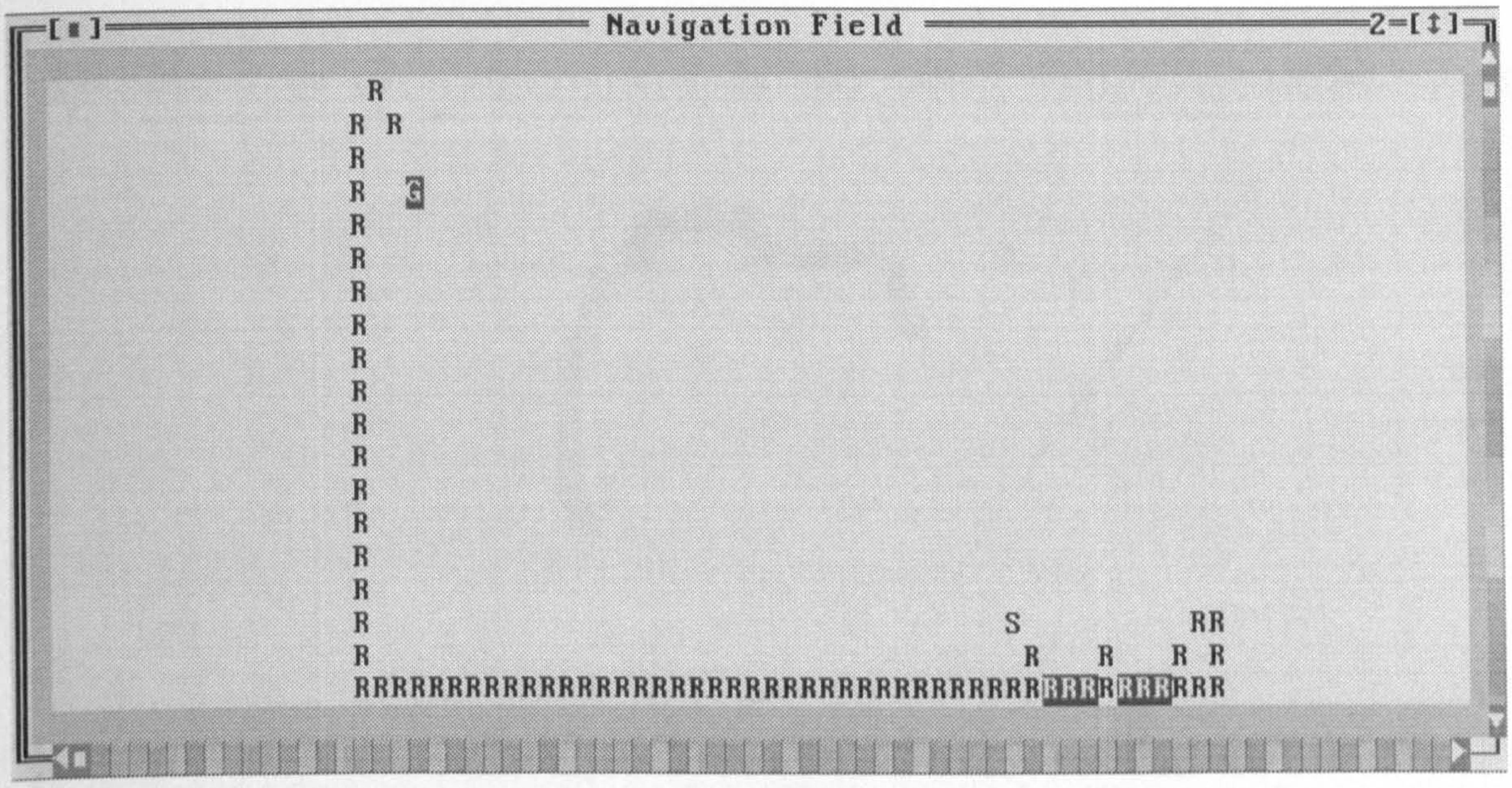
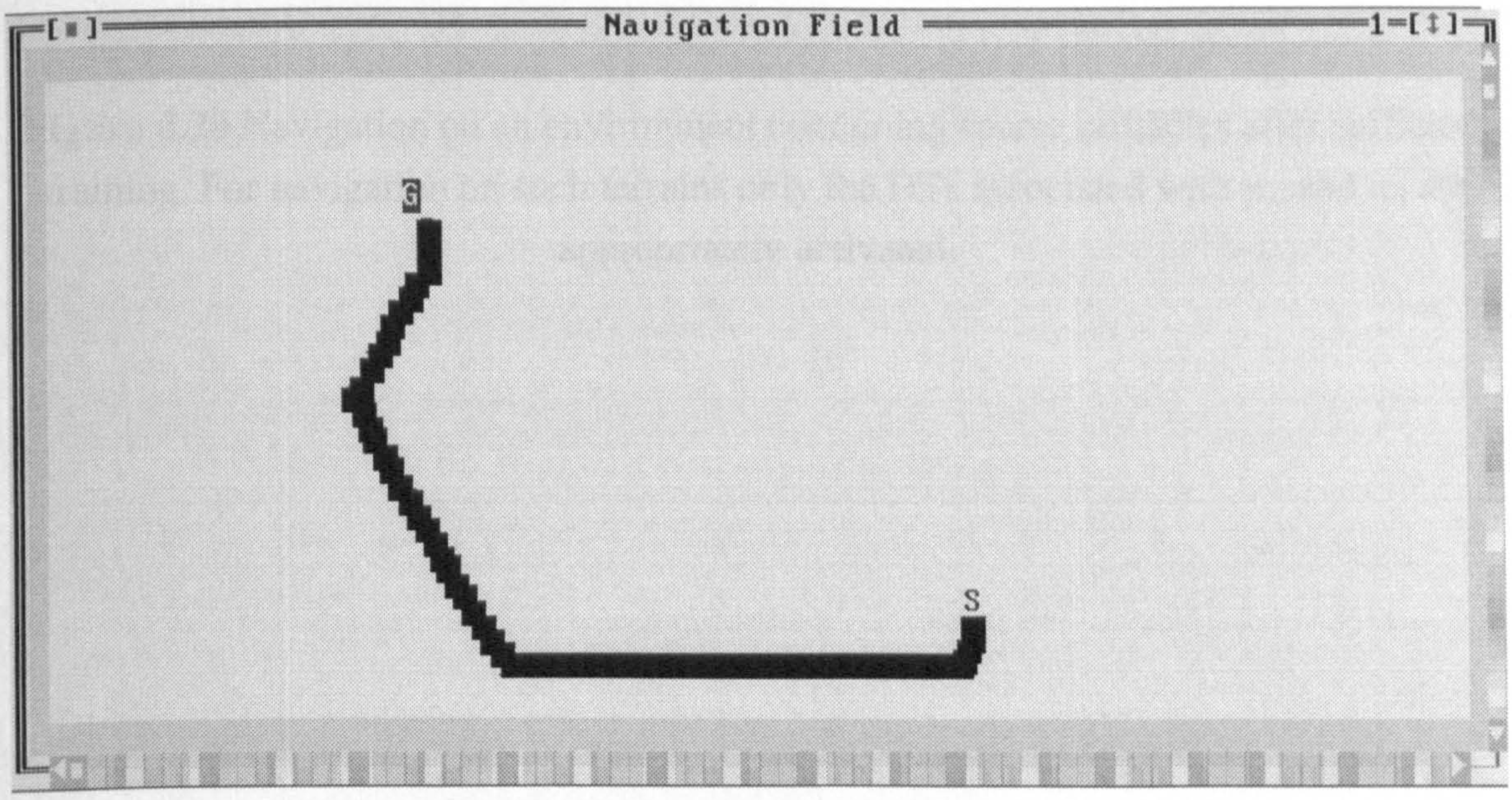


Figure 4.18 The post-pruned version of Figure 4.17 which replaces the sub-trees that are information theoretically less interesting with decision nodes.



(a)



(b)

Figure 4.19 Training the robot on the first layer of the hierarchy (w_0) with no obstacles to establish target-seeking behaviour, (a) where locations visited by the robot more than twice are highlighted. Figure (b) demonstrates a navigation with the same robot-goal setting after the robot has sufficiently been trained. This environment can be navigated using the DT of Figure 4.5.

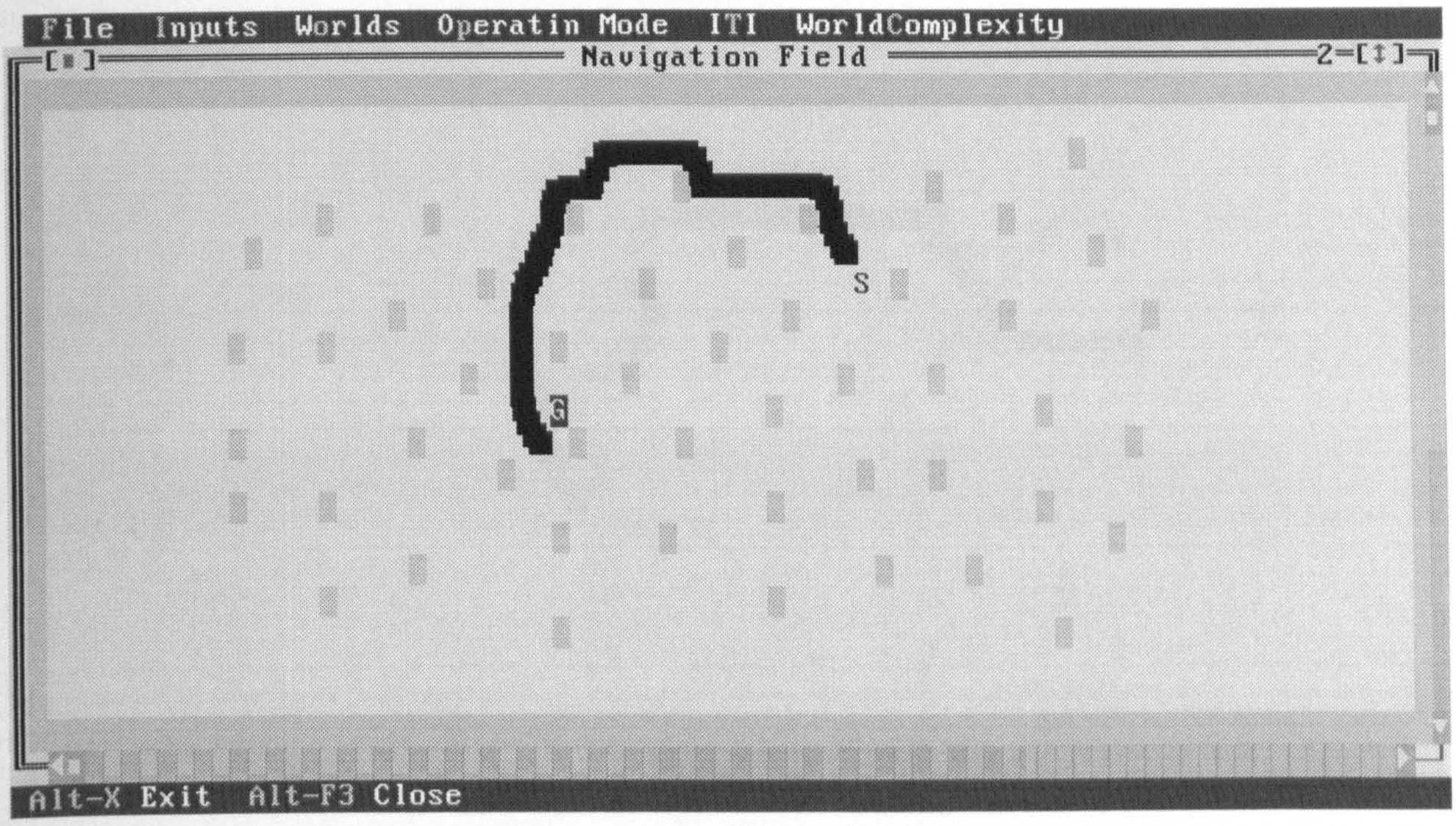


Figure 4.20 Navigation on an environment containing sparse obstacles after sufficient training. For navigation on such terrains only the DTs associated with w_0 and w_1 are appropriately activated.

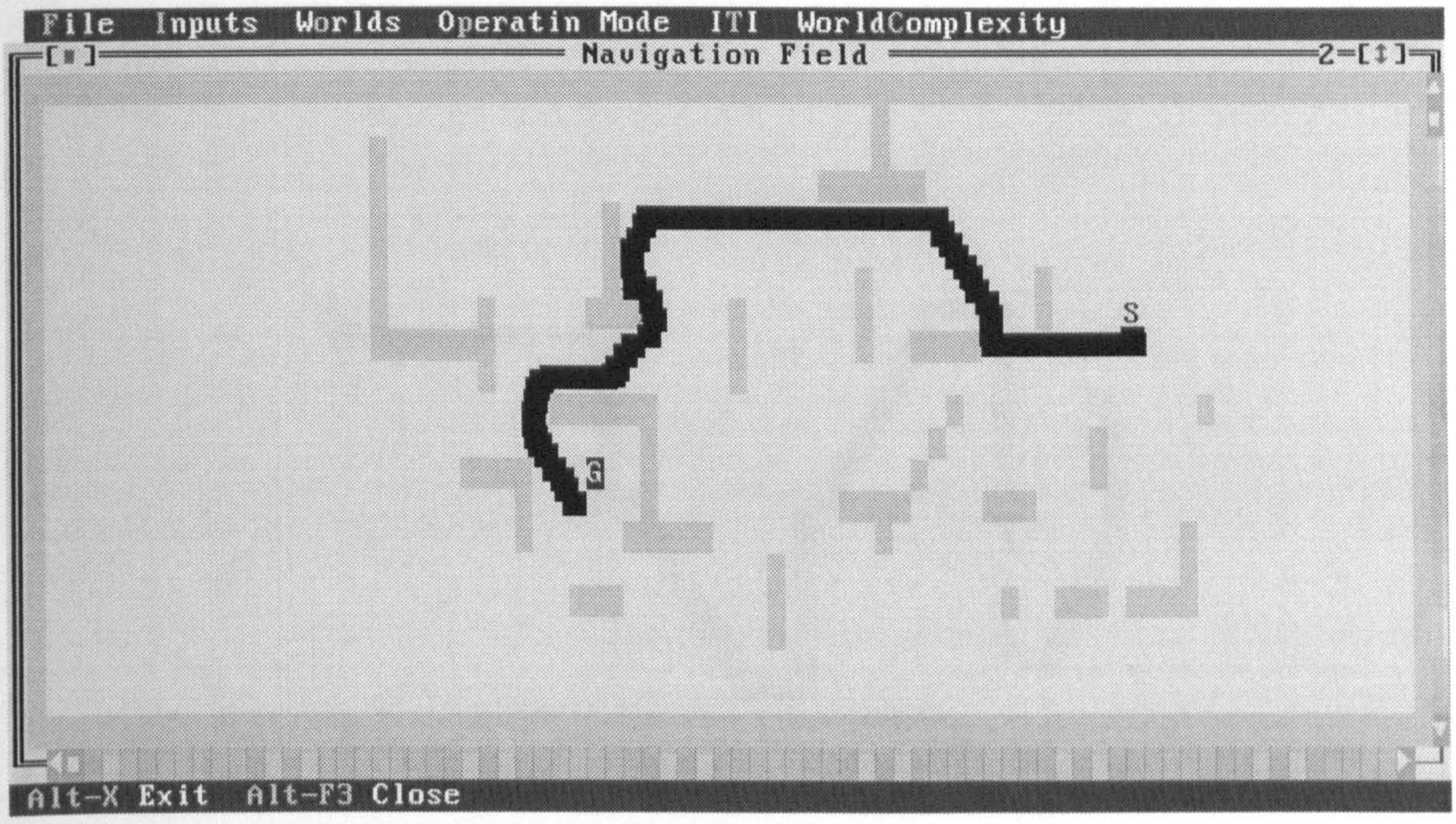


Figure 4.21 Target-seeking while avoiding obstacles in a complex environment.

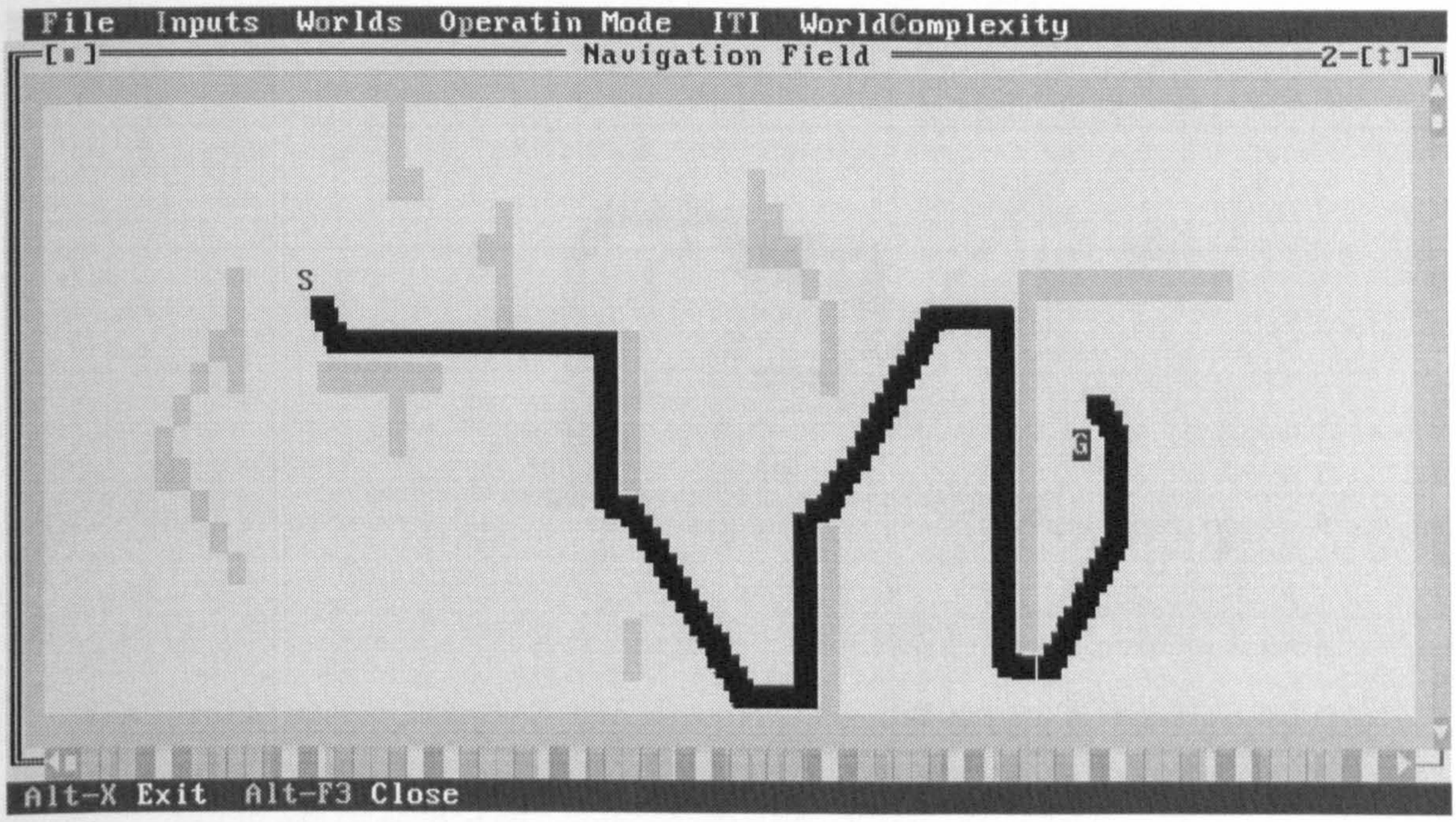


Figure 4.22 Navigation of a homing task showing reactivity and wall-following behaviour. In such environments the robot has access to the entire DTs in the hierarchy.

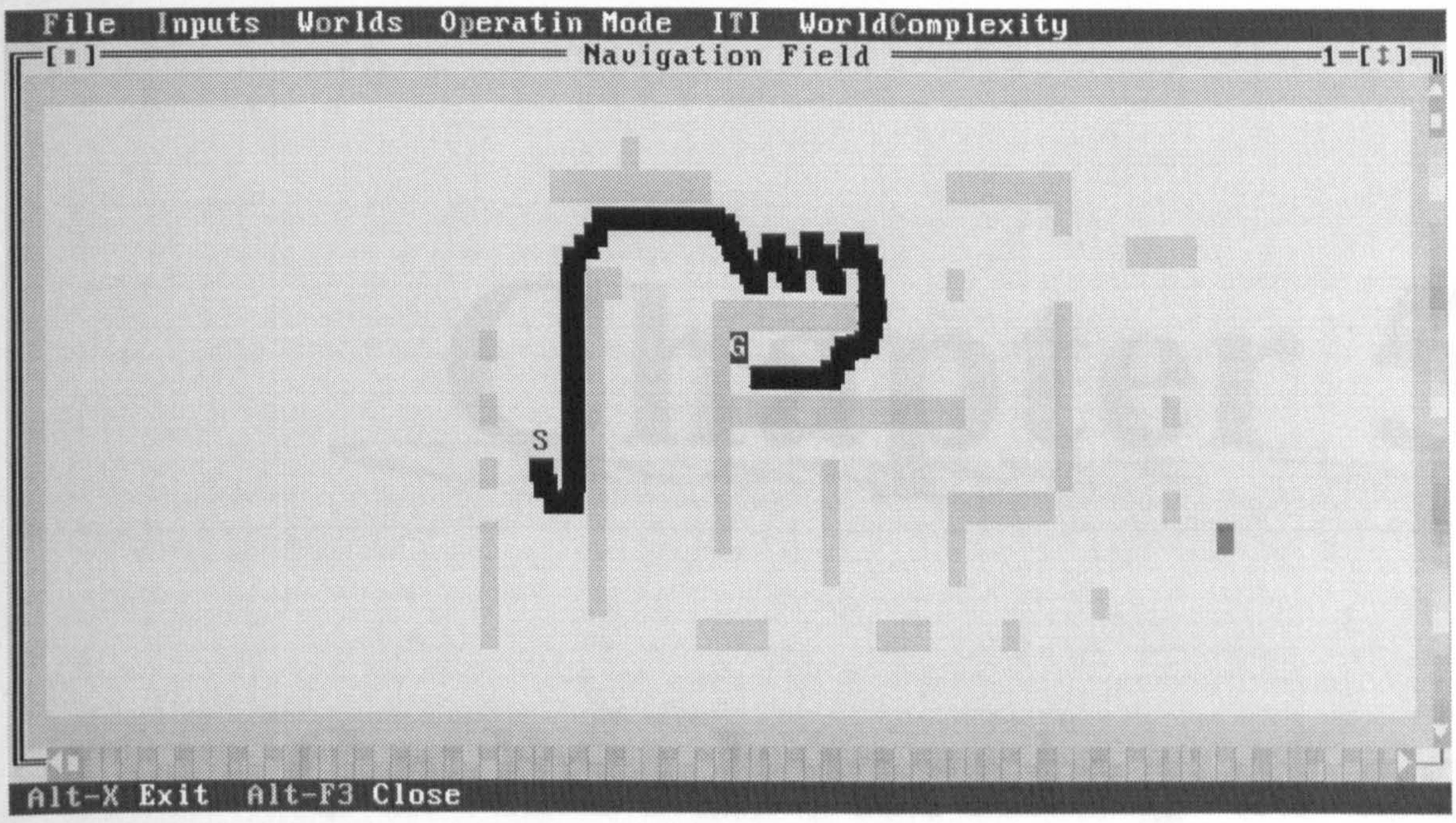


Figure 4.23 The zigzag trajectory is an example of behaviour which can result from transition between rule layers at either ends of the hierarchy due to behaviour dominance.

Chapter 5

On-line Learning of an Adaptive DT Array Applied to Robot Control:

Realistic Environments

To learn is to change, and to change is to learn.

Bart Kosko
Fuzzy Thinking

The previous chapter introduced an approach to behaviour learning in which the knowledge space was decomposed into a layered hierarchy of control rules each encoded in a distinct DT. The strengths of the proposed approach were shown to be the efficient management of the overall knowledge complexity, ease of implementation and the

generation of expressive control rules due to the intelligibility of induced knowledge which is characteristic to DTs. The experiments conducted were largely of qualitative significance in their demonstration of the feasibility of the approach, and were performed by making a number of simplifying assumptions about the robot and its environment in order to keep the dimensionality of the problem domain manageable and under control. This chapter takes further the approach introduced in the previous chapter by investigating the performance of the control concept in environments with realistic assumptions, as well as considering the characteristics of a physical robot with finite dimensions.

5.1 Introduction

This chapter implements the methodology of the hierarchical learning design introduced in the previous chapter, augments the method with incremental and on-line learning and considers a continuous perception-action space. This allows the quantitative assessment of the approach and its application to real world scenarios. Unlike the preliminary experiments in which the robot was considered as a point object (one of infinitesimally small physical dimensions), the perception-action was discretised and learning was performed off-line; this chapter is concerned with the application of the approach considering the following aspects:

- the robot has finite physical dimensions;
- the environment is continuous;
- the action space is continuous;
- the learning mode is incremental and on-line.

In contrast to the supposition of a discrete environment, the continuous perception-action space suggests an increase in the dimensionality of the DTs, and consequently the generation of a significantly larger rule space to search. This is because the number of possible states increases, which is directly related to the diversity of training examples which are supplied to the learning algorithm. Investigations into: how large the DTs will grow, whether they are able to suitably generalise new concepts, and whether DT truncation affects the generalisation capabilities of the DTs, will be carried out.

This chapter defines the relevant terminology used and introduces the features of the simulated mobile platform used in the experiments. The control architecture of the system is outlined and the method of development of the incremental tree induction is described. The concept of incremental evolution of DTs is illustrated in an example, illustrating the continuous adaptation of DTs as new knowledge is provided. The results are taken from a number of typical navigatory scenarios and include investigations of the efficiency and the limits of applicability of the approach. Possible improvements of the current technique are also discussed.

5.2 Terminology

As in the previous chapter, *world* is defined to be the instantaneous robot perception. The input perception P is defined to be that provided by the set of six circumference sensors as $P = \{S_0, S_1, S_2, S_3, S_4, S_5\}$ in which the range of each sensor value is defined by $S_i = \{0,1,2,3,\ldots,1023\}$ (to be discussed in section 5.3). However, the numerical outcome of each sensor is mapped on the symbolic set $S_{iS} = \{Y, N\}$, meaning that an obstacle is detected (Y), or no obstacle is detected (N). The experimental results demonstrated that $S_i = 750$ is a suitable threshold for classifying a numerical sensor value into either region of a decision plane (Figure 4.9 of chapter 4) in the same manner as discussed in section 4.8.1. Noise and uncertainty associated with sensor values and partitioning of the decision space is tackled in the same way as described in section 4.8.

Any perceived world is an element of $W = \{w_0, w_1, w_2, w_3, w_4\}$, where w_0 is the simplest and w_4 the most complex world. w_4 is not represented as an independent rule layer in the hierarchy as the action space in w_4 is limited to a single class regardless of the goal location, making the generation of a corresponding tree unnecessary. Training vectors need to have the format: $V = \{f_0, f_1, \ldots, f_i, \ldots, f_n, c_j\}$, where f_i is a feature with $f_i \in F$ and $F = \{f0, f1, \ldots, fi, \ldots, fn\}$ is the space of features (input variables) each being defined on a unique space of feature values such as $f_i = \{f_{0i}, f_{1i}, f_{2i}, \ldots, f_{mi}\}$. c_j is a class (control action) with $cj \in C$ from the set of output reflexes $C = \{c1, c2, \ldots, cj, \ldots, cn\}$.

5.3 Description of the robot

To verify the proposed control algorithm, the Khepera robot simulator is used [1]. Khepera is a commercially-available miniature mobile robot, shown in Figure 5.1, and Figure 5.2 shows the simulated counterpart. Khepera in its basic configuration has a diameter of 55mm, a height of 30mm and weighs 70g. The onboard computation contains a Motorola 68331 microprocessor with 256 Kbyte RAM and 256 Kbyte ROM. Khepera has access to an array of eight light sensors as well as eight infra-red proximity sensors for range measurement. The current work uses only the six frontal infra-red distance sensors to scan the areas to the front and sides of the robot for object detection and each returns a value in the range 0 to 1023 depending on the nature and the range of the detected objects. In general, the greater the magnitude of a sensor return value, the closer is the obstacle. For example, 0 means that no obstacle is detected while 1023 means that an obstacle is very close to the robot (almost collision). The Khepera simulator has been designed to incorporate realistic assumptions and to minimise the presence of unrealistic suppositions, facilitating the transfer of the simulation results without major change directly to the real Khepera robot [2,3]. For more information about the Khepera robot, the reader is referred to [1].

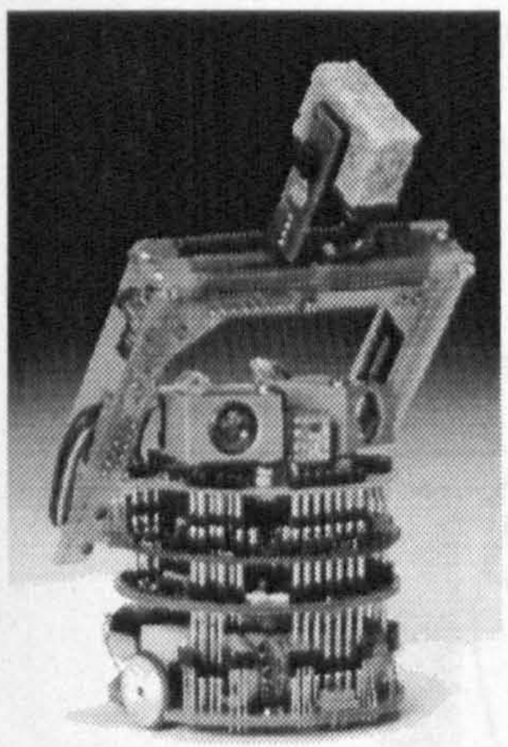


Figure 5.1 Actual Khepera robot equipped with gripper, vision and extra processing modules (Photo by Alain Herzog, Courtesy of EPFL Laboratory, Lausanne).

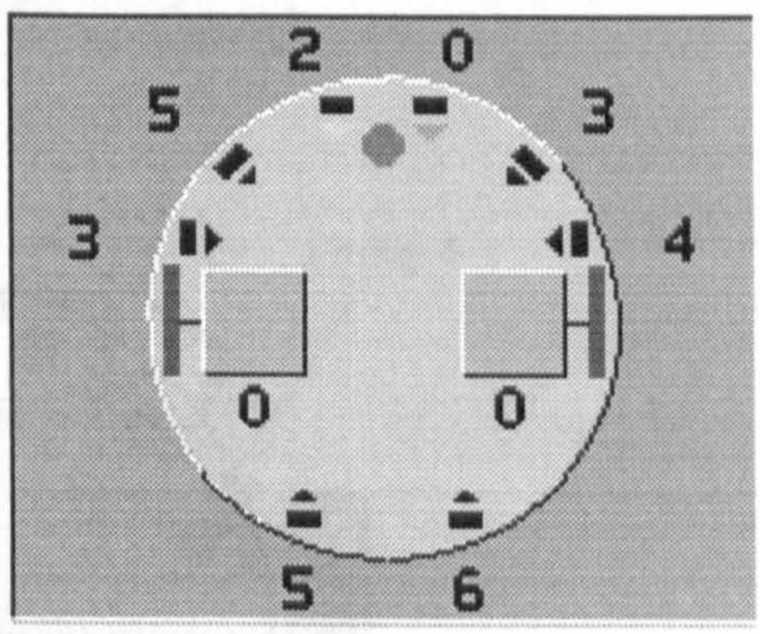


Figure 5.2 Simulated Khepera robot seen from above.

5.3.1 Robot positioning

The simulated robot is steered differentially by controlling the wheels individually. Robot positioning is performed by incremental encoders placed on each motor axis for step counting (dead-reckoning). This mechanism is used in the simulated counterpart for positioning the robot. The position vector of Khepera which is continuously changing consists of an x-value, a y-value and the direction-indicating angle α (the robot absolute direction). The current position (x_n, y_n, α_n) is calculated from the previous one (x_{n-1}, y_{n-1}, α_{n1}) using the number of wheel rotations n_L and n_R given by the incremental encoders placed on each motor axis.

5.3.2 Configuration of proximity sensors

In order to associate a certain perception with a world (defined in section 5.2) in the hierarchy and to use the same nomenclature as in chapter 4, frontal sensors are divided into five groups each containing two *adjacent* sensors as shown in Figure 5.3. Any two adjacent sensors comprise a sensor group to indicate the complexity of the world detected. In this configuration, each sensor group can be considered as a single sensor in order to be consistent with the sensor configuration introduced in chapter 4.

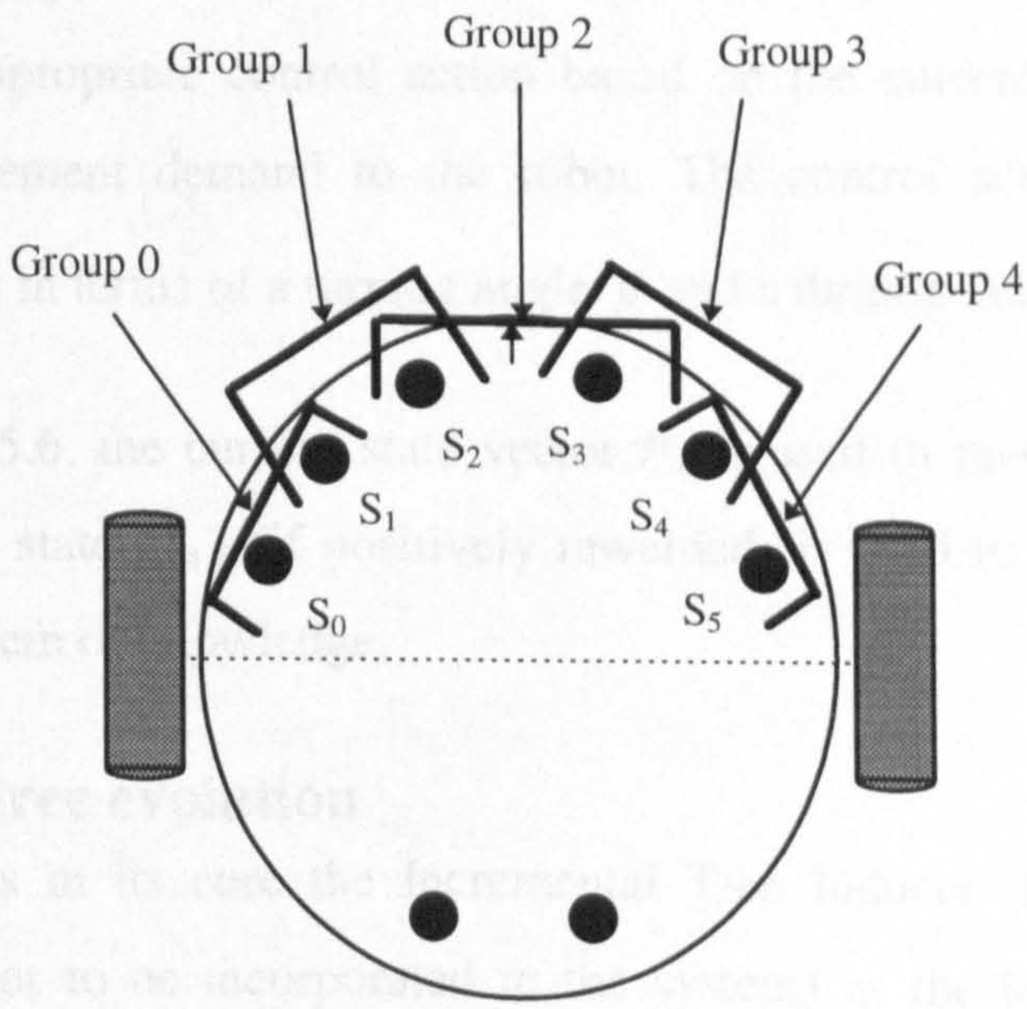


Figure 5.3 Frontal sensors (S_0, S_1, S_2, S_3, S_4, S_5) are divided into five sensor groups: Group 0 (Left), Group 1 (Front_Left), Group 2 (Front), Group 3 (Front_Right), and Group 4 (Right). In each case, two adjacent sensors form one sensor group.

The number of *sensor groups* indicating that obstacles are present determines the world complexity and their nomenclature in the hierarchy. This means that, for example, if only one sensor group detects an obstacle at a certain distance from the robot that entire perception is classified as w_1 to designate world one, and w_0 specifies that the instantaneous environment (world) has no obstacles. In a sensor group, the state of either sensor determines the state of the group whether the sensor group detects the presence of obstacles or no obstacles are apparent, and this is demonstrated in Figure 5.4.

5.4 Control system architecture

Figure 5.5 shows the schematic view of the system architecture. This is composed of three main modules:

1. *Environment.* The interface providing data from the physical world.
2. *Local Perception Space*. Each perception is mapped on a certain state that falls into a unique world category. The output is a state vector containing a finite number of state variables, and this is used to infer or re-infer tree networks.
3. *Controller*: This accommodates two sub-modules, namely high-level decision making (whose architecture is shown in Figure 5.6), and low-level control. The former performs predictions based on the current state vector, whereas the latter generates an appropriate control action based on the current decision in order to provide a movement demand to the robot. The control action $a(\beta, v)$ is a state variable defined in terms of a turning angle β and a turning velocity v.

As shown in Figure 5.6, the current state vector P_n is used to predict the robot motion, whereas the previous state, P_{n-1} (if positively rewarded) is used to restructure the tree to incorporate the new item of knowledge.

5.5 Incremental tree evolution

The algorithm houses in its core the Incremental Tree Inducer (ITI-2.8) [4] (which is modified by the author to be incorporated in the system) as the learning module. ITI is driven in its incremental mode and on-line. This means that appropriate knowledge entities are given to ITI either for incorporation into an appropriate existing decision tree or to instantiate a new tree if none exists already.

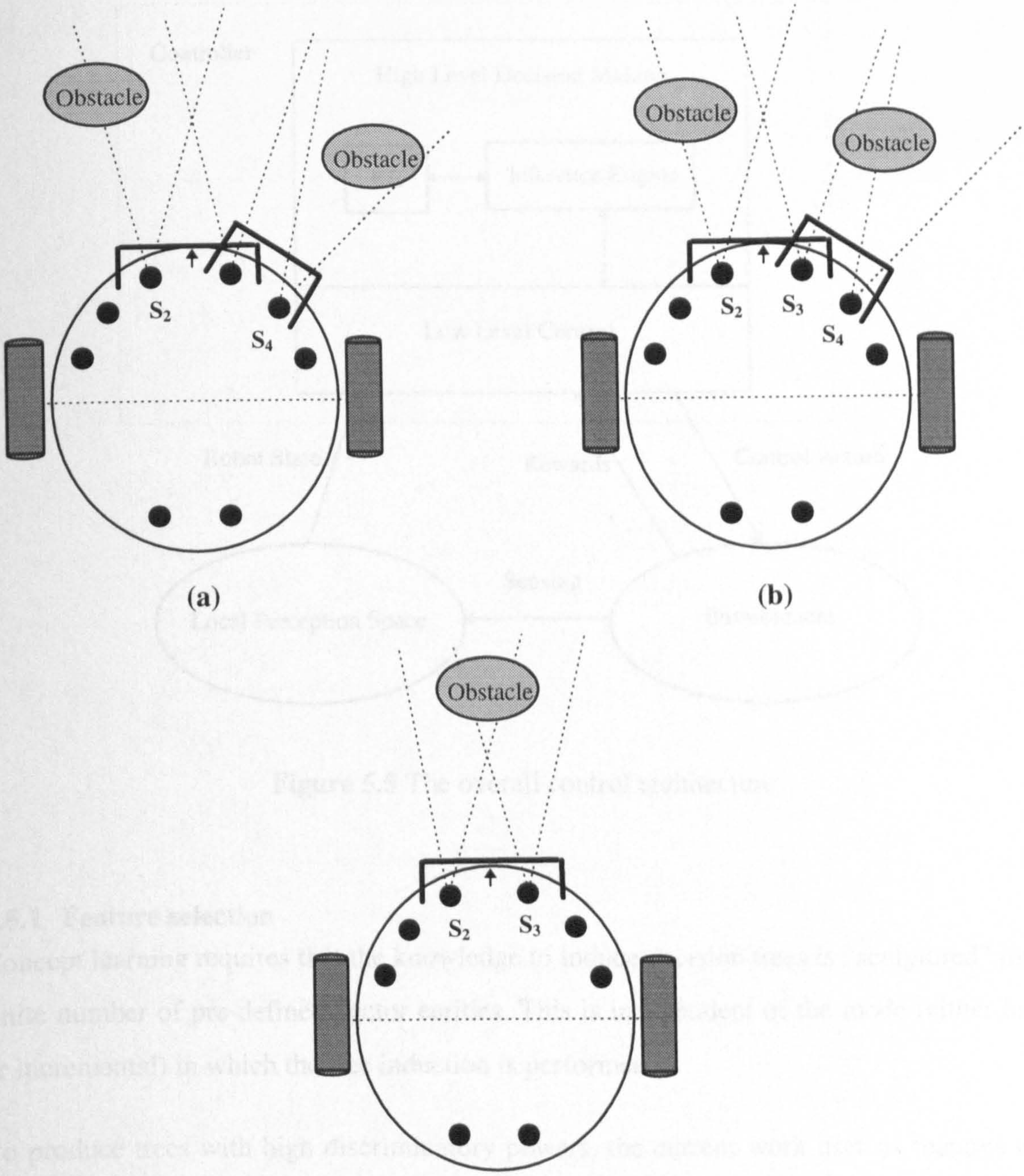


Figure 5.4 Different perceptual situations: (a) Two non-adjacent sensors S_2 and S_4 covered by *two groups* indicating w_2. (b) Three consecutive sensors S_2 and S_3 and S_4 detecting obstacles and represented by *two groups* indicating w_2. (c) Two adjacent sensors S_2, S_3 detecting obstacles and covered by *one* group specifying w_1.

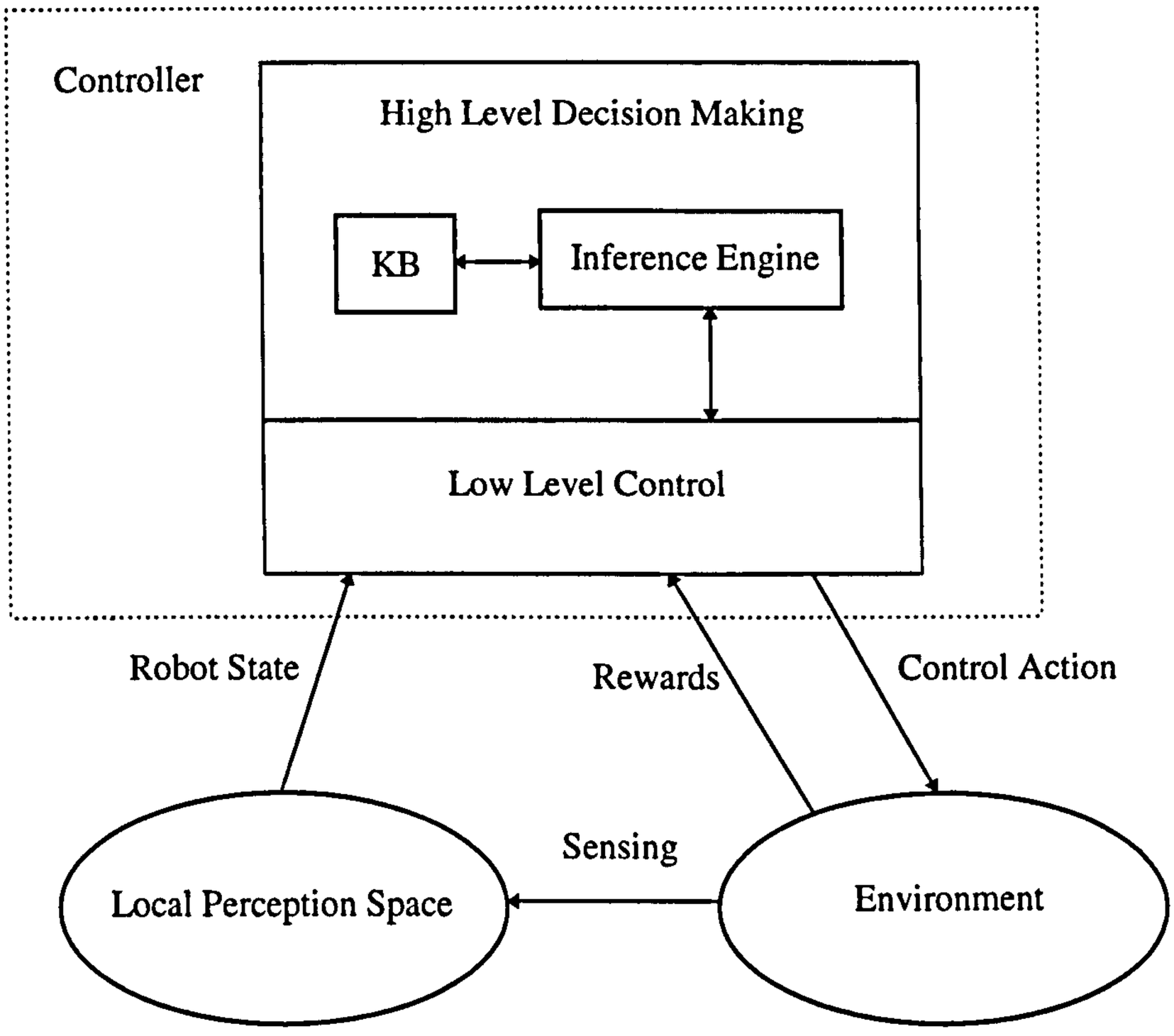


Figure 5.5 The overall control architecture

5.5.1 Feature selection

Concept learning requires that the knowledge to induce decision trees is "sculptured" into a finite number of pre-defined vector entities. This is independent of the mode (either batch or incremental) in which the tree induction is performed.

To produce trees with high discriminatory powers, the current work uses as features both the proximity sensor values and the relative location of the target to the robot. To restrict the dimensionality of the world-dependent trees, the format of the training examples is configured to be a function of the world complexity. This means that in world w_0 with no obstacles, each training example is reduced to the relative location of the goal (*GoalRelLoc*) and a single class, allowing the production of a linear decision tree which is highly goal oriented. In higher order worlds, proximity sensor values are added to the existing features resulting in the modified format of feature vector, $S_0, S_1, S_2, S_3, S_4, S_5, GoalRelLoc, c_i$. In the above, $GoalRelLoc \equiv \{N, NE, E, SE, SW, W, NW\}$

and $C \equiv \{F, FR, R, BR, BL, L, FL\}$ where F is "front", FR is "front right", BL is "back left", and so on.

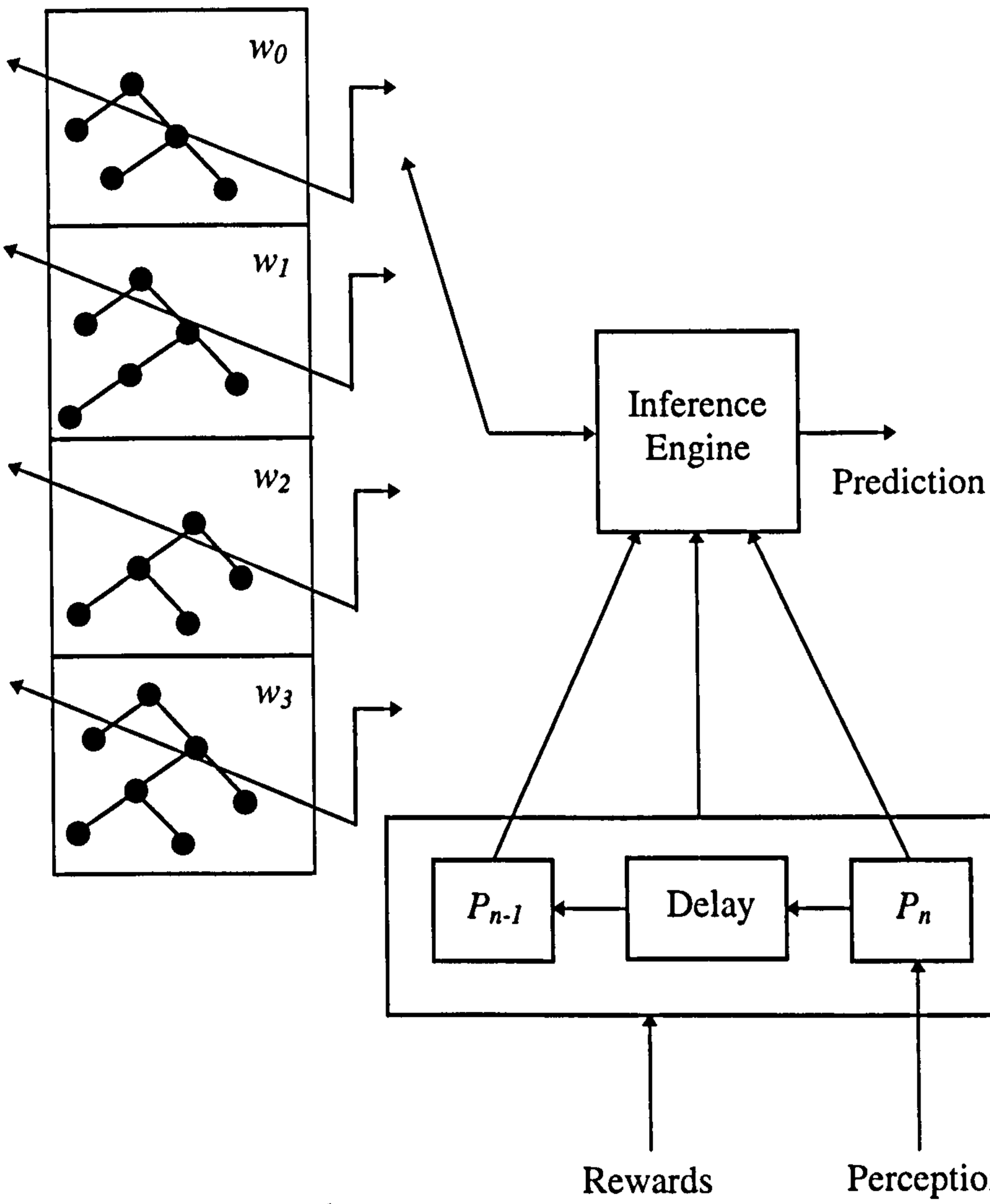


Figure 5.6 High level decision module with parallel co-existing decision trees as intelligent local planners.

5.5.2 Automatic knowledge acquisition and class prediction mechanism

One aspect of the approach that facilitates the population of the parallel co-existing trees in incremental mode is the automatic generation of training examples. This is carried out in order to collect the classification knowledge without intervention by human experts. Since the robot motion directions are generated randomly at the training stage, the usefulness $U = f(C)$ is defined as a measure of goodness of the performed actions. Following the

transition from the previous state $n-1$ to the current state n, the cost C associated with each motion is calculated according to an heuristic function (described in section 4.5 of chapter 4). Only those states delivering a usefulness of unity are positively reinforced by being remembered, the remainder are forgotten. Each individual remembered experience is supplied to ITI-2.8 as part of the training data for the appropriate tree.

5.5.3 Local independence and global coupling of DTs

Figure 5.6 illustrates that each tree network is an individual computational entity in itself and is able to make predictions when activated. In environments of greater complexity, with arbitrarily shaped walls and corners, a range of different trees will generally be sampled in order to generate the appropriate control actions, a suitable sequence of which forms the global path.

By analysing the process of tree generation in a finite window of time, one can observe how each individual tree evolves from a single-class entity to a mature decision tree capable of predictions, hence the notion of "tree evolution". Table A.1 of Appendix A provides an illustration of tree evolution, showing that the effect of behaviour learning is to improve the robot navigation performance incrementally over a finite number of iterations.

5.6 Algorithms

The complete system incorporates the two algorithms described below.

1. *Behaviour learning*. This algorithm precedes navigation in the executional context as it is able to set the robot in exploratory mode in order to acquire knowledge. It also generates a signal to invoke ITI for tree induction. The pseudocode for this algorithm is shown below.

IncrementalLearningMode (Tree, World)

Step 1: **IF** Goal is not reached
Step 2: **THEN** Choose a direction at random
Step 3: Evaluate the previous motion
Step 4: **IF** Previous motion interesting (output of the cost function)
Step 5: **THEN** Set up an appropriate training vector
Step 6: (Re-) Infer previous tree to absorb new knowledge
Step 7: Knowledge = UseKnowledge (Tree, World)
Step 8: **IF** Knowledge
Step 9: **THEN IF** Goal
Step 10: **THEN** Go to step 14
Step 11: **ELSE** Go to step 3
Step 12: **ELSE** Go to step 2
Step 13: **ELSE** Go to step 2
Step 14: **END.**

Note that in step 6 only *new* knowledge entities are evaluated for possible tree induction. The sub-routine UseKnowledge allows the tree to be searched concurrently with new knowledge being acquired, enabling the robot to have access to its previous experiences. This sub-routine is defined as follows.

UseKnowledge (Tree, World)

Step 1: **IF** Useful knowledge for the current world
Step 2: **THEN** Fetch knowledge
Step 3: Generate an appropriate control action using the tree
Step 4: **END.**

2. *Intelligent navigation*. This part implements the worlds w_0 to w_3 as finite state machines (FSMs) in which each world is represented as an independent state. The dynamic behaviour of the robot occurs as a result both of the actions of the decision trees and of

the transitions between trees, which are controlled by the FSMs. The procedure for the intelligent navigation process is shown below.

NavigationMode (Tree, World)

In state n world w_x is detected ($w_x \in$ **World**)
Step 1: **IF** Goal is not reached
Step 2: **THEN IF** tree x exists (**tree** $x \in$ **Tree**)
Step 3: **THEN IF** useful knowledge available for current world
Step 4: **THEN** Fetch knowledge
Step 5: Generate an appropriate control action
Step 6: NavigationMode (Tree, World)
Step 7: **ELSE**
Step 8: **WHILE** ($x > 0$)
Step 9: $x = x - 1$
Step 10: Search tree x for knowledge to train world w_{x+1}
Step 11: Go to step 3
Step 12: IncrementalLearningMode(Tree, World)
Step 13: **ELSE** Go to step 12
Step 14: **END.**

5.7 Illustration of a locally trained binary DT

In the vast majority of cases, a decision tree consists of a finite number of decision nodes and terminal nodes (classes) linked together to form a complete network. In extreme cases, the tree can lead to only a single class (see Table A.1 (a) of Appendix A). Decision nodes are generated in descending order from the root of the tree and each accommodates a particular feature with a unique outcome. The order of each test node in the hierarchy is arranged either information theoretically [5] or is based on other merits, for example Kalmogrov-Smirnoff distance in [6].

Unlike C4.5 [5], which is an ID3 descendant, the decision tree networks produced by ITI-2.8 are of a binary nature. This implies that, at any given test node, if the test result is

positive (yes) the tree branches to the left, otherwise the search is directed to the right. This process to find a match for a given pattern proceeds until the leaf is reached which provides a classification for the pattern.

Figures 5.7 to 5.10 show a number of local predictors (the DTs for the worlds) being used in combination to provide a plan for the global path. The behaviour dominance in the tree hierarchy changes in a complementary fashion, meaning that, if goal-orientedness is the dominant behaviour in a layer such as w_0, reactivity is of minor influence. Conversely, in a highly reactive layer such as w_3, target seeking behaviour is practically non-existent.

5.8 Action selection and conflict resolution

From a decision tree point of view, inconsistent training examples occur when more than one class would be able to classify the same input pattern [4]. In the current robotics application, for a given perceptual pattern more than one rule can be fired to generate a control action. To resolve the conflict, a single value is generated from a rectangular distribution to produce a percentage measure. ITI-2.8 also associates every rule with a *frequency tag*, and that rule is fired whose frequency tag matches the randomly generated frequency measure. This results in consistent action commands being sent to the actuation level to drive the robot forwards.

5.9 Dynamic rule inhibition

Another important aspect of the proposed learning algorithm is the monitoring and long term assessment of incrementally-learned rules. In the learning phase, when the goal direction is located between the lines of maximum sensitivity of two adjacent sensor groups, rules can be evaluated by the system as useful, although they would quantitatively be considered to deliver poor performance under normal navigation circumstances. This effect is also a source of multiple-class generation. To exclude these types of rule, each individual rule is assessed after it has been fired depending on whether the resulting performance was satisfactory and the performance is used to give an indication of its overall usefulness. If the overall usefulness drops below a pre-set level, the rule is flagged and inhibited. Even though inhibited rules physically exist in the network, they have no

actual contributions. These rules are shown shaded in Figures 5.7 to 5.10, where appropriate, to highlight the inhibiting mechanism.

5.10 Results and discussion

Each of the aforementioned modules, namely w_0 to w_3, is a unique computational entity with a dynamic life time. Some have a process-long life whereas other types of entity can be generated during the process, but may also be destroyed to avoid an unnecessary increase in computational cost. Table 5.1 shows that approximately 55 training epochs are needed to set up the DT hierarchy. It also indicates the average number of training examples needed to grow each individual DT to capture the representative vectors.

	Worlds				
	w_0	w_1	w_2	w_3	w_4
Average Number of Runs	13	12	11	9	10
Number of Training Examples	67	180	120	70	105
Total number of training runs to build up the hierarchy (Mean)					55

Table 5.1 The approximate number of training epochs and training examples needed to set up each individual DT of the hierarchy.

As discussed previously, global learning is initiated in w_0 and propagates up the hierarchy to the more complex worlds. This is carried out by adjusting the dimensions of the previous rule layer to adapt to new knowledge in the current layer. In all the examples, the robot is initially set at the starting point S and is expected to reach the light source G, also indicated by a star symbol. The navigation fields chosen consist of typical situations that can occur in a navigatory task, for example sharp corners and edges, right angle corners, and both straight and rounded walls with or without discontinuities. The different stages of Figure

5.11 demonstrate an example of the learning process in w_0: it illustrates how the learning is formed (Figure 5.11 (a)) and improved in two-leg (Figure 5.11 (b)) and in multi-leg trajectories (Figure 5.11 (c)). In w_0, the robot learns how to find its target (survival) in an environment with no disturbances. Figure 5.11 (c) demonstrates that after sufficient training the robot is able to reach the set target (using similar settings to those used in the production of Figures 5.11 (a) and (b)) in significantly fewer steps, producing a smooth trajectory.

The robot is then trained in environments with sparse obstacle population, namely in w_1. The knowledge gained in the previous environment is used as meta-knowledge to aid the robot to adapt to the new environment with a small degree of hostility, and consequently generating a new DT to accommodate the adapted knowledge. Figures 5.12 (a) and (b) show how the robot tries to adapt by producing a new behaviour while heading towards the target. This adaptation is more evident in Figure 5.12 (a) where the robot first encounters obstacles. Figure 5.12 (c) is a scenario where the robot is sufficiently trained in w_1 and homes in on the target.

Different stages of Figures 5.13 and 5.14 show learning and adaptation in more densely obstacle-populated environments, and also environments with arbitrarily shaped obstacles, corners and walls. The navigation performance in Figure 5.13 (c) shows a significant improvement compared with those shown in Figures 5.13 (a) and (b) which have the same settings (initial heading angle and position, same target location). Figures 5.14 (a), (b) and (c) show the robot behaviour while encountering long walls, edges, and corners in homing tasks.

The experimental results shown in the above figures, reveal two characteristics of the hierarchical learning system. Firstly, they illustrate an incremental and steady adaptation to new environments along with improved and smooth trajectories which are reflected in navigation tasks. Secondly, they demonstrate that the robot is always able to reach the target while avoiding obstacles of various configurations and shapes. A further characteristic is the presence of behaviour arbitration when the robot follows long walls, and this is reflected in the so-called "zigzag" trajectories. This is more evident in Figures

5.14 (a), (b) and (c) where the robot follows walls, and conflicting behaviours such as target-seeking and reactivity alternate.

Two possible sources for this zigzag motion are the presence of multiple-class rules and abrupt behaviour switching as control is moved between rule layers. In the current work, the effect of the former is not significant as the classes can generally be considered to have overlapping areas of operation. However, the effect of the latter source does significantly affect the time taken for the robot to reach its goal. In response to this drawback, chapter 7 introduces fuzziness to the rule hierarchy to blend behaviours and optimise global paths.

5.11 Comparison with related learning systems

This section provides a comparison between the current work and research found in the literature concerning single-strategy learning systems. The proposed learning system is operationally similar to that of Tani and Fukumura [7], in that both systems utilise local sensory information to produce general situation-action hypotheses to implement object-avoidance and target-seeking behaviours. The objective of the system introduced in [7] is to construct a hypothetical vector field (based on the information supplied by twenty range sensors) whose temporal flow was used to navigate the robot towards a set target. This system incorporates a composite neural network which consists of a Kohonen network and a three-layer feed-forward network that uses back-propagation learning. The Kohonen network employed is a three-dimensional lattice, and is used to fuse the input sensory data into fewer dimensions. The address of the output of the Kohonen network (described by three parameters in the lattice) at each stage provides three input elements (essential sensor information) which are presented along with their time-delayed version, to the feed-forward network. This network has two output nodes which supply the direction of motion and a stopping command when the goal is reached.

Tani and Fukumura [7] tested their system with a simulated robot which was supervised by a human operator who knew the optimal path leading to a set target. In each training epoch, the robot was guided by its supervisor towards the goal, thereby generating training examples for the learning system. They trained the robot in three consecutive epochs with 38, 86 and 195 training examples, respectively. As also stressed in the current work, they

confirmed that with little training data the robot may get lost or perform erratic movements by wandering around the terrain, as they observed that the desired vector field was largely constructed in the third stage of training where almost all representative vectors were learned. After sufficient training, the system was tested in three navigatory tasks in the same environment, however, with one additional polygonal obstacle placed at different locations. They report that the generated trajectories were nearly perfect and the robot could reach the target, except one situation that the robot was drawn into an unexpected trajectory.

An important aspect of the learning algorithm in the current work in contrast to the learning system in [7] is that in the former learning improves incrementally (for example, concluding legs of trajectories, even at the training stage, are smoother than the initial parts due to the incremental learning), and after sufficient training the system can be tested in completely unknown environments with arbitrarily shaped obstacles. This is in contrast to the learning system in [7], which is reported to be sufficiently robust to cope with only a restricted class of environmental changes. Tani and Fukumura outline the future direction of their research as to incorporate self-learning mechanisms without the need for human guidance - one of the main features of the current work.

The learning system described by Dubrawski and Crowley [8] shares a number of similarities with the current work, even though the control architecture integrates fuzzy logic into a neural learning mechanism to form a hybrid system. As performed in the current work, this system processes the local sensory information to generate reactive situation-action stimuli which are learnt from inception and in a self-supervising fashion. In a manner similar to that found in [7], Dubrawski and Crowley [8] fuse the readouts of 24 proximity sensors to derive seven overlapping sensor readings which cover front, left and right-hand sides of their robot simulator *Robuter*. These sensor readings, together with a set of sensor readings delayed by the sampling time, constitute the 14 input signals to an adaptive resonance theory-based [9] neuro-fuzzy classifier, whose outputs are directed to a single-layer neural associative memory. This network has three processing elements, each of which represents one distinct robot motion, namely move straight on, turn left and turn right. In effect, the adaptive resonance theory (ART) network "stimulates" an appropriate

processing element of the single-layered neural network, which in turn memorises the corresponding pattern and generates a control action. Certain properties of ART systems such as incrementality and fast learning are employed in this system. The aim of the experiments performed using Robuter was to investigate the efficiency of learning from scratch, and Dubrawski and Crowley found that in navigation tasks, the robot exhibited adequately both reactivity and target-seeking behaviour. However, the system showed unstable behaviour when the robot entered narrow perceptual regions and they also report that system may perform oscillatory motions with the robot facing long walls, and left and right turns have equal priority.

As far as the behavioural synthesis (reactivity and goal-seeking), sensor grouping, overall system performance and trajectory profile in navigation scenarios are concerned, the current work is similar to the LIFIA control system introduced by Reignier in [10] and other systems such as those described in [11,12]. The navigation algorithm in LIFIA is a set of manually engineered reactive rules which perform a mapping from the input space to control actions. Reignier reports that LIFIA is able to navigate among randomly scattered obstacles towards a set target. However, it shows oscillatory movements in front of long walls due to a behaviour arbitration, and sometimes falls into local minima.

Beaufrere and Zeghloul [11] implement a sensor-based navigation with an internal supervisor that determines the priority in behaviour activation (and to guide the robot through the shortest path) when obstacles are detected by the sonar sensors. They use two different sets of fuzzy rules for target-seeking and obstacle avoidance behaviour; but the number of such rules is not reported. This system performs local mapping by using fuzzy reasoning without incorporating learning into the algorithm.

The navigation algorithm introduced by Wu [12] treats the robot as a moving point with physical dimensions that are small in comparison with surrounding obstacles. The navigation algorithm is fuzzy based and assumes the environment is stationary, and this is in contrast to the learning algorithm introduced in this thesis. The system in [12] detects the vertices of polygonal obstacles and drives the robot along the line connecting consecutive vertices. In a manner similar to the approach taken in [11], this system performs an

environmental mapping without either exhibiting intelligent behaviour, or being able to cope with changing environments.

5.12 Summary

Inspired by survival instincts and based on DT learning, this chapter introduced to a practical robot environment a new approach to behaviour learning and global navigation. The robot environment is decomposed into a set of homogeneous perceptual worlds of differing complexity and the robot learning system uses the experience of exploring the robot environment to train each of the layers on-line and incrementally. At the navigation stage, each perceptual world is mapped to a unique rule layer which can be searched for prediction purposes.

The core of the system is composed of two algorithms which together carry out on-line learning and navigation, which are distributed over a string of four intelligent local predictors that are independent computational entities with dynamic life times. They are used in combination and are sampled continuously to perform globally-shaped motion predictions. An important attribute of the system is the dynamic rule inhibition which is based on a long term assessment of the performance of the rule layers in the navigation task. The control system ensures safe and globally tuned predictions: safety is achieved by individual local predictors and global shaping by their coupling. The symbolic nature of this technique gives the advantage of control laws which are intelligible to human users, in contrast to those obtained as a result of applying connectionist methods.

A shortcoming of the current implementation, which is exhibited in some of the homing tasks, is the unsmoothness of robot trajectories when following long walls. This effect has also been reported by Reignier in [10]. In the current work, it is the effect of the behaviour switching as control is moved between rule layers which contributes to this oscillatory motion. In response to this drawback, and also to compensate the overall noise in the system, chapter 7 will introduce fuzziness into the learning algorithm to blend behaviours, optimise global paths and exploit the approximate reasoning of fuzzy theory to cope with noisy input data. The following chapter is devoted to the introduction of the mathematical

and set-theoretical foundations of fuzzy logic in order to aid the understanding of the issue of discussion in chapter 7.

References

[1] Michel, Mage Team, I3s Laboratory, CNRS, University of Nice-Sophia Antipolis, France; Khepera Simulator was provided for the Official Khepera Contest at Evolution Artificial Conference (Nimes, 1997), downloadable from: http://alto.unice.fr/~om/khep-contest.html.

[2] O. Michel and P. Collard, "Artificial Neurogenesis, An Application to Autonomous Robotics", *Proceedings of the Eighth International Conference on Tools with Artificial Intelligence*, IEEE Computer Society Press, 1996, pp. 207-214.

[3] A. Löffler, J. Klahold and U. Rückert, "The Dynamical Nightwatch's Problem Solved by the Autonomous Micro-Robot Khepera", J. K. Hao, E. Lutton, E. Ronald, M. Schoenauer and D. Snyers (Eds.), Artificial Evolution, *Third European Conference AE'97*, Nîmes, France, October 1997.

[4] P.E. Utgoff, "Decision Tree Induction Based On Efficient Tree Restructuring", *Technical Report* 95-18, March 17, 1995.

[5] J.R. Quinlan, "C4.5: Programming For Machine Learning", Morgan Kaufmann Publishers, 1992.

[6] P.E. Utgoff and J.A. Clouse, "A Kolmogrov-Smirnoff Metric for Decision Tree Induction", *Technical Report* 96-3, 1996.

[7] J. Tani and N. Fukumura, "Learning Goal-directed Sensory-based Navigation of a Mobile Robot", *Neural Networks*, Elsevier Science Ltd, Vol. 7, No. 3, 1996, pp. 553-563.

[8] A. Dubrawski and J.L. Crowley, "Learning Locomotion Reflexes: A Self-supervised Neural System for a Mobile Robot", *Robotics and Autonomous Systems*, 12 (1994), ELSEVIER, pp. 133-142.

[9] G.A. Carpenter, S. Grossberg and D. Rosen, "Fuzzy ART: Fast Stable Learning of Analogue Patterns by an Adaptive Resonance System", *Neural Networks*, 4 (1991).

[10] P. Reigner, "Fuzzy Logic Techniques for Mobile Robot Obstacle Avoidance", *Robotics and Autonomous Systems*, 12 (1994), ELSEVIR, pp. 143-153.

[11] B. Beaufrer and S. Zeghloul, "Navigation Method for a Mobile Robot Using a Fuzzy Based Method: Simulation and Experimental Aspects", *International Journal of Robotics and Automation*, Vol. 10, Issue 3, 1995, pp. 106-113.

[12] C.J. Wu, "Fuzzy Robot Navigation in Unknown Environments", *The IEEE International Workshop on Emerging Technologies and Factory Automation*, August 11-14, 1992, pp. 624-628.

PAGE

MISSING

IN

ORIGINAL

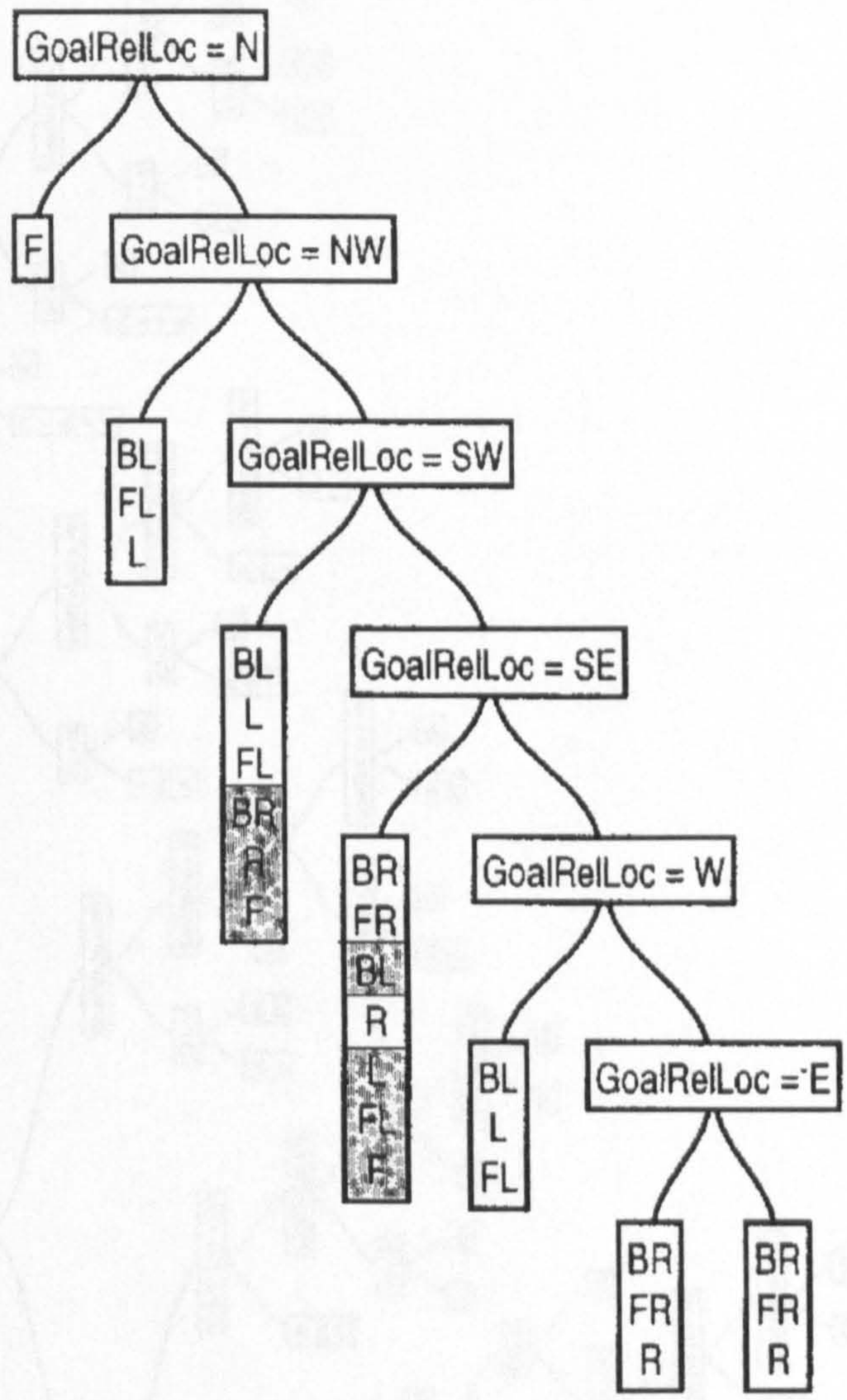


Figure 5.7 A linear decision tree representing w_0 (world zero of the hierarchy). This is used for prediction in environments without any objects. Rules following poor performance are inhibited during navigation (shaded).

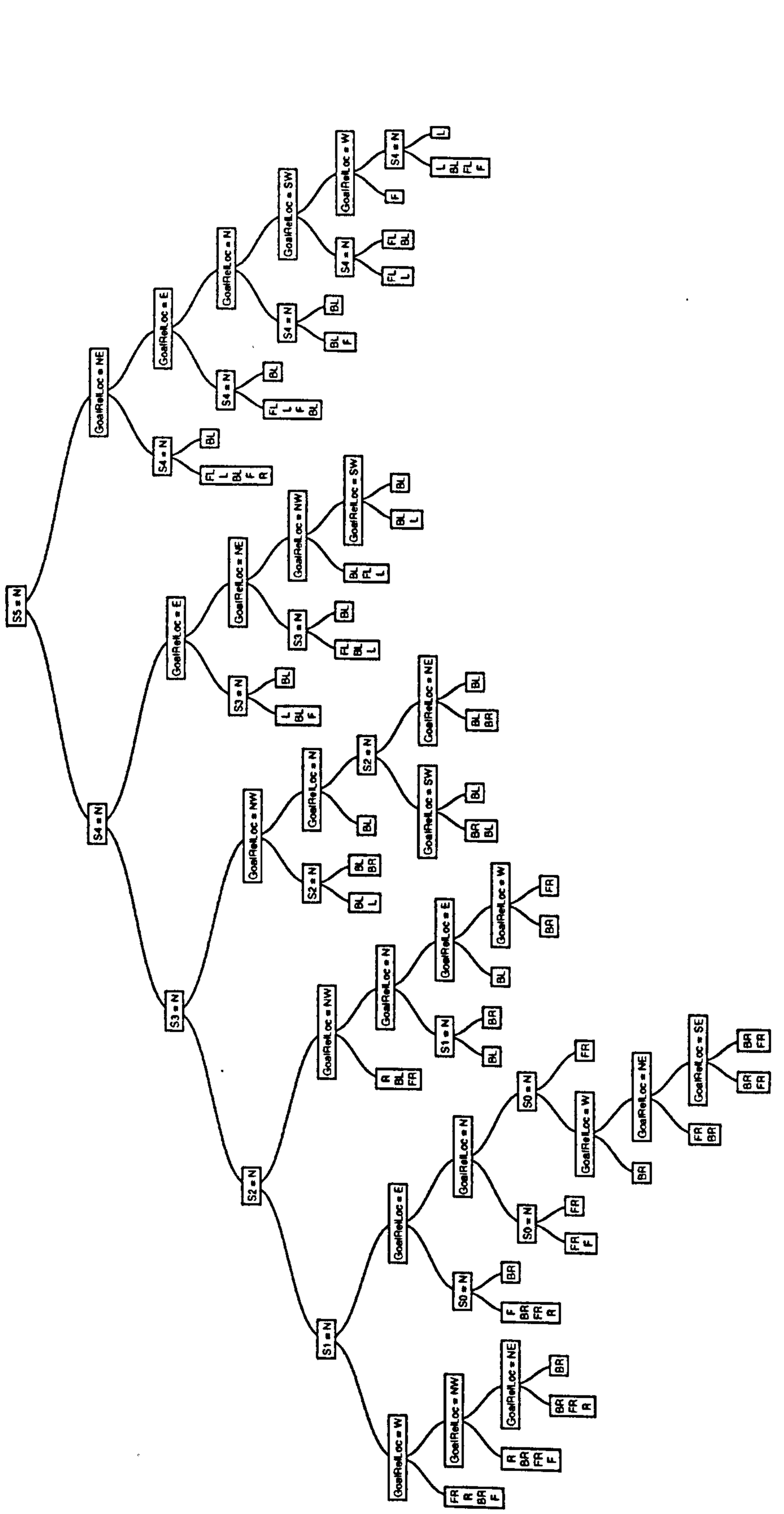


Figure 5.8 A multi dimensional tree network for perceptual mapping in w_1. This layer maps perceptual states where only one proximity sensor group detects objects.

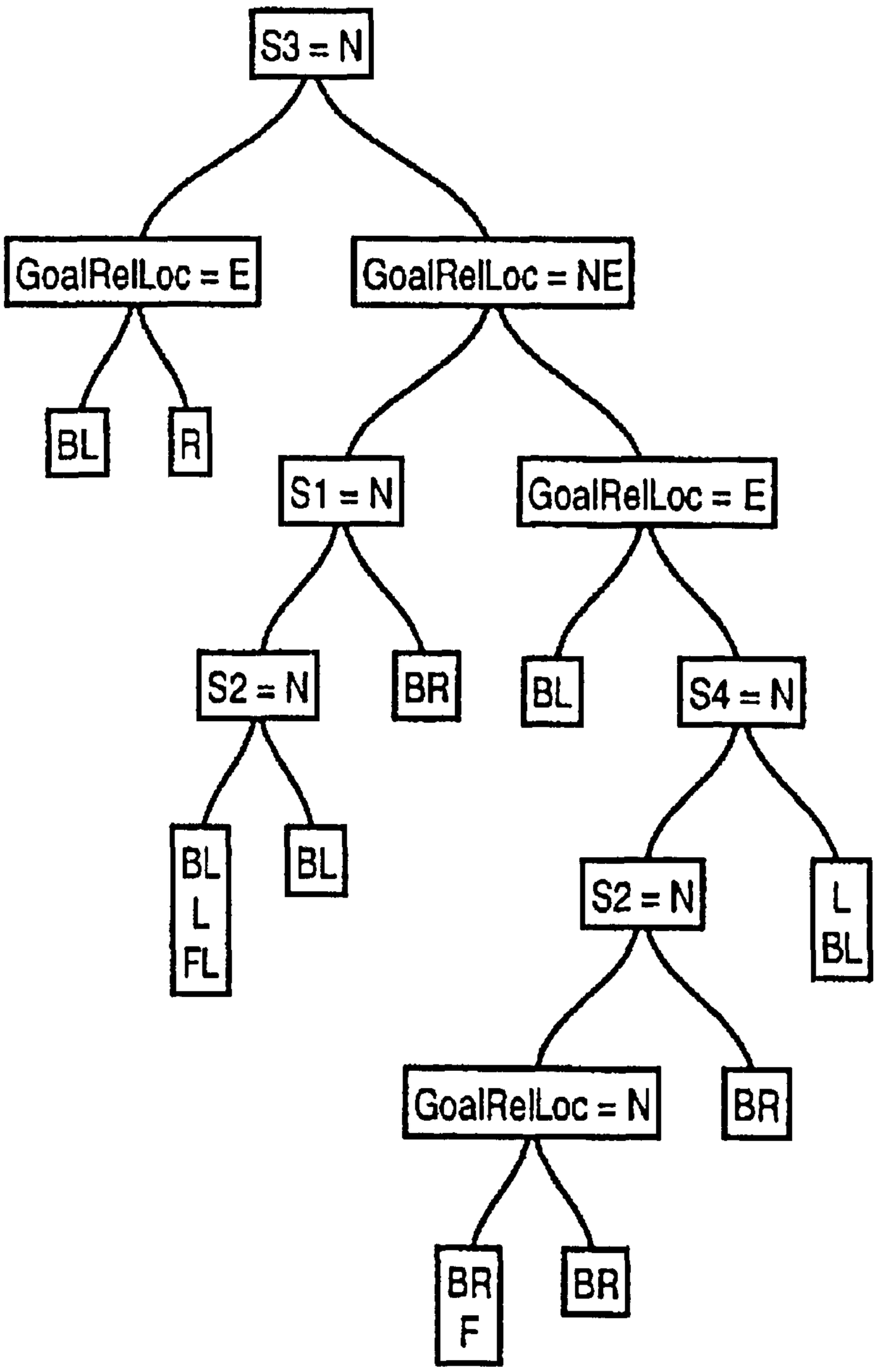


Figure 5.9 An intermediate stage to evolve the third rule layer of the hierarchy w_2.

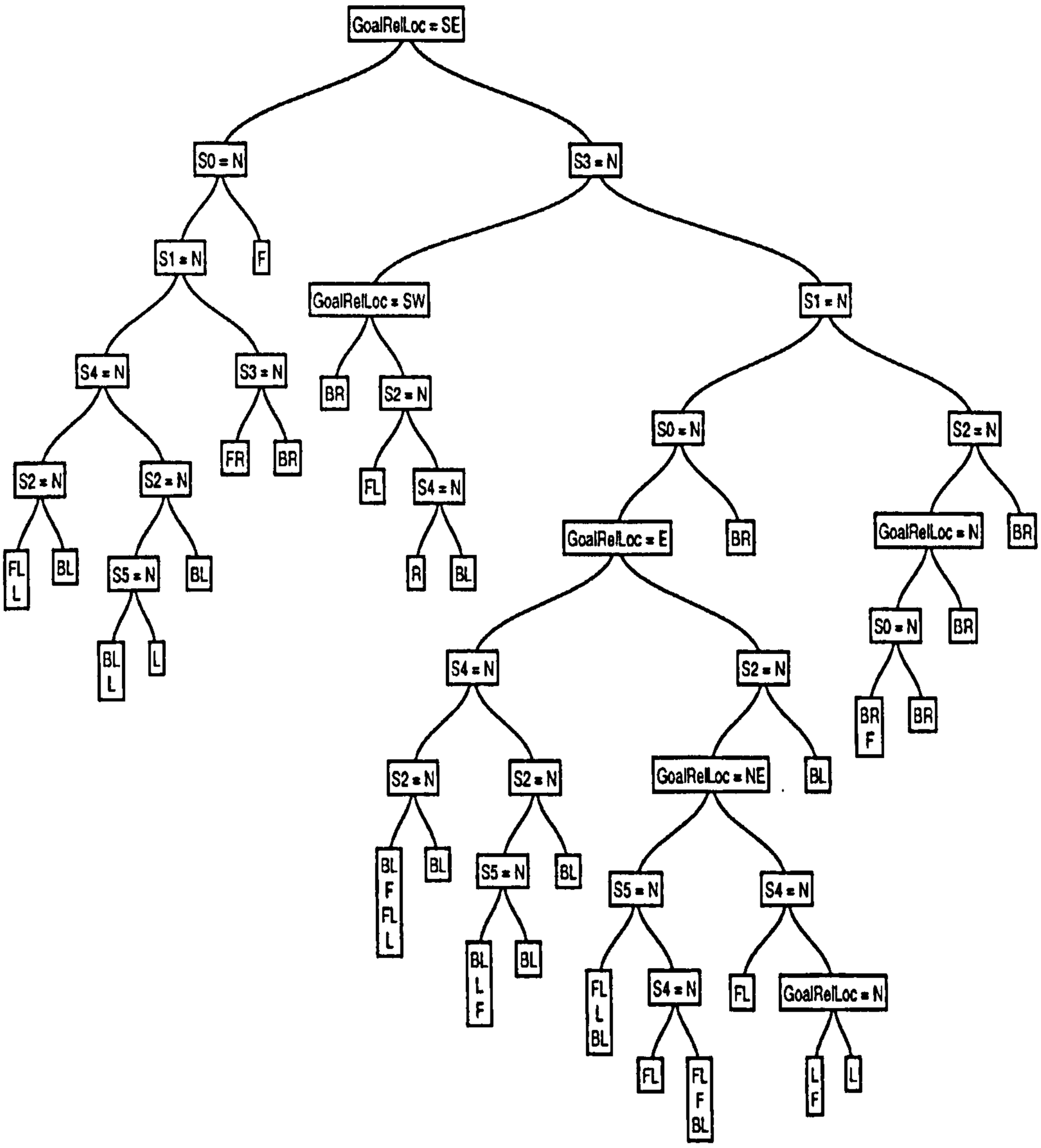
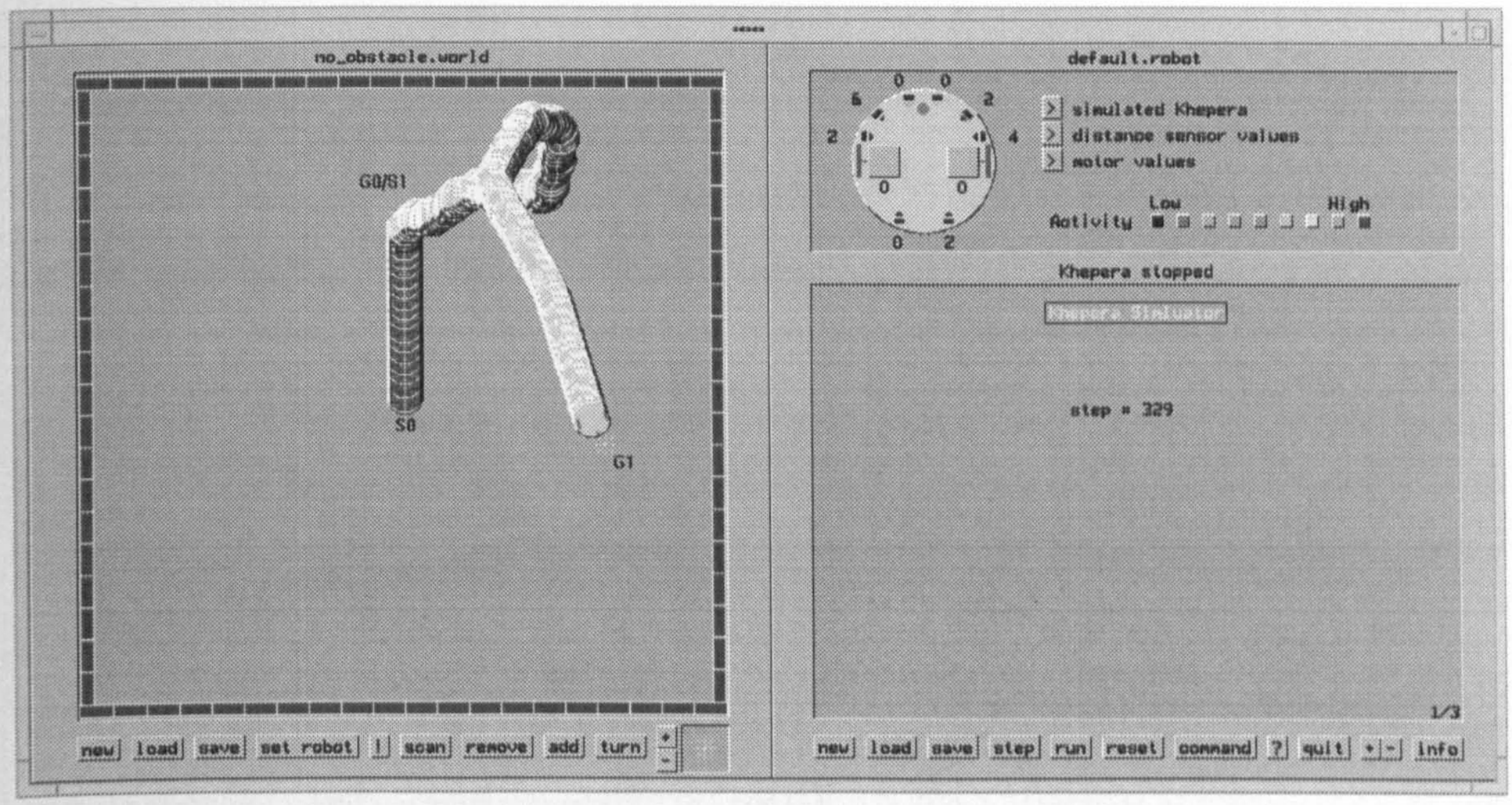
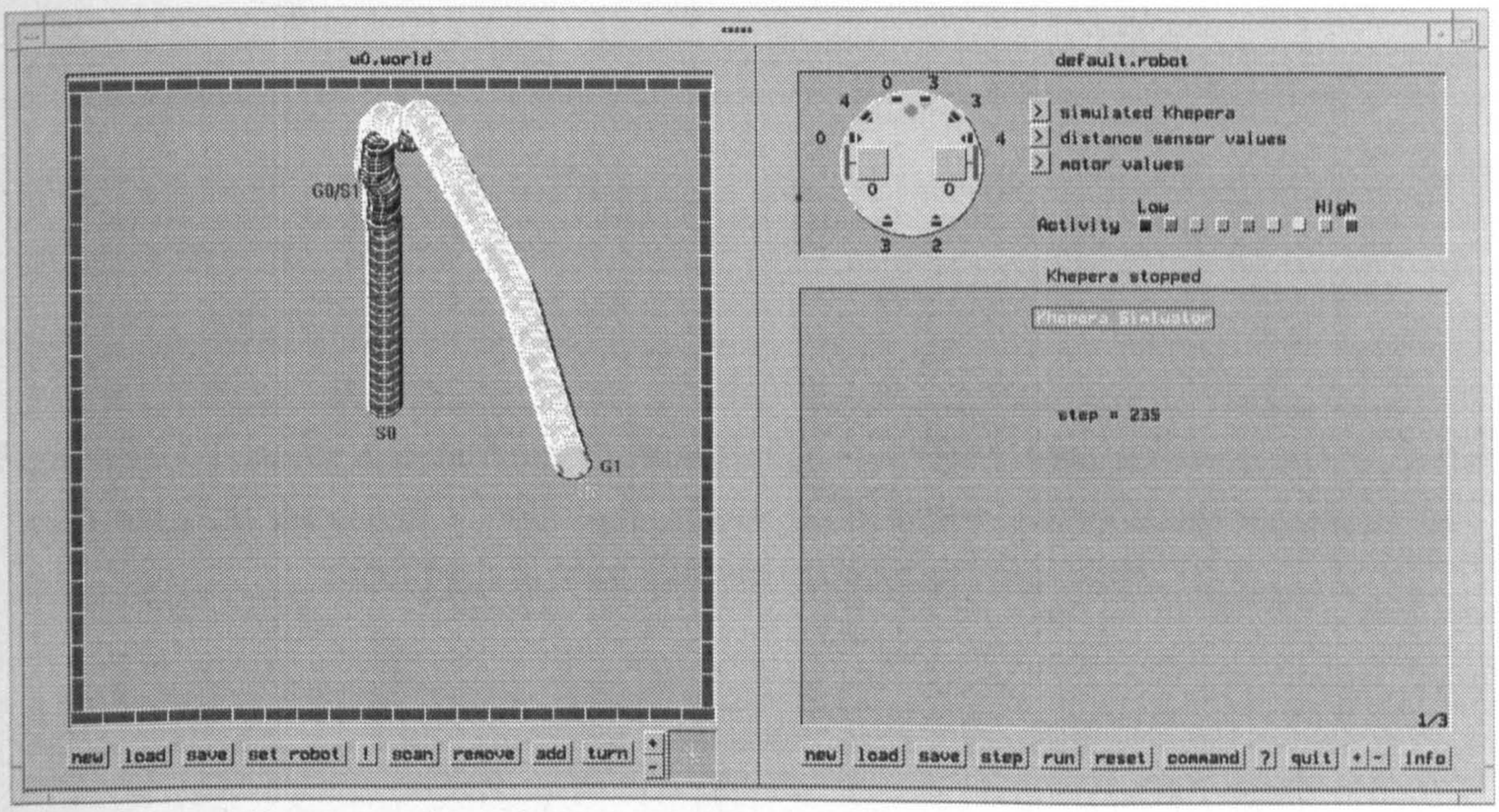


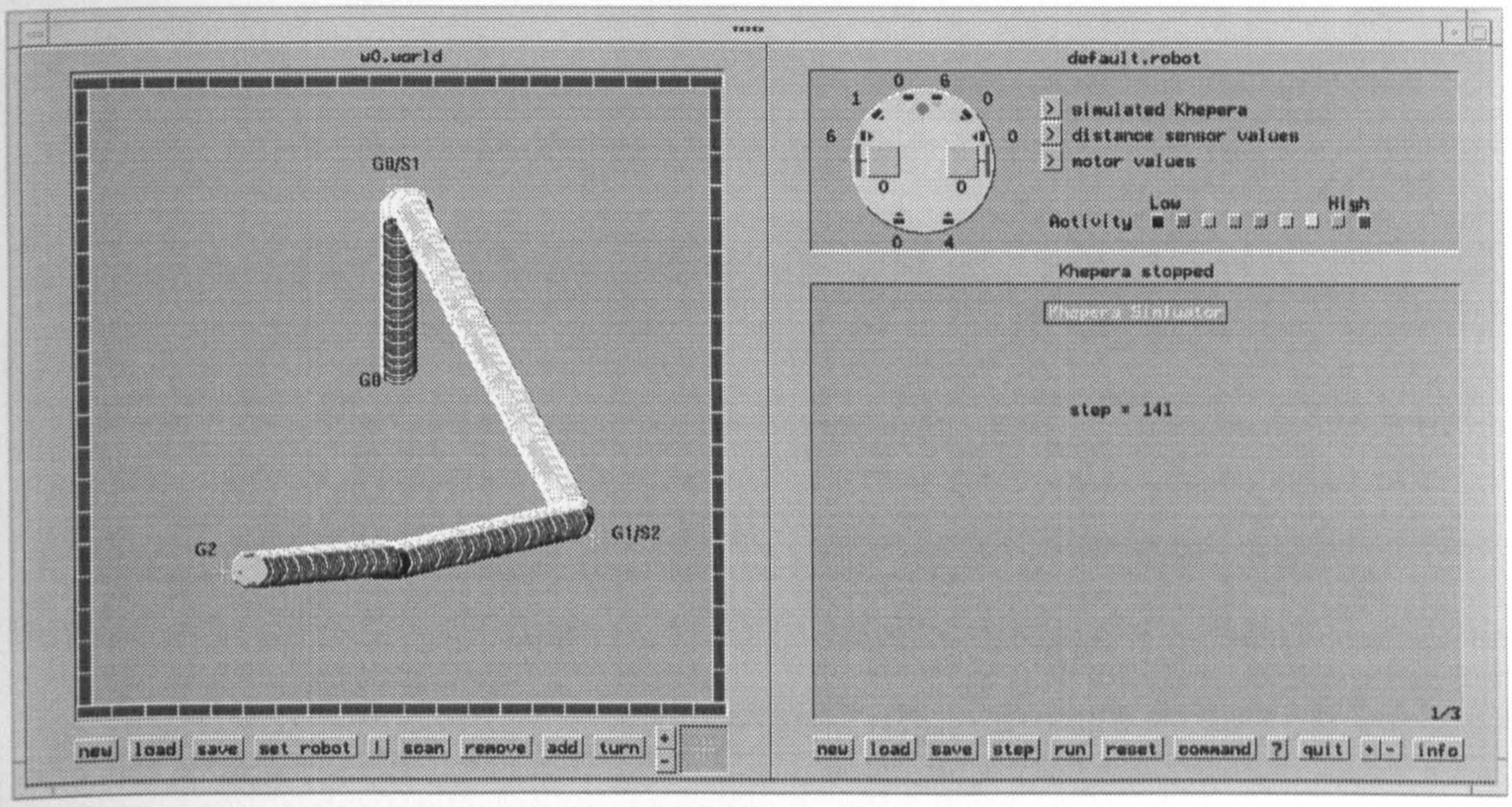
Figure 5.10 The third tree network from the universe of rule layers for predictions in more complex environments.



(a) First stage of learning in w_0 in a two-leg trajectory.

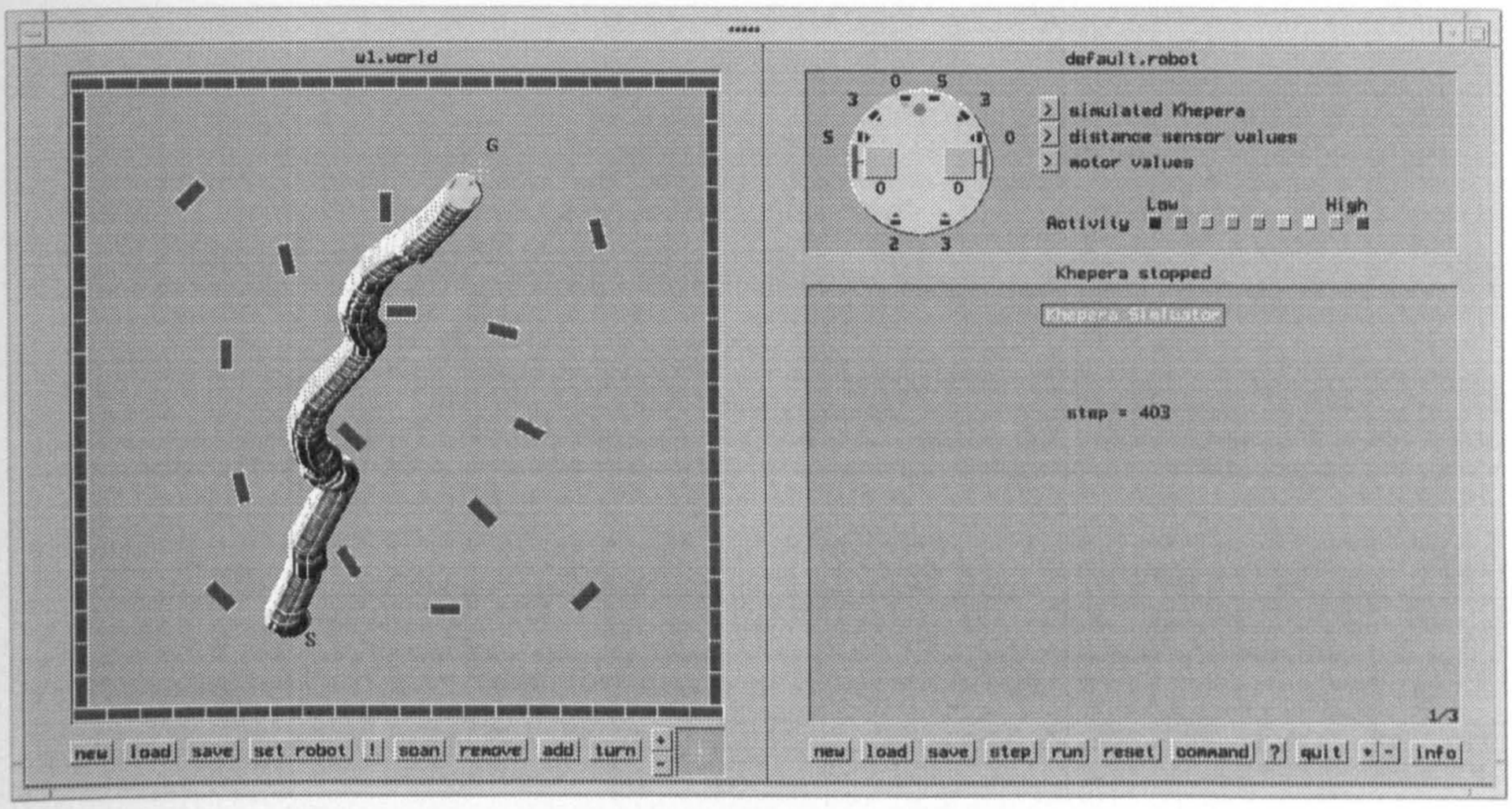


(b) An intermediate stage of learning in w_0 (no obstacles) in a two-leg trajectory leading to incremental improvement.

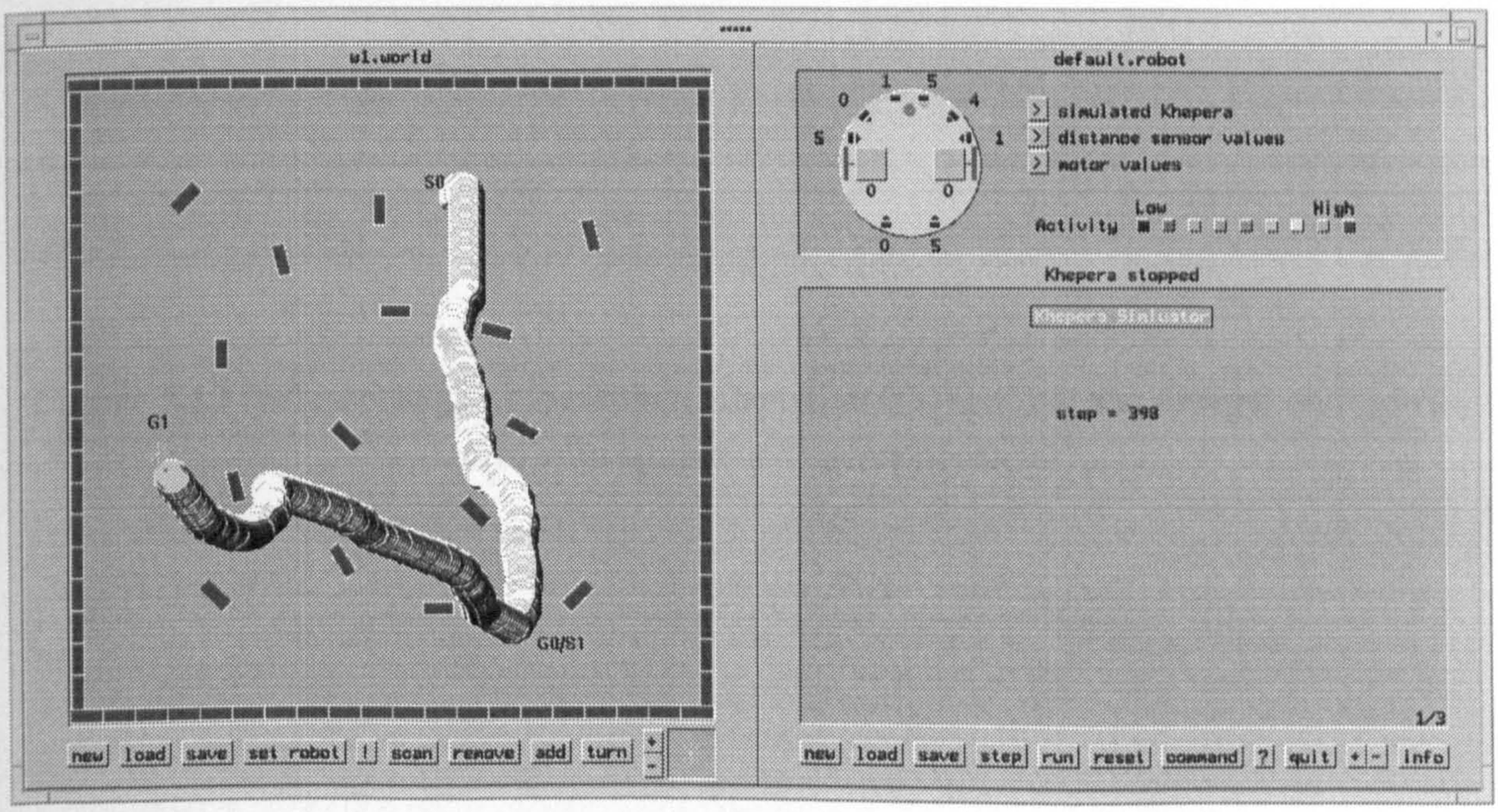


(c) A navigation example in w_0 in a three-leg trajectory.

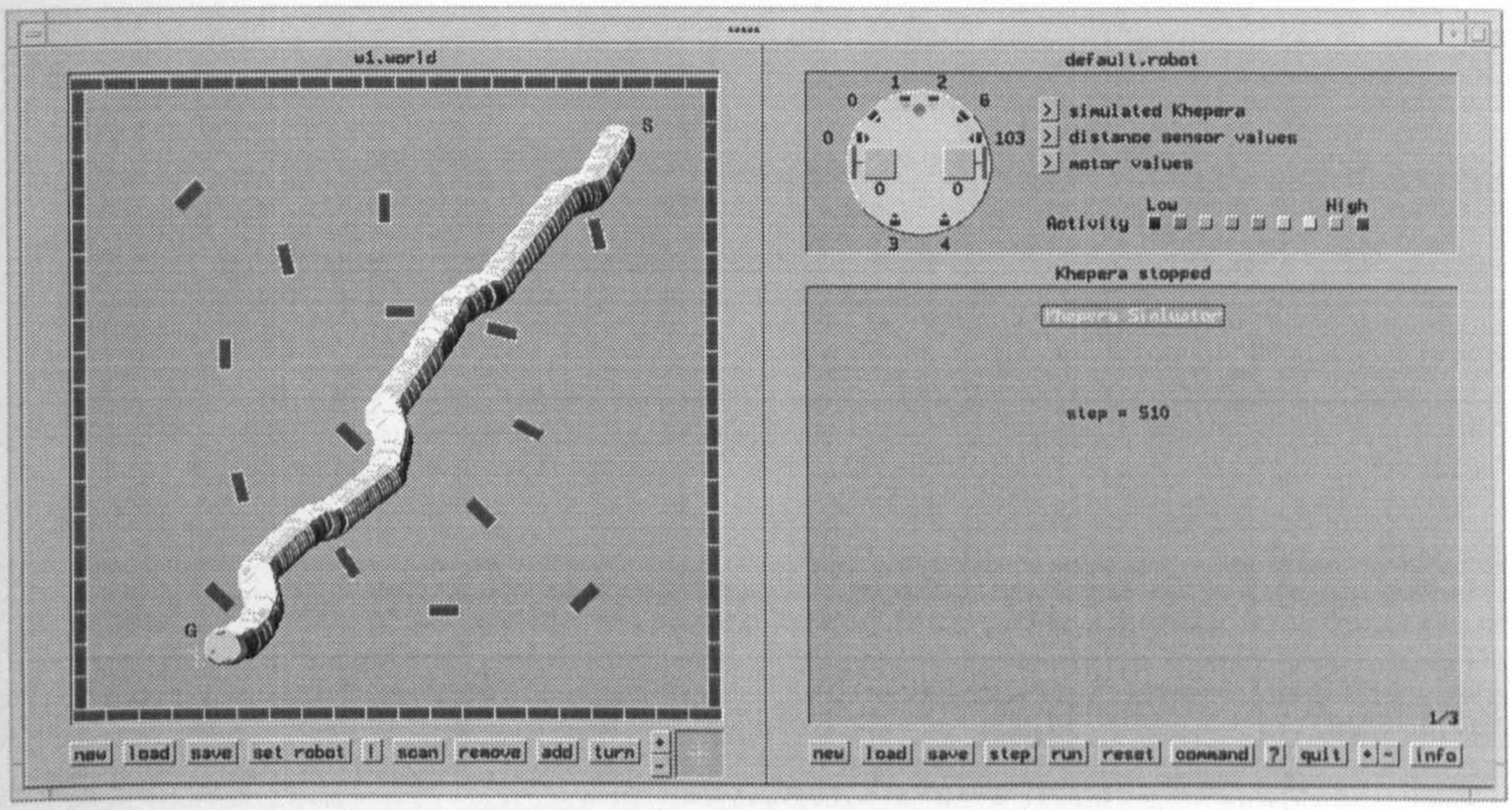
Figure 5.11 Training and initiating the knowledge base (KB) in w_0 (no obstacles). S_x is the start and G_x is the end of x^{th} trajectory. Figure (c) demonstrates the improvement of the robot performance as more knowledge is acquired.



(a) Example of learning in w_1. Learning to become reactive by avoiding simple obstacles scattered in the environment.

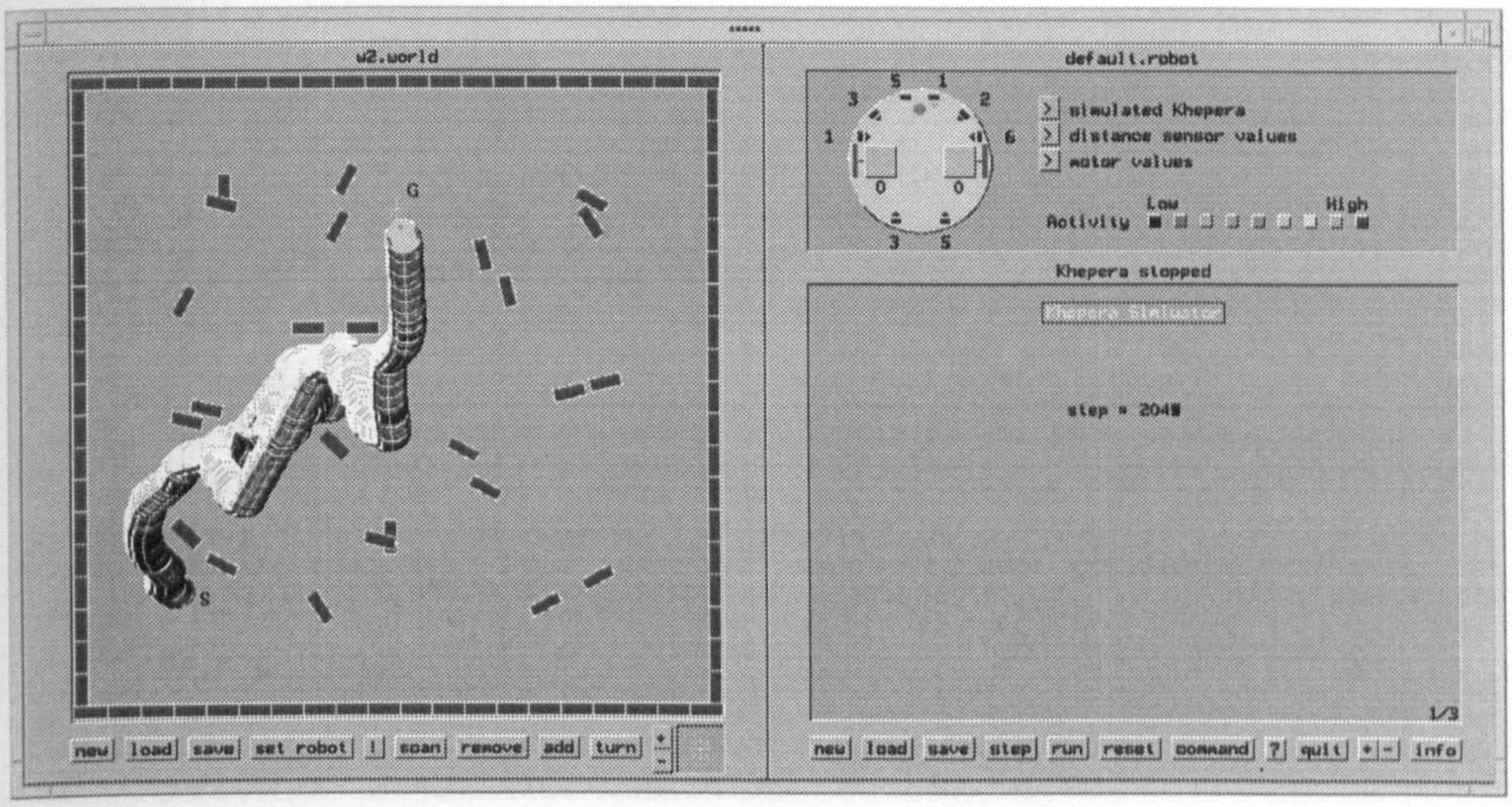


(b) An intermediate stage of learning in w_1 with the robot performing two consecutive homing tasks.

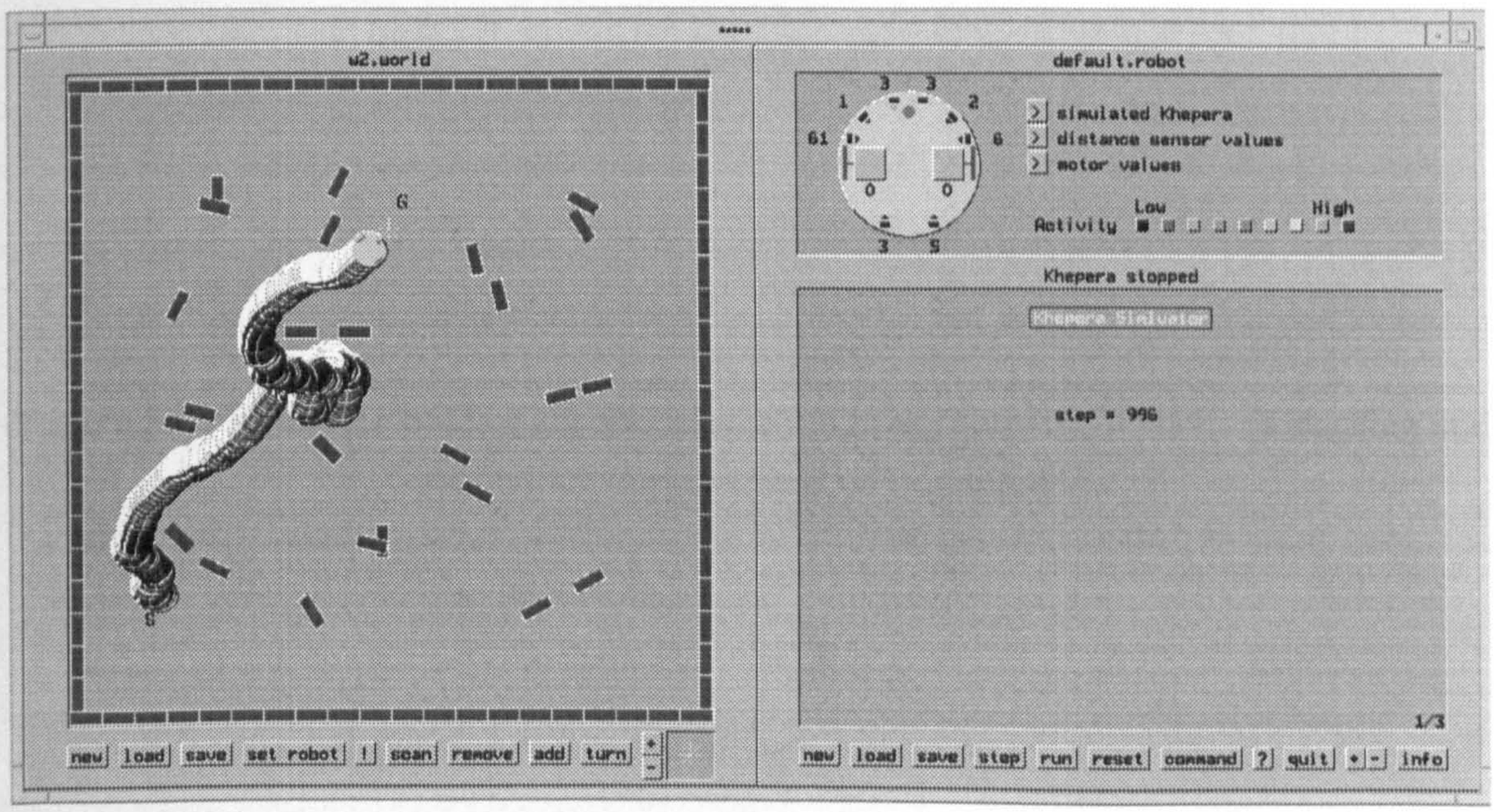


(c) A navigation example in w_1 in a single-leg trajectory. The robot demonstrates a significant improvement in manoeuvring past the obstacles while heading the light source.

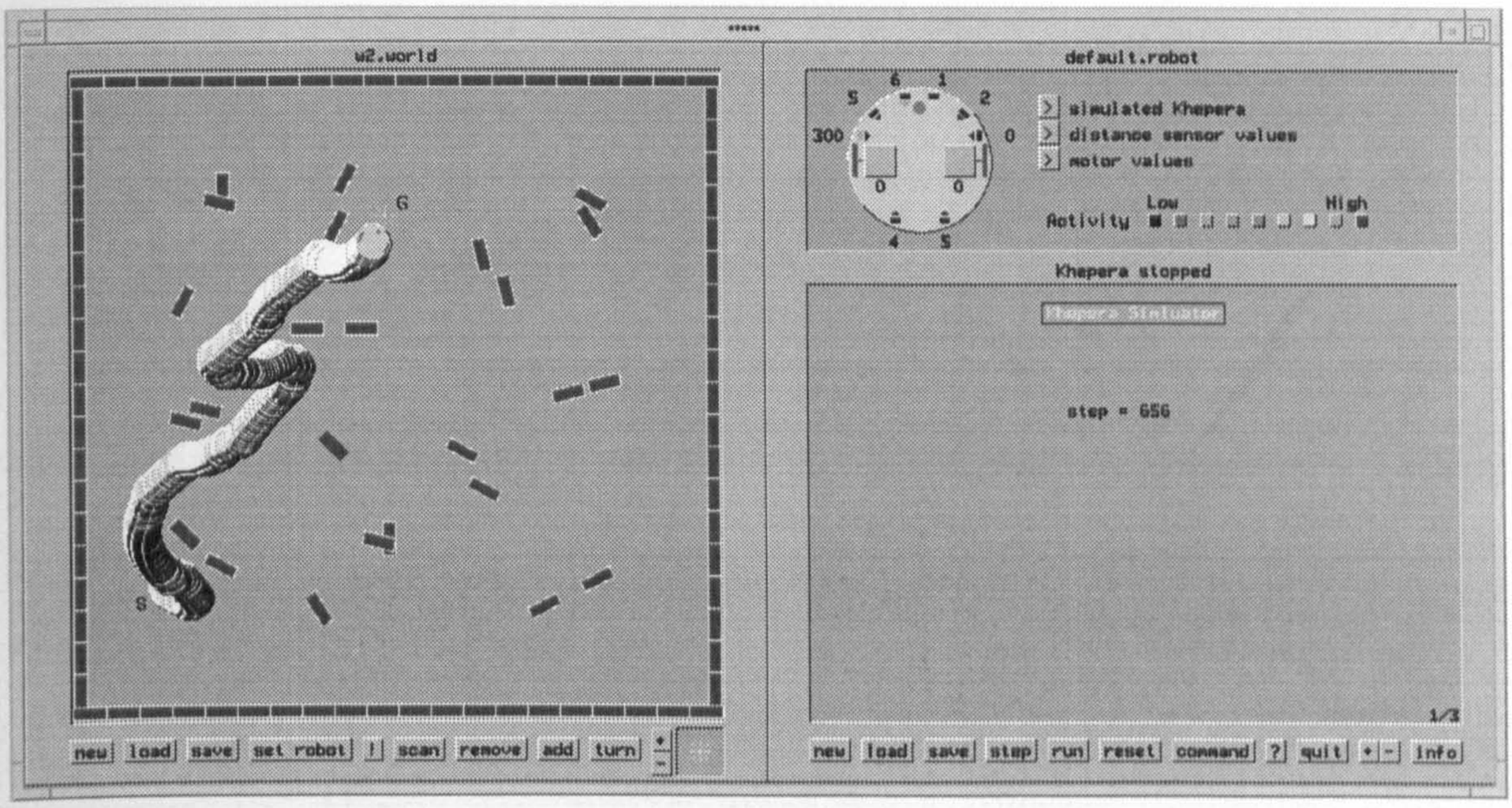
Figure 5.12 Different stages of learning, adaptation ((a) and (b)) and navigation (c) in w_1 (world with simple and sparse obstacles).



(a) Example of learning in and initiating w_2 by adapting to new environment.

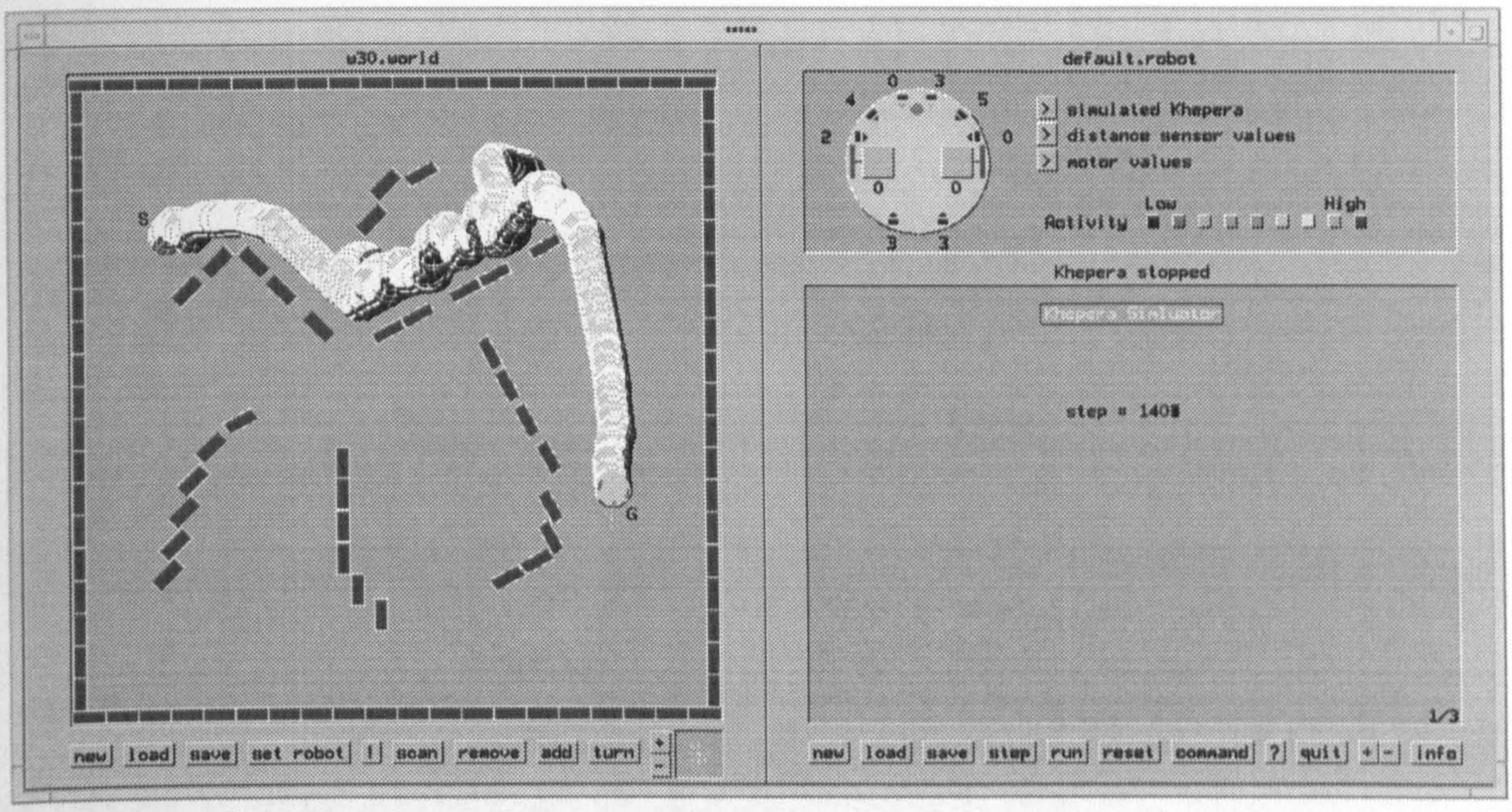


(b) An intermediate stage of learning in w_2 after representative knowledge vectors have been learned.

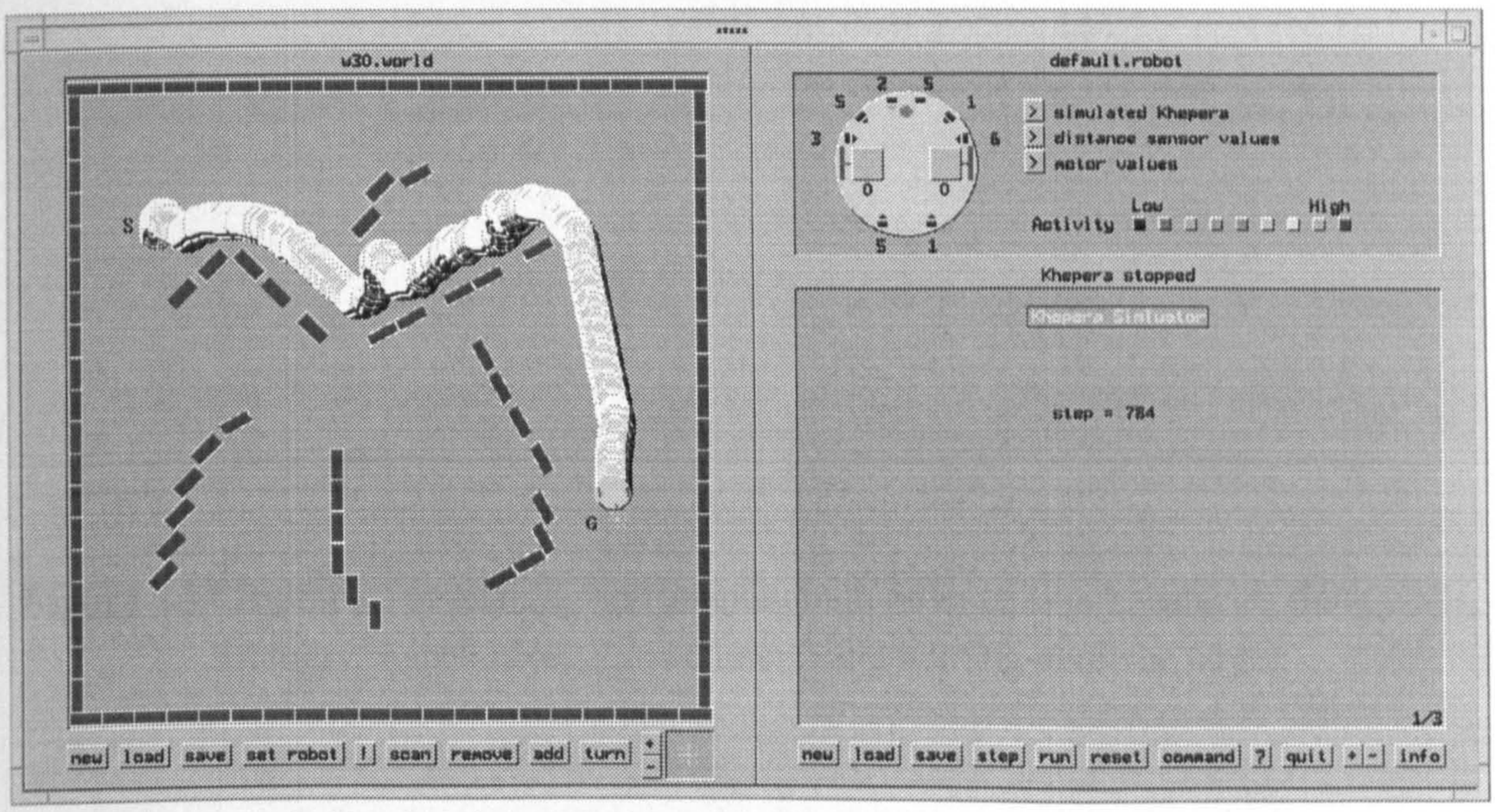


(c) A navigation example in w_2 in a single-leg trajectory. The robot demonstrates a smooth trajectory compared with Figures (a) and (b) in avoiding obstacles.

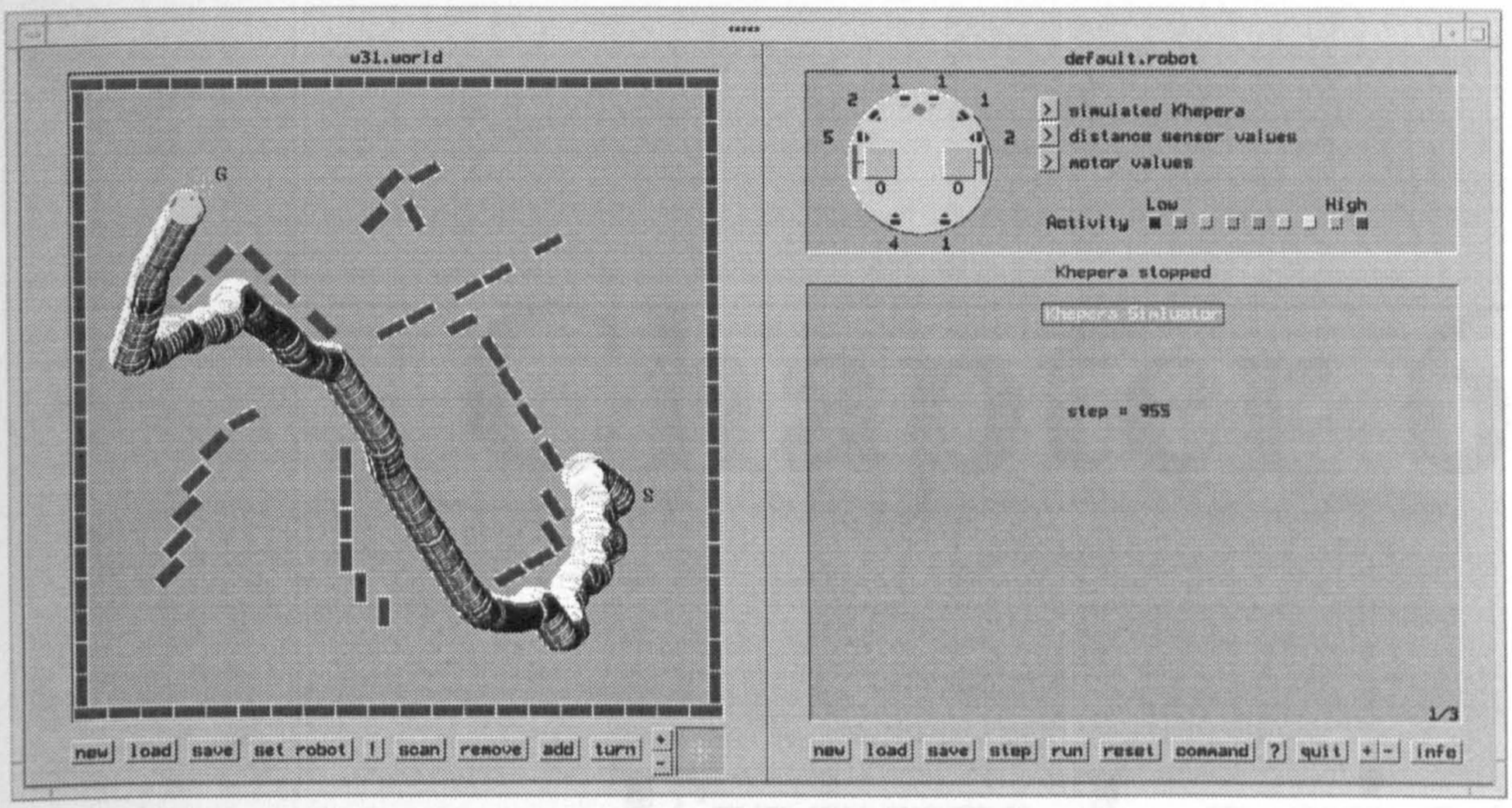
Figure 5.13 Examples of learning and adaptation in w_2, showing behaviour learning and performance improvement on homing tasks.



(a) An initial stage of learning to follow corners, edges and long walls.



(b) A typical example of incremental improvement in learning wall-following behaviour in complex environments with arbitrarily shaped-obstacles.



(c) A navigation example in w_3 in a single-leg trajectory. The robot demonstrates an improved behaviour, but the effect of behaviour arbitration (zigzag trajectory)is not removed.

Figure 5.14 Examples of learning and adaptation in order to avoid obstacles (long walls, arbitrary shaped objects and corners) and homing in on the target.

Chapter 6

Fuzzy Logic: Set Theoretical Foundations

When the only tool you have is a hammer,
everything begins to look like a nail.

L.A. Zadeh

This chapter provides an introduction to fuzzy logic (FL) without entering into its detailed mathematical foundations, as there exists a wealth of literature on this topic [1,2,3,4,5,6,7,8,9,10,11]. The primary objective, though, is to consider the applications of FL rather than analyse in detail the set theoretical foundations. The concept, basic principles, related terminologies and more importantly the engineering applications of fuzzy theory such as fuzzy logic control (FLC) are discussed and addressed, and where appropriate, mathematical derivations are also provided to aid the understanding of the

materials covered in the following chapter. The concluding section is devoted to FLC, in which cognitive information, human experience, heuristics and intuitions can be formulated in mathematical terms for process control whereas this would be unutilised in the context of conventional logic.

6.1 The history of fuzzy logic

Fuzzy set theory, an extension of conventional set theory, was introduced in 1965 after L.A. Zadeh of University of California at Berkeley published his paper on Fuzzy Sets [12] in the Journal of Information and Control. Fuzzy theory has had a prolonged and rather unfortunate childhood (mid 1960's to the early 1980's) in contrast to its booming youth (mid 1980's to present day). Although it is said that this paper has been completed more than two years before, though due to its unorthodox and radical ideas initially no technical authority dared to publish this paper. In later publications, Zadeh laid the foundations of fuzzy logic and approximate reasoning in complex and decision making processes [13]. The methodology advocated in this paper, namely the "principle of incompatibility", as termed by Zadeh, might have been the origin of fuzzy logic. The principle of incompatibility claims that precise and meaningful description of the system's behaviour becomes impossible as the complexity of the system exceeds a certain limit.

Initially, scientists and engineers were rather reluctant to investigate further its theoretical foundations or its possible applications to industrial problems. A turning point for fuzzy logic arrived in September 1974 when Ebrahim Mamdani of the Queen Mary College London applied for the first time fuzzy algorithms in the form of linguistic IF-THEN rules to control a laboratory steam engine [14]. This sparked off the first industrial application of fuzzy logic in 1980 by F.H. Smith of Denmark to control a cement kiln [10]. However, the forerunners (in the early 1980's) in industrial applications of fuzzy logic were Japanese companies such as Fuji Electric who applied fuzzy logic to control a water purification process [8], and Hitachi who transcribed the knowledge of a human expert into a fuzzy rule base for automatic control of the Sendai subway system[10,15]. This fuzzy controlled train is said to have the "smoothest ride on earth" and is able to stop within centimetres of the target [8].

The second International Fuzzy Systems Association (IFSA) conference (held in Tokyo, Japan, July 18, 1987) is widely recognised as a point of maturity for the application of FL in Japanese industries. The conference opened three days after the Sendai Subway opened [8], and control engineers witnessed a demonstration of how easily and inexpensively Takeshi Yamakawa's controller had achieved the fuzzy control of a self-balancing pole. Consumer electronics companies such as Mitsushiti subsequently launched one-button fuzzy washing machines, fuzzy vacuum cleaners, fuzzy rice cookers, fuzzy controlled anti-jittering camcorders, which helped the capture of the consumer products market. Zadeh was awarded the Honda Prize of around $77,000 in November 1989; at that time most western engineers had never heard of fuzzy logic [8].

Japanese engineers have more recently moved onto the fuzzy control of nuclear reactors, and Mitsubishi's engineers are currently developing a fuzzy controller for the cooling system of a nuclear reactor [8].

Fuzzy theory can be regarded now as an important part of the modern, intelligent and cost effective control systems.

6.2 Fuzzy sets and membership functions

A *fuzzy set A*, defined over a universe of discourse X (from which set elements are chosen), is a function that maps some elements of universe X on the real numbers in the closed interval [0,1]. The grade or the extent to which the members of universe X belongs to the fuzzy set A, is called the *membership* function and is usually shown as μ_A and defined as $\mu_A : X \rightarrow [0,1]$. For instance, the membership function of the fuzzy set A defined over universe X (when X is a continuous function) is defined as $\int \mu_A(x)/x$. This is graphically shown in Figure 6.1.

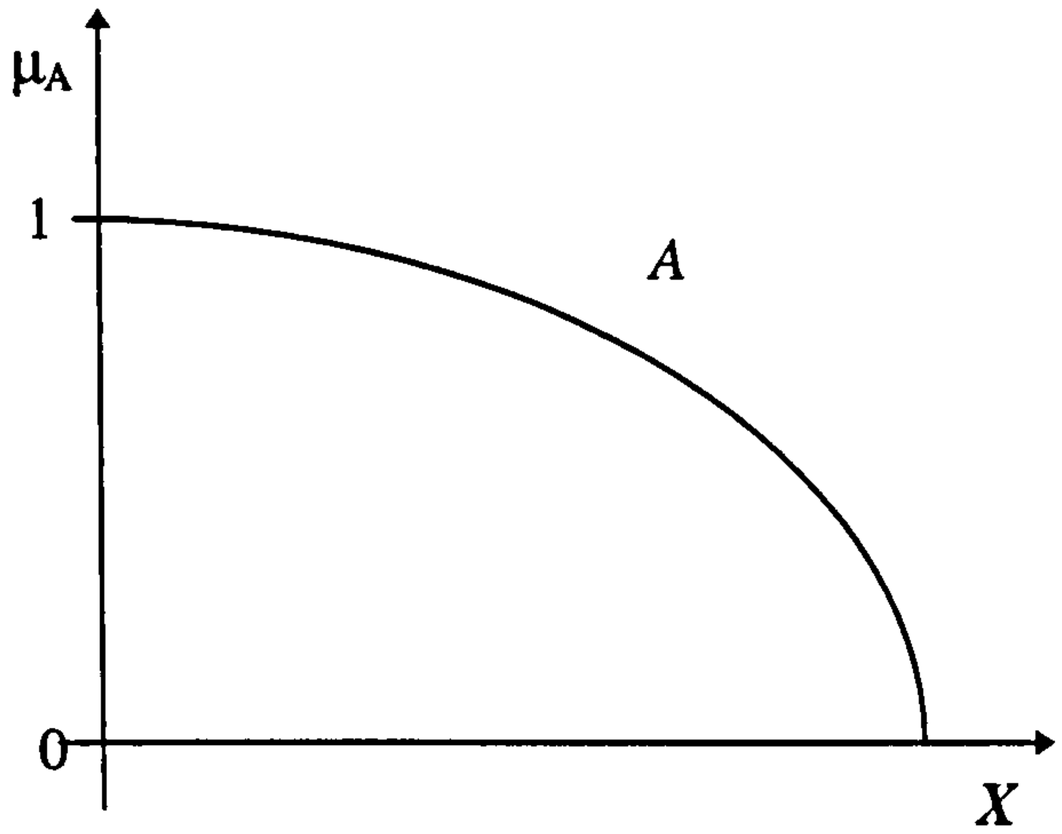


Figure 6.1 A continuous fuzzy set *A*

In situations where the universe X is quantised into *n* discrete levels, as shown in Figure 6.2, the above expression transforms to $\sum_i \mu_A(x_i)/x_i$. The exact definition of a fuzzy set *A* defined over the discrete universe X according to Zadeh [13] is defined as follows:

$$A = \sum_{i=1}^{n} \mu_A(x_i)/x_i \tag{6.1}$$

$$A = \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + \ldots + \mu_A(x_n)/x_n \tag{6.2}$$

Note that in the preceding equations, "$\sum$" and "/" do not imply any summation or division in the ordinary sense applied to conventional (crisp) sets. The same is also applicable to the integral sign in the first expression. To make these definitions more apparent, consider the following numerical examples.

Presumably, a given universe of discourse *X* is defined as $X = \{1, 2, \ldots, 15\}$, the following sets are valid fuzzy sets defined over *X*. They are also graphically depicted in Figure 6.2 and Figure 6.3.

A = {1, 2, 3, 4, 5, 6, 7} or A ≡ 0.25/1 + 0.5/2 + 0.75/3 + 1/4 + 0.75/5 + 0.5/6 + 0.25/7

B = {3, 4, 5, 6, 7} or B ≡ 0/2 + 0.5/3 + 1/4 + 1/5 + 1/6 + 0.5/7 + 0/8

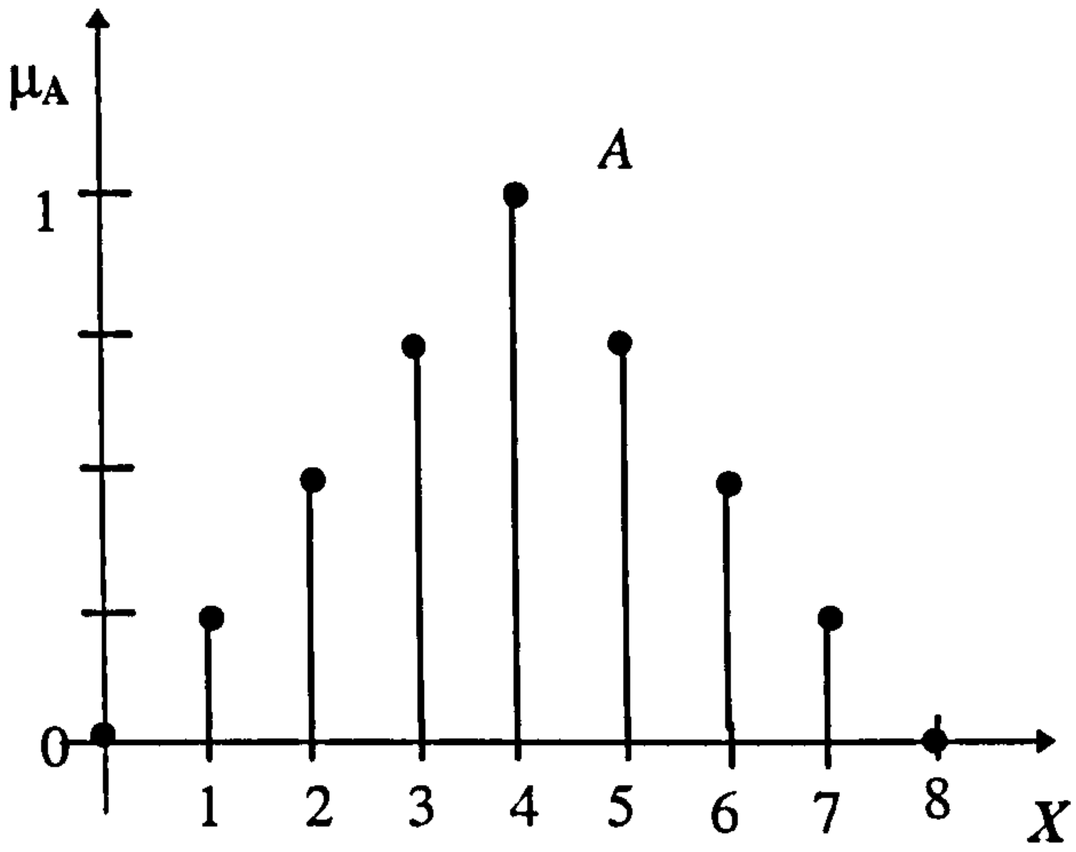


Figure 6.2 Finite expression of fuzzy set *A* (Triangular)

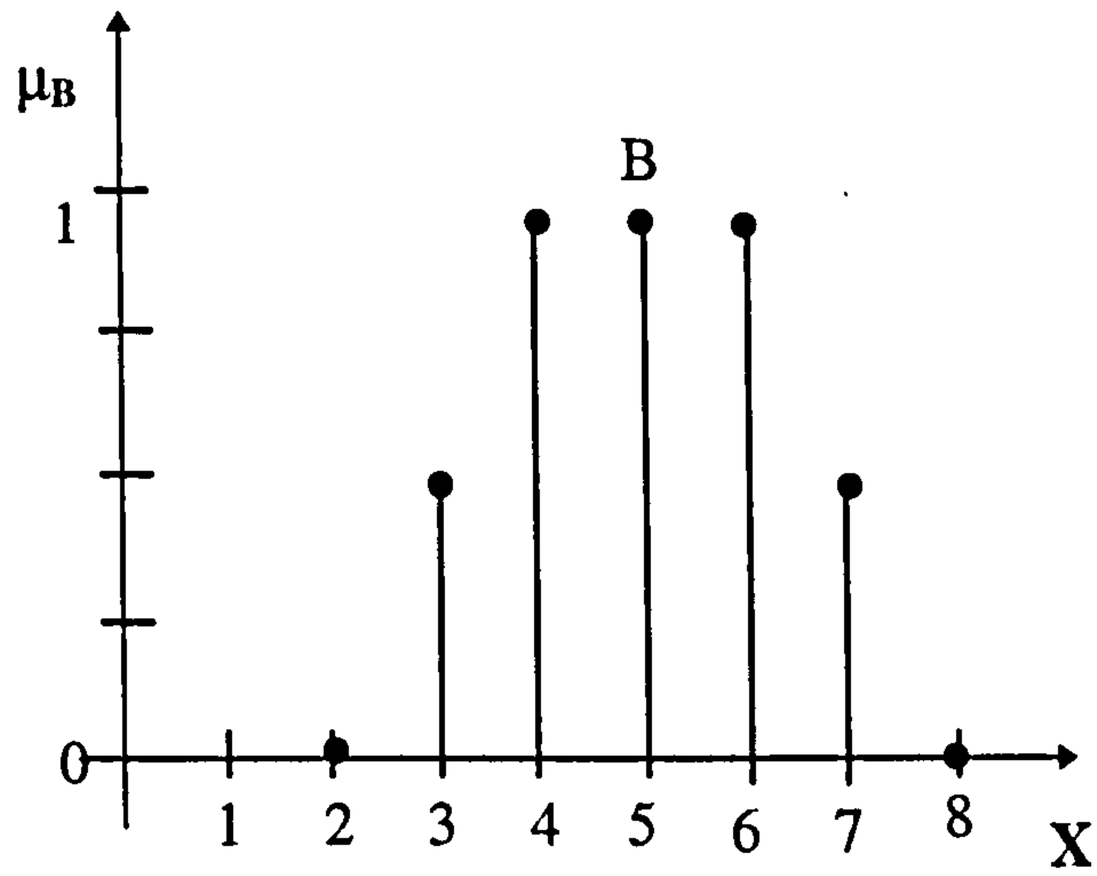


Figure 6.3 Definition of the quantised fuzzy set *B* over *X* (Trapezoidal)

6.2.1 Fuzzy numbers and linguistic variables

A fuzzy set *C* over a continuous universe of discourse *X* is defined to be a *fuzzy number C*, if and only if it satisfies the following conditions:

1. *C* is a convex fuzzy set.
2. *C* is a normal fuzzy set.

Convexity and normality of fuzzy sets can be mathematically expressed in terms of their membership functions as follows:

$$\begin{cases} \max_{x \in X} \mu_C(x) = 1 & (\textit{normality}) \\ \forall x_1, x_2, x_3 \in X, x_1 \le x_2 \le x_3 \Rightarrow \mu_C(x_2) \ge \min\left(\mu_C(x_1), \mu_C(x_3)\right) & (\textit{convexity}) \end{cases} \tag{6.3}$$

Therefore, it can be inferred that every fuzzy number is a fuzzy set whereas the reverse is not valid. Figure 6.4 shows some examples of fuzzy numbers and non-fuzzy numbers.

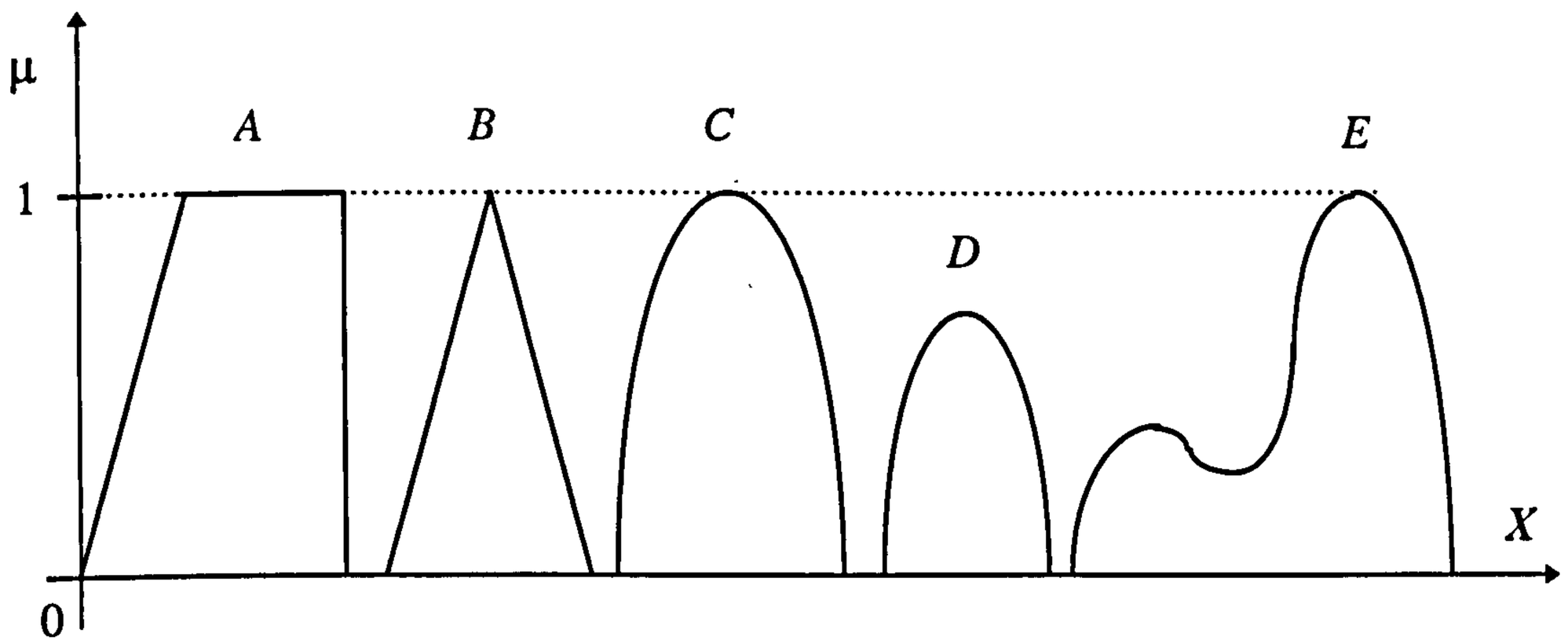


Figure 6.4 An example of fuzzy numbers (*A*, *B* and *C*) and non fuzzy-numbers (*D*, *E*)

It is common practice to swap semantically fuzzy sets with fuzzy numbers, since fuzzy sets such as *D* and *E* (shown in Figure 6.4) are hardly used in engineering applications. Therefore, in the following chapters fuzzy sets are presumed to be both convex and normal (fuzzy numbers). Fuzzy sets provide a basis for manipulation of vagueness and imprecision. To operate and deal with this type of uncertainty, this can be represented by variables whose values are either fuzzy numbers or linguistic terms. These variables are called *linguistic variables*. They are characterised by the triple $(x, T(x), X)$ [16]. Above, x is a variable, *T(x)* is the term set of the variable, i. e. the set of linguistic values of the linguistic variable x with each value being a fuzzy number defined over the universe X.

An example

To clarify the above definitions and to demonstrate the practical significance of fuzzy sets to deal with uncertainty in various application domains, we consider the scenario of human

perception when giving information about the room temperature. In this case *Temperature* is the linguistic variable. Its term set T (*Temperature*) may take on, for instance, the following values:

T(*Temperature*) = {cold, quite cold, warm, very warm, hot}. The vagueness in these terms arising from human cognition gives rise to interpret them as fuzzy sets rather than crisp values. Each fuzzy set in T (*Temperature*) is characterised by its membership function and is defined over the universe of discourse U = {0,...,50}°C. Figure 6.5 shows the distribution of fuzzy sets describing the linguistic variable *Temperature*.

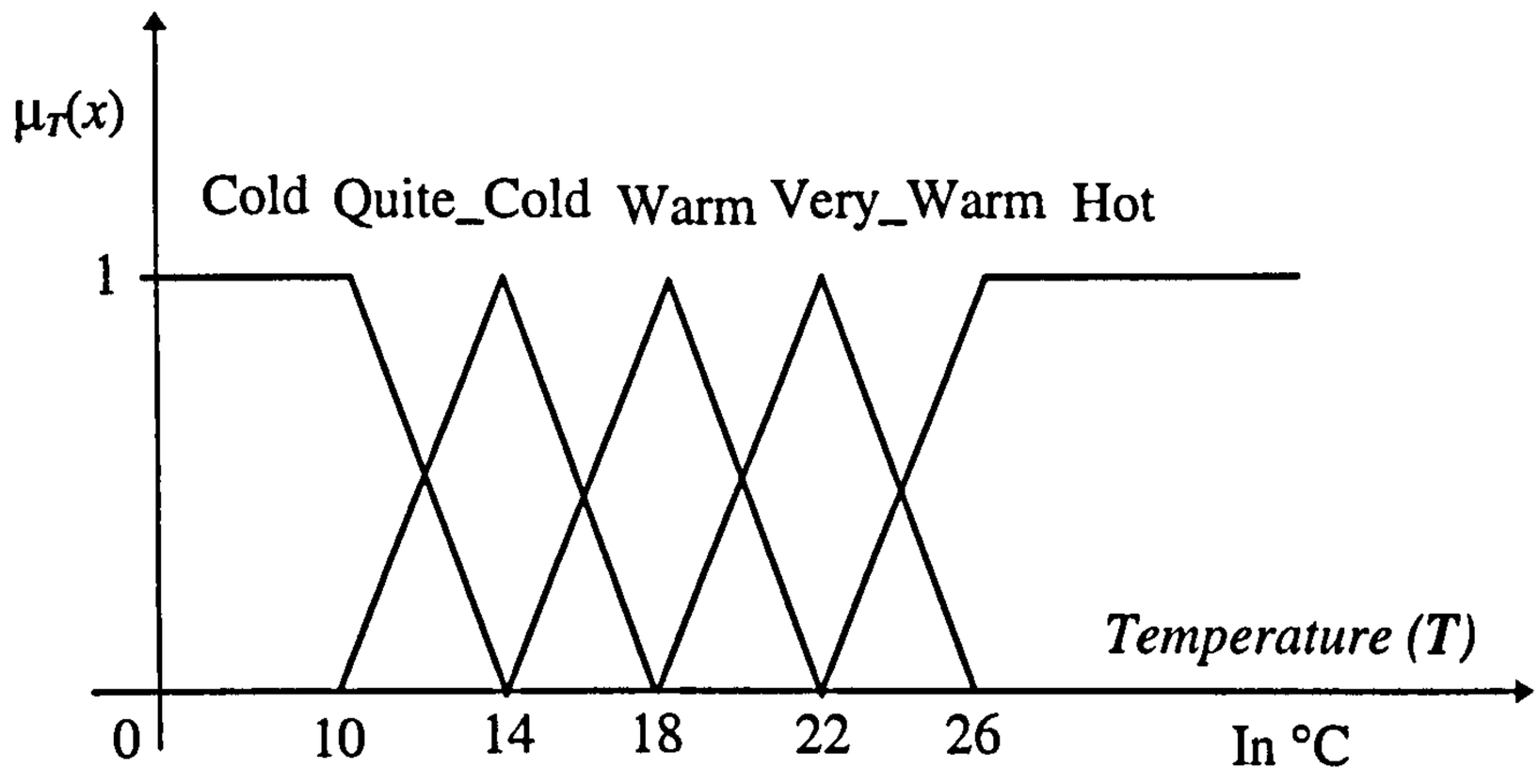


Figure 6.5 Representation of the linguistic variable *Temperature* in terms of its fuzzy values Cold, Quite Cold, Warm, Very Warm and Hot.

6.2.2 Fundamental operations on fuzzy sets

Since fuzzy set theory can be considered as the extension of classical set theory, set theoretic operations on fuzzy sets obey the same principles as in the case of crisp sets. However, the extension to fuzzy sets is not uniquely defined, as $\mu_A(x)$ and $\mu_B(x)$ can take on an infinite number of values from the interval of real numbers [0,1]. Thus, there exists an infinite number of possible definitions to represent fuzzy *union* and *intersection*. *Triangular norms* (T-norms) and *triangular conorms* (T-conorms or S-norms) are different representations to implement fuzzy intersection and union, respectively, introduced by different researchers. The definitions of these operations are given below along with the T-norms and T-conorms introduced by Zadeh [12]. In this thesis, the entire operations on

fuzzy sets will be based on these implementations. For a comprehensive list of T-norms and T-conorms the reader is referred to [17,18]. In the following, we also discuss briefly the complement operation extended to fuzzy sets. However, for more detailed discussions and derivations on properties of fuzzy sets see [10,16,18].

If we generalise the membership functions of two fuzzy sets A and B, $\mu_A(x)$ and $\mu_B(x)$, *union*, *intersection* and *complement* of the above sets are then defined as shown below.

Union of fuzzy sets

Considering A and B to be fuzzy sets defined over universe X, the union of these fuzzy sets, $A \cup B$, is itself a fuzzy set over X defined by its membership function as:

$$\mu_{A\cup B}(x) = \mu_A(x) \vee \mu_B(x) \text{ or}$$

$$= \max\left(\mu_A(x), \mu_B(x)\right) \tag{6.4}$$

where

$$\max\left(\mu_A(x), \mu_B(x)\right) = \begin{cases} \mu_A(x) & \text{if} \quad \mu_A(x) \geq \mu_B(x) \\ \mu_B(x) & \text{if} \quad \mu_A(x) < \mu_B(x) \end{cases} \tag{6.5}$$

Intersection of fuzzy sets

The intersection of two fuzzy sets A and B, $A \cap B$, is a fuzzy set over X defined by its membership function as:

$$\mu_{A\cap B}(x) = \mu_A(x) \wedge \mu_B(x) \text{ or}$$

$$= \min\left(\mu_A(x), \mu_B(x)\right) \tag{6.6}$$

where

$$\min\left(\mu_A(x),\mu_B(x)\right)=\begin{cases}\mu_A(x) & \text{if} \quad \mu_A(x)\le\mu_B(x)\\ \mu_B(x) & \text{if} \quad \mu_A(x)>\mu_B(x)\end{cases} \tag{6.7}$$

Fuzzy complement

The complement of a fuzzy set A, $\overline{A}$, is a fuzzy set over X defined in terms of its membership function as:

$$\mu_{\overline{A}}(x)=1-\mu_A(x) \tag{6.8}$$

Note that the only properties of crisp sets which are, in general, not valid for fuzzy sets are:

- the law of the excluded middle:

$$A\cup\overline{A}\neq X \tag{6.9}$$

- the law of contradiction:

$$A\cap\overline{A}\neq\varnothing \tag{6.10}$$

where $\varnothing$ is an empty set. More details on related materials can be found in [10,18,19].

6.3 Fuzzy relations

The dynamics of the fuzzy-based systems are described by fuzzy relations. They play a key role in these systems, because they characterise the interrelationships between fuzzy sets or more generally fuzzy variables.

Since fuzzy relations are directly coupled with another concept, namely fuzzy *Cartesian product*, this is first defined to provide a better understanding of underlying principles associated with fuzzy relations.

Fuzzy Cartesian product

Let $A_1, \ldots, A_n$ be fuzzy sets in $X_1, \ldots, X_n$, respectively. The Cartesian product of $A_1, \ldots, A_n$ is a fuzzy set in the product space $X_1 \times X_2 \times \ldots \times X_n$ and is defined as follows:

$$A_1 \times A_2 \times \ldots \times A_n = \int_{X_1 \times X_2 \times \ldots \times X_n} \min\left(\mu_{A_1}(x_1), \ldots, \mu_{A_n}(x_n)\right) / (x_1, \ldots, x_n) \tag{6.11}$$

Where the membership function of this product is defined as:

$$\mu_{A_1 \times A_2 \times \ldots \times A_n}(x_1, x_2, \ldots, x_n) = \min\left(\mu_{A_1}(x_1), \mu_{A_2}(x_2) \ldots, \mu_{A_n}(x_n)\right) \tag{6.12}$$

An *n-ary fuzzy relation* R in the Cartesian product space $X_1 \times X_2 \times \ldots \times X_n$ is a fuzzy set whose membership function has n variables and is defined as follows:

$$R = \left\{ \mu_R(x_1, x_2, \ldots, x_n) / (x_1, x_2, \ldots, x_n) \middle| x_1 \in X_1, x_2 \in X_2, \ldots, x_n \in X_n \right\} \tag{6.13}$$

or alternatively

$$R = \int_{X_1 \times X_2 \times \ldots \times X_n} \mu_R(x_1, x_2, \ldots, x_n) / (x_1, x_2, \ldots, x_n) \tag{6.14}$$

where μ_R is the membership function of R and is a mapping function given by:

$$\mu_R: \; X_1 \times X_2 \times \ldots \times X_n \rightarrow [0,1] \tag{6.15}$$

A practical example of fuzzy relations is that they describe a fuzzy set in a multi-dimensional space such as a Cartesian product space. This has a central meaning, since fuzzy relations can be utilised to model linguistic associations, correlations and relations between the elements of the product space in practical applications. These include, for instance, statements such as "*slightly smaller than*", "*about the same as*", "*fairly close to*", etc.

To visualise the importance of fuzzy relations and their significance in dealing with vague information, a pictorial image of fuzzy relations is demonstrated by considering a hypothetical example.

Figures 6.6(a) and (b) show two fuzzy sets "High Temperature" and "Medium Pressure", taken from the term set of Temperature and Pressure defined in a unique universe X and Y, respectively. However, Figure 6.7 demonstrates the fuzzy set describing the interaction between Temperature and Pressure in a two-dimensional space. This set is called a fuzzy relation and is defined in the Cartesian product space of $X \times Y$.

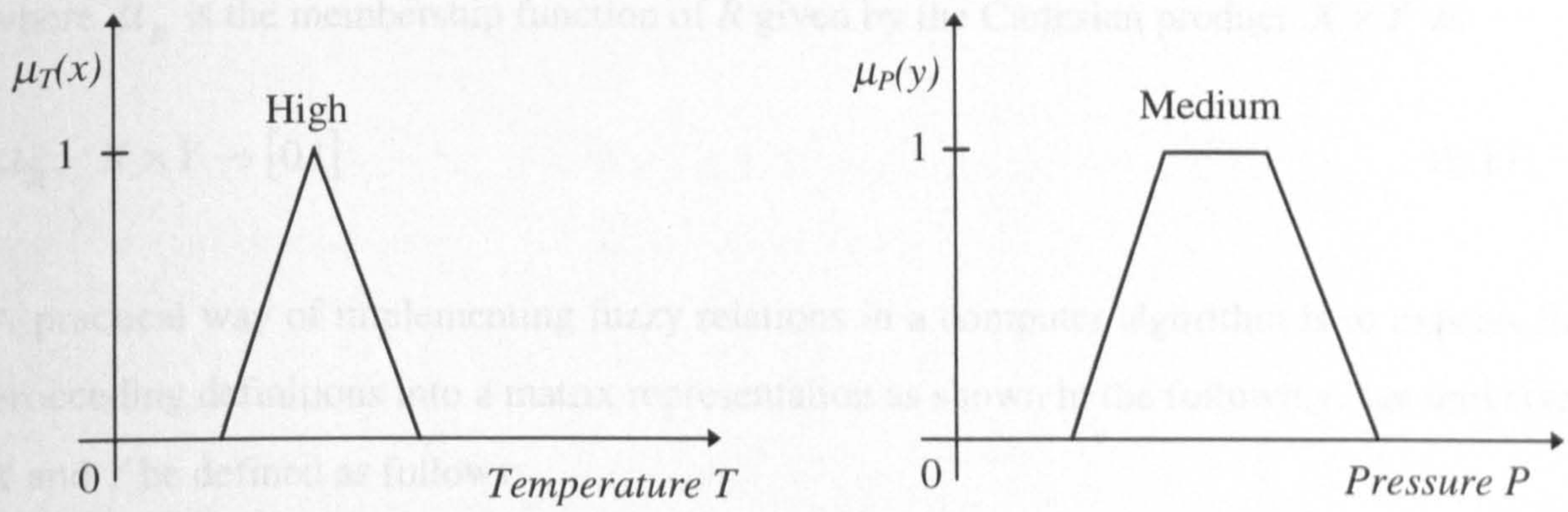


Figure 6.6 Individual fuzzy sets defined in independent universes X an Y.

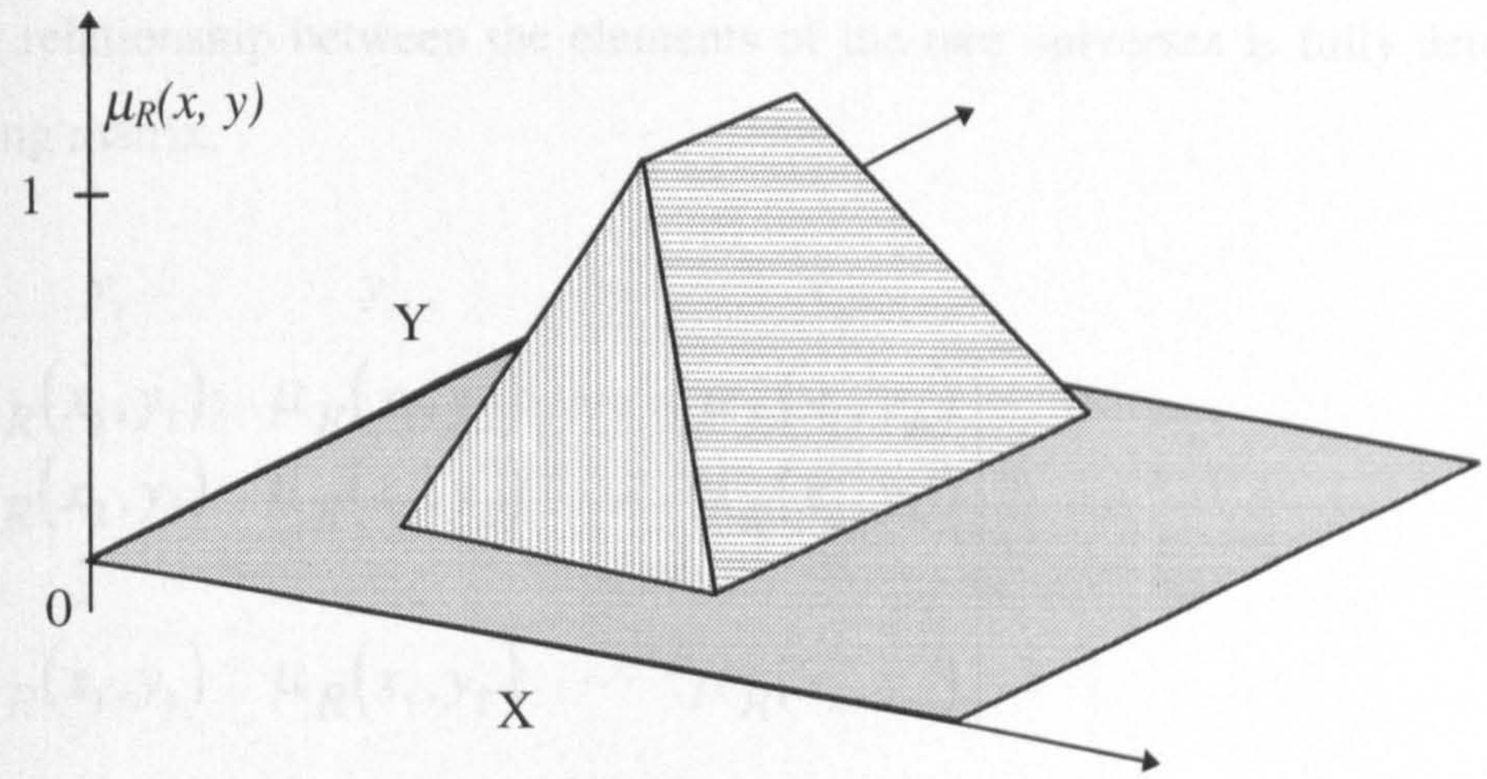


Figure 6.7 A fuzzy relation in the multi-dimensional space of Temperature and Pressure characterising the interaction between the two universes.

In most practical applications, the dimensionality of *n-ary* fuzzy relations is reduced to two or more dimensions such as binary relations for two-dimensional cases. In binary relations, the expressions (6.14) and (6.15) are reduced to the following forms:

$$R = \int_{X \times Y} \mu_R(x,y)/(x,y) \tag{6.16}$$

where μ_R is the membership function of R given by the Cartesian product $X \times Y$ as:

$$\mu_R: \ X \times Y \rightarrow [0,1] \tag{6.17}$$

A practical way of implementing fuzzy relations in a computer algorithm is to express the proceeding definitions into a matrix representation as shown in the following. Let universes X and Y be defined as follows:

$$\begin{cases} X = \{x_1, x_2, \ldots, x_n\} \\ Y = \{y_1, y_2, \ldots, y_n\} \end{cases} \tag{6.18}$$

The binary relationship between the elements of the two universes is fully determined by the following matrix.

$$R = \begin{array}{c} \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{array} \begin{array}{c} \begin{array}{cccc} y_1 & y_2 & \cdots & y_m \end{array} \\ \begin{bmatrix} \mu_R(x_1,y_1) & \mu_R(x_1,y_2) & \cdots & \mu_R(x_1,y_m) \\ \mu_R(x_2,y_1) & \mu_R(x_2,y_2) & \cdots & \mu_R(x_2,y_m) \\ \vdots & \vdots & \ddots & \vdots \\ \mu_R(x_1,y_1) & \mu_R(x_1,y_2) & \cdots & \mu_R(x_n,y_m) \end{bmatrix} \end{array} \tag{6.19}$$

The above matrix is also called a *fuzzy matrix* and is fundamental to the entire operations on fuzzy reasoning.

Since a relation itself is a set, all operations on fuzzy sets and their properties are entirely extendible to fuzzy relations. For further details, the reader is referred to [20,21 23].

6.4 Compositions of fuzzy relations

In many applications *composition*, symbolised by the operator $\circ$, plays an important role. Composition is a binary operation and can operate on two fuzzy relations as well as fuzzy sets and relations. In the following, the general definition of composing two fuzzy relations is given.

Let R be a fuzzy relation on $X \times Y$ and S a fuzzy relation on $Y \times Z$. The composition of R and S is a fuzzy relation on $Y \times Z$ and is defined as:

$$Q = R \circ S \tag{6.20}$$

or in terms of its membership function:

$$\mu_Q(x,z) = \sup_{y \in Y} \left(\mu_R(x,y) * \mu_S(y,z) \right) \qquad x \in X, y \in Y, z \in Z \tag{6.21}$$

Above, "$*$" could be any operator from the set of triangular norms such as minimum (*min*) or algebraic product ($\wedge$) [22]. Expression (6.21) is also called *sup-star* composition in the literature.

The minimum operator (*min*) is the most commonly used T-norm (proposed by Zadeh [12]) to represent conjunctions in control applications throughout this thesis. Consequently, equation (6.21) transforms to the following expression.

$$\mu_Q(x,z) = \sup_{y \in Y} \min\left(\mu_R(x,y), \mu_S(y,z) \right) \qquad x \in X, y \in Y, z \in Z \tag{6.22}$$

Nevertheless, equations (6.20) and (6.21) reduce to the following forms, in the case when the composition operates on the combination of a fuzzy set and a fuzzy relation rather than solely on fuzzy relations.

$$B = A \circ R \tag{6.23}$$

$$\mu_B(y) = \sup_{x \in X} \min\left(\mu_A(x), \mu_R(x,y)\right) \qquad x \in X, y \in Y \quad \textit{and} \quad A \subset X, B \subset Y \tag{6.23}$$

In the latter equation, A and B are both fuzzy sets with B being the induced outcome of the composition operation defined on Y.

In software, compositions are commonly implemented as matrix operations which have the basic properties of matrix manipulations, but where algebraic operators are substituted by their set theoretic counterparts. The induction of new relations by means of composition is block-diagrammatically depicted in Figure 6.8.

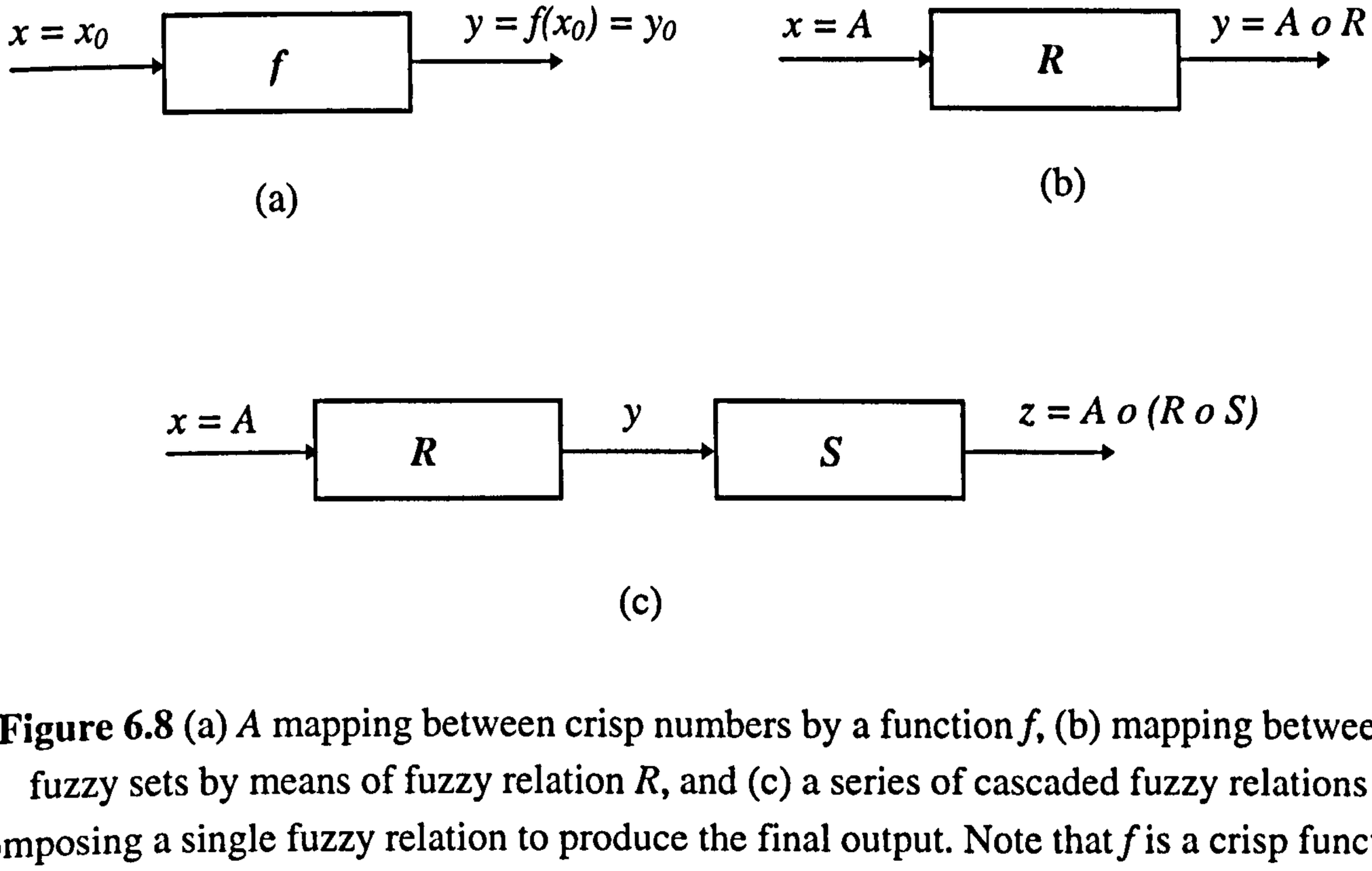


Figure 6.8 (a) *A* mapping between crisp numbers by a function *f*, (b) mapping between fuzzy sets by means of fuzzy relation *R*, and (c) a series of cascaded fuzzy relations composing a single fuzzy relation to produce the final output. Note that *f* is a crisp function with *R* and *S* both being fuzzy relations defined over multi-dimensional spaces.

6.5 Fuzzy reasoning and inference mechanism

A basic, though important, concept in fuzzy logic is *fuzzy propositions*. A fuzzy proposition represents, in general, a statement such as "*x* is *A*", where x is a linguistic

variable and A is a linguistic label, namely a fuzzy set defined over a universe of discourse. For example, the above proposition may take on specific forms such as:
"Temperature is high", "Speed is moderate", etc., where Temperature and Speed are fuzzy variables and high and moderate are their associated lables. Fuzzy propositions can be combined using the connectives AND or OR implemented by T-norms and T-conorms, respectively. Where the propositions are related to different universes, a logical connective results in a fuzzy relation. For example, in the following two-dimensional proposition

x is A **AND** y is B

where A and B are characterised by the membership functions $\mu_A(x)$ and $\mu_B(x)$, the proposition can then be represented by the following fuzzy relation:

$$\mu_R(x,y) = T\left(\mu_A(x), \mu_B(y)\right) \tag{6.24}$$

In the above equation, T can be any T-norm for modelling the connective AND.

6.5.1 Fuzzy rules and implication

To be able to perform fuzzy reasoning, there should exist fuzzy inference rules which in turn must be represented by an implication function. In binary logic, the implication function is defined by

$$A \rightarrow B \tag{6.24}$$

is the mathematical representation of the statement

IF A **THEN** B

In fuzzy logic, statements such as that above, where A and B are fuzzy propositions, are called *fuzzy if-then rules* or simply *fuzzy rules*. A is the antecedent and B is the consequent of the fuzzy rule. In most applications, the antecedents of fuzzy rules contain more than one proposition combined by the logical connective AND such as follows:

IF x is A **AND** y is B **THEN** z is C

The above statement can be represented by the fuzzy relation R, as shown below.

$$R = I\big(T(A,B),C\big) \tag{6.25}$$

Above, T is a T-norm to model the conjunction and I is an implication function to model the implication *if-then*. Equation (6.25) can be formulated as a fuzzy relation in terms of its fuzzy propositions, as shown below:

$$\mu_R(x,y,z) = I\Big(T\big(\mu_A(x),\mu_B(y)\big),\mu_C(z)\Big) \tag{6.26}$$

The implication function I is commonly denoted by $I(a,b)$ where $a,b \in [0,1]$. There exists a variety of representations for the implication function I in the literature. However, the most popular implications in engineering applications are the Mamdani's and Larsen's implication as shown below:

$$I(a,b) = \min(a,b) \quad \text{Mamdani} \tag{6.27}$$

$$I(a,b) = ab \quad \text{Larsen} \tag{6.28}$$

Since Mamdani's implication is computationally easier to implement and faster in run-time and also has been more widely used in control applications, implication operations on fuzzy relations in this thesis will adopt Mamdani's approach.

An explicit and direct application of the composition of fuzzy relations, namely the inference of a single fuzzy rule has been demonstrated. This was introduced by Zadeh [13] and was termed the *compositional rule of inference*. This has been expressed as a fuzzy rule in equation (6.26), but in the following section, this mechanism is discussed in terms of *modus ponens* and *modus tollens* of inference mechanism found in classical logic.

6.5.2 Inference mechanism

In classical logic, reasoning is based on modus ponens and modus tollens and is defined as follows:

- *Modus ponens:*

$$\begin{aligned} &\textit{Rule:} \quad A \rightarrow B \\ &\textit{Fact:} \quad A \\ &\text{----------} \\ &\therefore \quad B \end{aligned} \tag{6.29}$$

- *Modus tollens:*

$$\begin{aligned} &\textit{Rule:} \quad A \rightarrow B \\ &\textit{Fact:} \quad \neg B \\ &\text{----------} \\ &\therefore \quad \neg A \end{aligned} \tag{6.30}$$

In the preceding expressions, we interpret $A \rightarrow B$ as "if A is true, then B is true" where the operator $\rightarrow$ denotes implication.

In fuzzy logic, the aforementioned expressions are expanded to fuzzy sets and termed as the *generalised modus ponens* (GMP) and the *generalised modus tollens* (GMT) by Zadeh [13] and defined as:

- *GMP:*

$$\begin{aligned} &\textit{Rule:} \quad \textit{if } x \textit{ is } A \textit{ then } y \textit{ is } B \\ &\textit{Fact:} \quad x \textit{ is } A' \\ &\text{----------} \\ &\therefore \quad y \textit{ is } B' \end{aligned} \tag{6.31}$$

- *GMT:*

$$\begin{array}{l} \textit{Rule: if } x \textit{ is } A \textit{ then } y \textit{ is } B \\ \textit{Fact: } y \textit{ is } B' \\ \hline \therefore \quad x \textit{ is } A' \end{array} \tag{6.32}$$

where A, A', B and B' are fuzzy sets. A significant difference between GMP and the conventional modus ponens is that fuzzy sets A and A' in the first and second premise of expression (6.31) do not need to be precisely the same whereas in binary logic they must be identical. This has given rise to an alternative terminology for fuzzy reasoning, namely *approximate reasoning* [10,16].

In most practical applications, the antecedent of a fuzzy rule is multitudinous in the number of propositions to assure flexible reasoning. In the following, we consider two somewhat different approaches to the formulation of fuzzy reasoning. The first approach implements fuzzy reasoning in a rather graphical manner (Mamdani's method) whereas the second implementation which was first introduced by Zadeh [13] is based on the compositional rule of inference. This converts entire fuzzy rules to corresponding fuzzy relations which are then aggregated and utilised to induce new knowledge by means of matrix operations.

Mamdani's method of inference

In order to describe the underlying mechanism of Mamdani's method, consider the following simple rule base consisting of two fuzzy rules.

$\Re_1$: **IF** x is A_1 **AND** y is B_1 **THEN** z is C_1

$\Re_2$: **IF** x is A_2 **AND** y is B_2 **THEN** z is C_2

Above, x, y and z are fuzzy linguistic variables with A_i, B_i and C_i their associated fuzzy sets, where $x \in X$, $y \in Y$ and $z \in Z$. Mamdani's reasoning process is composed of three steps to induce new knowledge which is, in general, represented as a fuzzy set. These steps are discussed below and illustrated in Figure 6.9.

Step 1: Calculate the *firing strength* of each individual rule, α_i, for given inputs.

Assuming, the inputs to the system are x_0 and y_0, as shown in Figure 6.9, the associated firing strengths are:

$$\alpha_1 = \min\left(\mu_{A_1}(x_0), \mu_{B_1}(y_0)\right) \tag{6.33}$$

$$\alpha_2 = \min\left(\mu_{A_2}(x_0), \mu_{B_2}(y_0)\right) \tag{6.34}$$

α_1 and α_2 are shown in Figures 6.9 (a) and (d), respectively.

Step 2: Apply the firing strength α_i to the consequence of fuzzy rule $\Re_i$, to calculate the intermediate output, $\mu_{C_i'}(z)$, for the number of given fuzzy rules on the rule list. This operation truncates the output fuzzy set to a fuzzy sub-area whose height equals α_i. These areas are shown shaded in Figures 6.9 (c) and (f) and are represented by the following equations, respectively.

$$\mu_{C_1'}(z) = \min\left(\alpha_1, \mu_{C_1}(z)\right) \tag{6.35}$$

$$\mu_{C_2'}(z) = \min\left(\alpha_2, \mu_{C_2}(z)\right) \tag{6.36}$$

Step 3: Aggregate the intermediate fuzzy outputs associated with each rule to the final fuzzy output set, $\mu_C(z)$. Mamdani's approach implements this operation using a fuzzy logical OR (max-operation) to compile the final fuzzy output set as follows:

$$\mu_C(z) = \max\left(\mu_{C_1'}(z), \mu_{C_2'}(z)\right) \tag{6.37}$$

Equation (6.37) is graphically shown as the shaded concave area in Figure 6.9 (g). This is a fuzzy set describing the induced data.

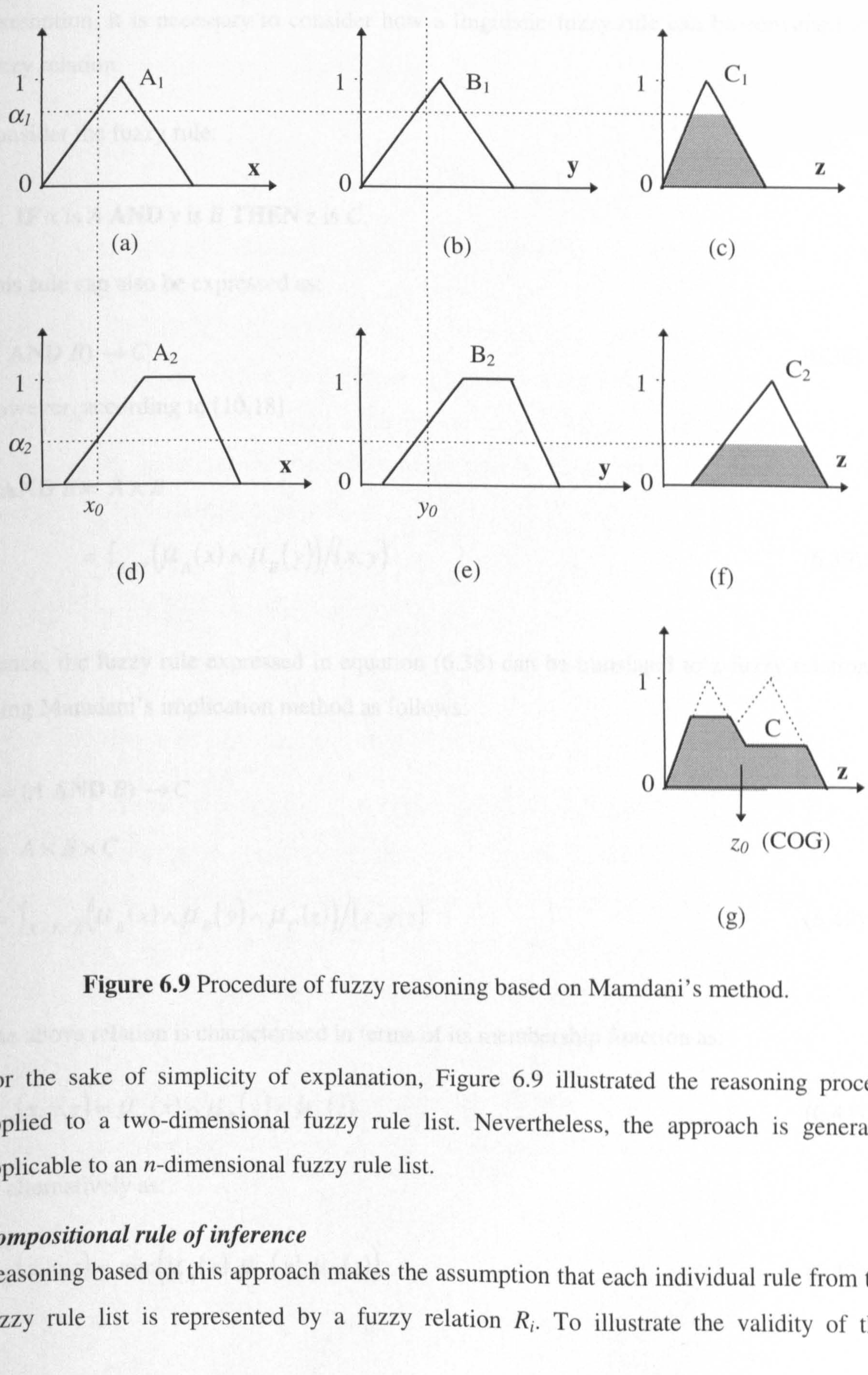


Figure 6.9 Procedure of fuzzy reasoning based on Mamdani's method.

For the sake of simplicity of explanation, Figure 6.9 illustrated the reasoning process applied to a two-dimensional fuzzy rule list. Nevertheless, the approach is generally applicable to an n-dimensional fuzzy rule list.

Compositional rule of inference

Reasoning based on this approach makes the assumption that each individual rule from the fuzzy rule list is represented by a fuzzy relation R_i. To illustrate the validity of this

assumption, it is necessary to consider how a linguistic fuzzy rule can be converted to a fuzzy relation.

Consider the fuzzy rule:

$\Re$: **IF** x is A **AND** y is B **THEN** z is C.

This rule can also be expressed as:

$$(A \textbf{ AND } B) \rightarrow C \tag{6.38}$$

However, according to [10,18]

$$A \textbf{ AND } B = A \times B$$

$$= \int_{X \times Y} \left(\mu_A(x) \wedge \mu_B(y) \right) / (x, y) \tag{6.39}$$

Hence, the fuzzy rule expressed in equation (6.38) can be translated to a fuzzy relation R using Mamdani's implication method as follows:

$$R = (A \textbf{ AND } B) \rightarrow C$$

$$= A \times B \times C$$

$$= \int_{X \times Y \times Z} \left(\mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z) \right) / (x, y, z) \tag{6.40}$$

The above relation is characterised in terms of its membership function as:

$$\mu_R(x, y, z) = \mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z) \tag{6.41}$$

or alternatively as:

$$\mu_R(x, y, z) = \min\left(\mu_A(x), \mu_B(y), \mu_C(z) \right) \tag{6.42}$$

Now that we are able to capture a fuzzy linguistic rule in a fuzzy relation R, the compositional rule of inference is defined as [13]:

$$C' = A'o(B'oR) = B'o(A'oR) \tag{6.43}$$

This equation is the mathematical representation of the GMP expressed in (6.31) expanded to three-dimensional space. This means, having the knowledge of the input-output relationship, R, new information, C', can be induced when two inputs, A' and B' are presented to this inference system. Equation (6.42) demonstrates that the resultant fuzzy relation is three-dimensional when a fuzzy rule has two input variables. Each individual element of the three-dimensional membership function of the fuzzy relation is calculated according to the following scheme:

$$\begin{cases} \mu_R(x_i, y_j, z_k) = \mu_A(x_i) \wedge \mu_B(y_j) \wedge \mu_C(z_k) \\ i, j, k = 1,2,\ldots,n \end{cases} \tag{6.43}$$

where n is the number of elements of each fuzzy set.

The fuzzy relation R in Equation (6.43) is the representation of a single fuzzy rule. However, in most applications the reasoning is based on a set of fuzzy rules forming the fuzzy *knowledge base* (KB) or the fuzzy rule list of a system. That means, generally, relation R represents a finite set of fuzzy rules each expressed as a fuzzy relation R_i. These relations are then subsumed with a representative relation R. This operation is called *aggregation.*

Mamdani's inference mechanism implements the aggregation operation as a logical OR, which is the union of n fuzzy relations. The aggregated relation R is then defined as follows:

$$R = R_1 \cup R_2 \cup \ldots \cup R_n = \bigcup_{i=1}^{n} R_i \tag{6.44}$$

For further details and also other aggregation operators, the reader is referred to [10,22,23]. Compositional rule of inference is then carried out following the three steps listed below:

Step 1: Convert each individual linguistic rule to a corresponding fuzzy relation.

Step 2: Aggregate the fuzzy relation using a suitable aggregation operation.

Step 3: Compose the available data (usually a fuzzy set) with the aggregated fuzzy relation to induce new data.

Having illustrated various ways of reasoning about fuzzy concepts, a pertinent question is how the performances of the various approaches compare. Since there exists no scientific proof or evidence that one method is superior to the other or vice versa, one should perhaps choose the method which suits best the objectives of the work. It has generally been found that Mamdani's approach to reasoning is the most suitable when the number of rules is relatively small [10], but the choice generally depends on empirical results.

6.6 Fuzzy logic control (FLC)

In the previous sections, was shown that the compositional rule of inference [12] performs a non-linear mapping in that it transforms a multi-dimensional fuzzy vector (whose elements are fuzzy sets) into a single fuzzy vector (the output space). In this operation, the fuzzy relation describing the input/output relationship is crucial and central to this operation. In most practical applications, this relationship is the representative of a finite number of fuzzy rules composing the fuzzy rule base or fuzzy knowledge base (KB) of a fuzzy system.

Fuzzy reasoning was first applied by Mamdani [14] to control a simple experimental control system. In this approach, the fuzzy output vector was converted into a scalar (rather than crisp) number which was then used to control the plant. The output of the plant (control parameter) which was a crisp number needed to be converted into a fuzzy set to perform fuzzy reasoning. The KB of his system consisted of a number of heuristic rules extracted from the knowledge of a human who was expert in controlling such a plant.

This section describes an FLC system in terms of its constituent units and the way they are interconnected to form a working control system.

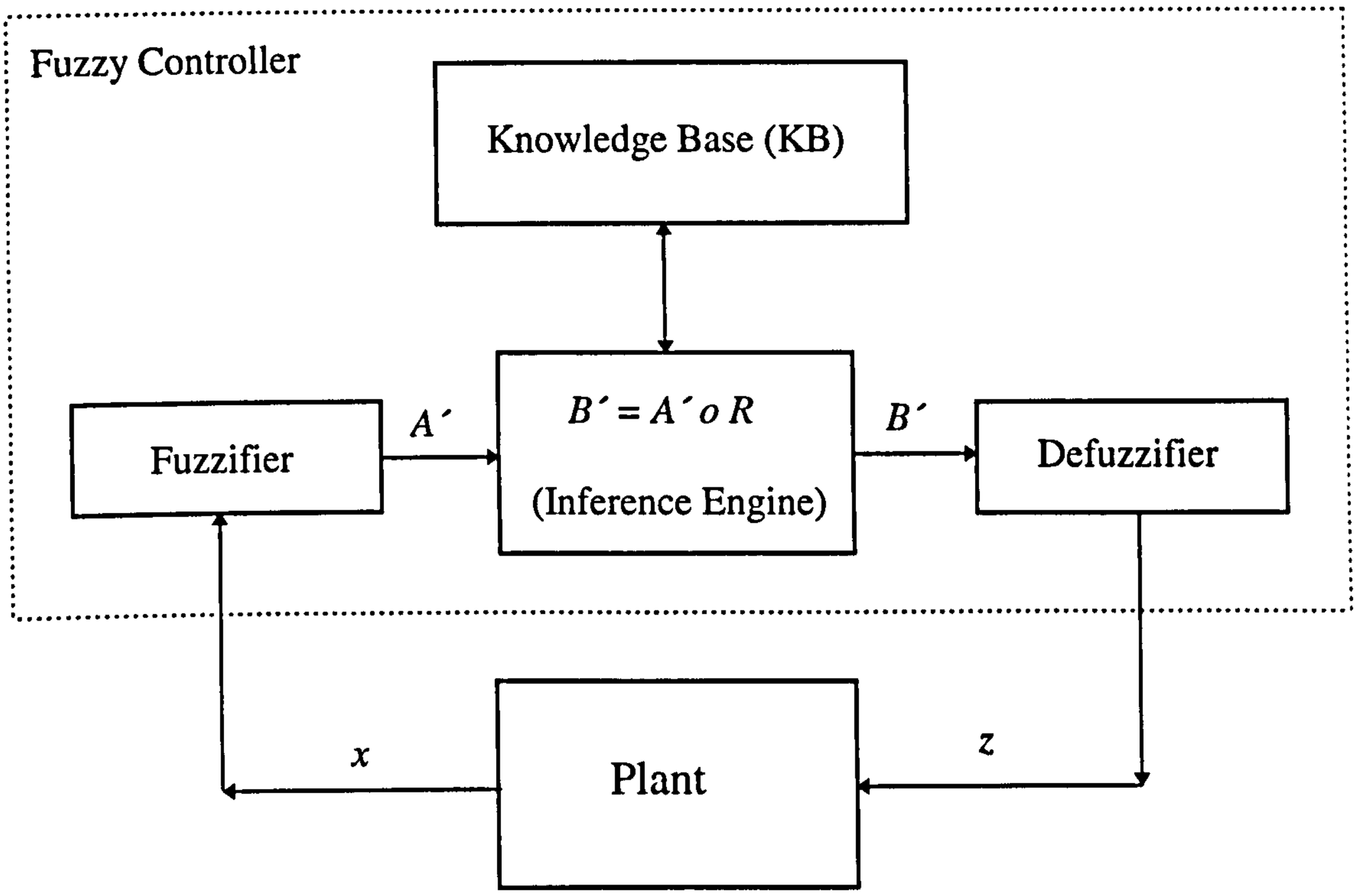


Figure 6.10 Block diagrammatic representation of a fuzzy logic system (FLS) incorporating a fuzzy controller.

As depicted in Figure 6.10, the fuzzy controller is composed of four modules:

- Fuzzifier
- Inference engine
- Knowledge base (KB) or fuzzy rules
- Defuzzifier

The inference engine performs fuzzy reasoning on the input data, A', to generate an output, B', by means of linguistic knowledge formulated in the fuzzy rules. KB is, technically speaking, the control algorithm in the from of IF-THEN fuzzy rules which are in turn extracted from human domain expertise. Fuzzification and defuzzification convert crisp numbers to fuzzy sets and vice versa. Both operations explained in detail in the following sub-sections.

6.6.1 Fuzzification

The process of mapping input data into fuzzy sets is called *fuzzification* and defined as:

$$A' = fuzzifier\ (x_0) \tag{6.45}$$

where x_0 is the crisp input, A' is the associated output and the *fuzzifier* operator is a mapping function. In fuzzy control applications, the input data are crisp numbers (sensor values) and fuzzification involves matching the sensor measurements against the associated fuzzy sets [19,22,24] to obtain the degree of membership in a fuzzy set. This is demonstrated in Figure 6.11.

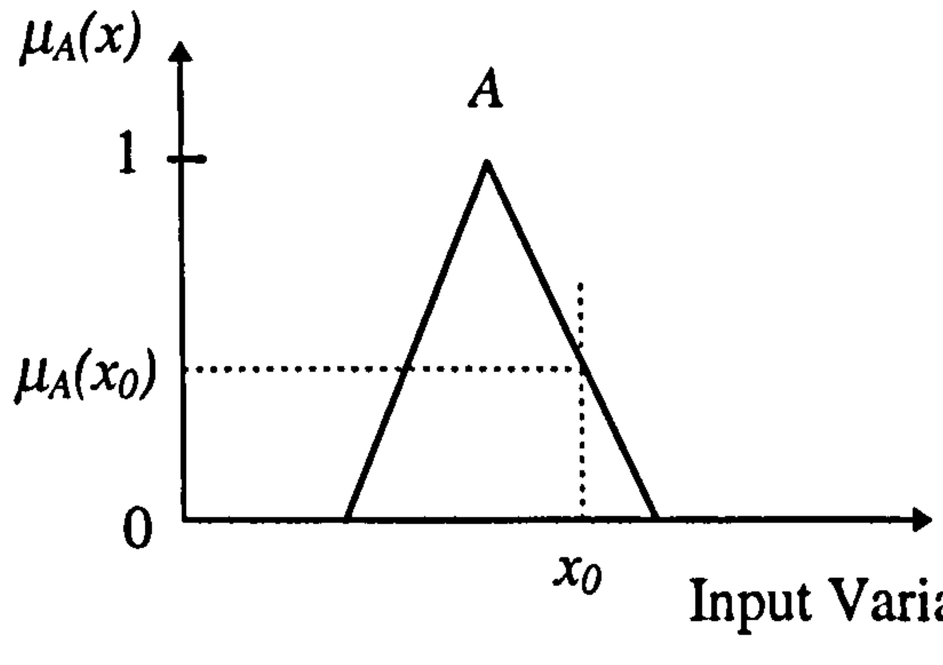


Figure 6.11 Mapping the crisp value x_0 to its membership degree, $\mu_A(x_0)$, by means of the membership function $\mu_A(x)$.

Figure 6.11 states that a certain crisp value x_0 from a universe of discourse X is associated with the fuzzy set A with the degree of membership $\mu_A(x_0)$.

6.6.2 Defuzzification

As depicted in Figure 6.10, the output of the inference unit is the fuzzy set B' which represents the possibility distribution of the output parameter. However, the control action applied to the plant needs to be a crisp value, for example z_0. The operation that determines a representative for the output set is called *defuzzification* and is defined as:

$$z_0 = defuzzifier\ (B') \tag{6.46}$$

where *defuzzifier* can be any defuzzification operator. There exists a number of different defuzzification approaches which have been suggested by different researchers, but the one most widely used in control applications is the centre of gravity (COG) method.

The centre of gravity (COG) method

Let the possibility distribution of the output fuzzy set be defined by the membership function $\mu_C(z)$. The COG method calculates literally the geometrical centre of this set which is a crisp value in the universe of definition. Having a discrete universe of discourse Z, the COG method is calculated as follows:

$$z_0 = \frac{\sum_{i=1}^{n} z_i \, \mu_C(z_i)}{\sum_{i=1}^{n} \mu_C(z_i)} \tag{6.47}$$

Above, z_0 is the crisp control action and n is the number of quantisation levels of the output. The method is graphically shown in Figure 6.12 which is adapted from the reasoning process described in section 6.5.2.1.

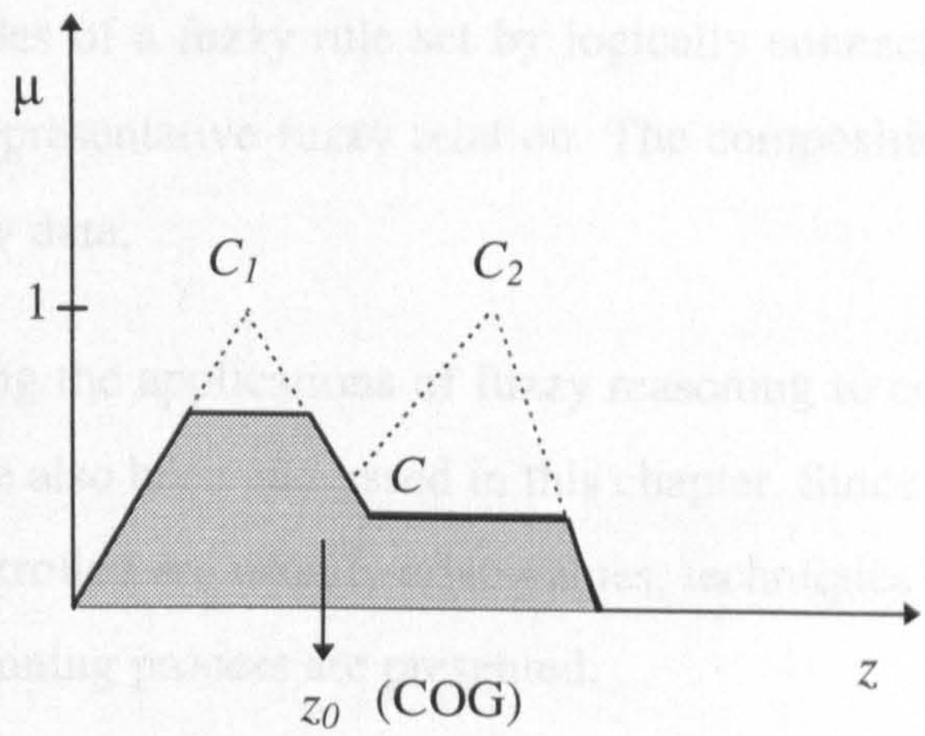


Figure 6.12 Producing a crisp control action, z_0, by applying the centre of gravity (COG) method to defuzzify the output fuzzy set C.

For further material on this subject and also other defuzzification strategies the reader is referred to [17,19,22,24].

6.7 Summary

The extension of the classical binary logic to multi-valued logic and the ability to reason with vagueness and uncertainty constitute fuzzy logic (FL). FL attempts to mimic the way humans think and reason with linguistic concepts. Linguistic variables are the basic ingredients of fuzzy propositions and fuzzy propositions of different universes are combined to form the antecedents (commonly in multitude) and the consequences of fuzzy *if-then* rules.

Fuzzy rules are the verbal representations of fuzzy relations describing the correlation, association or the interaction of fuzzy sets. Fuzzy relations are implication functions which can be implemented in several ways depending on their suitability for the application under consideration.

Inference is used to obtain new knowledge by composing fuzzy relations with the data being used. The inference mechanism is usually performed on a set of parallel fuzzy rules. Two approaches to the inference can be distinguished; namely local and global inference [17]. The former performs inference with individual fuzzy rules and combines the intermediate outcomes to form the final result. This method is known as Mamdani's direct method of inference, which can readily be illustrated graphically. The latter, however, aggregates the entire rules of a fuzzy rule set by logically connecting their corresponding fuzzy relations into a representative fuzzy relation. The compositional rule of inference is then applied to infer new data.

The principles underlying the applications of fuzzy reasoning to control problems, namely fuzzy logic control, have also been addressed in this chapter. Since input/output parameters of the process to be controlled are usually crisp values, techniques to fuzzify and defuzzify these values for the reasoning process are presented.

The material discussed in this chapter provides the mathematical foundations and theoretical knowledge of FL to aid the understanding of the next chapter which applies FL to build a hybrid learning system.

References

[1] Fuzzy Sets and Systems, *An International Journal*, North-Holland Publishing Company, Amsterdam.

[2] Fuzzy Systems, *IEEE Transactions*.

[3] L.A. Zadeh and J. Kacprzyk, Eds., "Fuzzy Logic for the Management of Uncertainty", John Wiley and sons, ISBN: 0-471-54799-9.

[4] A. Kaufmann, "Introduction to the Theory of Fuzzy Subsets", Vol. I, Academic Press, ISBN: 0-12-402301-0.

[5] E.H. Mamdani and B.R. Gaines, Eds., "Fuzzy Reasoning and its Applications", Academic Press, 1981, ISBN: 0-12-467750-9.

[6] B. Kosko, "Fuzzy Thinking", Harper Collins Publishers, 1993, ISBN: 0 00 255352 X.

[7] R.R. Yager, S. Ovchinnikov, R.M. Tong and H.T. Nguyen, Eds., "Fuzzy Sets and Applications, Selected Papers by L.A. Zadeh", John Wiley and Sons, Inc., 1987, ISBN: 0-471-85710-6.

[8] D. McNeill and P. Freiberger, "Fuzy Logic", Simon & Chuster, 1993, ISBN: 0-671-73843-7.

[9] J. Yen, R. Langari and L.A. Zadeh, "Industrial Applications of Fuzzy Logic", *IEEE Press*, 1995, ISBN: 0-7803-1048-9.

[10] K. Tanaka, Translated by T. Niimura, "An Introduction to Fuzzy Logic for Applications", Springer Verlag, 1997, ISBN: 0-387-94807-4.

[11] G.J. Klir and T.A. Folger, "Fuzzy Sets, Uncertainty and Information", Prentice Hall, 1988, ISBN: 0-13-345984-5.

[12] L.A. Zadeh, "Fuzzy Sets", *Information and Control*, Vol. 8, New York: Academic Press, 1965, pp. 338-352, Fuzzy Sets and Applications: Selected Papers by L.A. Zadeh, (Edit.) R.R. Yager, S. Ovchinnikov, R.M. Tong and H.T. Nguyen, pp. 29-44, ISBN 0-471-85710-6.

[13] L.A. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Processes", *IEEE Transactions on Systems, Man and Cybernetics*, SMC-3 (1973), pp.28-44, Fuzzy Sets and Applications: Selected Papers by L.A. Zadeh, (Edit.) R.R.

Yager, S. Ovchinnikov, R.M. Tong and H.T. Nguyen, pp. 105-146, ISBN 0-471-85710-6.

[14] E.H. Mamdani, "Applications of Fuzzy Algorithms for Control of Simple Dynamic Plant", *Proceedings of IEE, Control and Science,* Vol. 121, No. 12, December 1974, pp. 1585-1588.

[15] J. Yen, R. Langari and L.A. Zadeh (Eds.), "Industrial Applications of Fuzzy Logic and Intelligent Systems", *IEEE Press*, ISBN 0-7803-1048-9.

[16] C.C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller-Part I", *IEEE Transactions on Systems, man and Cybernetics*, Vol. 20, No. 1, March./April 1990.

[17] R. Jager, "Fuzzy Logic in Control", *PhD Thesis*, 1995, ISBN 90-9008318-9.

[18] J.S.R. Jang, C.T. Sun and E. Mizuyani, "Neuro-Fuzzy and Soft Computing, A Computational Approach to Learning and Machine Intelligence", Prentice Hall Upper Saddle River, NJ 07458, 1997, ISBN 0-13-261066-3.

[19] R. Fuller, "Lecture Notes on Fuzzy Decision Making", Available to download from WWW at: http://www.tucs.abo.fi/courses/95-96/material/fuzzydec.html.

[20] G.J. Klir and B. Yuan, "Fuzzy Sets and Fuzzy Logic", Theory and Applications, Prentice Hall, ISBN: 0-13-101171-5.

[21] G. Klir and T.A. Folger, "Fuzzy Sets, Uncertainty and Information", Prentice Hall, 1988, ISBN 0-13-345984-5 025.

[22] C.C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller-Part II", *IEEE Transactions on Systems, man and Cybernetics*, Vol. 20, No. 1, March./April 1990.

[23] T.J. Ross, "Fuzzy Logic with Engineering Applications", McGraw-Hill, Inc., ISBN 0-07-053917-0.

[24] H.R. Berenji, "An Introduction into Fuzzy Logic Applications in Intelligent Systems", R.R. Yager an L.A. Zadeh (Eds.), Kluwer Academic publishers, ISBN: 0-7923-9191-8.

Chapter 7

Hybrid Learning: Self-organising Fuzzy Decision Trees Applied to Robotic Environments

So far as the laws of mathematics refer to reality, they are not certain.
And so far as they are certain, they do not refer to reality.

Albert Einstein
Theoretical Physist and Nobel laureate
"Goemetrie und Erfahrung", Lecture to Prussian Academy, 1921

With the aid of the theory presented in the preceding chapter, this chapter introduces a hybrid learning technique in which fuzzy logic (FL) is incorporated into DTs to produce a multi-strategy learning system incorporating the benefits of the individual

techniques. The resulting fuzzy decision trees (FDTs) are demonstrated in their application to navigation and, in particular, to control a robot in dynamic and unstructured environments.

Section 7.1 outlines the objectives of fusing the two methodologies in the context of robot navigation. Section 7.2 introduces the physical nature and sensor configuration of the simulated mobile platform used for experimental results. The remaining sections, namely 7.3 to 7.13 present the author's original work (except most of section 7.6 which has been didactically placed in this chapter) and develop the application of FDTs to robot control.

7.1 Objectives

Chapter 5 discussed a control architecture in which a hierarchy of self-organising decision trees (DTs) was utilised to navigate a robot in unstructured environments. The system architecture combined the characteristics of behaviour-based systems [1,2,3] and reactive systems [4,5] to form a hybrid architecture. The resultant system was able to respond to immediate robot perceptions (reactivity) to avoid obstacles while incorporating a distributed mechanism of behaviour execution. Navigation experiments, however, revealed that the robot performs oscillatory movements when approaching and following long walls. This problem is inherent to reactive systems and is produced as the result of conflicting behaviours which are also reported in [4].

As discussed in the preceding chapter, the compositional rule of inference is based on a finite set of contributive fuzzy rules populating the fuzzy rule list. The idea of partial contribution of fuzzy rules which underlies the aggregation operation and also the qualitative reasoning inherent to fuzzy logic [6] together provide flexible control and allow the merging of conflicting behaviours [7,8]. This chapter is devoted to test the validity of the foregoing statement by applying fuzzy reasoning to the inductive symbolic knowledge encoded in the DTs to synthesise fuzzy control rules. The new hybrid learning system, namely fuzzy decision trees should similarly provide robust reasoning and smooth control in the presence of partial information and uncertainty in sensory data [9,10,11,12,13,14]. These are the two primary objectives of constructing fuzzy-based systems, in particular, the hybrid system addressed in this chapter.

In an attempt to automate the process of fuzzy rule generation, Hsu *et al* in [15] and Hall and Lande in [16] applied decision tree learning. In both approaches a crisp decision tree was produced which was then transformed into a set of fuzzy rules. However, compared to these two methods, the approach described in this chapter is a significant improvement in two respects:

- In addition to automatic *fuzzy rule* generation, the approach is incremental and capable of automatic *fuzzy data* acquisition which imparts the capability of *on-line* fuzzy-rule learning to the algorithm.
- Both data acquisition and rule generation are performed automatically and operate on fuzzy data without the need for intermediate transformation.

This work uses the problem of behaviour learning and robot path planning as a testbed for FDTs. These are applied to navigate a robot in unseen and unstructured environments. This chapter relies on the methodology and architecture of fuzzy logic control systems and presents the analysis and development of FDTs in the automatic synthesis of linguistically formulated fuzzy rules. Figure 7.1 is the schematic representation of the control architecture used to demonstrate the subsequent development steps (modules 1 to 6) from crisp data acquisition (module 1) to FDT generation (module 4). The results of on-line experiments conducted on this controller are presented in this chapter.

7.2 The mobile platform and sensor arrangements

The learning algorithm to be described is implemented as an extension to the mobile platform shown in Figure 7.2, namely the simulated counterpart of the real miniature robot Khepera [17], Figure 7.3. Khepera has two drive wheels and an array of eight infra-red circumferential sensors to detect the proximity of objects. Robot positioning, which is a measure of target location, is performed by wheel encoded dead reckoning. The input variables of the system consist of the six frontal sensors, the wall location in situations where the robot follows long walls and the relative location of the target is derived from the instantaneous robot position (see chapter 4 for the derivation method). Drive wheels are steered differentially to generate an appropriate control action.

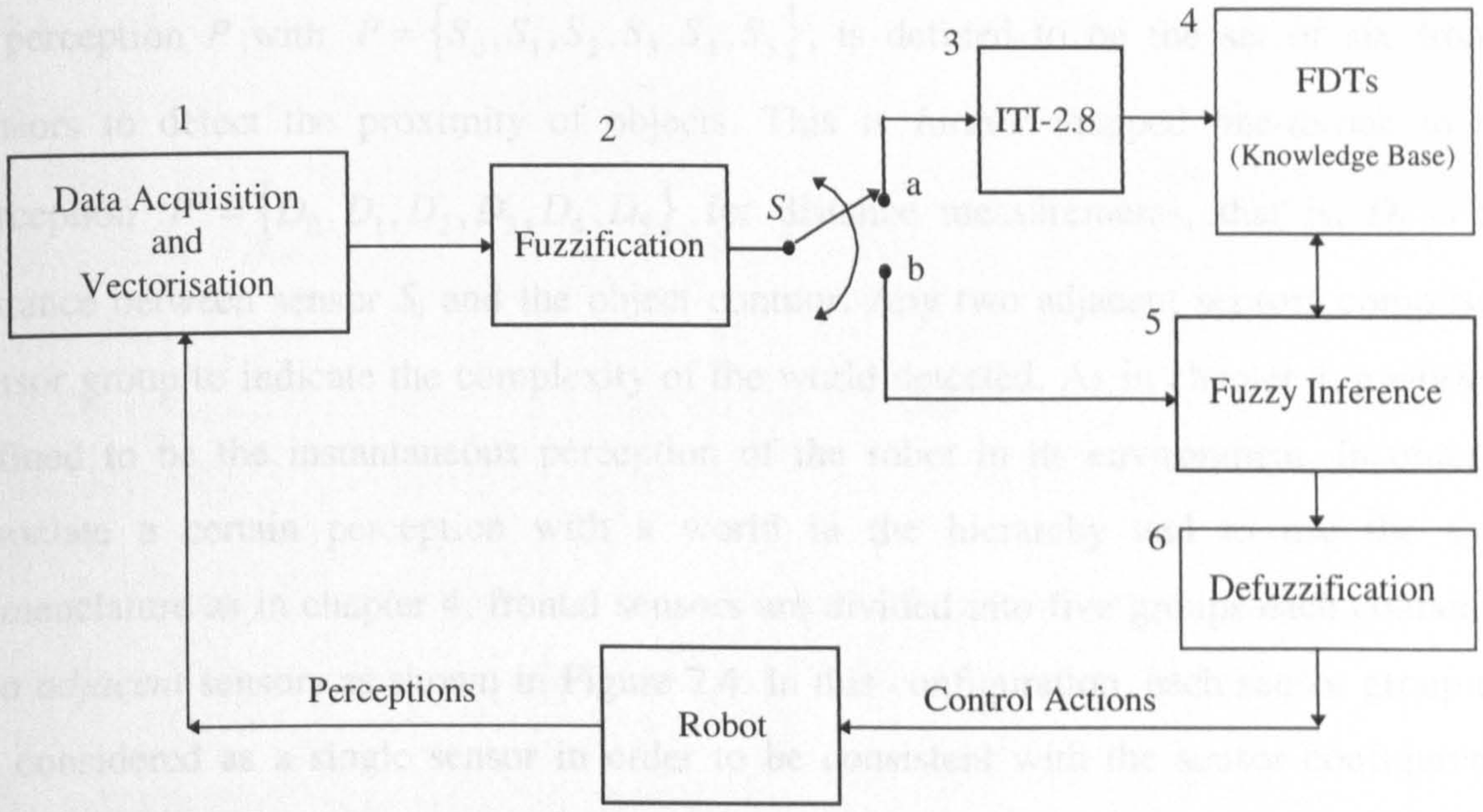


Figure 7.1 The block diagram representation of the control system in terms of individual working units. The numbers associated with the units are used for reference in the text.

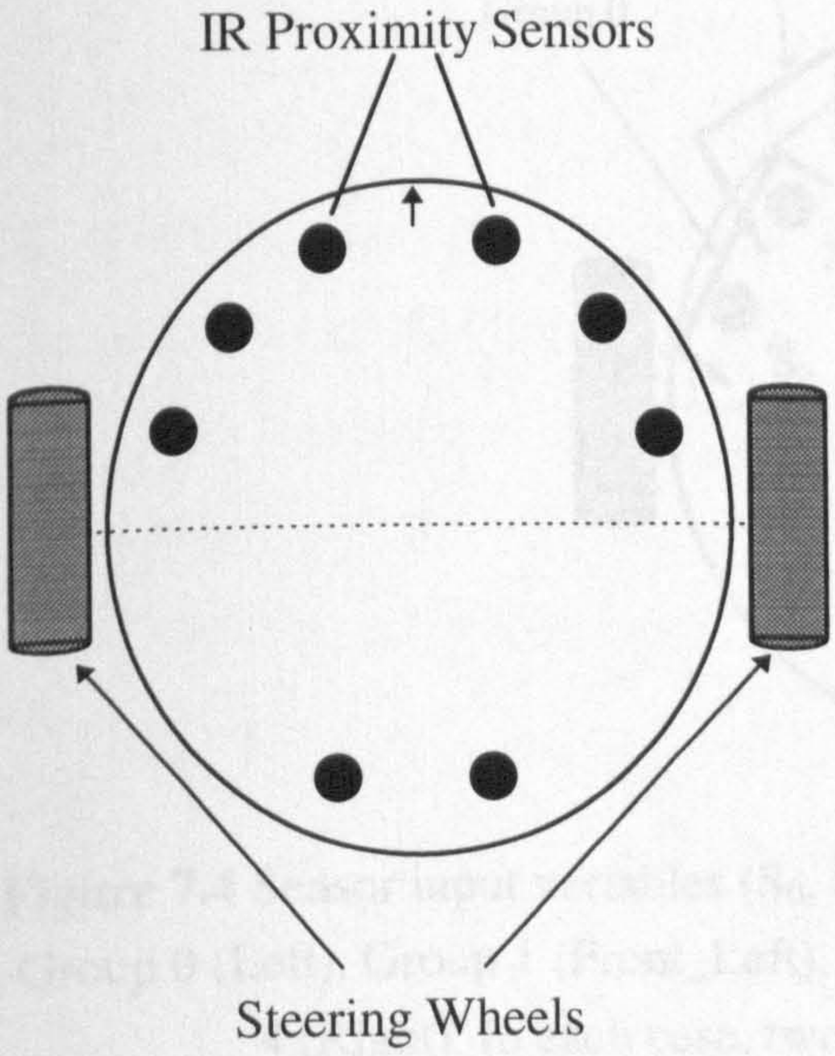


Figure 7.2 The top-view of the simulated Khepera with infra-red proximity sensors.

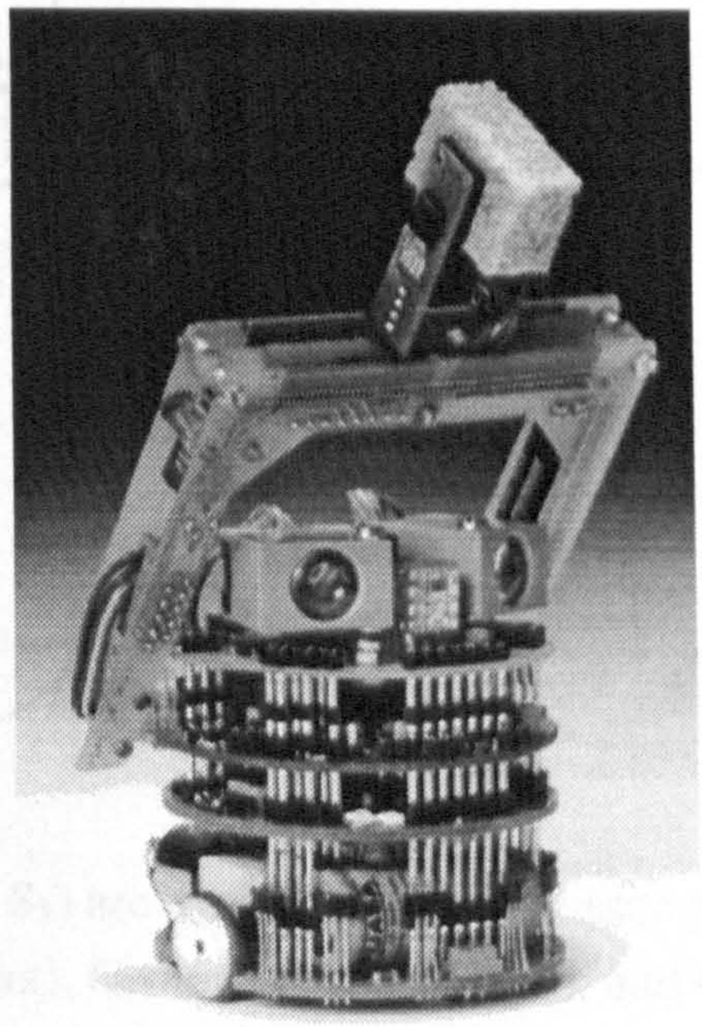


Figure 7.3 Khepera robot equipped with gripper and vision (Photo by Alain Herzog, Courtesy of EPFL Laboratory, Lausanne)

A perception P with $P = \{S_0, S_1, S_2, S_3, S_4, S_5\}$, is defined to be the set of six frontal sensors to detect the proximity of objects. This is further mapped one-to-one to the perception $P' = \{D_0, D_1, D_2, D_3, D_4, D_5\}$ for distance measurements, that is, D_i is the distance between sensor S_i and the object contour. Any two adjacent sensors comprise a sensor group to indicate the complexity of the world detected. As in chapter 4, a *world* is defined to be the instantaneous perception of the robot in its environment. In order to associate a certain perception with a world in the hierarchy and to use the same nomenclature as in chapter 4, frontal sensors are divided into five groups each containing two *adjacent* sensors as shown in Figure 7.4. In this configuration, each sensor group can be considered as a single sensor in order to be consistent with the sensor configuration introduced in chapter 4.

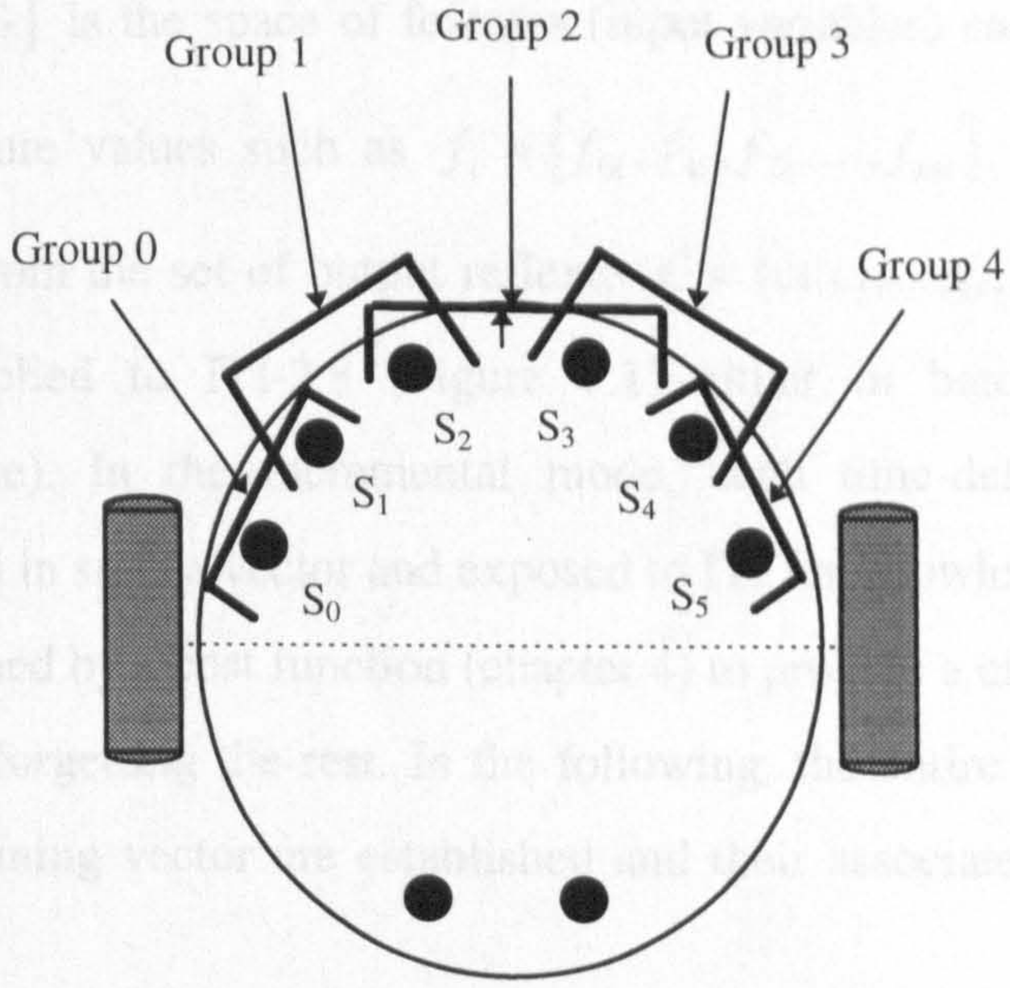


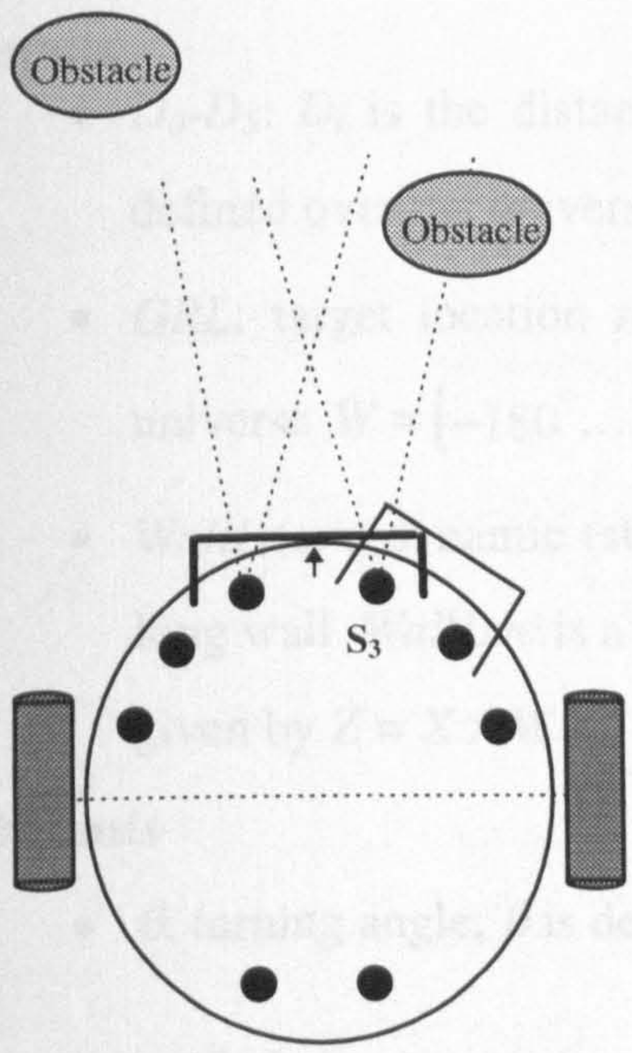
Figure 7.4 Sensor input variables (S_0, S_1, S_2, S_3, S_4, S_5) are divided into five sensor groups: Group 0 (Left), Group 1 (Front_Left), Group 2 (Front), Group 3 (Front_Right), and Group 4 (Right). In each case, two adjacent sensors form one sensor group.

The number of *sensor groups* indicating that obstacles are present determines the world complexity and their nomenclature in the hierarchy. This means that, for example, if only one sensor group detects an obstacle at a certain distance from the robot that entire perception is classified as w_1 to designate world one, and w_0 specifies that the instantaneous environment (world) has no obstacles. In a sensor group, the state of either sensor determines the state of the group whether the sensor group detects the presence of obstacles or no obstacles are apparent, and this is demonstrated in Figure 7.5.

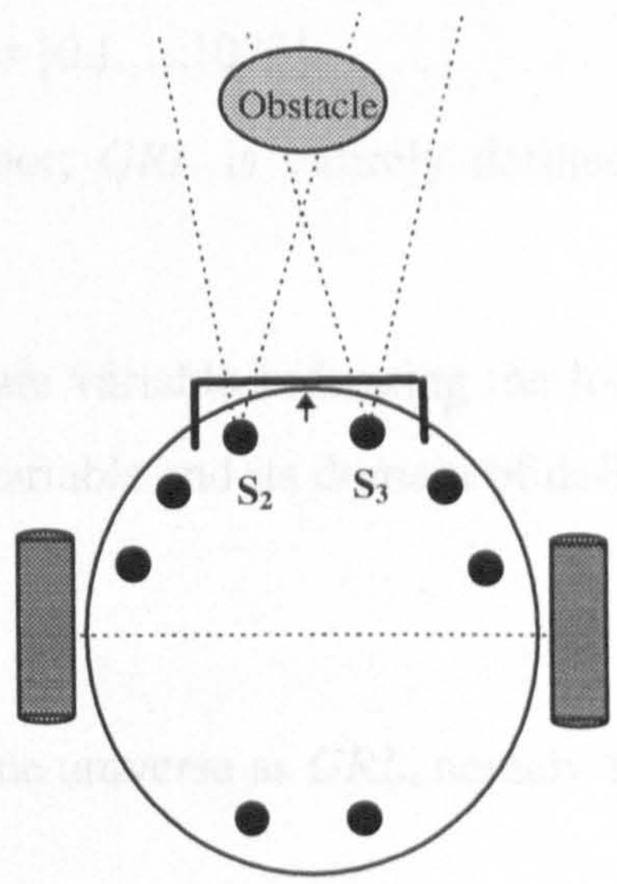
Reignier [4] took a similar approach to grouping proximity sensors in order to reduce the dimensionality of the perceptual space and to locate obstacle positions.

7.3 Automatic fuzzy data acquisition

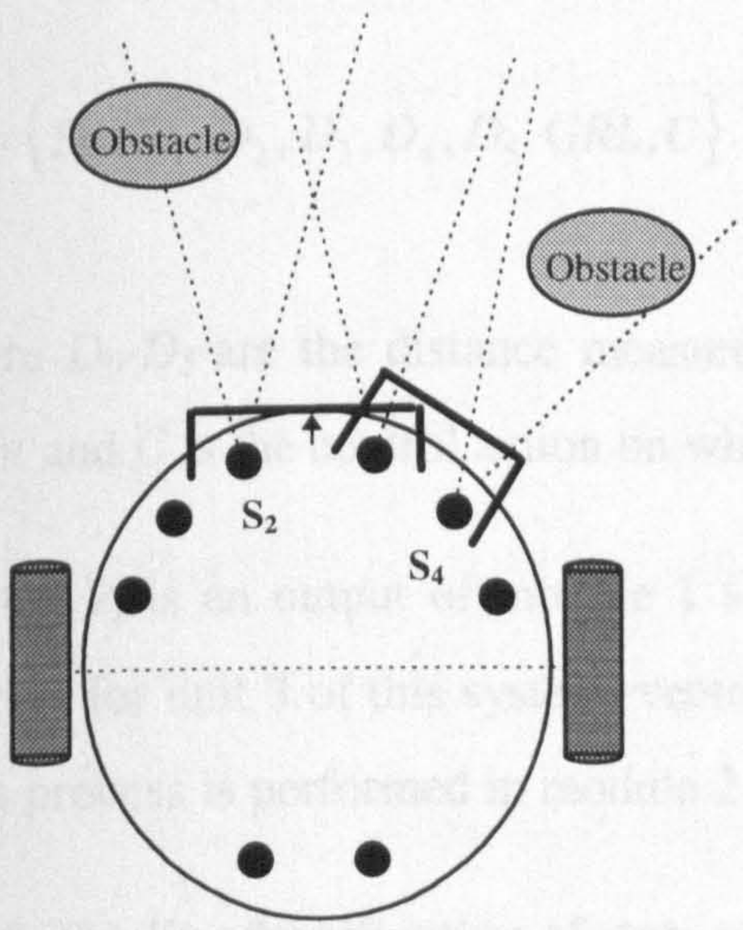
To acquire knowledge by means of DTs, concept learning requires formatted data as vector entities such as: $V = \{f_0, f_1, \ldots, f_i, \ldots, f_n, c_j\}$, where f_i is a feature with $f_i \in F$ and $F = \{f_0, f_1, \ldots, f_i, \ldots, f_n\}$ is the space of features (input variables) each being defined on a unique space of feature values such as $f_i = \{f_{0i}, f_{1i}, f_{2i}, \ldots, f_{mi}\}$. c_j is a class (control action) with $c_j \in C$ from the set of output reflexes $C = \{c_1, c_2, \ldots, c_j, \ldots, c_n\}$. These training vectors can be supplied to ITI-2.8 (Figure 7.1) either in batch-mode (off-line) or incrementally (on-line). In the incremental mode, each time-delayed robot rewarded experience is encoded in such a vector and exposed to ITI for knowledge extraction. Action assessment is performed by a cost function (chapter 4) to provide a class c_j by remembering useful actions while forgetting the rest. In the following, the entire input/output variables which compose a training vector are established and their associated domains of validity are defined.



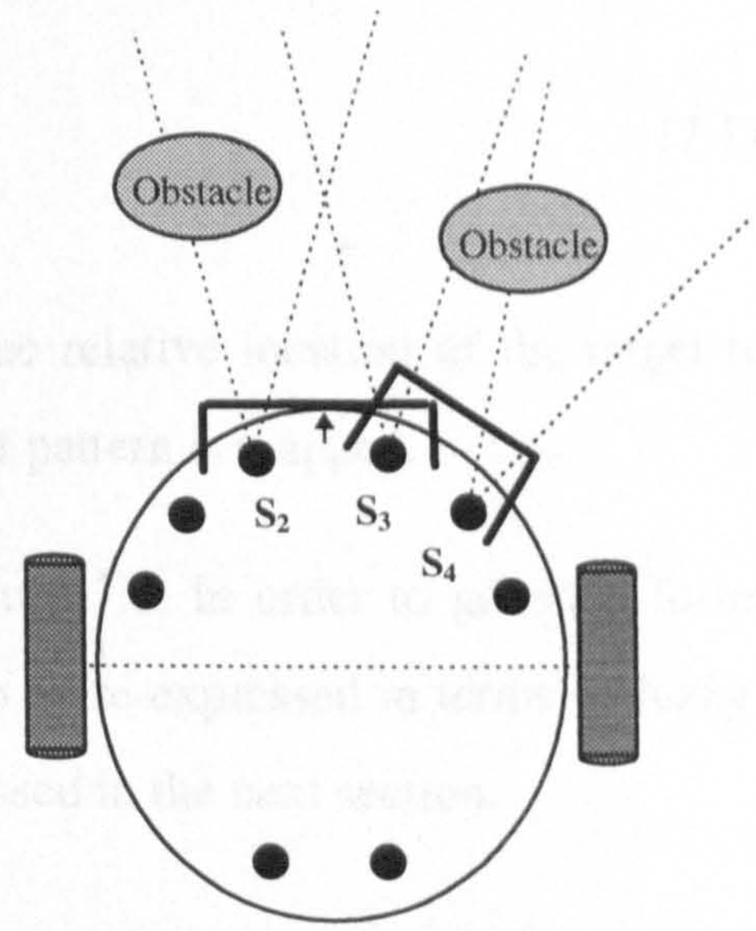
(a)



(b)



(c)



(d)

Figure 7.5 Different perceptual situations: (a) Two separate groups in which only one sensor detects obstacles is recorded as w_1 (the second obstacle is assumed to be outside the range of detection of the sensors). (b) Two adjacent sensors S_2 and S_3 detecting obstacles and represented by *one group* indicating w_1. (c) Two non-adjacent sensors S_2 and S_4 covered by *two groups* indicating w_2. (d) Three sensors S_2, S_3 and S_4 detecting obstacles and covered by two groups specifying w_2.

- **Inputs**
  - D_0-D_5: D_i is the distance from the object associated with sensor S_i. They are defined over the universe of discourse $X = [0,1,\ldots,1023]$.
  - *GRL*: target location relative to the robot; *GRL* is entirely defined over the universe $W = [-180^{\circ},\ldots,180^{\circ}]$.
  - *WallLoc*: a dynamic (state-dependent) state variable indicating the location of a long wall. *WallLoc* is a compound state variable and its domain of definition Z is given by $Z = X \times W$.
- **Outputs**
  - θ: turning angle; θ is defined over the same universe as *GRL*, namely *W*.

Having specified the input/output variables, a training example (feature vector) has the following general format:

$$V_i = \{D_0, D_1, D_2, D_3, D_4, D_5, GRL, C\} \tag{7.1}$$

where D_0-D_5 are the distance measures, *GRL* is the relative location of the target to the robot and C is the control action on which this input pattern is mapped.

Vector V_i is an output of module 1 shown in Figure 7.1. In order to generate formatted vectors for unit 3 of this system, vector V_i needs to be re-expressed in terms of fuzzy sets. This process is performed in module 2 and is discussed in the next section.

7.3.1 On-line fuzzification of state variables

In order to supply data increments to the ITI-module (module 3), the input domain needs to be fuzzified in terms of suitable fuzzy linguistic variables. A fuzzy variable is fully defined as the triple $(x, T(x), U)$ [18,19] in which x is the name of the variable, $T(x)$ the term set of x, that is, the set of names of the linguistic values of x with each value being a fuzzy number defined on U. Each fuzzy linguistic variable can be described as a set of some fuzzy numbers (labels assigned to this variable) each characterised by a membership function. It is also necessary to establish the shapes and the regions of those membership functions. The assignment of the fuzzy sets to the fuzzy linguistic variables is an *ad hoc*

design and intuitive. In the current work, this is based on our contextual and semantic knowledge of the system. However, the shapes of the fuzzy sets on a creation universe are empirical and based on experimental results. Subsequently, the term sets of input/output variables are defined in terms of their associated fuzzy sets and the domain of their validity.

- **Inputs**
  - D_0-D_5: D_i is the distance from the object associated with sensor S_i with the term set of D_i defined as $T(D_i) = \{VF, SF, SC, CL, VC\}$ on the universe of discourse $X = [0,1,\ldots,1023]$.
  - *GRL*: target location relative to the robot; the term set of *GRL* is defined as $T(GRL) = \{LB, LE, SL, FR, SR, RI, RB\}$ over the universe of discourse $W = [-180^{\circ},\ldots,180^{\circ}]$.
  - *WallLoc*: the location of a perceiving long wall detected on either side of the robot; *WallLoc* is a compound state variable and its term set *T(WallLoc)* is defined to be $T(WallLoc) = \{LE, RI\}$ over Z which is in turn defined as :

    $Z = X \times W$.

- **Outputs**
  - θ: turning angle; θ is defined over the same universe as *GRL*, namely W with its term set being $T(\theta) = \{TLB, TLE, TSL, TFW, TSR, TRI, TRB\}$.

Figures 7.6 to 7.8 represent the normalised input/output variables and the membership functions of their corresponding fuzzy sets. These membership functions are used for on-line fuzzification and also defuzzification of the output sets.

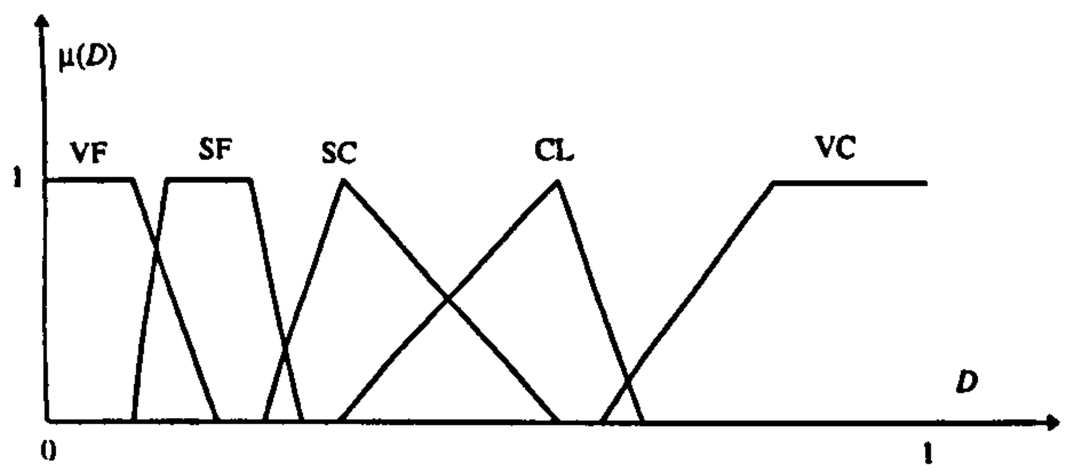


Figure 7.6 Membership functions for the normalised distance between the robot and the target on fuzzy sets: VF (Very Far), SF (Slightly Far), SC (Slightly Close), CL (Close) and VC (Very Close)

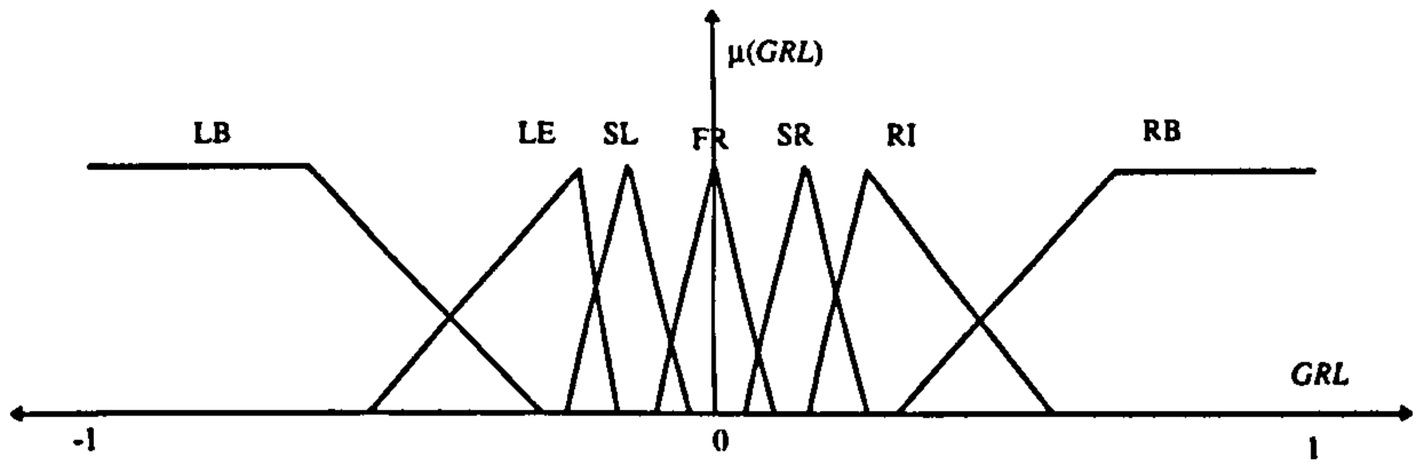


Figure 7.7 Membership functions for the normalised robot position on fuzzy sets: LB (Left Big), LE (Left), SL (Slightly Left), FR (Front), SR (Slightly Right), RI (Right) and RB (Right Big)

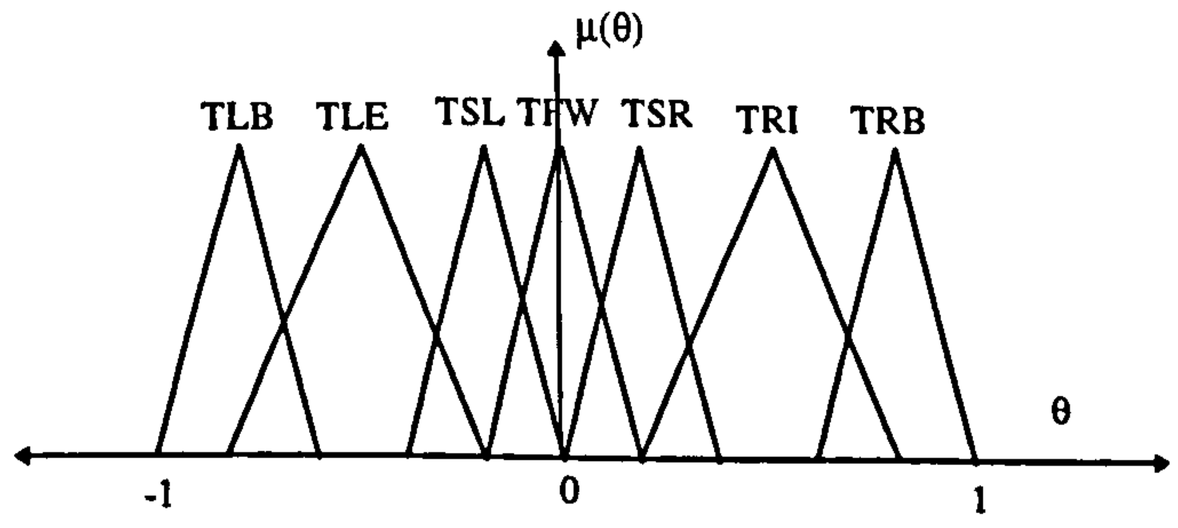


Figure 7.8 Membership functions for the normalised turning angle on fuzzy sets: TLB (Turn Left Big), TLE (Turn Left), TSL (Turn Slightly Left), TFW (Turn Forward), TSR (Turn Slightly Right), TRI (Turn Right) and TRB (Turn Right Big)

As in the manner described in section 6.6.1, fuzzification is used to transform the sensor readings to relevant corresponding linguistic labels. Since all state variables assume crisp values, then fuzzification is merely the matching of an input variable to the membership function of the corresponding linguistic label. This process is illustrated in Figure 7.9

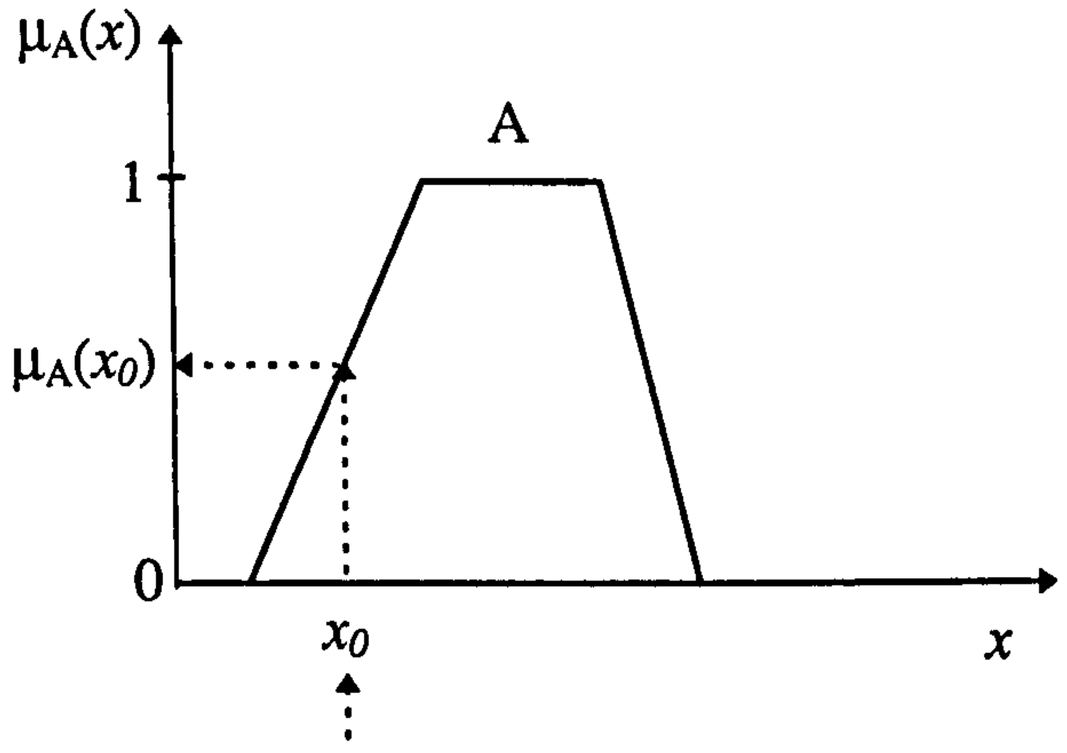


Figure 7.9 Fuzzification of the value of a state variable x by means of a fuzzy set A to achieve the degree of membership of x_0 in A.

In Figure 7.9, the crisp value of a state variable x, namely x_0, falls in the region described by the fuzzy set A. Fuzzification of x_0 involves matching x_0 against the fuzzy set A to find the degree to which x is a member of A. This is represented as $\mu_A(x_0)$ and is linguistically expressed as "x is A".

Joint fuzzy sets

In the current work, due to the incremental nature of fuzzy rule generation, the fuzzification process is performed on-line and iteratively using the membership functions shown in Figures 7.6 to 7.8. The fuzzification process serves the following two distinct purposes (illustrated by switch S in Figure 7.1).

- Fuzzification of state variables to produce fuzzy propositions such as: "x is A" in order to compose a fuzzy vector to be supplied to ITI-2.8 (state a of switch S). This mode is responsible for on-line generation of fuzzy associative memories (FAMs) which make up the KB of the system (unit 4 in Figure 7.1).

- Fuzzification of state variables to produce a perception dependent fuzzy proposition such as "*y* is *B*" to be set theoretically composed with appropriate fuzzy rules encoded in the knowledge base (KB) of the system (state b of switch *S*). The composition operation is performed in the unit "Fuzzy Inference" and its output represents the fuzzy control action space.

State variables can assume values which may fall into the overlapping area of two adjacent fuzzy sets, as depicted in Figure 7.10.

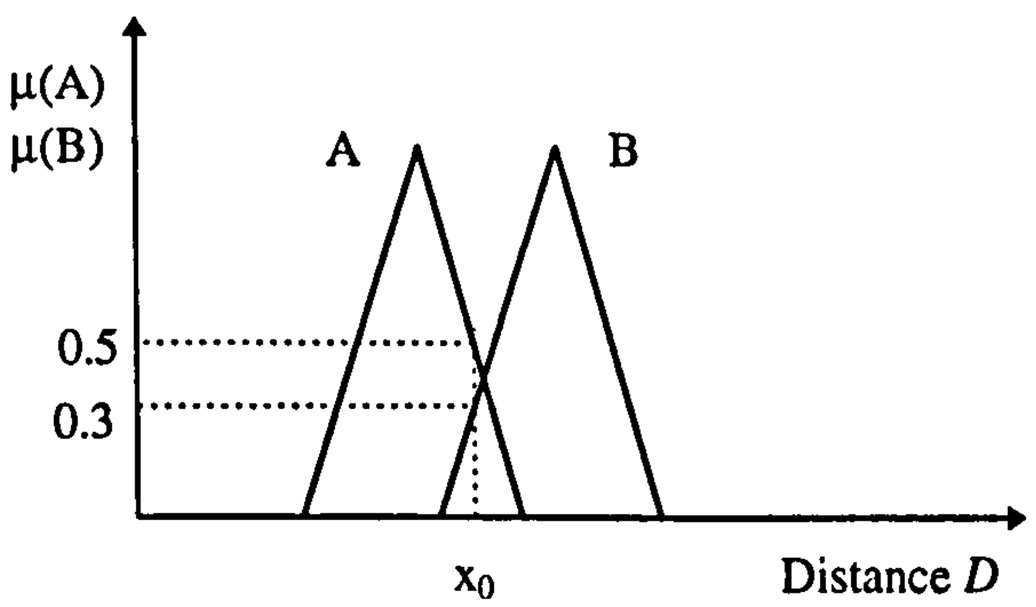


Figure 7.10 Resolving ambiguity in fuzzification of joint sets

In such circumstances, on-line fuzzification involves joint fuzzy sets rather than a single fuzzy set. To deal with this problem, the membership function of the resultant fuzzy set *C* associated with the crisp value x_0 is defined to be

$$\mu_C(x_0) = \boldsymbol{max}\ \{\mu_A(x_0), \mu_B(x_0)\} \tag{7.2}$$

which is a single fuzzy set.

7.4 Fuzzy knowledge generation

As described in chapter 6, the core of any fuzzy reasoning system consists of a repertoire of fuzzy linguistic rules stored in and managed by fuzzy associative memories (FAMs) (also known as the system KB or the fuzzy rule list) and a mechanism performing the compositional rule of inference (inference engine). This implies that the FAMs are *static* and their existence is a pre-requisite to fuzzy reasoning.

Unlike conventional methods where FAMs are configured *prior* to process control, in this work fuzzy rules are learned and generated in an incremental manner. This implies that world-specific FAMs need to be generated on-line, requiring the implementation of dynamic FAMs. By virtue of having multi-dimensional input training vectors, multi-dimensional decision trees are generated in a binary structure which is specific to ITI-2.8. If a world specific decision tree does not already exist, it is initiated and evolves to convergence as new knowledge is acquired. Decision trees are then interpreted into a hierarchy of perceptual fuzzy rules by searching the space of trees. This is conceptually equivalent to searching a hierarchy of nested two-dimensional FAMs. Nesting two-dimensional FAMs to cope with higher-dimensional fuzzy rules in a multiple fuzzy logic system has been addressed and implemented in [20] to resolve the non-linear problem of flexible pole-cart balancing.

Figure 7.11 (see page 198) demonstrates a non-linear multi-dimensional FAM encoded in a binary decision tree and can be compared to an array of two-dimensional FAMs shown in Figure 7.12 (see page 198). The former highlights the intelligibility and expressive power of DT-based FAMs as opposed to conventional FAMs as the dimensionality of the input vector increases. Figures 7.13 to 7.17 (see pages 199 to 203) show the multi-dimensional FAMs encoded in binary DTs. They have been grown incrementally while training the robot in different worlds, and classified as FDT-0 to FDT-4 in the hierarchy. They are then searched to synthesise appropriate control rules.

7.5 The control system architecture

The previous section addressed the nature and the mechanism by which FAMs are generated in the current work. This implies that the FAMs are no longer static nor fixed in size or dimensionality. A change in size or dimensionality occurs each time a new data increment is supplied to ITI. Since the DTs from incremental ITI are inclined to generalise the concept they are constantly learning, they re-organise the tree structure continuously to incorporate new data to augment the learning behaviour. Hence the FAMs used in this work are both dynamic and self-organising.

Figure 7.18 shows the overall control system architecture. As depicted, this is a modified version of the traditional fuzzy-based two-level architectures [8,12,21] for autonomous robots. This approach is somewhat similar to the approach proposed in [21] in the way that both architectures utilise a combination of high-level symbolic planning techniques with low-level continuous control. The author's approach, however, merges the planner and the rule base of the fuzzy controller of these architectures into a hierarchy of dynamic FAMs that evolve progressively and on-line. Together with ITI-2.8 and the fuzzy inference engine, they compose the high level fuzzy decision making unit of the system. From an internal operational view, complex tasks are learned and decomposed into simpler ones and sub-tasks are each encoded and assigned to a certain FAM. However, from the view of the symbolic control action (planner's output), FAMs appear to perform in a similar way to ordinary high-level planners as far as behaviour activation is concerned.

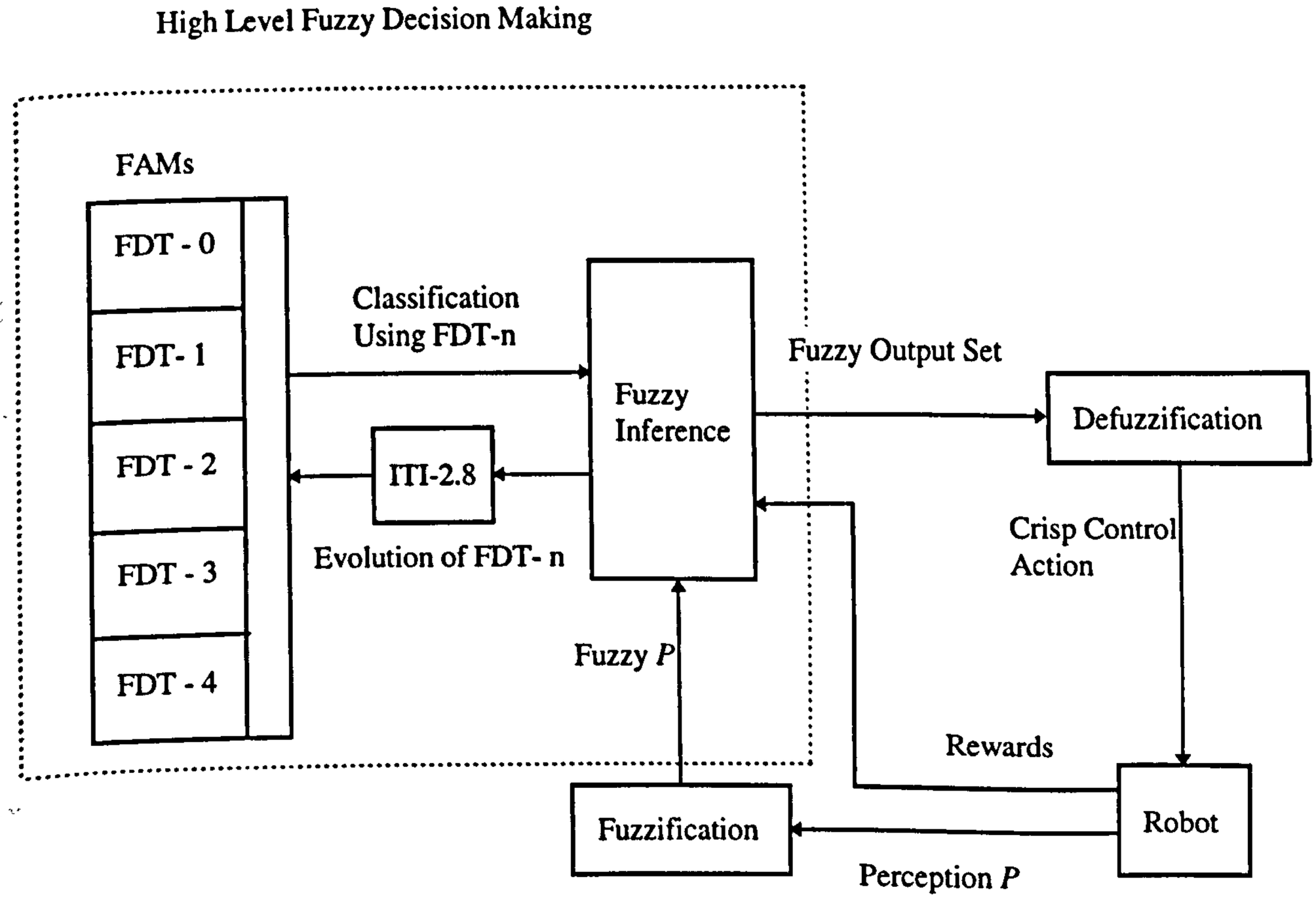


Figure 7.18 The overall system architecture

7.6 Fuzzy reasoning and inference mechanism

The fuzzy reasoning mechanism adopted in this work follows that of Mamdani [22] and is generally termed local inference. The process of reasoning is accomplished in module 5 of Figure 7.1. The mechanism of local inference was introduced in chapter 6 and is developed further below.

Consider a *world* specific fuzzy decision tree mapped on a set of rules

$$\Re = \{\Re_1, \Re_2, \ldots, \Re_i, \ldots, \Re_m\},$$

where

$\Re_i$: **IF** x_1 is A_{1i} and x_2 is A_{2i} and ... and x_n is A_{ni} **THEN** z is C_i.

$\Re_i$ can be implemented by the following fuzzy relation R_i:

$$R_i(X_1, X_2, \ldots, X_n, Z) = (A_{1i} \times A_{2i} \times \ldots \times A_{ni} \rightarrow C_i)(X_1, X_2, \ldots, X_n, Z) \quad (7.3)$$

or

$$R_i(X_1, X_2, \ldots, X_n, Z) = [A_{1i}(X_1) \wedge A_{2i}(X_2) \wedge \ldots \wedge A_{ni}(X_n)] \rightarrow C_i(Z) \quad (7.4)$$

where n is the dimension of the input vector. To classify an n-dimensional fuzzy input vector $\bar{x}$ with $\bar{x} = \{\bar{x}_{10}, \bar{x}_{20}, \ldots, \bar{x}_{n0}\}$ and where $\bar{x}_{i0}$ is the fuzzified crisp value x_{i0}, into a fuzzy output class $C(z)$, we need to compose first the input vector $\bar{x}$ with the calculated fuzzy relation R_i to produce the intermediate result

$$C_i' = (\bar{x}_{10} \times \bar{x}_{20} \times \ldots \times \bar{x}_{n0}) \circ R_i \quad (7.5)$$

C_i' is the output of the i^{th} rule and is set-theoretically defined as:

$$C_i'(z) = \left[A_{1i}(x_{10}) \wedge A_{2i}(x_{20}) \wedge \ldots \wedge A_{ni}(x_{n0})\right] \rightarrow C_i(z) \tag{7.6}$$

Then, all C_i' are combined by some aggregation operator to obtain the overall system output as follows:

$$C = \underset{i=1}{\overset{m}{\cup}} C_i' = \underset{i=1}{\overset{m}{\cup}} \left(\left[A_{1i}(x_{10}) \wedge A_{2i}(x_{20}) \wedge \ldots \wedge A_{ni}(x_{n0})\right] \rightarrow C_i(z)\right) \tag{7.7}$$

where m is the number of contributing rules on the list. If we relate the j^{th} fuzzy set of the i^{th} fuzzy rule, $A_{ji}(x_{ji})$, to its membership function, $\mu_{ji}(x_{ji})$, and model the fuzzy implication by Mamdani's *min* operator and also interpret the logical OR of rules by the *max* operator, equations (7.3) and (7.7) can be rewritten using *min-max* composition in terms of their membership functions as:

$$\mu_{R_i}(x_{1i}, x_{2i}, \ldots, x_{ni}) = \min_{j=1}^{n} \left[\mu_{A_{ji}}(x_{ji})\right] \tag{7.8}$$

and

$$\mu_C(y) = \max_{\substack{i=1 \\ x \in U}}^{m} \left[\min_{j=1}^{n} \left[\mu_{A_{ji}}(x_{ji}), \mu_{R_i}(x_1, x_2, \ldots, x_n, y)\right]\right] \tag{7.9}$$

where m is the number of *effectively* contributive rules. The underlying principles of labelling some fuzzy rules as effectively contributive are discussed in details in section 7.8. The above expression characterises the membership function of the output set which determines the space of possible outputs (control actions).

7.7 A practical approach to defuzzification

The output of the decision making module, $\mu_C(y)$, is a fuzzy set specifying the possibility distribution of the control action. Therefore, the process of reducing this fuzzy set to a crisp single-valued output is termed defuzzification. Since no defuzzifier has been derived from first principles [23] and due to the need for the computational simplicity, this work uses the

centroid method to generate a crisp control action. This specifies the centre of gravity of the output fuzzy set, $C(z)$, as z_0 defined as follows:

$$z_0 = \frac{\int_S z\mu_C(z)dz}{\int_S \mu_C(z)dz} \tag{7.10}$$

where S spans over the support of $\mu_C(z)$ with $\mu_C(z)$ being continuous. However, for an efficient implementation of the above equation, the author implements the following equation:

$$z_0 = \frac{\sum_{i=1}^{n}\left(A_i\,\mu_{C_i}(z)M_i\right)}{\sum_{i=1}^{n}\left(A_i\,\mu_{C_i}(z)\right)} \tag{7.11}$$

where

$$A_i = 0.5(a+b) \tag{7.12}$$

is the area of the i^{th} output fuzzy set, M_i is the parallel through the centre of the i^{th} output fuzzy set, with a and b its corresponding base, if we assume the general shape of fuzzy outputs is trapezoidal, whereas $\mu_{C_i}(z)$ is the firing strength of the i^{th} output fuzzy set. All areas, A_i, are calculated during initialisation to provide computationally faster defuzzification during training.

7.8 Classification by isolating contributive fuzzy rules

As demonstrated in Figure 7.11, the DTs generate multi-class leaves [24] that assign a certain perceptual pattern to more than one class. To investigate this multi-dimensionality in both input and output parameters and their consequence with regard to the dimensionality of the fuzzy rules which *effectively* contribute to the output, the following hypothetical scenario is first considered.

In a certain state, *GRL* may have an actual crisp value, x_0, which falls in the overlapping area between fuzzy sets SL and LE and D_1 with an actual crisp value of x_1 falling in the overlapping area between fuzzy sets SF and SC. According to the linear FAM shown in Figure 7.19, this perception activates the following four fuzzy rules to contribute collectively towards a single control action.

$\Re_1$: IF *GRL* is SL and D_1 is SF THEN θ is TSL.

$\Re_2$: IF *GRL* is SL and D_1 is SC THEN θ is TRI.

$\Re_3$: IF *GRL* is LE and D_1 is SF THEN θ is TSL.

$\Re_4$: IF *GRL* is LE and D_1 is SC THEN θ is TSR.

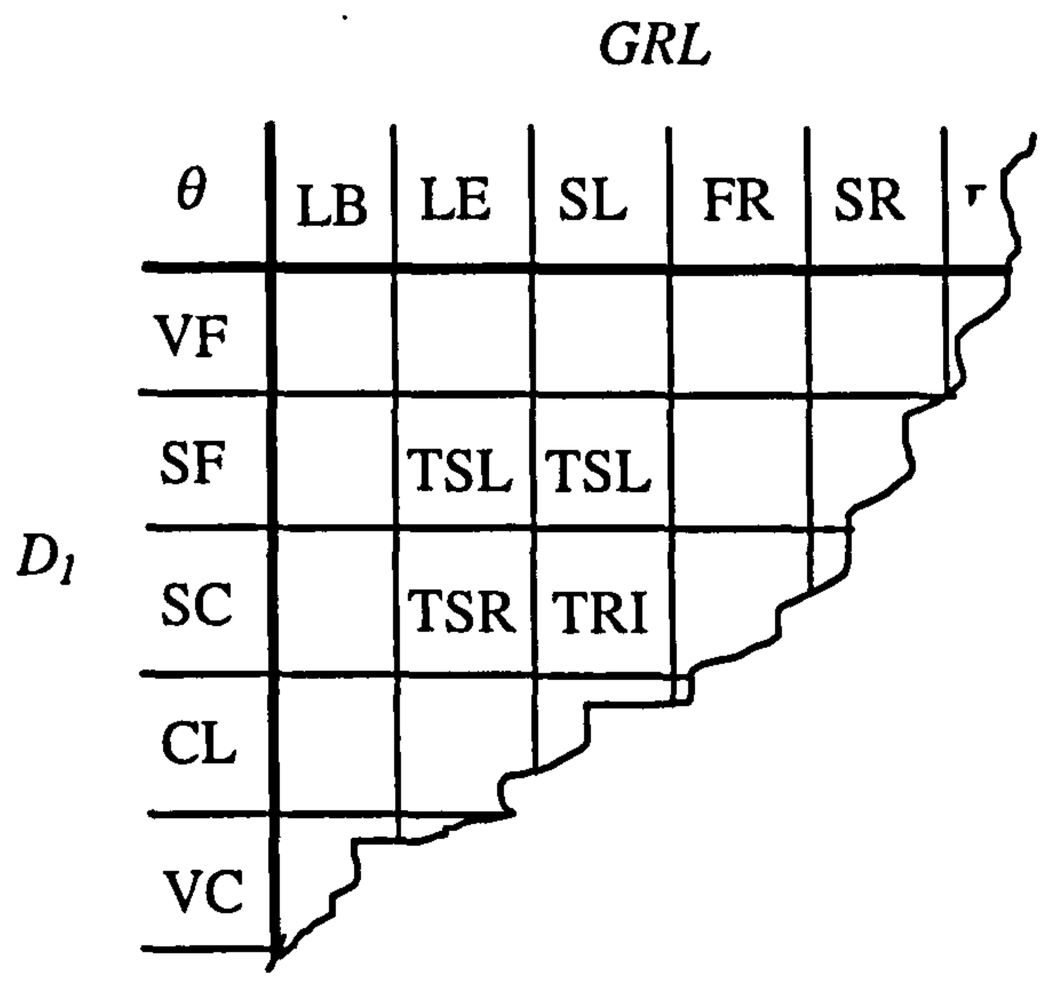


Figure 7.19 An example of a linear 2D-FAM

The above rule list illustrates that a two-dimensional input vector with a single output would create four fuzzy rules. However, the pertinent question is whether the dimensionality of the rule lists is linearly proportional to the combination of the input-output vectors' dimensions, as far as multi-dimensional input/output DTs are concerned. In the next section, it is demonstrated that the length of any perceptual fuzzy rule list, such as above, is directly proportional to the total number of classes in the leaf which classifies that perception.

7.8.1 An analogous problem

To demonstrate the validity of the statement mentioned above, a physical analogy shown in Figure 7.20 is considered.

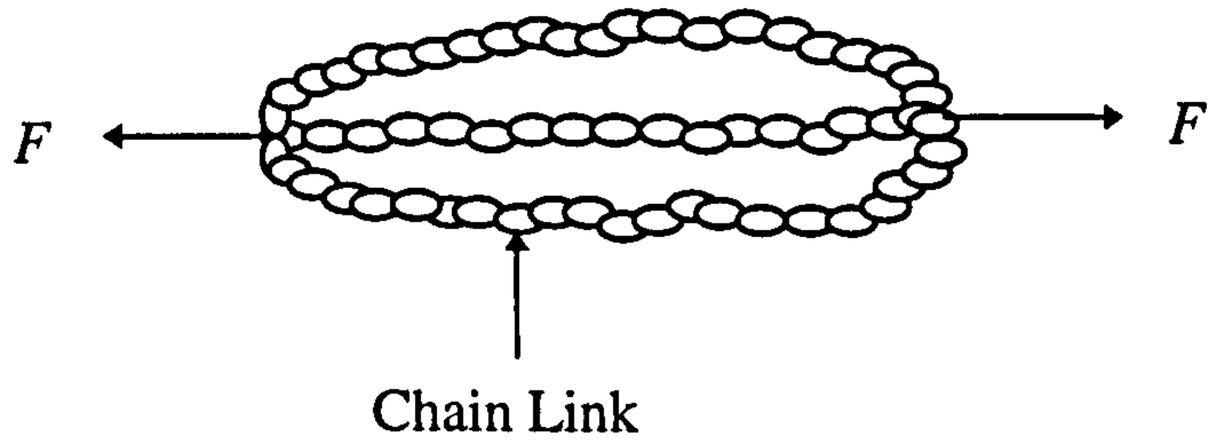


Figure 7.20 An analogous example describing *min-max* operation

As shown above, three physically separate chain branches are placed together at their ends to make up a chain system. Each branch consists of a finite number of chain links. To assess the strength of one branch, one would exert suitable equal and opposite forces, F, at either end of that branch. Therefore, the strength of the branch would be dominated by its weakest link. That is, the *minimum* (*min*) resistance of an entire *branch* in the presence of an exerting force is determined by its weakest chain link. Now, consider an increasing tensile force exerted at either end of the entire chain system. Weaker links would break down one after the other as the force increases further along. Hence, the strength of the entire chain system is governed by the strongest chain branch. In other words, the *maximum* (*max*) resistance of the entire *chain system* under an increasingly exerting force is dictated by the strongest branch.

This analogy is utilised to filter the effectively contributing fuzzy rules from the list of contributive fuzzy rules.

7.8.2 A robot perception example

This process can be made further apparent by considering an example of the robot perception in w_3 (shown in Table 7.1) and observing how the corresponding tree would be searched to classify this pattern.

Perception P D0,D1,D2,D3,D4,D5,GRL (Actual crisp values)	Class (Fuzzy output)
310,150,165,255,110,140,25	TFW,TSR

Table 7.1 An instantaneous perception of the robot

This perceptual state can be classified by searching the tree along the path shown in Figure 7.16. This path can be automatically transformed into the following *generic* rules:

$\Re_1$: **IF** D_2 = SF $\wedge$ D_1 = SF $\wedge$ D_3 = $\neg$VF $\wedge$ D_4 = VF **THEN** θ = TFW.

$\Re_2$: **IF** D_2 = SF $\wedge$ D_1 = SF $\wedge$ D_3 = $\neg$VF $\wedge$ D_4 = VF **THEN** θ = TSR.

Considering the actual crisp value of D_3 (255) and replacing the fuzzy set ($\neg$VF) with its complement, the above rules are altered as shown in Table 7.2.

$\Re_1$: **IF** D_2 = SF $\wedge$ D_1 = SF $\wedge$ D_3 = SF $\wedge$ D_4 = VF **THEN** θ = TFW.
$\Re_2$: **IF** D_2 = SF $\wedge$ D_1 = SF $\wedge$ D_3 = SC $\wedge$ D_4 = VF **THEN** θ = TSR.

Table 7.2 The generic rule set resulting from the search for effective contribution to generate the final control action

As depicted in Table 7.2, the crisp values of input parameters fall into the overlapping areas of two adjacent fuzzy sets activating an array of 16 fuzzy rules (considering only one class) and a total of 32 fuzzy rules (16 rule combinations and 2 classes), as illustrated in Appendix A. To combine such a number of rules to generate one control action is computationally expensive due to the complex defuzzification involved and may well result in a delayed response to such perceived situations. Also, the antecedent of each individual rule, $\Re_i$, on the list contributes a degree of membership, $\mu_{C_i}(y)$, which determines the extent to which the corresponding fuzzy output set is truncated (termed as firing strength). This implies that from the set of two arbitrary membership functions, $\mu_{C_1}(y)$ and $\mu_{C_2}(y)$ with $\mu_{C_1}(y) > \mu_{C_2}(y)$, the former dominates the effect of the latter.

Expanding the analogy of the chain system to the list of 16 fuzzy rules to compute the highest membership degree $\mu_C(y)$, each individual rule, $\Re_i$, is analogous to each chain branch (with *min* operating on antecedents of each rule) and the *maximum* contribution (with *max* operating on 16 fuzzy rules) of the entire list of 16 fuzzy rules would be equivalent to the strength of the entire chain system. The expression $\mu_C(y)$ of Appendix A indicates the highest firing strength that the fuzzy rule list can contribute. This implies that on applying this strategy, regardless of the total number of the *generated* rules on the list (16 rules), the maximum firing strength is computed and applied to all available classes in that leaf. Consequently, the total number of the *effectively* contributive rules is equal to the number of classes found in that leaf, as shown in Table 7.2.

7.9 Overfitting inception and tree pruning

Individual decision trees are grown to a size dependent upon the diversity in the patterns of their training vectors and the number of training vectors supplied to ITI. The experimental results demonstrate that w_0 (the world with no obstacles) and w_3 tend to generate smaller trees than the others (w_1, w_2 and w_4). This is due respectively to the small number of features and the limited number of directions available to the robot.

Highly non-linear training vectors demand a large number of training examples to produce trees with appropriate accuracy, whereas large numbers of training examples endanger the generalisation capability of the tree and may lead to overfitting. In this case, the dimensionality of trees grows unreasonably and trees tend to specialise each individual pattern. Consequently, generating appropriately-sized decision trees is a matter of trade-off between the two factors.

To suppress possible overfitting without compromising the generalisation nor the classification accuracy, an algorithm has been developed to perform automatically pruning or unpruning of the active tree.

STEP 1: Detect the active FDT

STEP 2: **IF** the threshold on the number of training examples exceeded

STEP 3: **THEN** Prune FDT

STEP 4: **IF** Accuracy of pruned FDT falls below desired threshold

STEP 5: **THEN** Unprune FDT

STEP 6: GOTO Step 8

STEP 7: **ELSE** Save pruned FDT

STEP 8: **END.**

7.9.1 Classification accuracy

A reliable estimate of classification accuracy of the generated DTs is the statistical method of *n*-fold cross-validation [25,26]. The training vectors are split into *n* blocks of approximately the same length and class distribution. To test each training vector only once, for each block a DT is constructed from the remaining training vectors and tested on the hold-out block. The error rate of a classifier constructed in such a way is the ratio of the total number of errors in the hold-out blocks to the total number of training vectors. The average error rate of cross-validations is a relatively reliable estimate (compared to the performance of this classifier on a set of new data) of the error rate of a single classifier produced from these training vectors [26].

When prompted, the integrated ITI in the current work performs a 15-fold cross-validation on the available training vectors to produce the classification accuracy of the active FDT. Figures 7.21 to 7.25 (see pages 204 to 208) show the pruned versions of FDTs depicted in Figures 7.13 to 7.17 (see pages 199 to 203) which are unpruned and 15-fold cross-validated. Table 7.3 demonstrates the prediction accuracy of the different FDTs before and after post-pruning.

It can be observed that pruning not only has achieved a significant reduction on tree size, but also it generally enhances the generalisation ability of trees which in turn increases the prediction accuracy. After pruning the tree, a 15-fold cross-validation was run two consecutive times on the tree. Table B.1 and Table B.2 of Appendix B contrast the prediction accuracy before and after pruning by conducting two runs of 15-fold cross-

validation on the FDT representing w_1, shown in Figure 7.14. It is evident that the average classification accuracy of the pruned tree has been enhanced compared to that of the unpruned version, with correct classification being improved from 81.76% to 83.31% of cases.

	Average Prediction Accuracy after 5 Runs with 15-fold Cross-Validation in %				
Worlds	w_0	w_1	w_2	w_3	w_4
Before Pruning	70.20	81.52	78.60	96.67	69.66
After Pruning	69.25	83.01	79.50	96.67	69.22

Table 7.3 The prediction accuracy of 15-fold cross-validated FDTs before and after exposing to pruning in percentage.

7.10 A detailed example of the novel approach

As discussed in the foregoing sections, the process of generating FDTs involves the following stages:

1. Automatic fuzzy data acquisition
2. Generation of fuzzy FDTs
3. Generation of fuzzy rules

To demonstrate the above mechanisms, we consider a simple example in which a small number of training vectors is collected, the tree is initiated and incrementally re-inferred; the tree is then searched to classify a robot perception.

7.10.1 Automatic fuzzy data acquisition

In the training mode, the algorithm evaluates in state *n* the performed robot motion of state *n-1* using the cost function. This mechanism is preceded by fuzzification of the entire perceptual state variables, section 7.3.1. If it results in a positive action, this perception is kept in a continuously up-dated memory store which is accessed directly by ITI when a new perception arrives. Appendix C is a collection of 40 such training vectors (describing partially w_2) whose elements are fuzzy sets and are the results of positively rewarded robot motions.

7.10.2 Generation of FDTs

The process of growing FDTs is incremental and is triggered when a new training vector is supplied. This means that FDTs adapt and alter their structure to incorporate the incoming data. The learning algorithm presented in this work, is able to search the space of FDTs in the intervals between two successive tree manipulations. Figure 7.26 shows the FDT which is grown on the fuzzy data acquired in the previous section.

7.10.3 Generation of fuzzy rules

Synthesis of fuzzy rules is accomplished on-line and concurrently with searching the trees. A certain robot perception can activate one or more fuzzy rules which are then aggregated to a single fuzzy output set and defuzzified to generate a suitable control action. For instance, consider the following robot perception shown in Table 7.6.

State Variables (Crisp Values)							Corres. World
D_0	D_1	D_2	D_3	D_4	D_5	*GRL*	
220	120	180	310	102	110	-30	w_2

(a)

State Variables (Fuzzified)							Corres. World
D_0	D_1	D_2	D_3	D_4	D_5	*GRL*	
SF	VF	SF	SC	VF	VF	SL	w_2

(b)

Table 7.4 An instantaneous robot state perceived from w_2 in terms of (a) its crisp values and (b) after fuzzification.

To classify the robot state, the FDT shown in Figure 7.26 is followed along the directed path to the terminal node containing the fuzzy output sets. This results in activating one fuzzy rule, as in shown in Table 7.5.

IF $D_2 = \neg$VF $\wedge$ $D_1 = \neg$ SF $\wedge$ $D_2 = \neg$SC $\wedge$ $D_4 = \neg$SC **THEN** C = TLB.

Table 7.5: The synthesised fuzzy rule after searching the FDT to classify the robot perception shown in Table 7.4 (b)

In Table 7.5, D_i is the i^{th} sensor and C is the class or the control action. Note that the FDT depicted in Figure 7.26, does not include all representative training vectors as it is rather "immature" and may misclassify certain robot states. Therefore, this example should only highlight the essential steps involved in the learning algorithm without presenting specific details of its development process.

7.11 Simulated behaviour learning

The hierarchical approach facilitates global learning which is initiated in w_0 and propagates up the hierarchy to more complex worlds. This is performed in such a way that the knowledge in the previous layer is modified to adapt to the current perception in order to grow knowledge for the current layer. The on-line learning technique presented in this work has been verified using the Khepera robot simulator. Due to the realistic assumptions made in the design of the simulator, this facilitates the transfer of the simulation results without major alterations to the real Khepera robot [27].

In all homing tasks, the robot is set at a starting point S with an arbitrary heading angle and is expected to reach a target labelled G, Figure 7.27 (see page 210). The simulation results aim to illustrate the feasibility of overall knowledge decomposition into a hierarchy of fuzzy decision trees that are trained and developed locally. Qualitative reasoning with fuzzy symbolic data in behaviour learning such as object avoidance, target seeking and wall following behaviour are demonstrated in the next section.

7.12 Results and discussion

The principal strength of the technique proposed is incremental behaviour learning in the frame of a reactive architecture. The hierarchy is set up to integrate two fundamental behaviours (object avoidance and target seeking behaviour) in such a way that dominance of one behaviour in the hierarchy reduces the presence of the other behaviour. A further behaviour, namely the wall-following behaviour, is also integrated in the hierarchy, and is activated when local minima are detected [28]. This behaviour is encoded in w_4. For instance, w_0 (perception with no obstacles) is highly goal-oriented as opposed to w_3 in which reactivity is the dominant behaviour (see also Figure 4.3 in chapter 4). This implies that certain environmental configurations (obstacle positions) can activate one of these two behaviours immediately *prior* to the second. This can then lead to oscillatory trajectories which are inherent to reactive strategies [4]. Table 7.6 shows the average number of runs needed to train FDTs to the size incorporating the representative feature vectors. It also demonstrates that the number of training examples varies in different FDTs depending on the world-specific obstacle configurations.

Figures 7.27 (a) and (b) demonstrate the first stages of learning in which the robot is trained in the world with no obstacles. In this world, the robot establishes the goal-seeking behaviour which is encoded in w_0. The gradually smoothing trajectories demonstrate the incremental nature of the learning algorithm. After sufficient training, the robot is able to reach the set target (using similar settings to those used in the production of Figures 7.27 (a) and (b)) in significantly fewer steps, Figures 7.28 (a) and (b).

Having access to the knowledge learned in w_0, the robot is trained in a slightly more complex world, w_1, which is sparsely populated with obstacles. Figures 7.29 (a) and (b) show how the robot tries to adapt by producing a new behaviour while heading towards the target. This adaptation is more evident, particularly in Figure (a) where the robot first encounters obstacles. Figures 7.30 (a) and (b) are navigation scenarios in the same unseen environment where the robot utilises the learned knowledge to home in on the goal.

Figures 7.31 (a) and (b) show an intermediate stage of learning (w_2) where the robot is trained in a more "hostile" environment in which obstacles are more densely situated. The

robot learns how to avoid larger obstacles. Once the robot has been sufficiently trained in this environment, it is navigated in a modified terrain. The robot can bypass the obstacles and demonstrates a smooth behaviour, particularly, in front of the disjoint long obstacles, as shown in Figures 7.32 (a) and (b).

	Worlds				
	w_0	w_1	w_2	w_3	w_4
Average No. of Runs	15	13	10	8	12
Number of Training Examples	78	223	118	72	212
Total no. of training runs to build up the hierarchy (Average)					58

Table 7.6 The average number of trials and the training examples in different worlds needed to set up the FDT hierarchy

A third behaviour which is also integrated in the hierarchy and encoded in w_4, is that of wall-following. This behaviour is triggered in situations when the robot follows long parallel walls or the algorithm detects local minima. This happens when the target is located behind long walls or nested walls, corners and dead-ends, in which case the robot would perform oscillatory movements in infinite loops. Figures 7.33 (a), (b) and (c) are examples where the robot is presented with long walls and corners to adapt to the aforementioned situations.

Figure 7.34 (a) illustrates a navigation scenario using pure DT learning where the two behaviours (target-seeking and obstacle avoidance) alternate. This gives rise to the oscillatory path which is typical in situations where the robot follows long walls or parallel

corridors of walls to reach a set target. This shortcoming has been overcome, as demonstrated in Figure 7.34 (b), by introducing fuzziness into the DT-based hierarchy, thereby merging the conflicting behaviours and assuring smooth trajectories in the presence of long walls. However, the variations remaining are due to the physical discontinuities of the long wall and are removed, as shown in Figure 7.34 (c), when the wall is physically augmented.

Figures 7.35 to 7.38 are various scenarios to demonstrate the performance of the robot in navigatory tasks where the robot has access to the knowledge in the entire hierarchy. Figures 7.37 and 7.39 show that reactive path planners are not always time-efficient as far as the length of the generated trajectories is concerned. They are, however, effective and can generate the shortest path, as shown in Figures 7.35 and 7.38.

7.13 Comparison with other learning systems

In this section, a comparison is made between the author's approach to the development of a hybrid learning system and previous work carried out in the frame of reactive architecture.

Since the learning algorithm in this work is that of a self-learning system in which the entire hierarchy is learned incrementally and the knowledge is acquired automatically, the new algorithm is able to learn automatically from its environment, providing a significant training advantage when compared with supervised learning systems. In these systems, the intervention (partially or completely) of a teacher or a trainer is inevitable in that the robot is either taught to learn [29] or its actions are punished or rewarded. The control architecture of the AuotonoMouse (a mouse-like robot) in [29] embodies a three-stage *developmental* learning, namely baby stage (in which the assistance of a trainer is needed), young stage and adult stage. The author's work can be considered to bypass the baby stage is implemented by the learning system.

Since reactivity directly transforms perceptions to actions, artificial neural networks (ANN) provide an obvious method for learning reactive transformations from perception to action. ANN systems have been developed which implement either a reinforcement algorithm [30,31] or a back propagation algorithm for learning and adaptation [32]. A drawback of

the learning techniques developed for ANN systems is that they are often very dependent on the presentation of meta-knowledge in order to achieve fast convergence, as can be observed in [30,33]. A similarity to the current work that the connectionist architecture, called TESEO, in [30] exhibits, is that it has been designed to overcome perceived limitations of the reinforcement learning technique, such as slow convergence and lack of incremental improvement. A significant architectural difference, however, between TESEO and the current work is that in the former the learning is augmented by built-in primitives (simple stimuli-responses) operating as background knowledge, whereas in the author's approach the learning is performed uniformly from inception to convergence with no *a priori* knowledge. The experimental results conducted on TESEO have demonstrated that the robot, after 10 training epochs, learns how to reach a target on a smooth trajectory and with the shortest path. The algorithm in this work, though, may not always generate the shortest path (as shown in Figures 7.34 and 7.38)due to its reactivity, as does TESEO, but it is effective and capable of always reaching the target. However, once trained, TESEO appears to learn a specific environmental structure of the navigation terrain; requiring re-training if the environment changes significantly. In the author's work, the algorithm is globally tuned, in that it does not require re-training for new environments.

The hybrid approach proposed in [33] which is a neuro-fuzzy robot learning system, also relies heavily on *a priori* knowledge. In an example of this approach, the knowledge encoded in an expert system is used to initialise learning which consequently updates the meta-knowledge of the expert systems. As far as the comprehensibility of the learned knowledge is concerned, it is very hard to understand the learned "rules" of a neural system, since the knowledge of the neural nets resides in the connections between the processing units (neurons) in the form of real numbers (weights), whereas the internal representation of DTs is highly intelligible to human users due to their symbolic nature.

The author's approach also improves on previous work in that it addresses a number of non-trivial real world issues and constraints such as sensor noise, reactivity, incrementality, short training times and robustness harnessing the power of decision trees and fuzzy theory for learning and adaptation in a robotic system.

7.14 Summary

A new approach for setting up an array of fuzzy associative memories (FAMs) in the context of hybrid learning has been introduced. FAMs are engineered on-line and incrementally without human intervention. This multi-strategy learning technique unites the features of inductive learning and fuzzy techniques to cope with inherent uncertainty in sensory data and to reason on high-level symbolic data. Fuzzy logic is also used to merge conflicting behaviours to assure smooth trajectories.

The novelty of the proposed hybrid technique is three-fold: automatic fuzzy data acquisition, automatic generation of fuzzy decision trees (FDTs) from inception and the introduction of DT-based FAMs. The latter feature facilitates non-linear and multi-dimensional FAMs accommodating fuzzified symbolic knowledge while offering intelligibility and expressive power (inherent to decision trees) to the fuzzy control rules. As demonstrated, this would be virtually equivalent to searching the space of a hierarchy of nested linear FAMs.

Behaviour learning and dominance is decomposed into a hierarchy of locally tuned FAMs each encoded in an individual FDT as physically isolated computational entities with dynamic life times. Global path planning can also be accomplished by coupling an array of locally tuned FAMs.

References

[1] Maes and R.A. Brooks, "Learning to Co-ordinate Behaviours", *Proceedings of AAAI'91*, Boston, MA, 1991, pp. 796 - 802.

[2] M.J Mataric, "Behaviour-Based Control: Main Properties and Implications", *Proceedings of IEEE International Conference on Robotics and Automation*, *Workshop on Architectures for Intelligent Control Systems*, Nice, France, May 1992.

[3] M.J. Mataric, "Behaviour-Based Control: Examples from Navigation, Learning, and Group Behaviour", *Journal of Experimental and Theoretical Artificial Intelligence*, Special Issue on Software Architectures for Physical Agents, Vol. 9, Nos. 2-3, Hexmoor, Horswill, and Kortenkamp, Eds., 1997, pp. 323-336.

[4] P. Reignier, "Fuzzy Logic Techniques For Mobile Robots Obstacle Avoidance", *Robotics and Autonomous Systems,* 12 (1994), pp. 143-153, ELSEVIER.

[5] D.T. Lawton, R.C. Arkin and J.M. Cameron, "Qualitative Spatial Understanding and Reactive Control for Autonomous Robots", *The IEEE Workshop on Intelligent Robots and Systems*, 1990, Vol. 2, pp. 709-714.

[6] A. Saffiotti, "The Uses of Fuzzy Logic in Autonomous Robot Navigation: a Catalogue Raisonne'", *Soft Computing*, Vol. 1, No. 4, 1997, Springer Verlag.

[7] A. Saffiotti, E.H. Ruspini and K. Konolige, "Blending Reactivity and Goal-Directedness in a Fuzzy Controller", *The Second IEEE Conference on Fuzzy Systems*, San Francisco, CA, March 1993, pp. 134-139.

[8] A. Saffiotti, E.H. Ruspini and K. Konolige, "Robust Execution of Robot Plans using Fuzzy Logic", *Lecture Notes in Artificial Intelligence 847*, Springer Verlag, pp.24-37.

[9] A. Saffiotti and E.H. Ruspini, "Using Fuzzy Logic for Mobile Robot Control", *International Handbook of Fuzzy Sets and Possibility Theory*, D. Dubois, H. Prade and H.J. Zimmermann, Eds., 1997, Kluwer Academic.

[10] A.M. Gonsalz de Miguel, P. Vacher, P. Collinwood and R.Osborn, "An Avoidance Fuzzy Logic Controller for the Mobile Robot Khepera", *Fifth International Workshop on Advanced Robotics and Intelligent Machines*, University of Salford, UK, March 1997.

[11] H. Rak and H.S. Cho, "A Sensor-Based Navigation for a Mobile Robot Using Fuzzy Logic and Reinforcement Learning", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 25, No. 3, March 1995.

[12] H. Surmann, J. Huser and L. Peters, "A Fuzzy System for Mobile Robot Navigation", *The Fourth IEEE Conference on Fuzzy Systems*, Yokohama, Japan, March 1995, pp. 83-86.

[13] G.H. Shah-Hamzei and D.J. Mulvaney, "Self-organising Fuzzy Decision Trees for Robot Navigation: An On-line Learning Approach", *IEEE International Conference on Systems, Man and Cybernetics SMC'98*, October 11-14, 1998, San Diego, CA, USA, (Forthcoming).

[14] M. Benreguieg, P.Hoppenot, H. Maaref, E. Colle and C. Barret, "Fuzzy Navigation Startegy: Application to Two Distinct Autonomous Robots", *Robotica*, Vol. 15, 1997, Cambridge University Press, pp.609-615.

[15] S.C. Hsu, J.Y. Hsu and I.J. Chiang, "Automatic Generation of Fuzzy Control Rules by Machine Learning Methods", *IEEE International Conference on Robotics and Automation*, 1995, pp. 287-292.

[16] L.O. Hall and P. Lande, "Generating Fuzzy Rules from Data", *Fifth IEEE International Conference on Fuzzy Systems*, 1996, New Orleans, Vol. 3, pp. 1757-1762.

[17] O. Michell, Mage Team, 13s Laboratory, CNRS, University of Nice-Sophia Antipolis, France; Khepera Simulator was provided for the Official Khepera Contest at Evolution Artificial Intelligence Conference (Nimes, 1997), Downloadable from http://alto.unice.fr/~om/khep-contest.html.

[18] C.C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller-Part II", *IEEE Transactions on Systems, man and Cybernetics*, Vol. 20, No. 1, March./April 1990.

[19] C.C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller-Part I", *IEEE Transactions on Systems, man and Cybernetics*, Vol. 20, No. 1, March./April 1990.

[20] E.P. Dadios, "Non-Conventional Control of the Flexible Pole-Cart Balancing Problem", *PhD Thesis*, Department of Manufacturing Engineering, Loughborough University, 1996.

[21] E.H. Ruspini, A. Saffiotti and K. Konolige, "Progress in Research on Autonomous Vehicle Motion Planning", J. Jen, R. Langari and L.A. Zadeh (Eds.), *Industrial Applications of fuzzy Logic and Intelligent Systems*, IEEE Press, 1994, ISBN: 0-7803-1048-9, pp. 157-190.

[22] E.H. Mamdani, "Applications of Fuzzy Algorithms for Control of Simple Dynamic Plant", *Proceedings of IEE, Control and Science*, Vol. 121, No. 12, December 1974, pp. 1585-1588.

[23] J.M. Mendel, "Fuzzy Logic Systems for Engineering: A Tutorial", *Proceedings of the IEEE, Special Issue on Engineering Applications of Fuzzy Logic*, Vol. 83, No. 3, march 1995, pp. 343-377.

[24] G.H. Shah-Hamzei, D.J. Mulvaney and I.P.W. Sillitoe, "Multi-layer Hierarchical Rule Learning in Reactive Robot Control using Incremental Decision Trees", *The International Journal of Intelligent and Robotic Systems*, Kluwer Academic Publishers (In Press).

[25] J.R. Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers, 1992.

[26] J.R. Quinlan, "Using C5.0: An Informal Tutorial", Available to download on WWW at: http://www.rulequest.com/tutorial.html.

[27] O. Michell and P. Collard, "Artificial Neurogenesis: An Application to Autonomous Robotics", *Proceedings of the Eighth International Conference on Tools with Artificial Intelligence*, IEEE Computer Society Press, 1996, pp. 207 - 214.

[28] G.H. Shah-Hamzei and D.J. Mulvaney, "System Instability and Oscillation Resolution in Reactive Robotics using DT-based Approach to Learning", *International Conference on Artificial Intelligence and Soft Computing*, Banff, Canada, July 27 - August 1, 1997, pp. 411 - 414.

[29] M. Dorigo and M. colombetti, "The Role of the Trainer in Reinforcement Learning", *MLC-COLT Workshop on Robot Learning*, July 10, 1994, Rutgers University, new Brunswick, N.J, pp.37-45.

[30] J.R. Milan and C. Torras, "Efficient Reinforcement Learning of Navigation Strategies in an Autonomous Robot", *IEEE Conference on Intelligent Robots and Systems*, September 1994, Munich, Germany, pp. 15-22.

[31] A.H. Fagg, D. Lotspeich and G.A. Bekey, "A Reinforcement Approach to Reactive Control Policy Design for Autonomous Robots", *The IEEE International Conference on Robotics and Automation*, May 1994, San Diego, CA, pp. 39-44.

[32] W. Li, "Neuro-Fuzzy Systems for Intelligent Robot Navigation and Control under Uncertainty", *IEEE Proceedings*, ISBN: 0-7803-2461-7, 1995, pp. 1747-1754.

[33] A. Kandel, G. Langhlz and M. Schneider, "Artificial Fuzzy Neural Networks and Their Application to Intelligent Robot Control Systems", *The Sixth IFAC Symposium on Large Scale Systems*, August 1992, Beijing, pp. 391-396.

PAGE

MISSING

IN

ORIGINAL

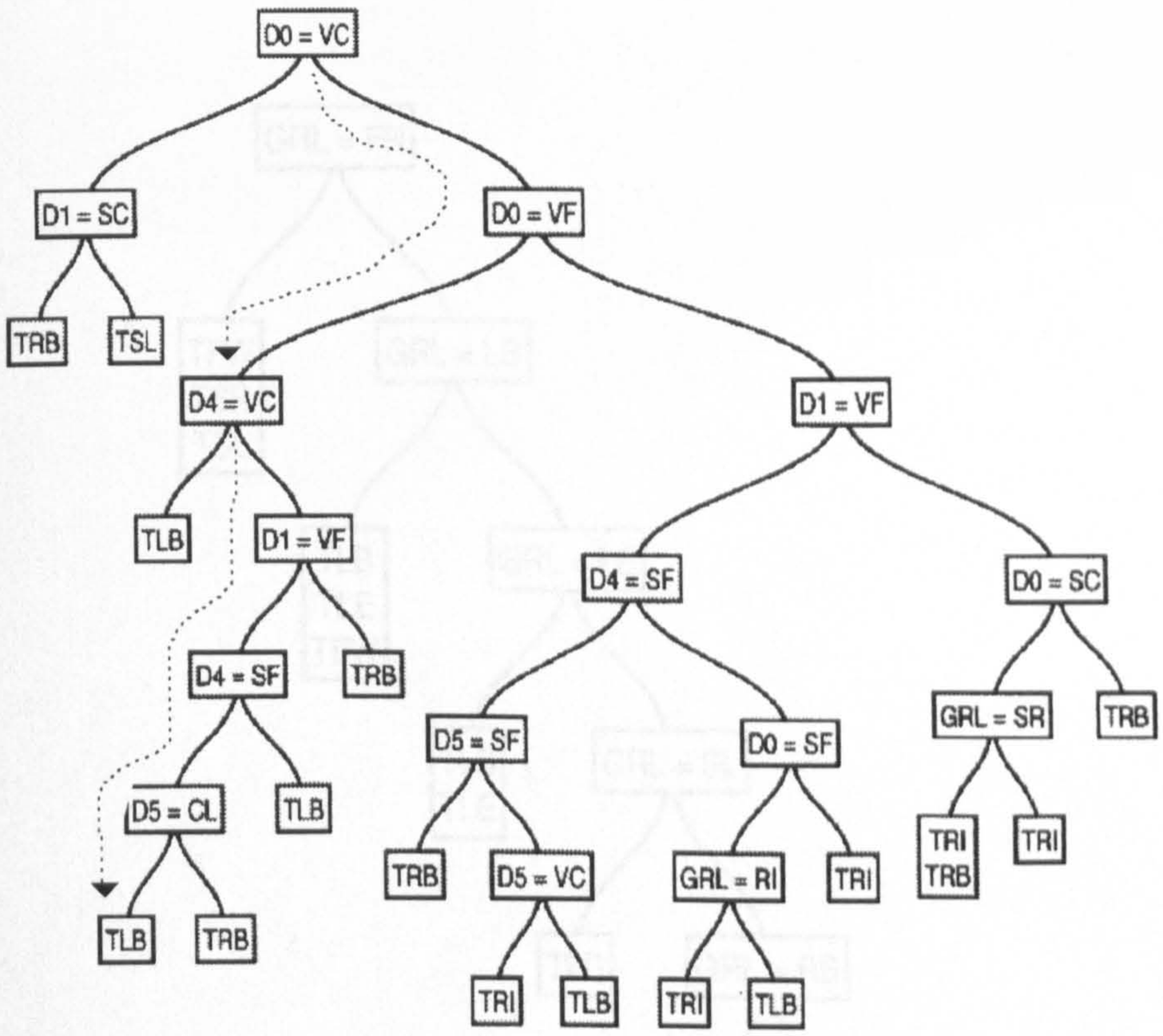


Figure 7.11 Searching the space of a DT-based FAM (representing a complete w_2). The broken directed lines show an example path to classify a perceptual pattern.

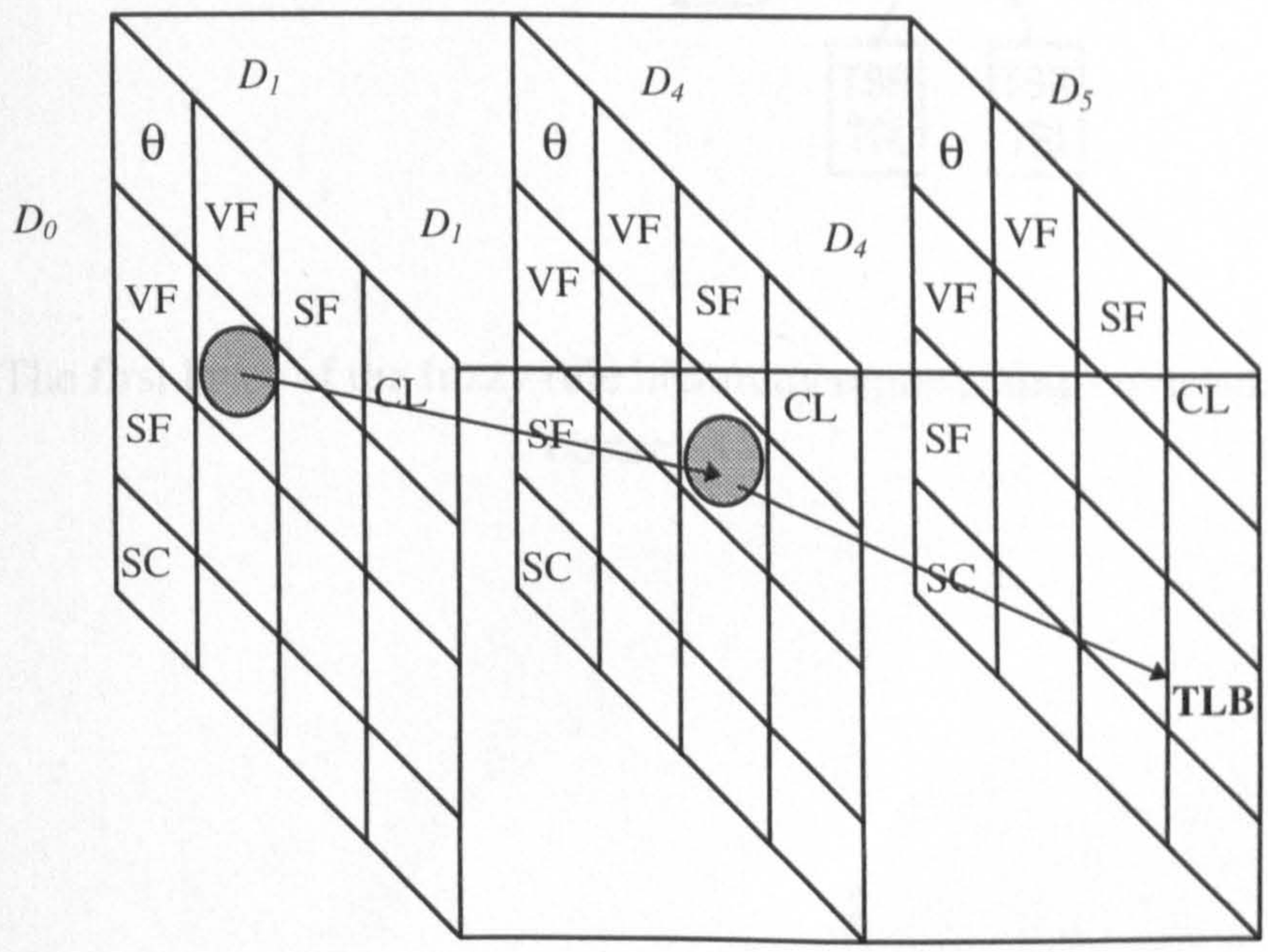


Figure 7.12 Mapping the multi-dimensional input vector of Figure 7.11 (directed path) on a hierarchy of nested 2D-FAMs (representing a portion of w_2)

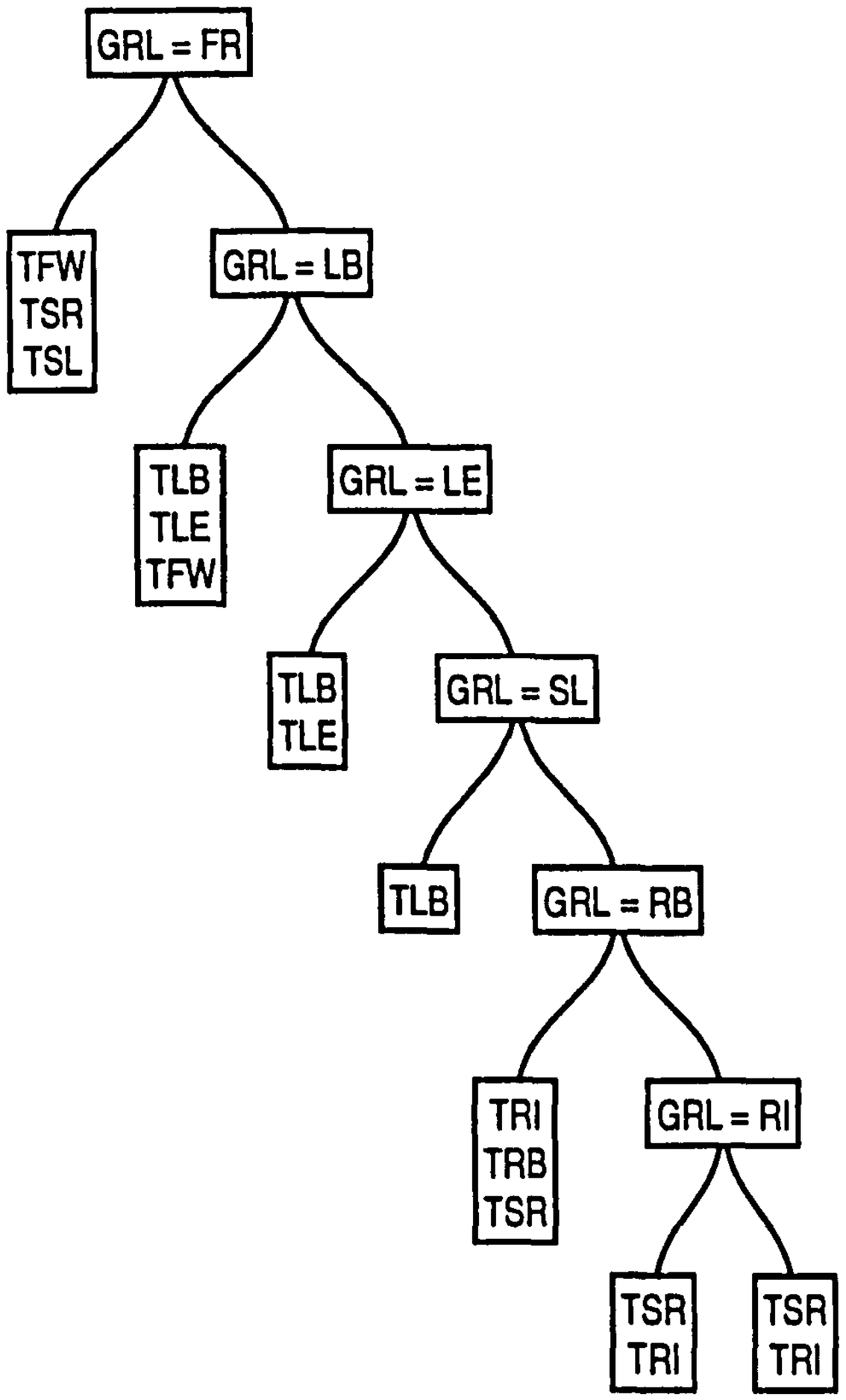


Figure 7.13 The first layer of the fuzzy rule hierarchy representing environments with no obstacles

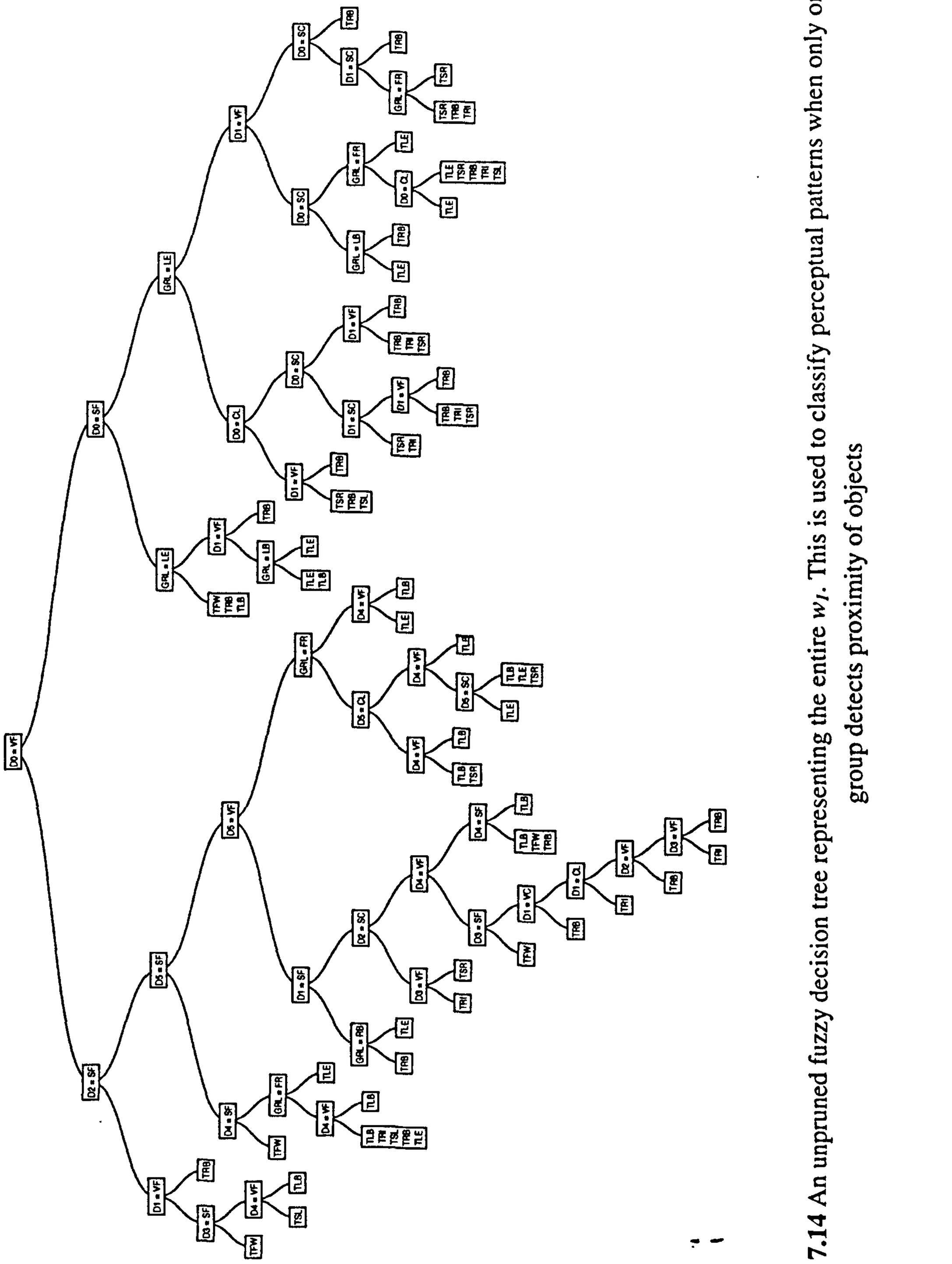


Figure 7.14 An unpruned fuzzy decision tree representing the entire w_1. This is used to classify perceptual patterns when only one sensor group detects proximity of objects

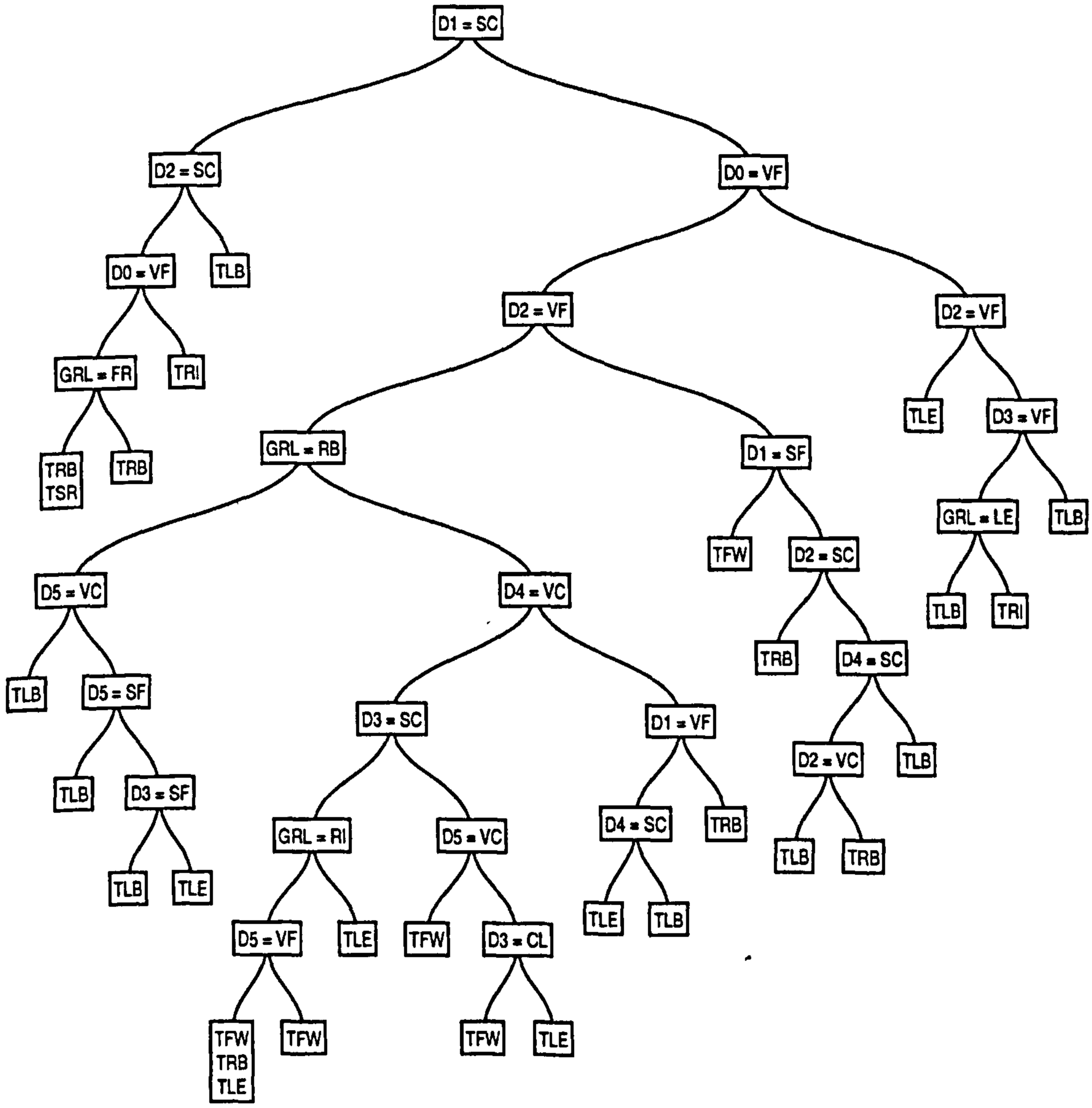


Figure 7.15 An unpruned version of the FDT representing w2

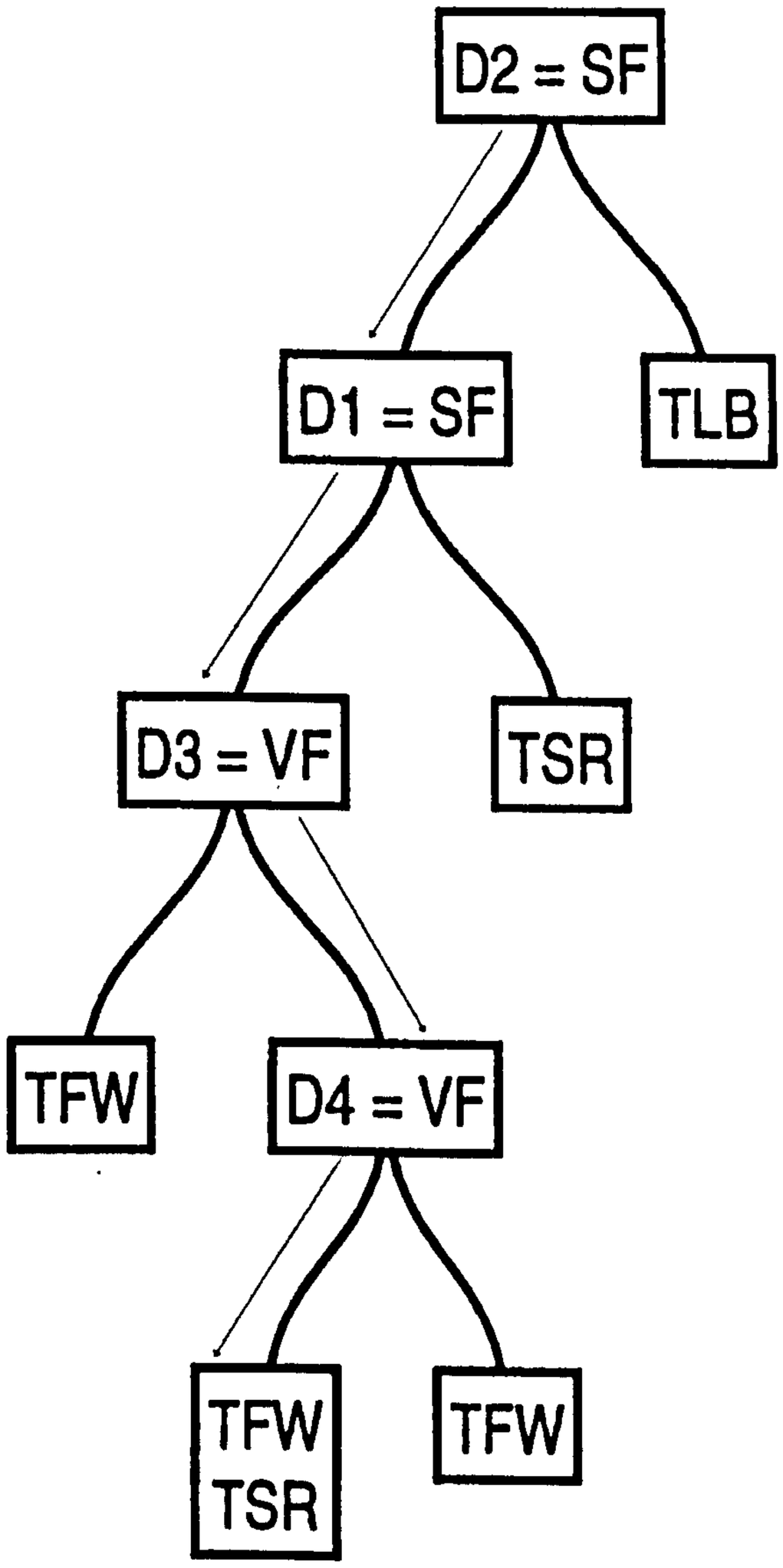


Figure 7.16 A more generalised FDT for the environments with more complex-shaped obstacles and configurations, namely w_3

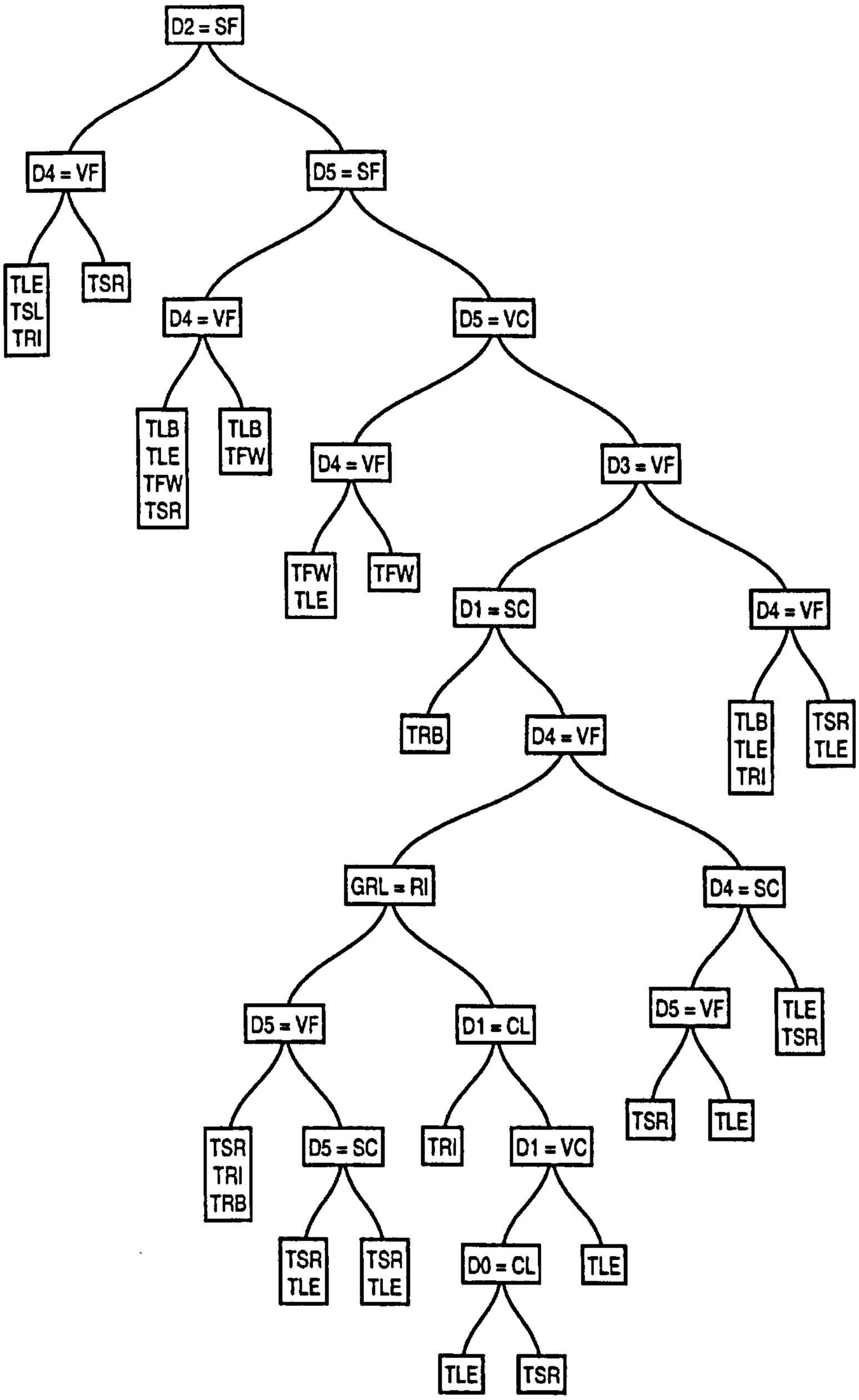


Figure 7.17 An unpruned version of the FDT used for wall-following scenarios after a long wall is detected, and this describes w_4.

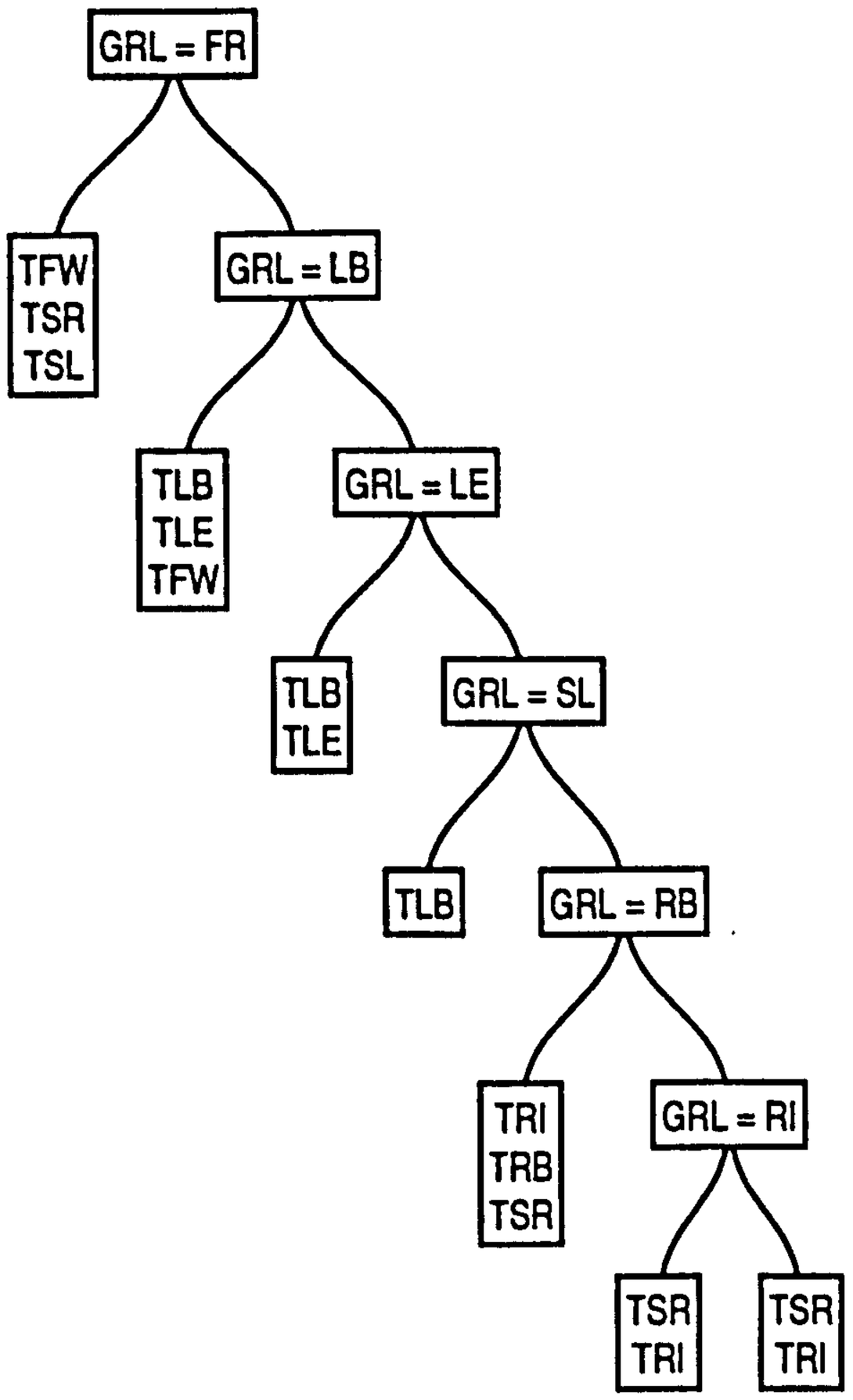


Figure 7.21 An FDT representing w0 after performing virtual pruning. It is evident that pruning has not affected this FDT as it is in its general form.

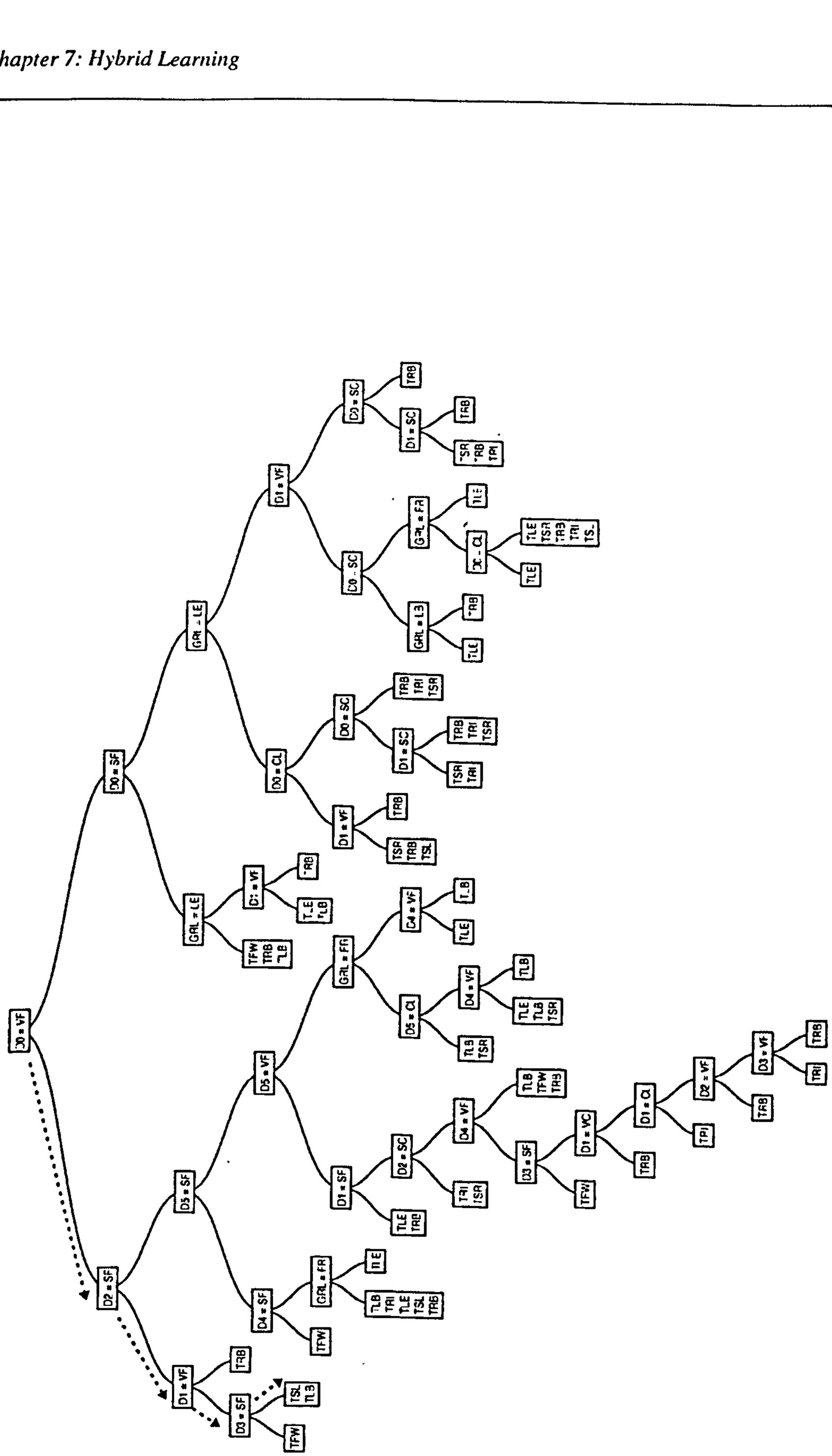


Figure 7.22 The pruned version of the fuzzy decision tree shown in Figure 7.14 which is 15-fold cross validated and is used for perception classification in sparsely obstacle-populated environments

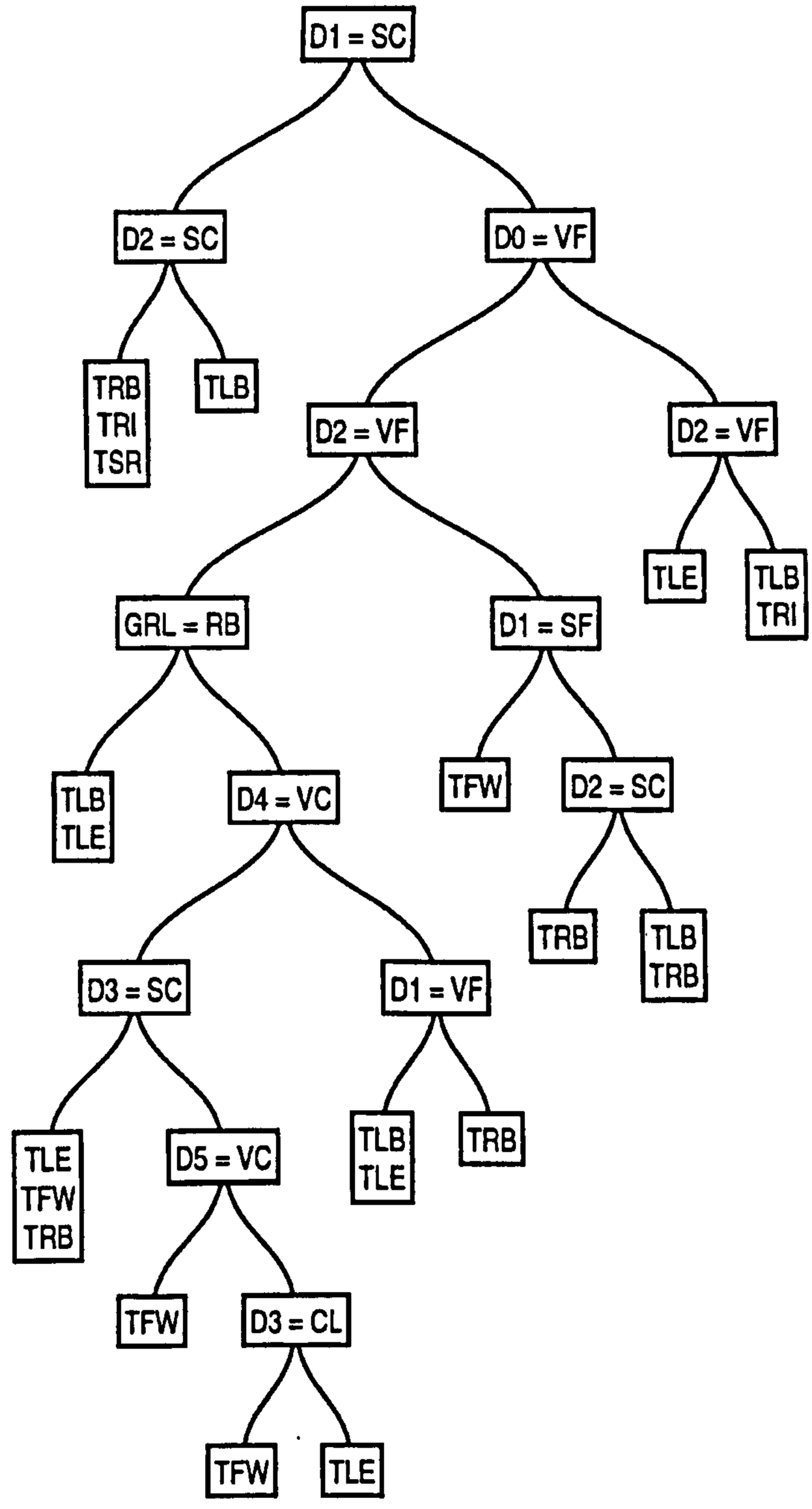


Figure 7.23 The truncated version of FDT shown in Figure 7.15 representing w_2

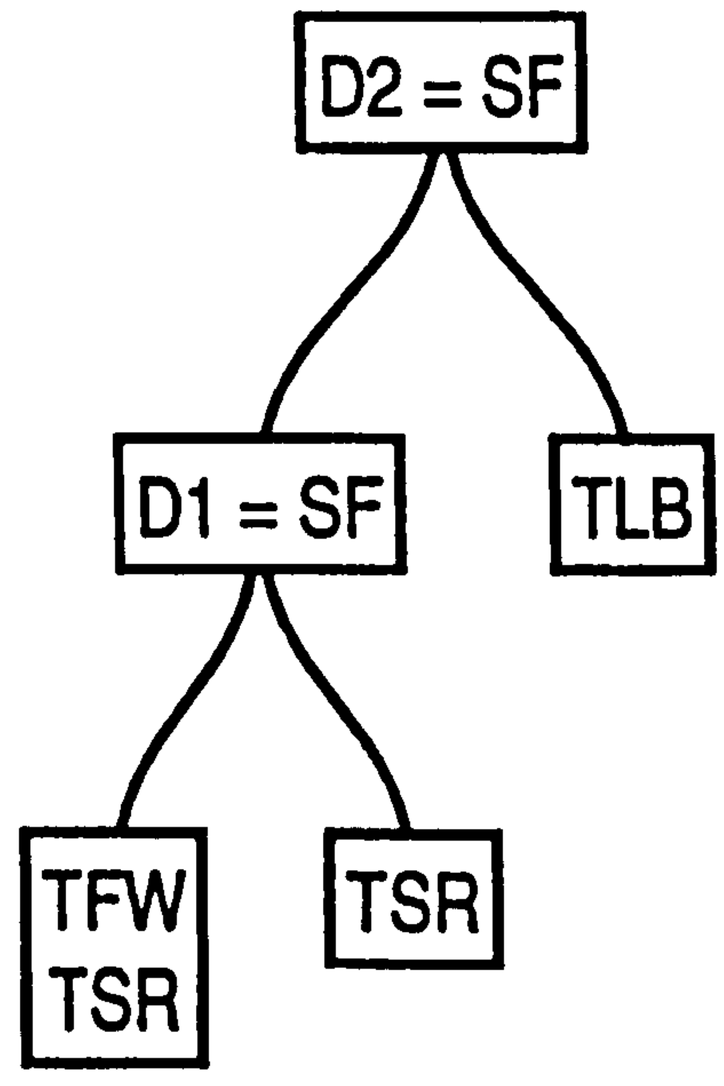


Figure 7.24 The truncated version of FDT shown in Figure 7.16 representing w_2. It is evident that this FDT tend to overgeneralise.

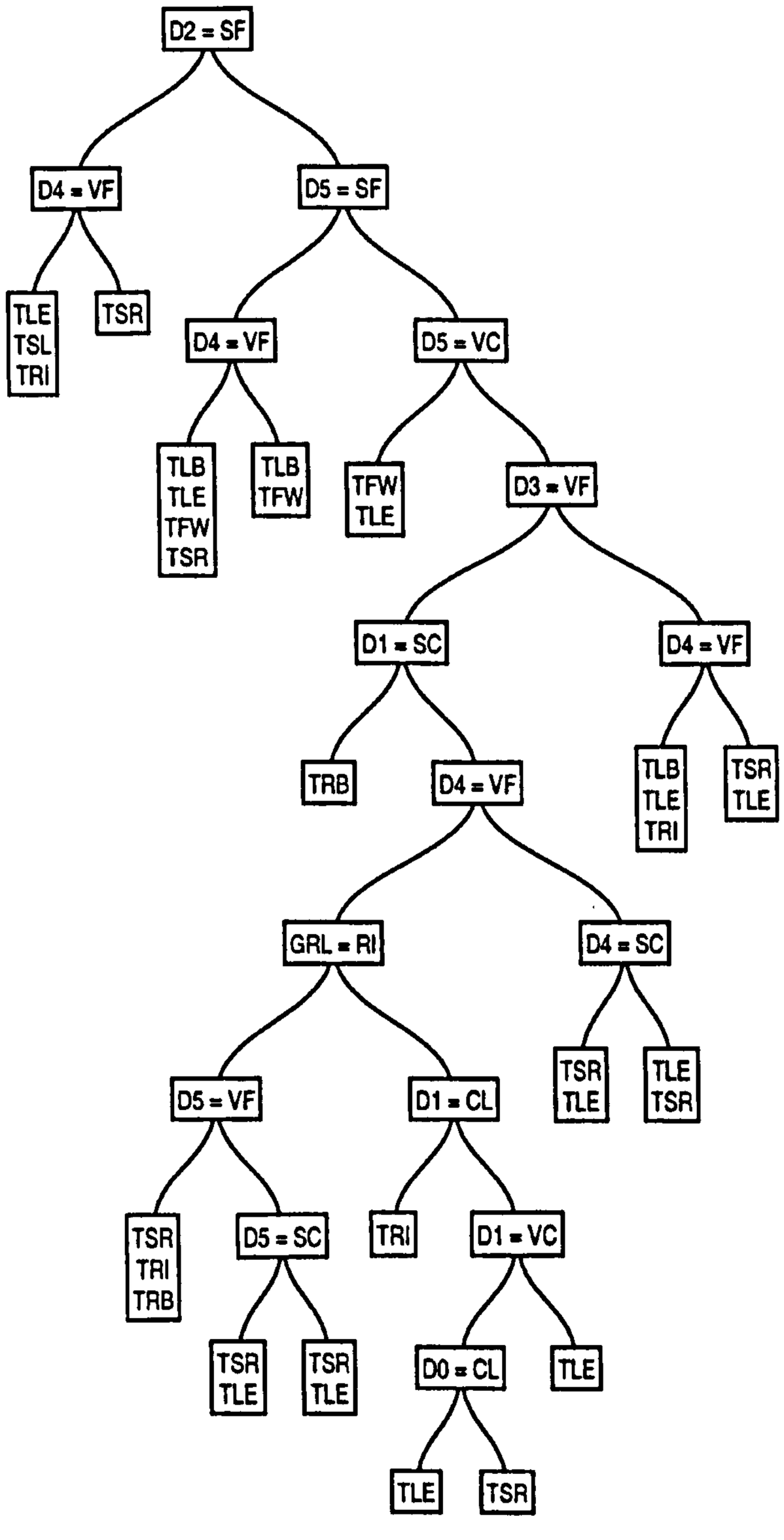


Figure 7.25 The FDT activated in wall-following situations after pruning

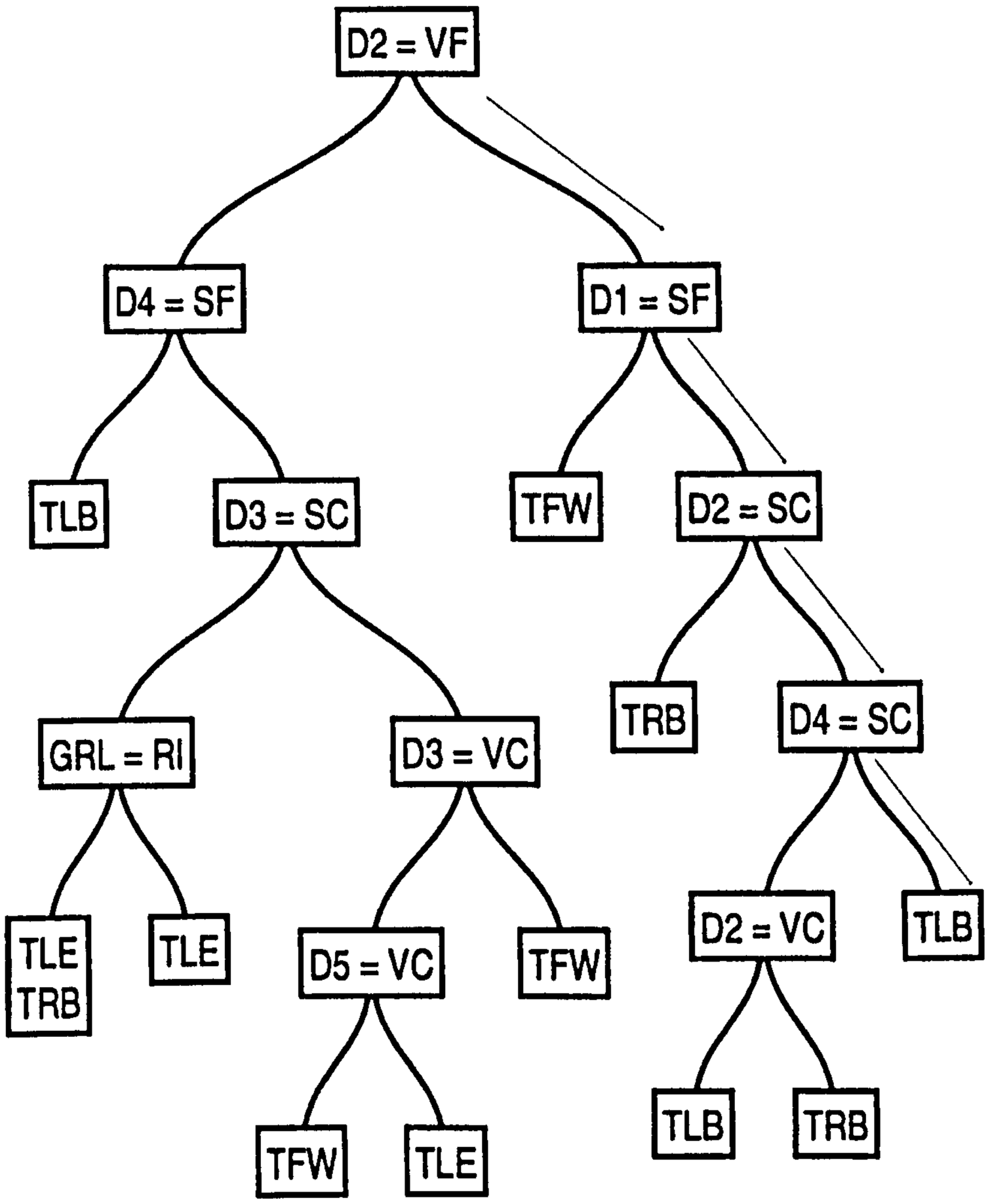
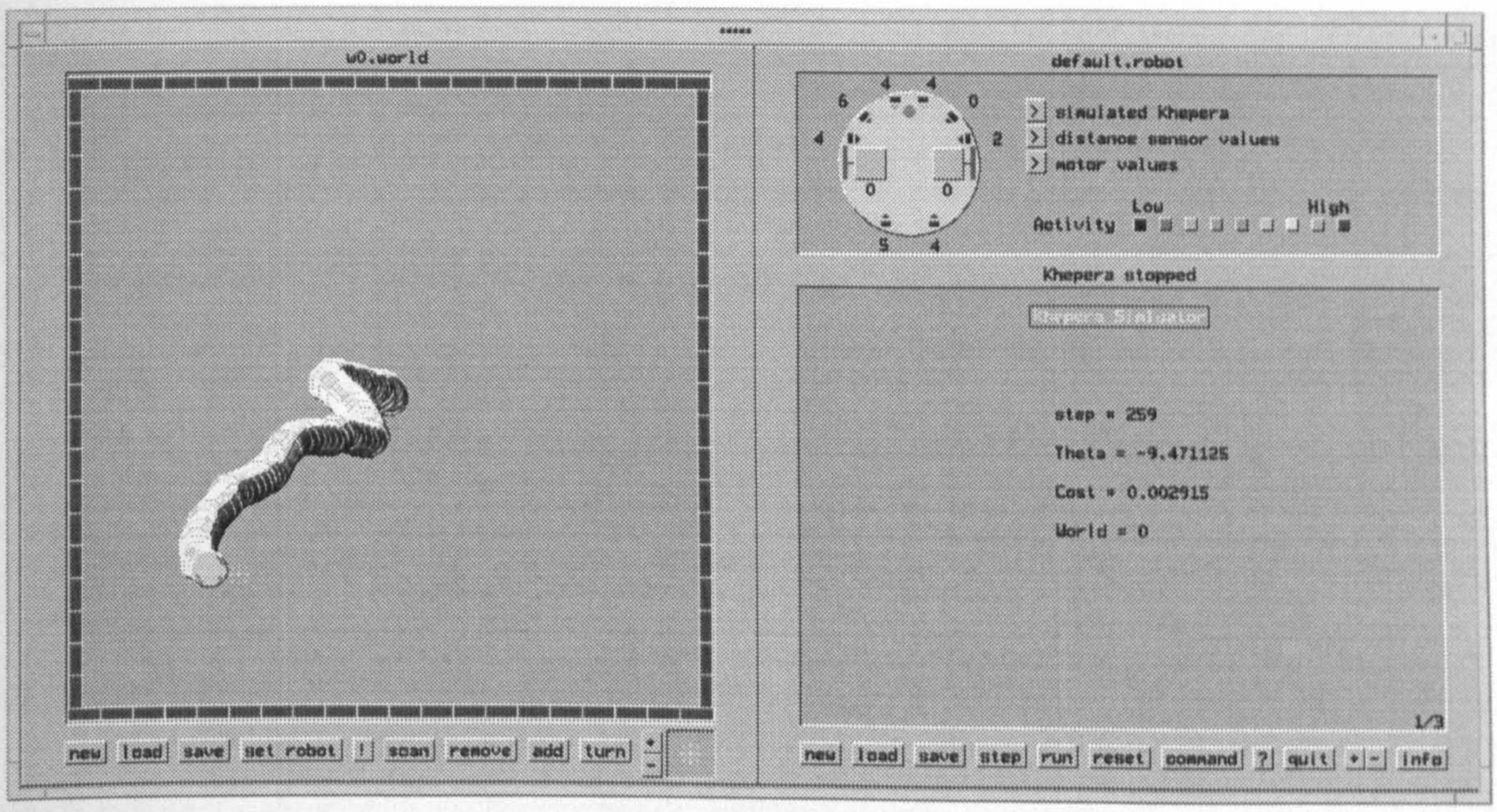
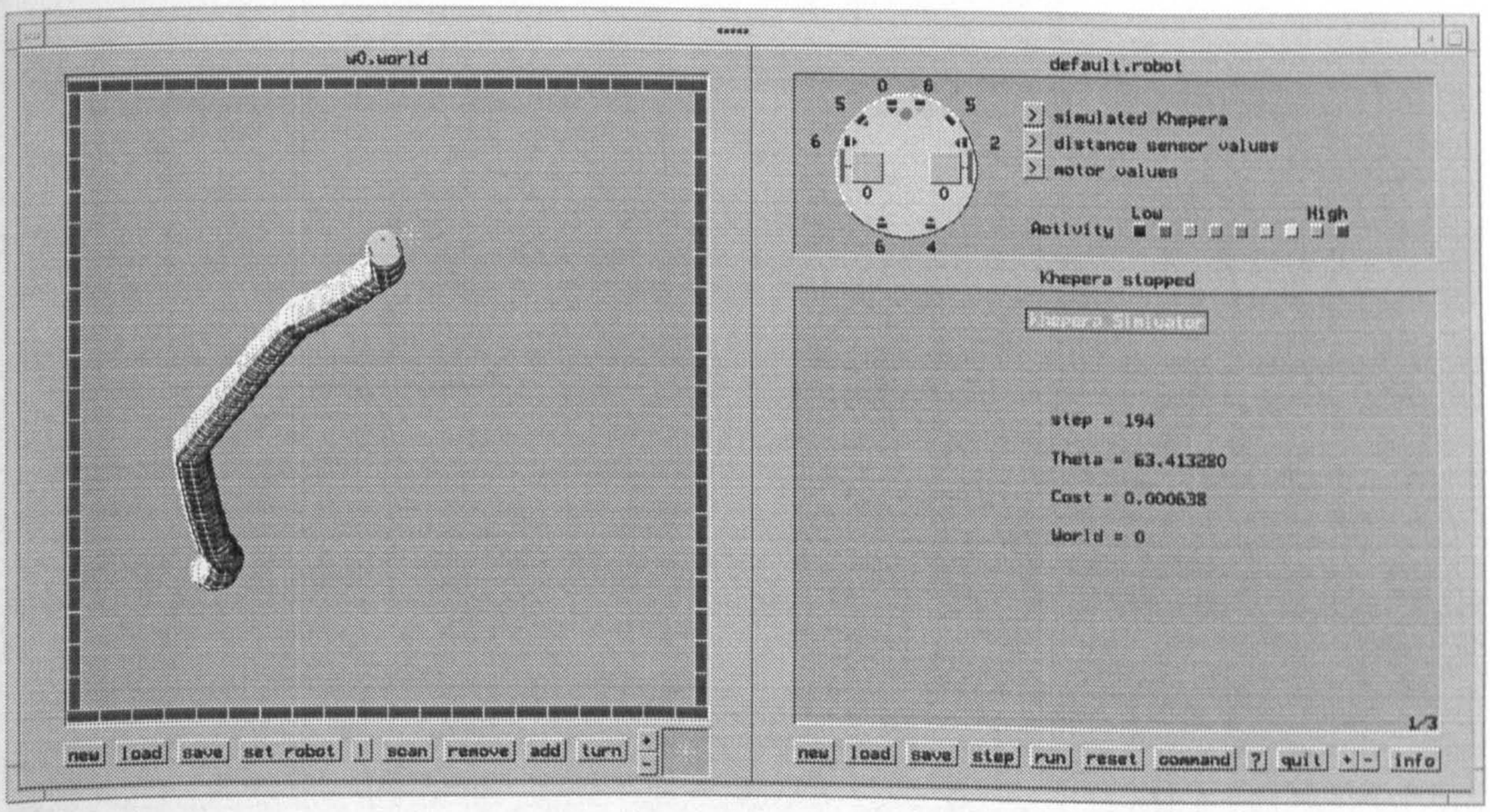


Figure 7.26 The FDT grown on a small batch of fuzzy data. The searched path is specified by a directed line.

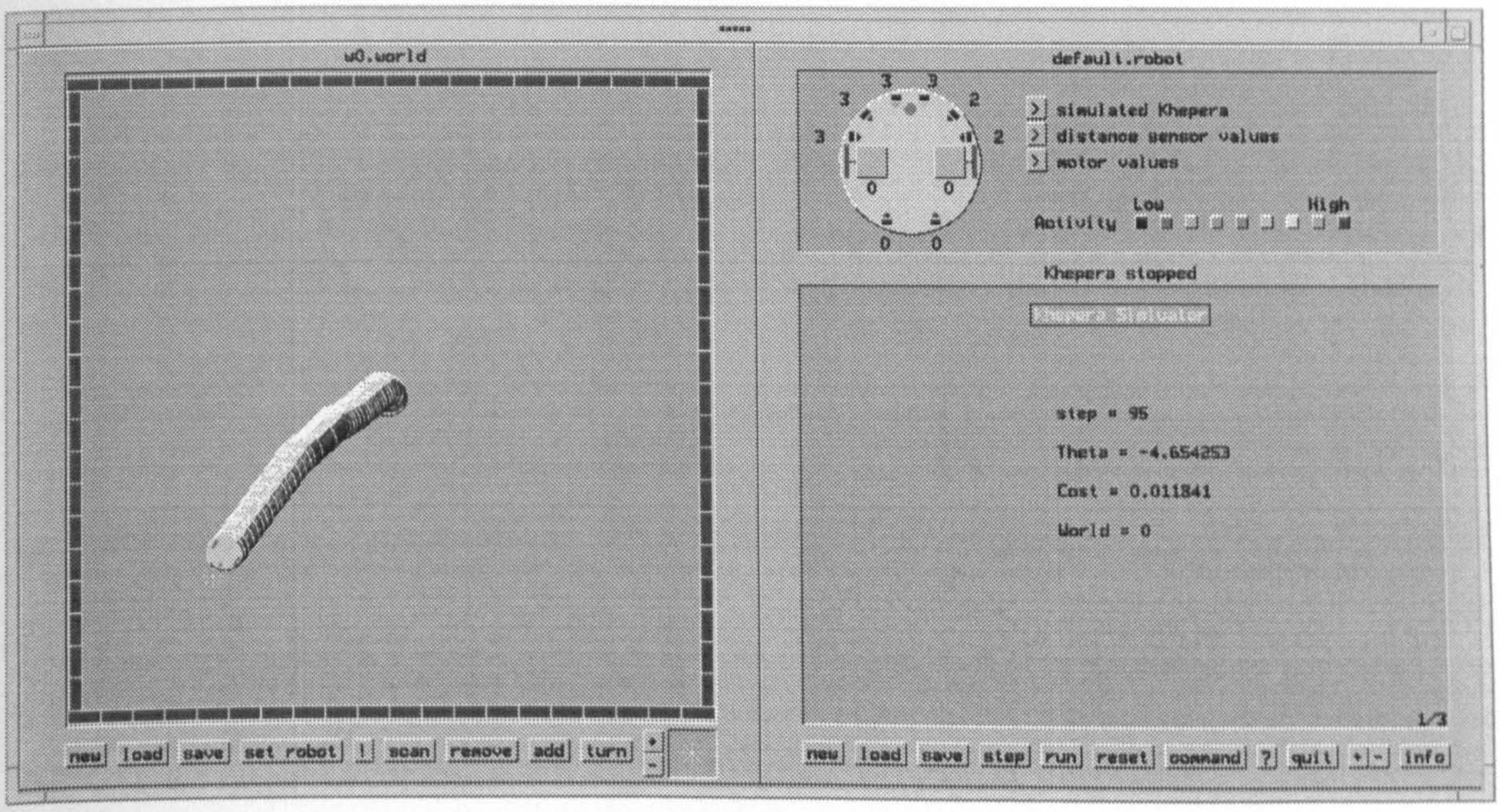


(a) The first stage of learning from inception and initiating the FDT hierarchy

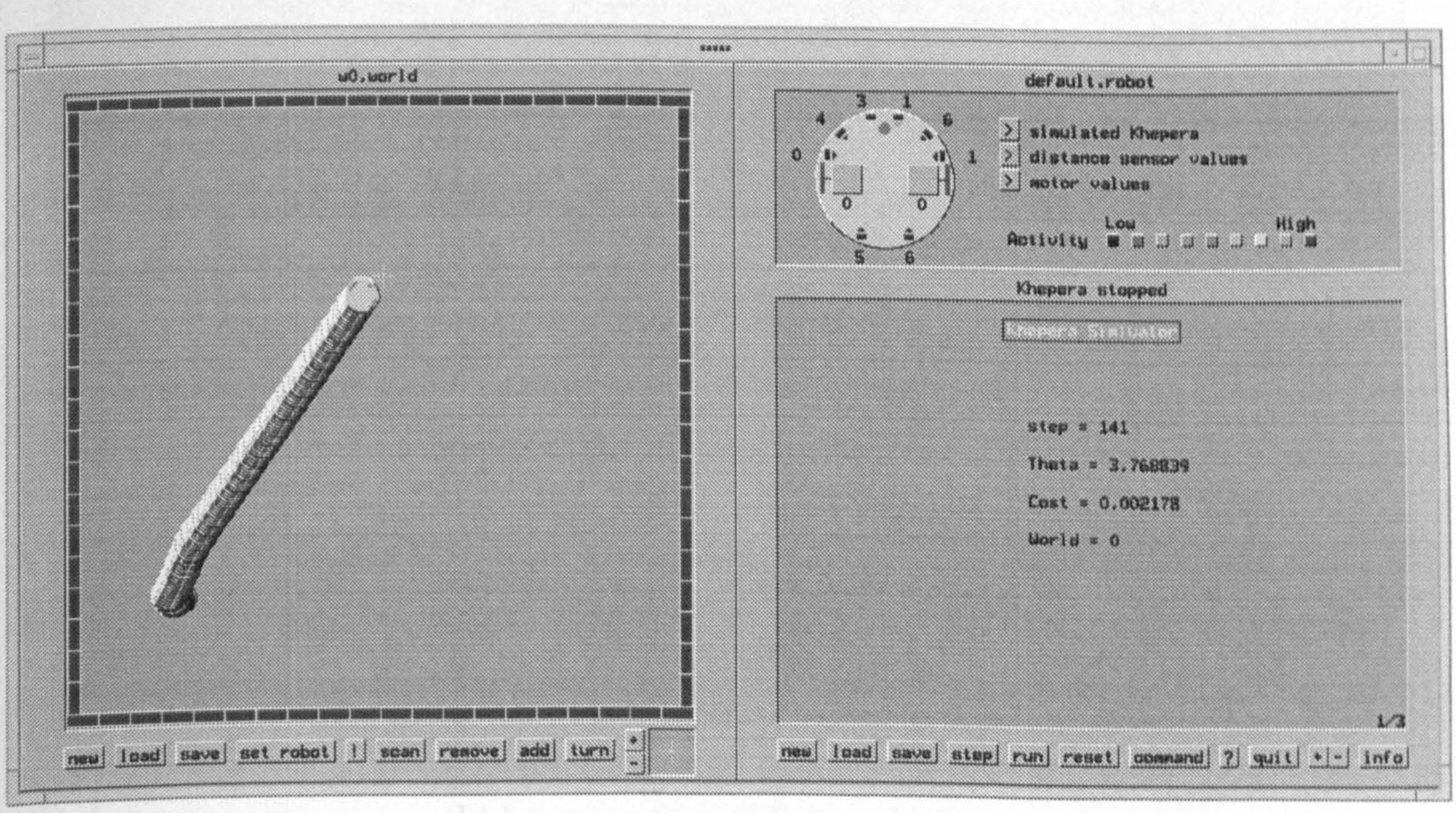


(b) An intermediate stage of learning in w0 (environments without obstacles)

Figure 7.27 Training in the world without obstacles

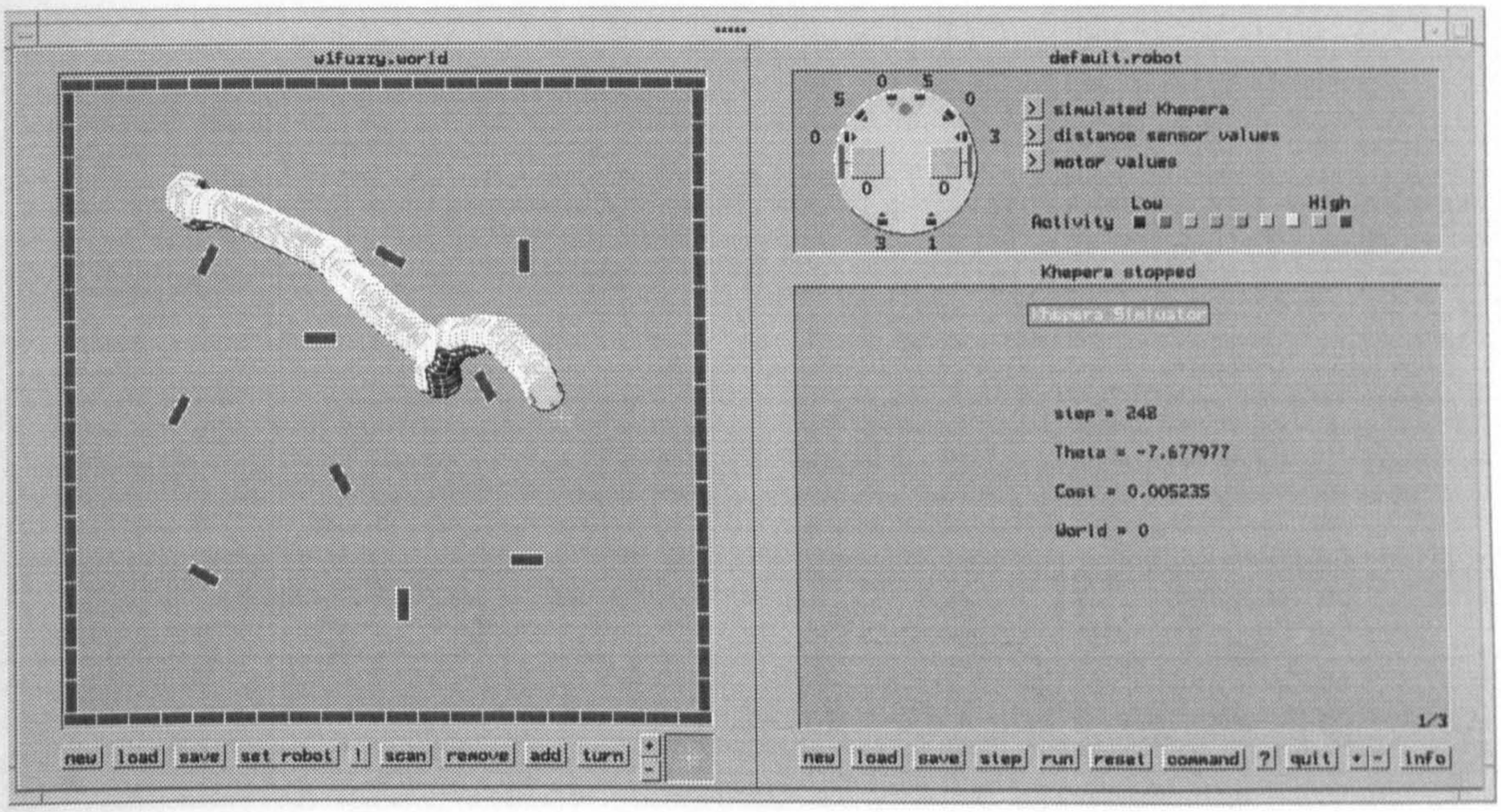


(a) Navigation in w_0 using the generated FDT

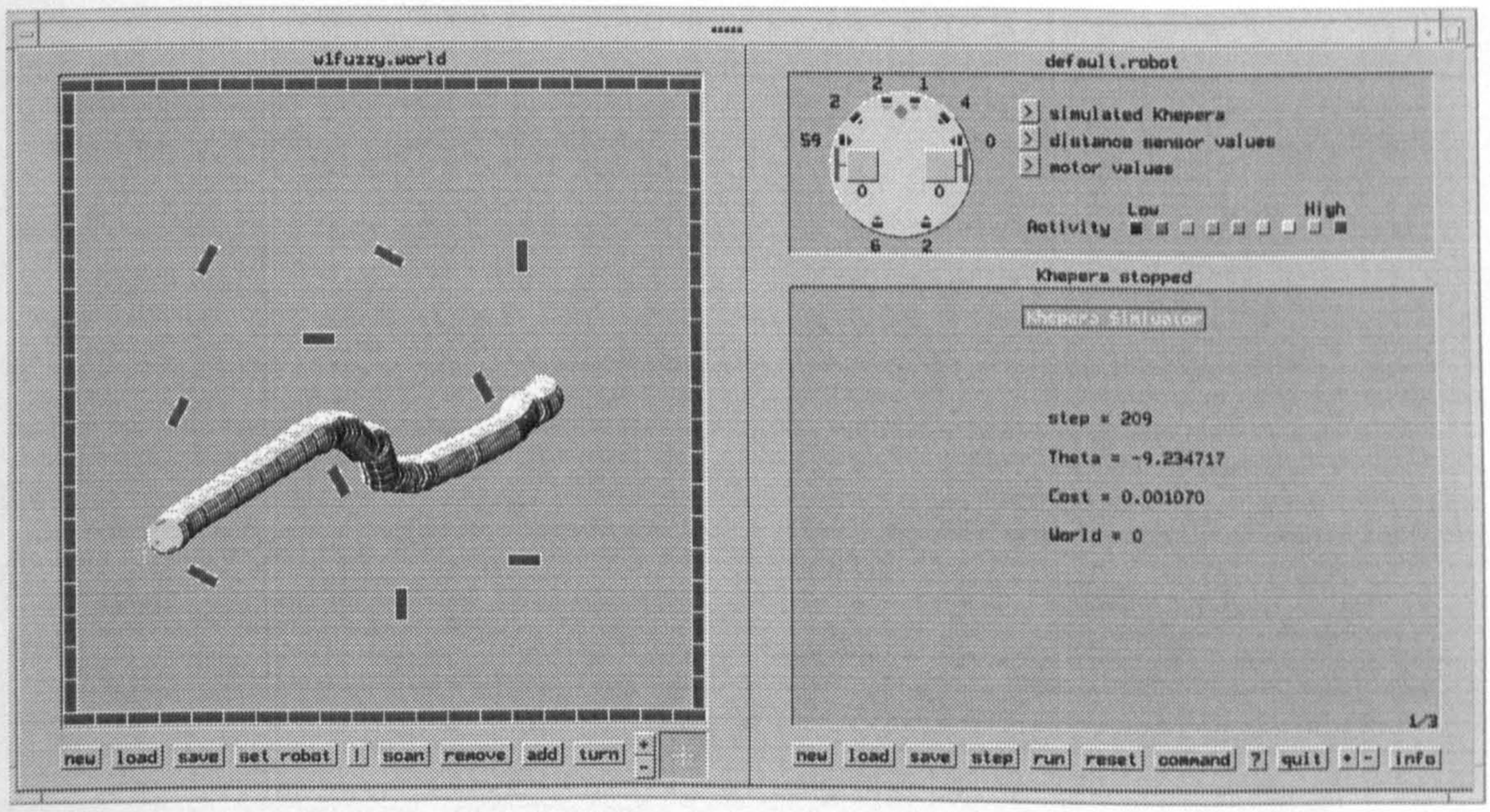


(b) Another scenario of homing task in w_0

Figure 7.28 Homing tasks in an environment without obstacles after the FDT-0 has been grown

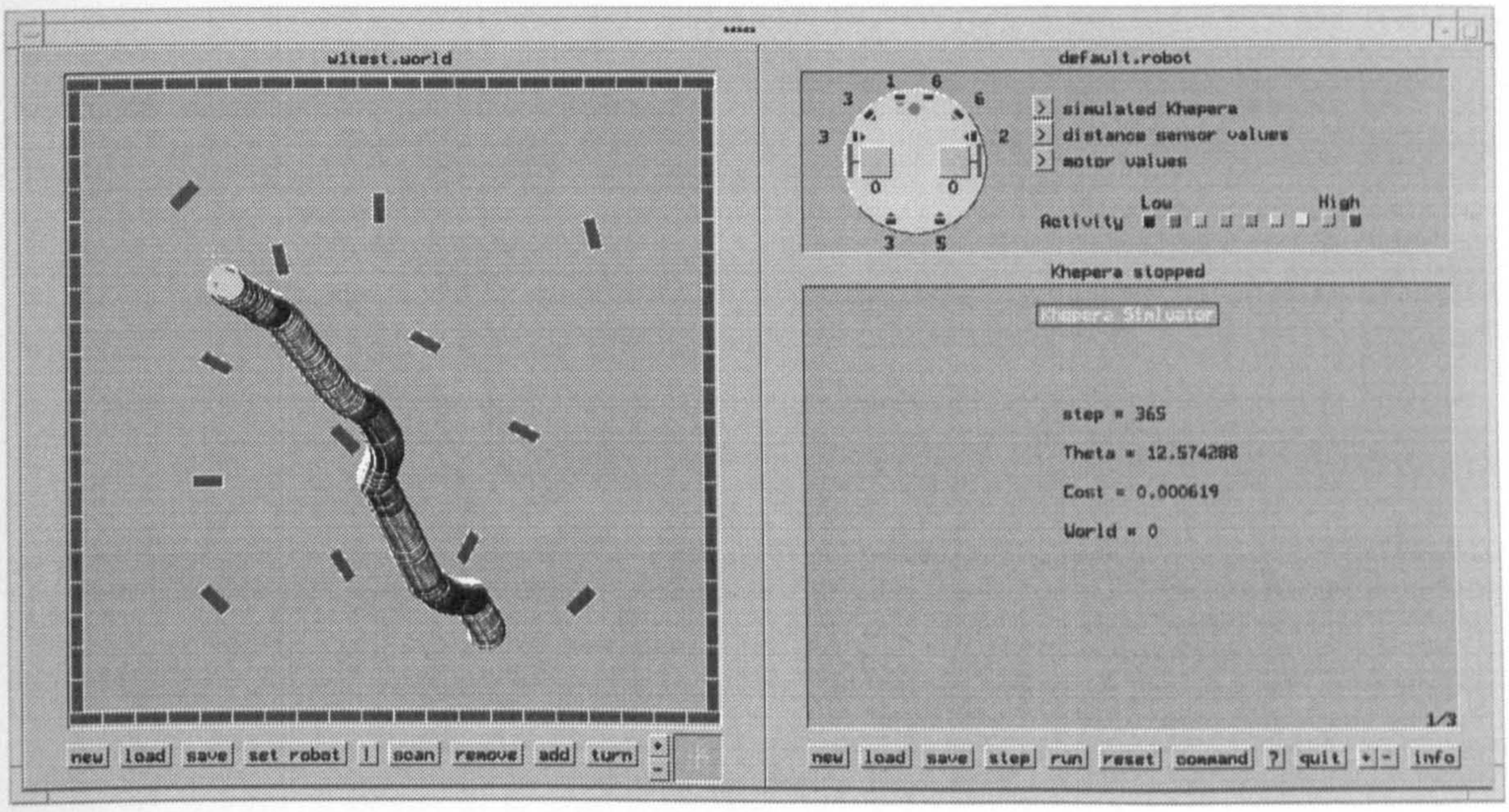


(a) A typical initial stage of learning in w_1

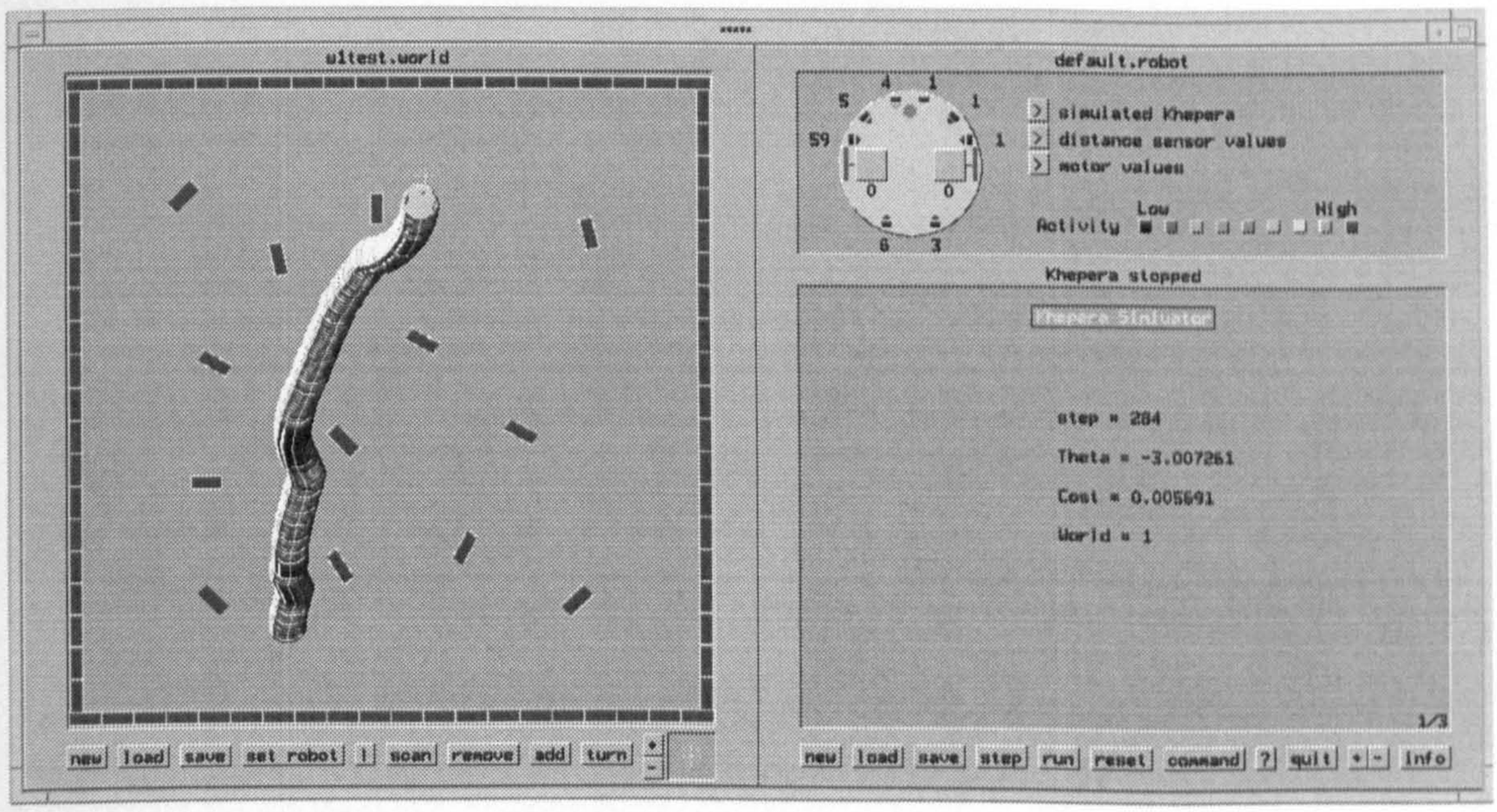


(b) Learning to avoid obstacles in w_1

Figure 7.29 Learning to avoid obstacles while homing on a set target. This is the first stage of learning to grow FDT-1.

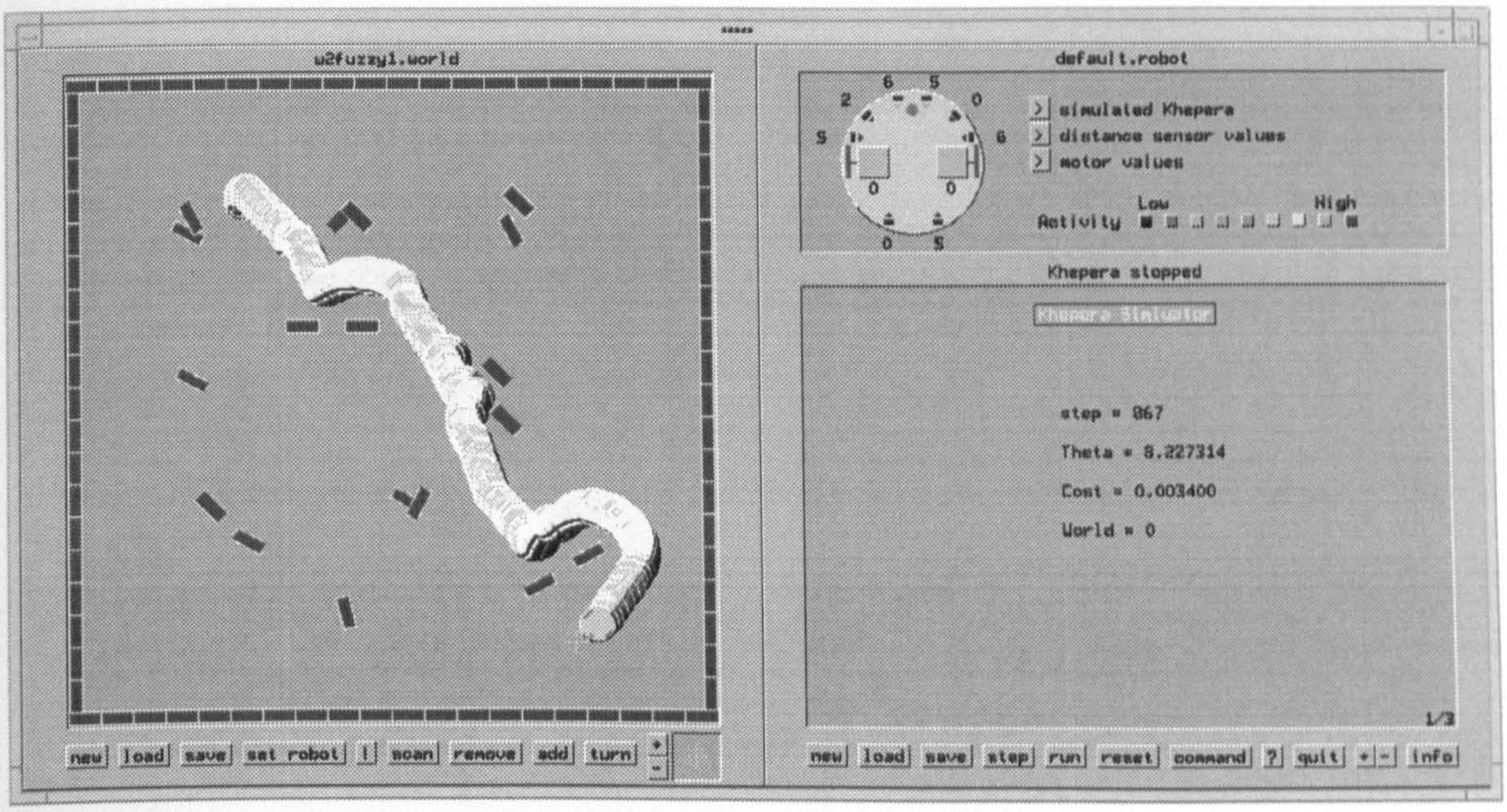


(a) Navigation after learning in w_1 is established

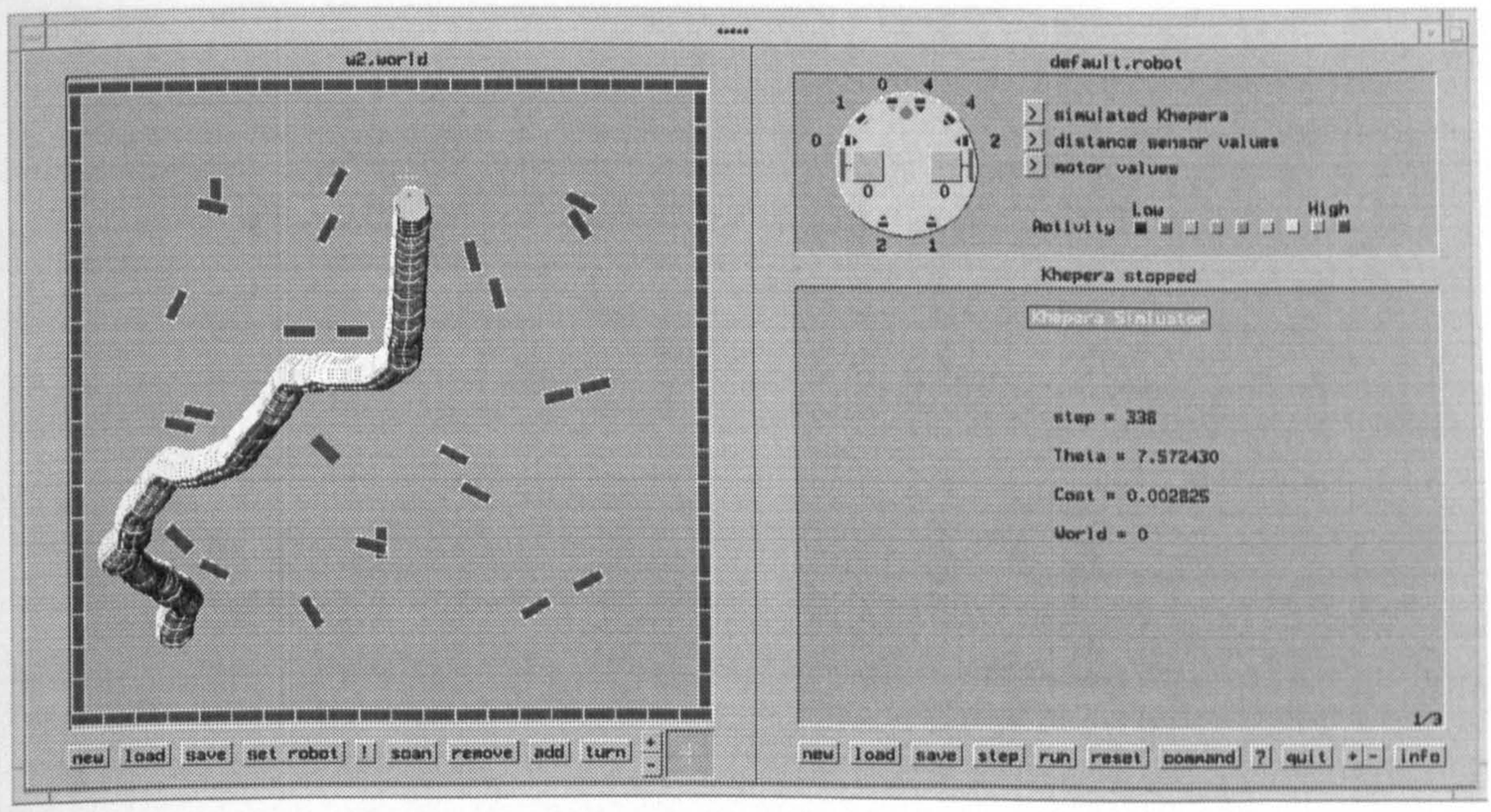


(b) Improvement in target-seeking behaviour in w_1

Figure 7.30 Navigatory tasks in w_1 when the robot has access to both FDT-0 and FDT-1

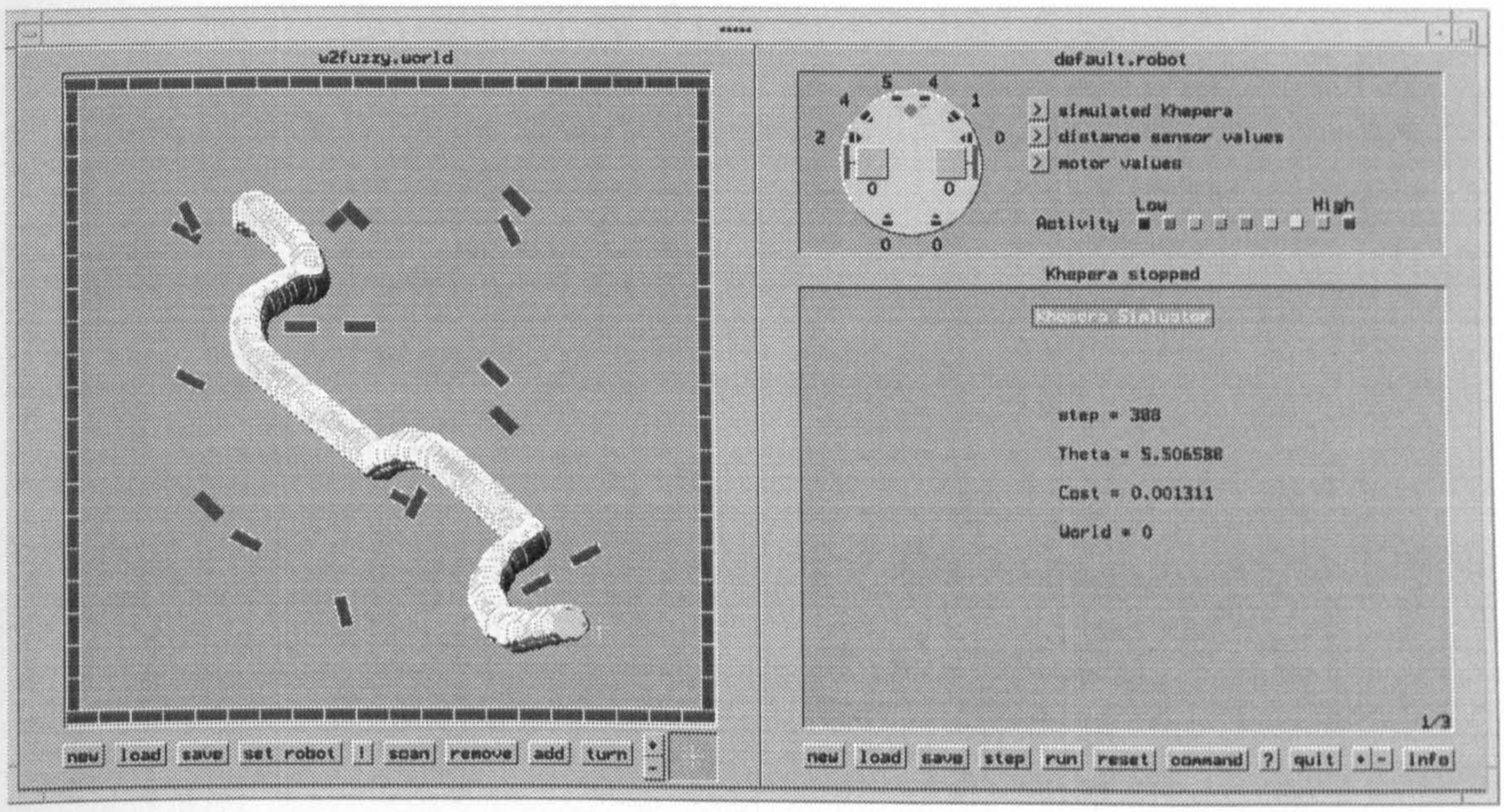


(a) Learning to improve reactivity in w_2

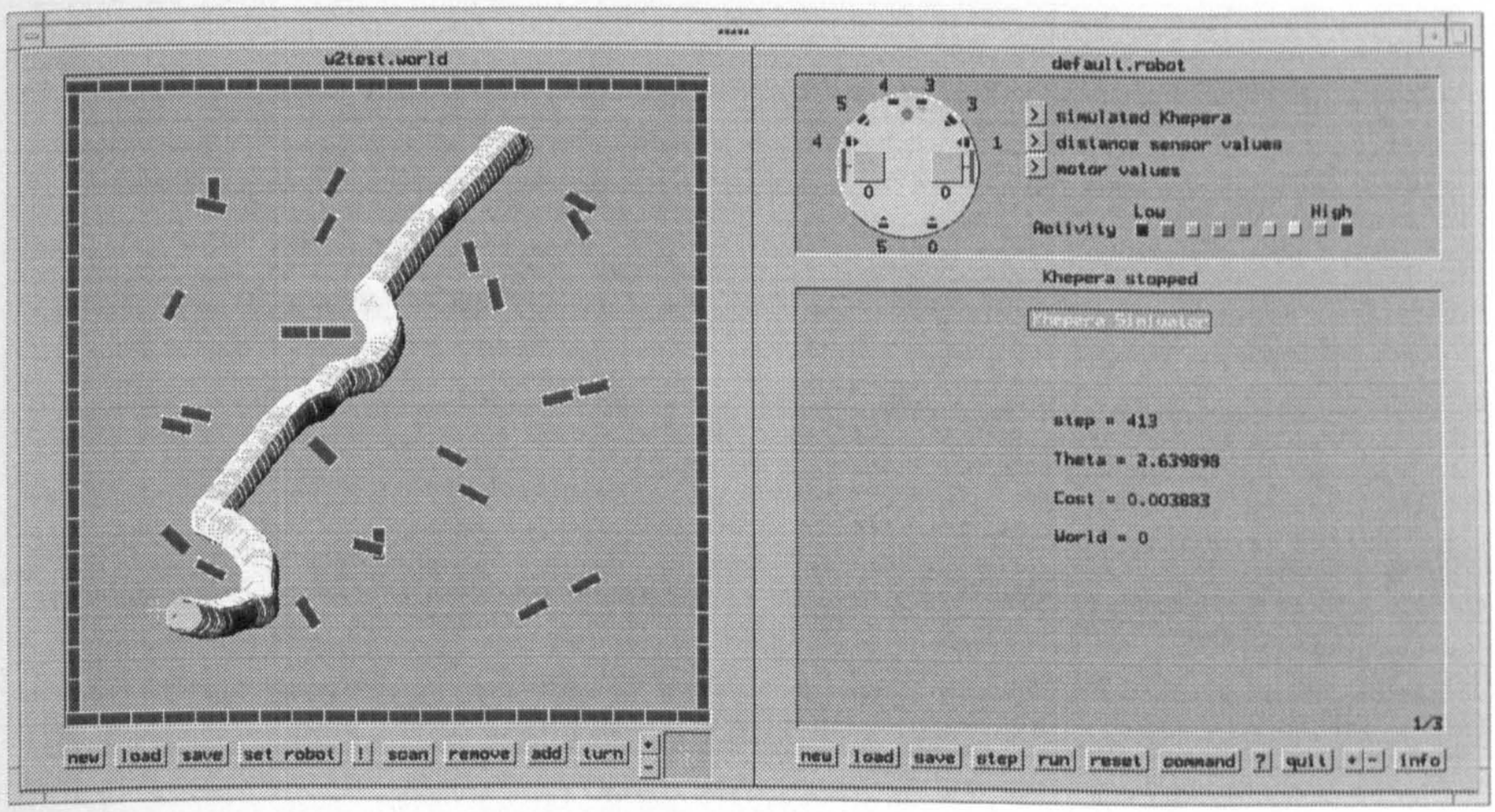


(b) A preliminary stage of learning and adaptation in w_2

Figure 7.31 Training the robot to cope with more complex object configurations to build FDT-2

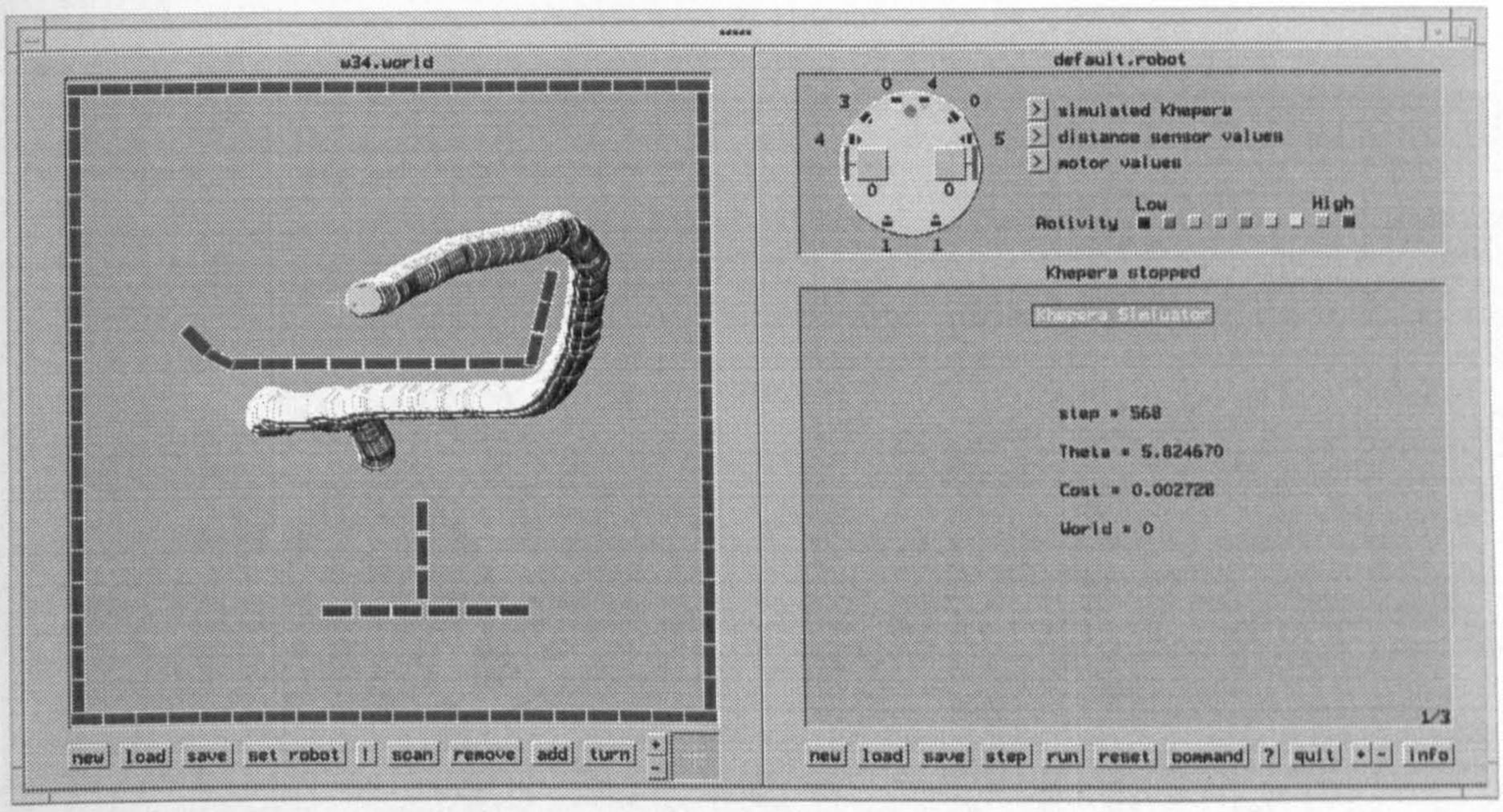


(a) A Navigation task in w_2

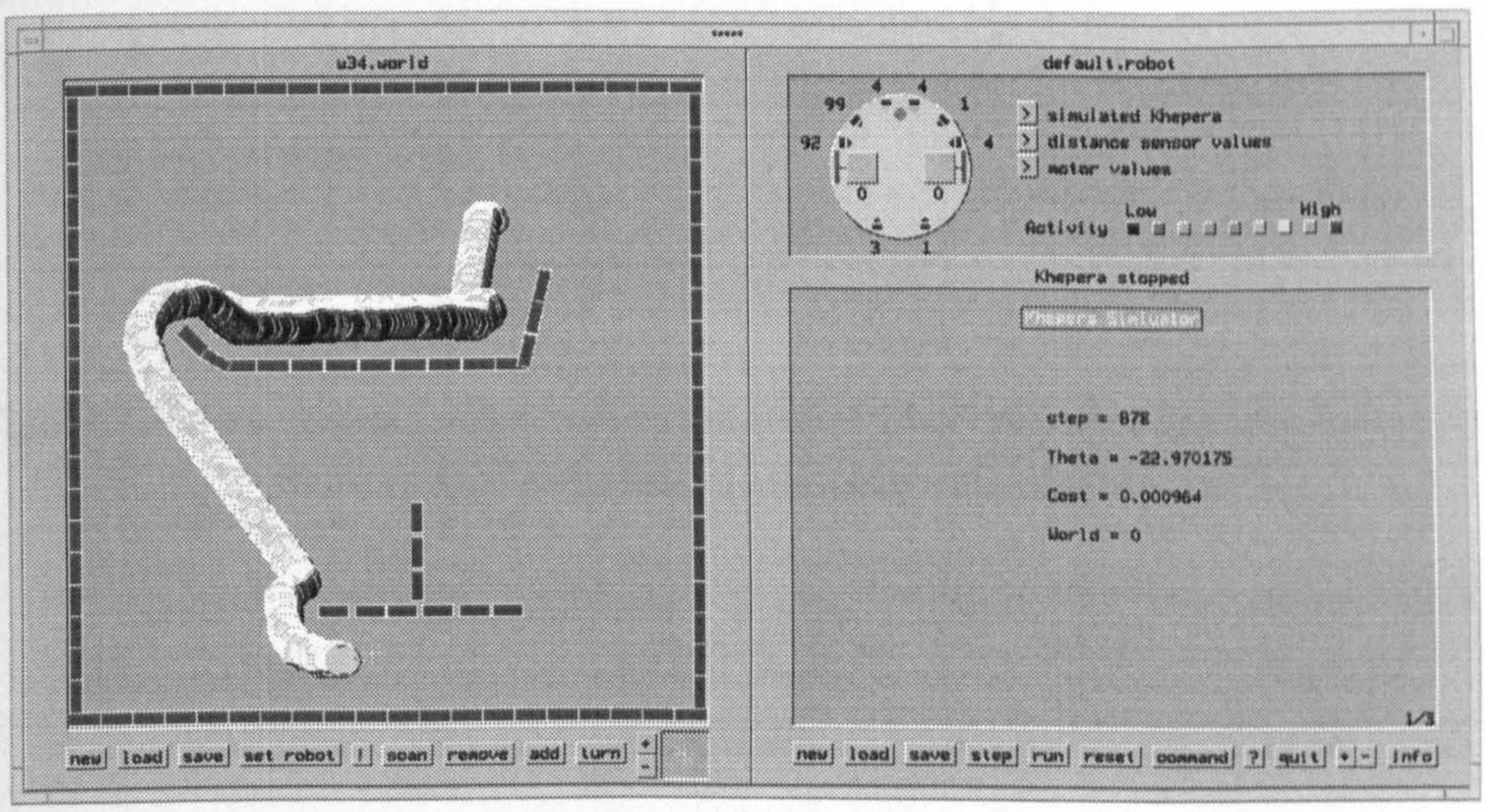


(b) Another example of homing task after FDT-2 has been grown

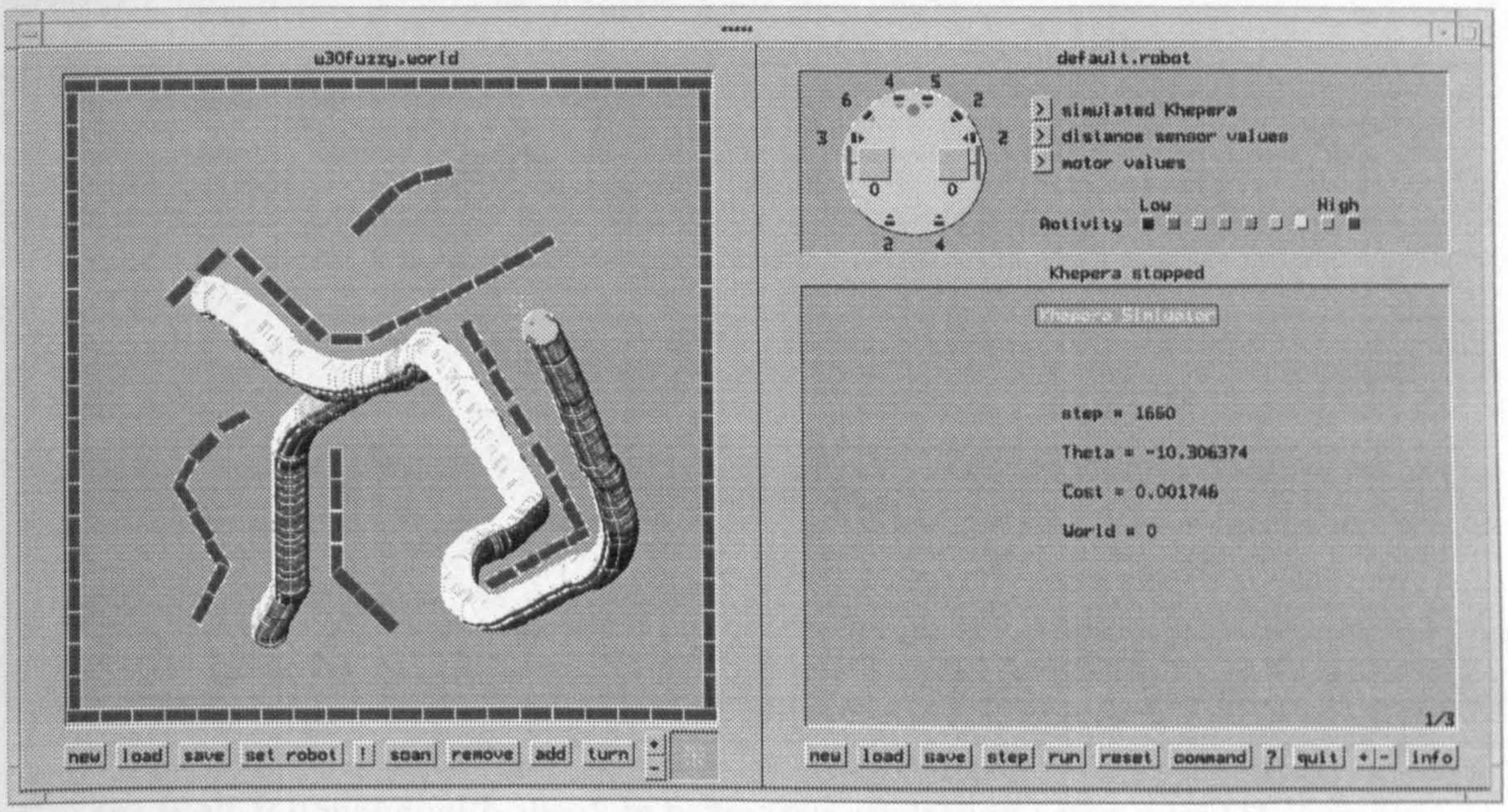
Figure 7.32 Example of homing tasks in more densely object-populated environments



(a) Learning to follow walls

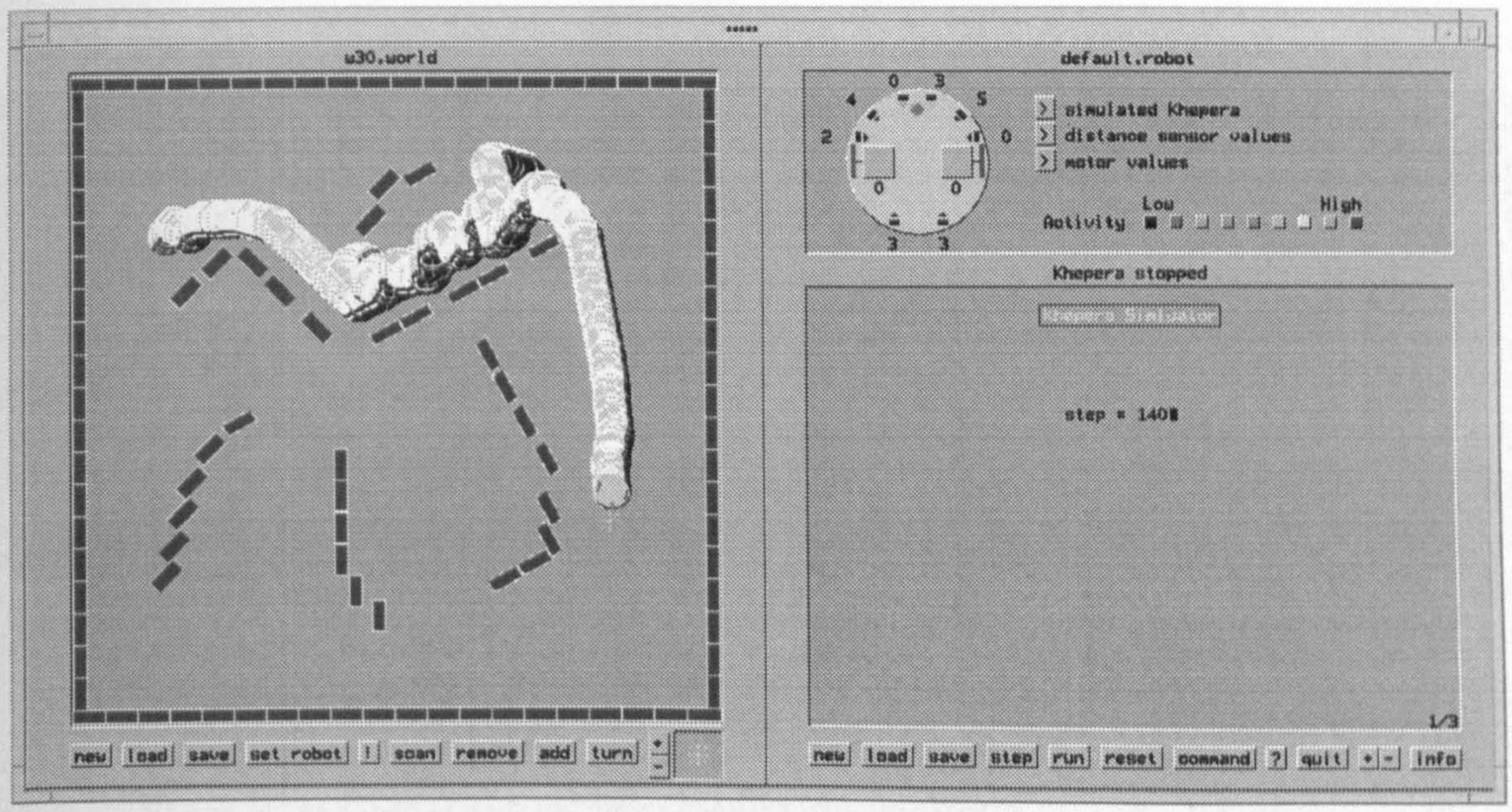


(b) Learning to follow long walls and edges

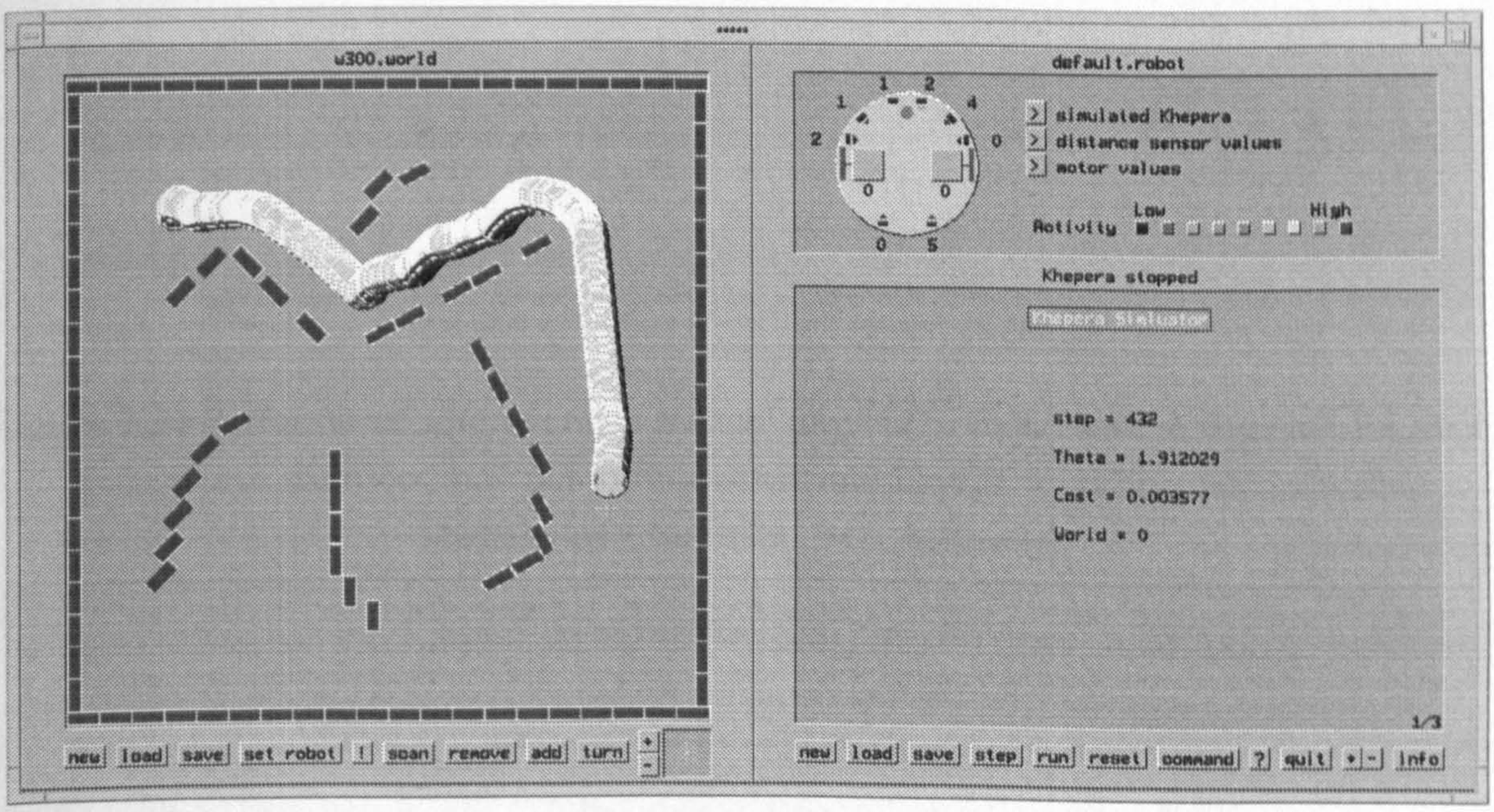


(c) Learning in an environment with arbitrary obstacle configurations (edges, walls and corners)

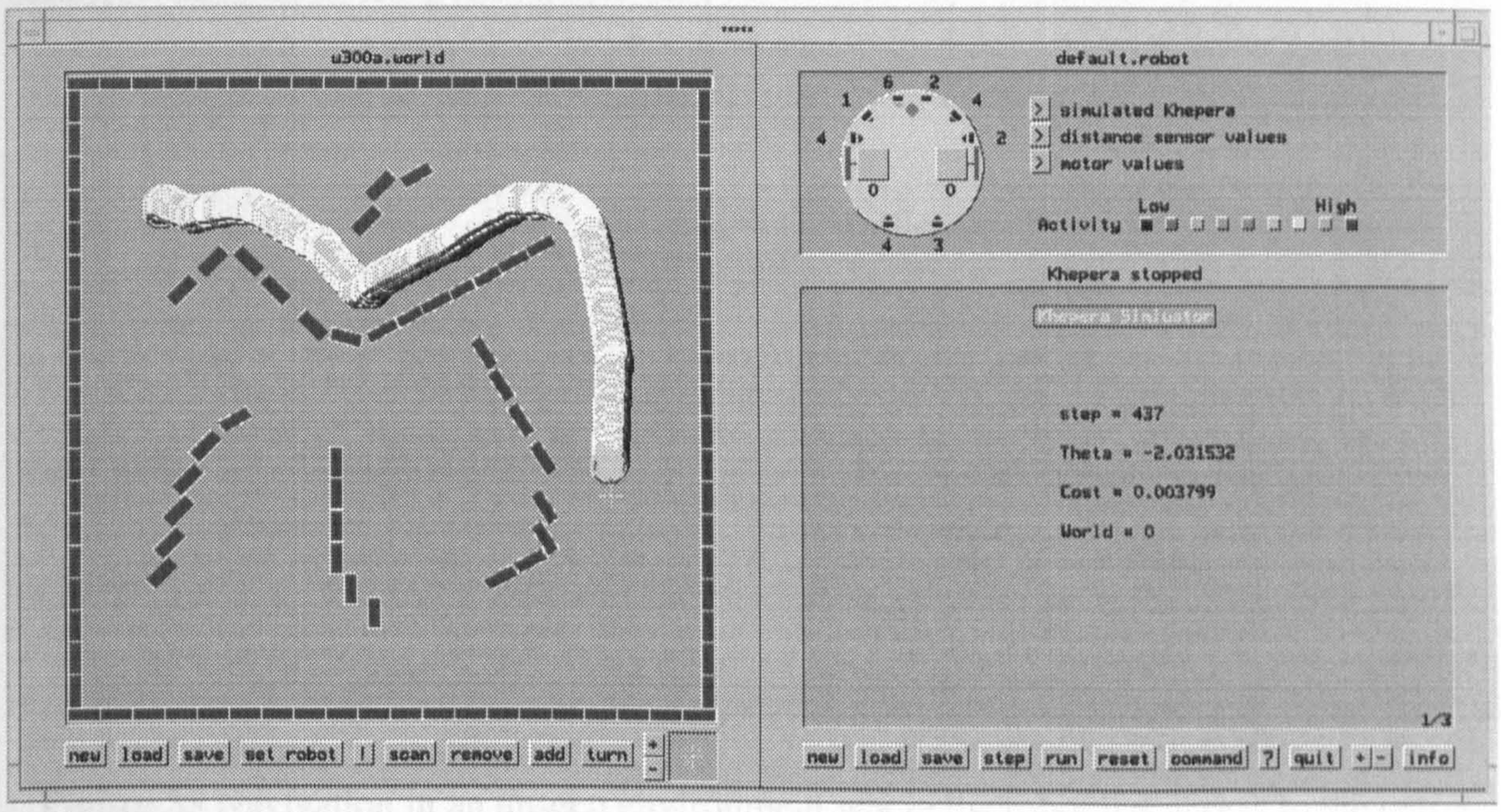
Figure 7.33 Learning examples from environments where the robot learns to follow long walls, corners and edges of arbitrarily-shaped obstacles



(**a**) Wall-following with absolute behaviour arbitration using pure DT learning



(**b**) Performance improvement in wall-following behaviour using FDT learning



(c) Smooth trajectory and behaviour in wall-following scenarios when the discontinuities are augmented

Figure 7.34 Performance improvement by applying FDTs to navigation and wall-following tasks where the robot may fail to follow smoothly longs walls due to the absolute behaviour arbitration in the hierarchy

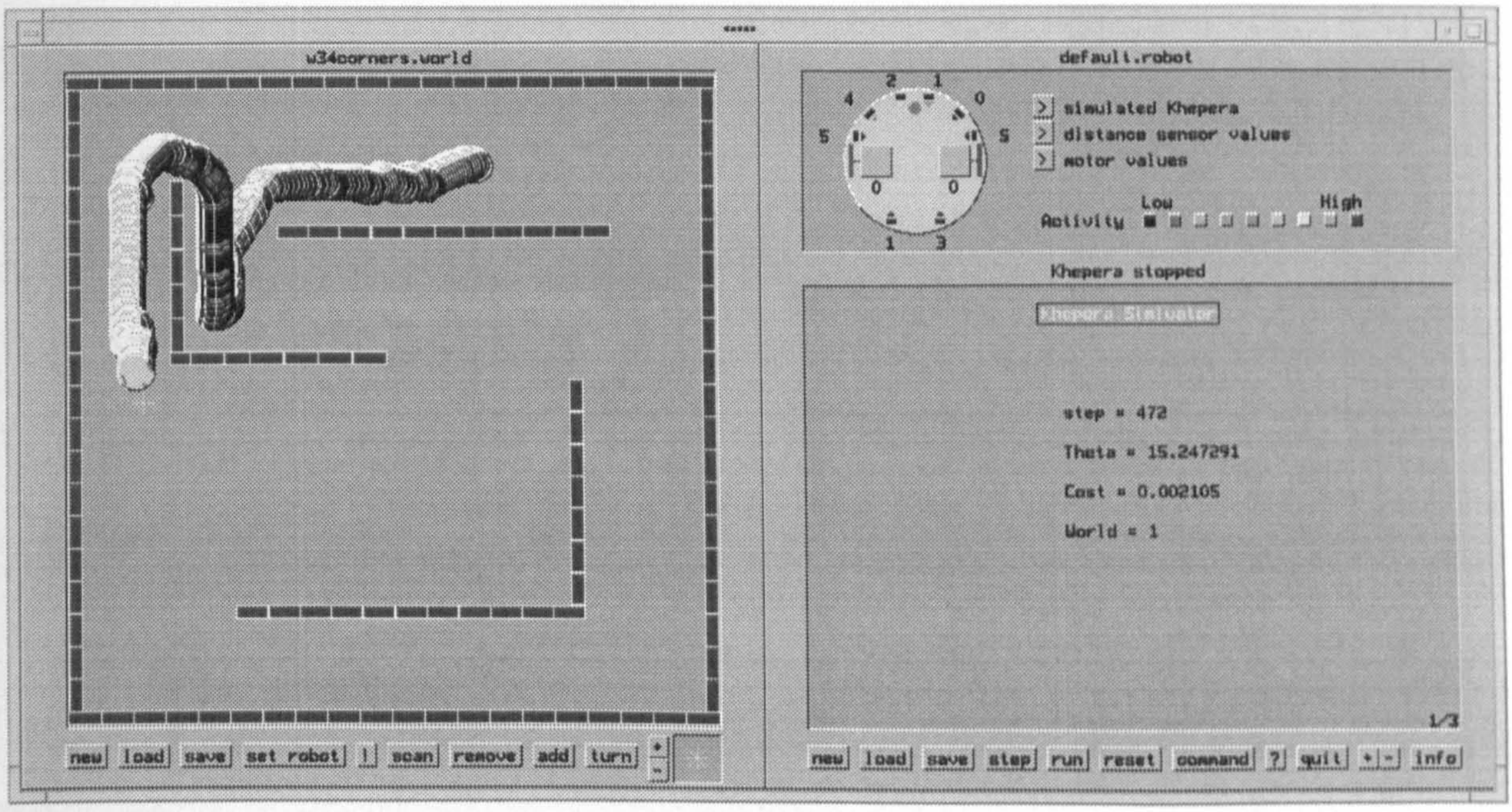


Figure 35 Navigation in an unseen environment where the robot has access to the entire FDT-hierarchy

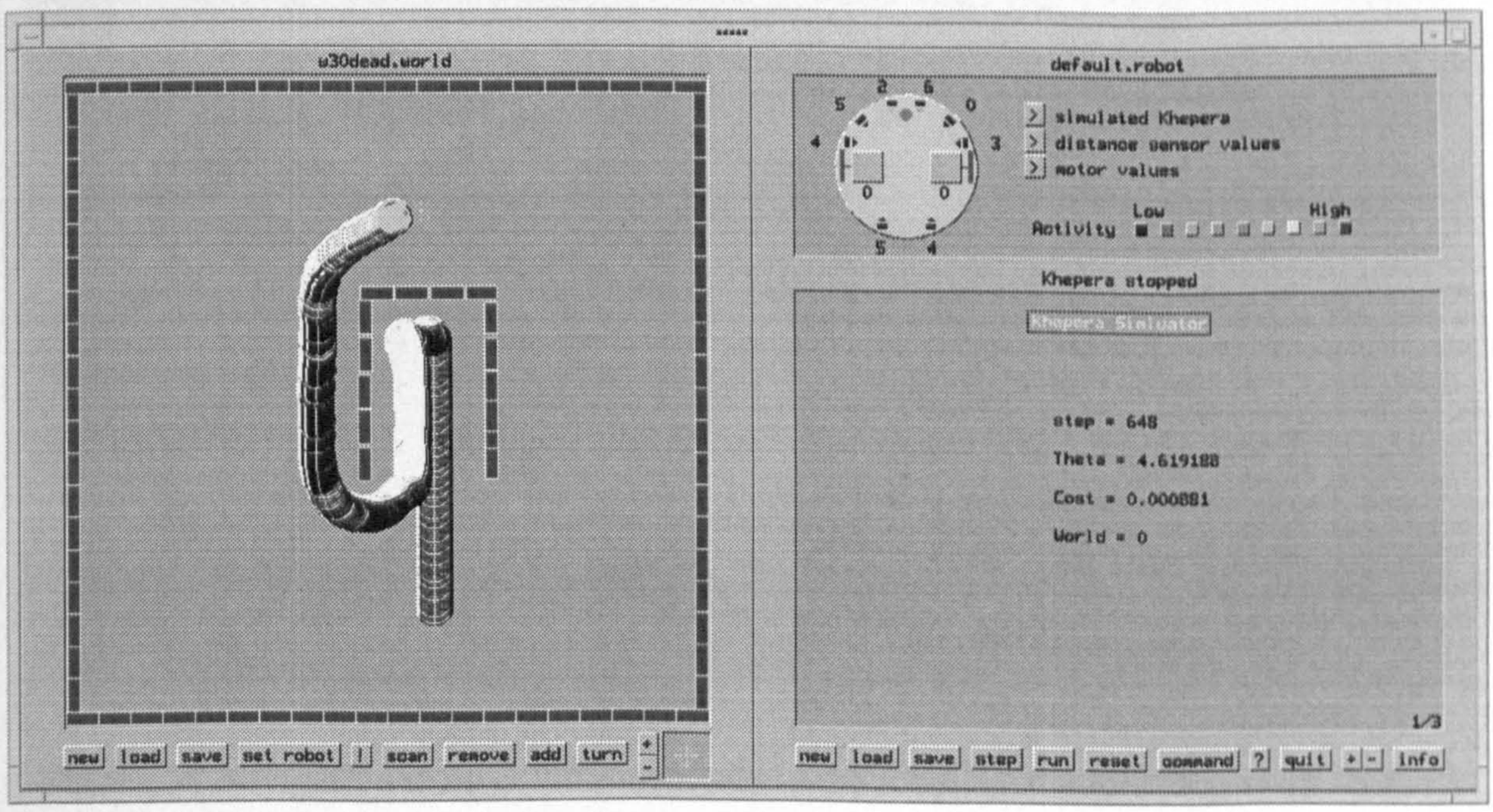


Figure 36 Target-seeking while avoiding dead-ends using FDTs

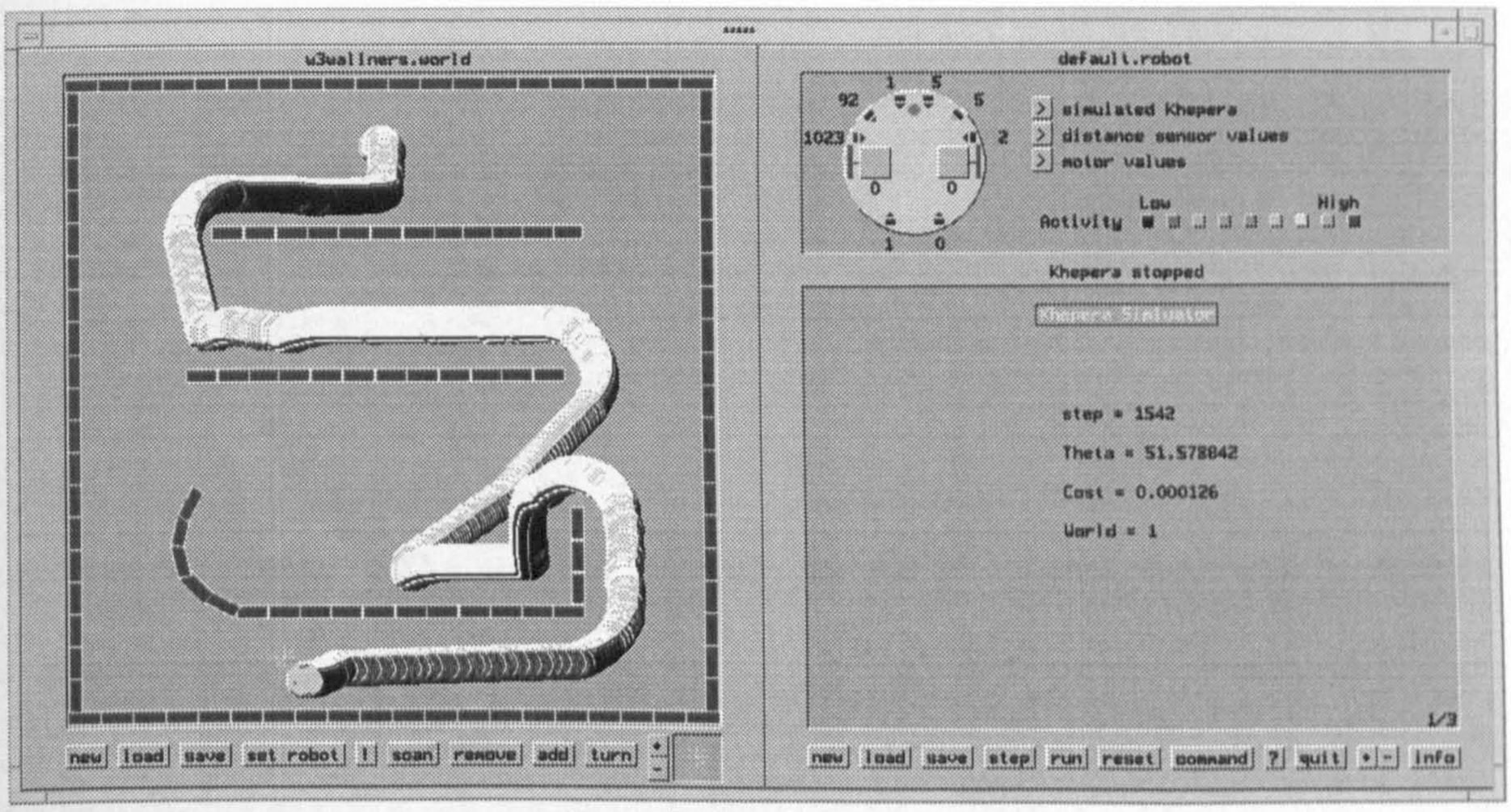


Figure 37 Target-seeking in an unseen environment

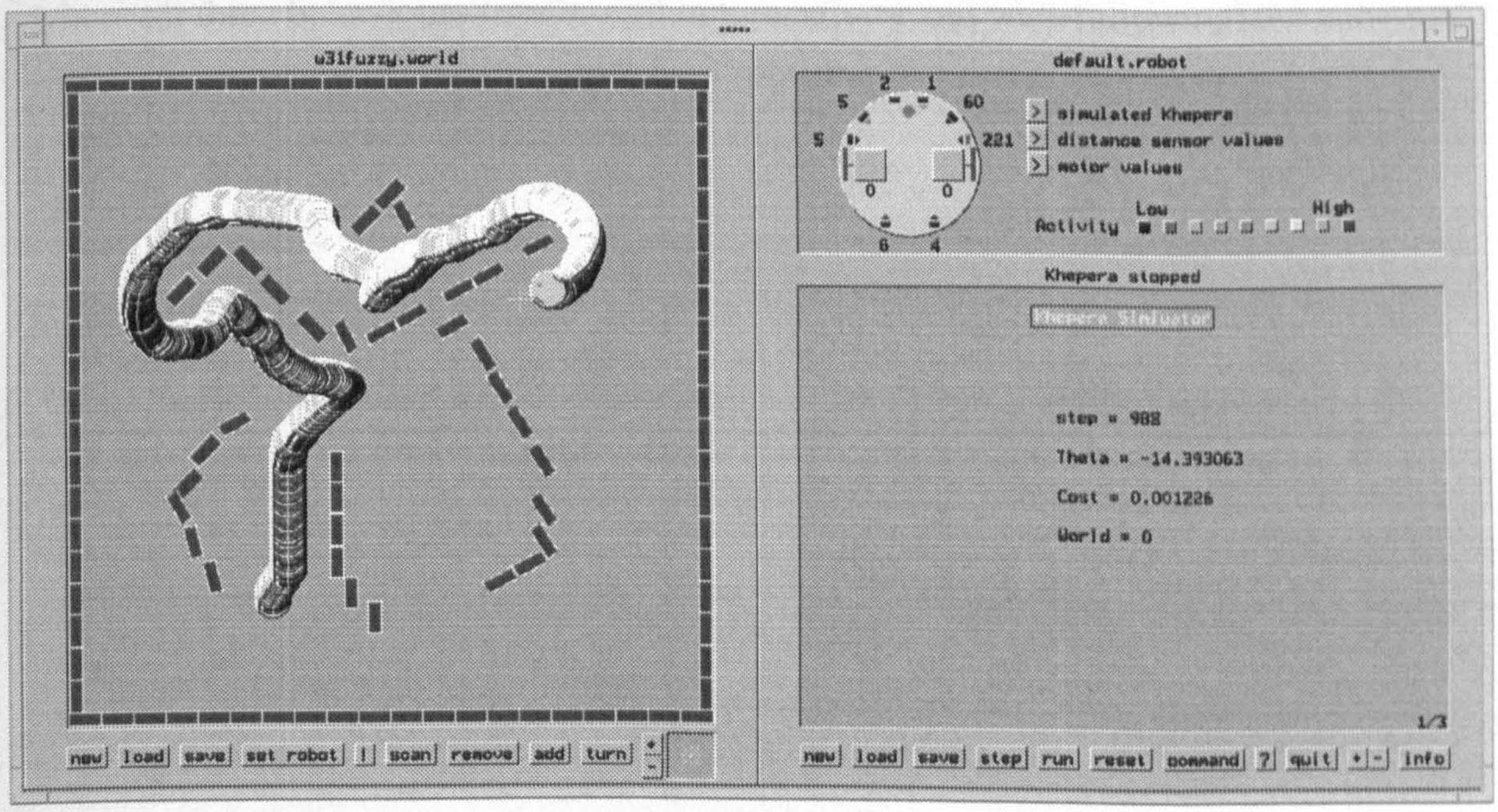


Figure 38 Navigation in an environment where the robot encounters long walls, corners and edges, and is able to manoeuvre around towards the target

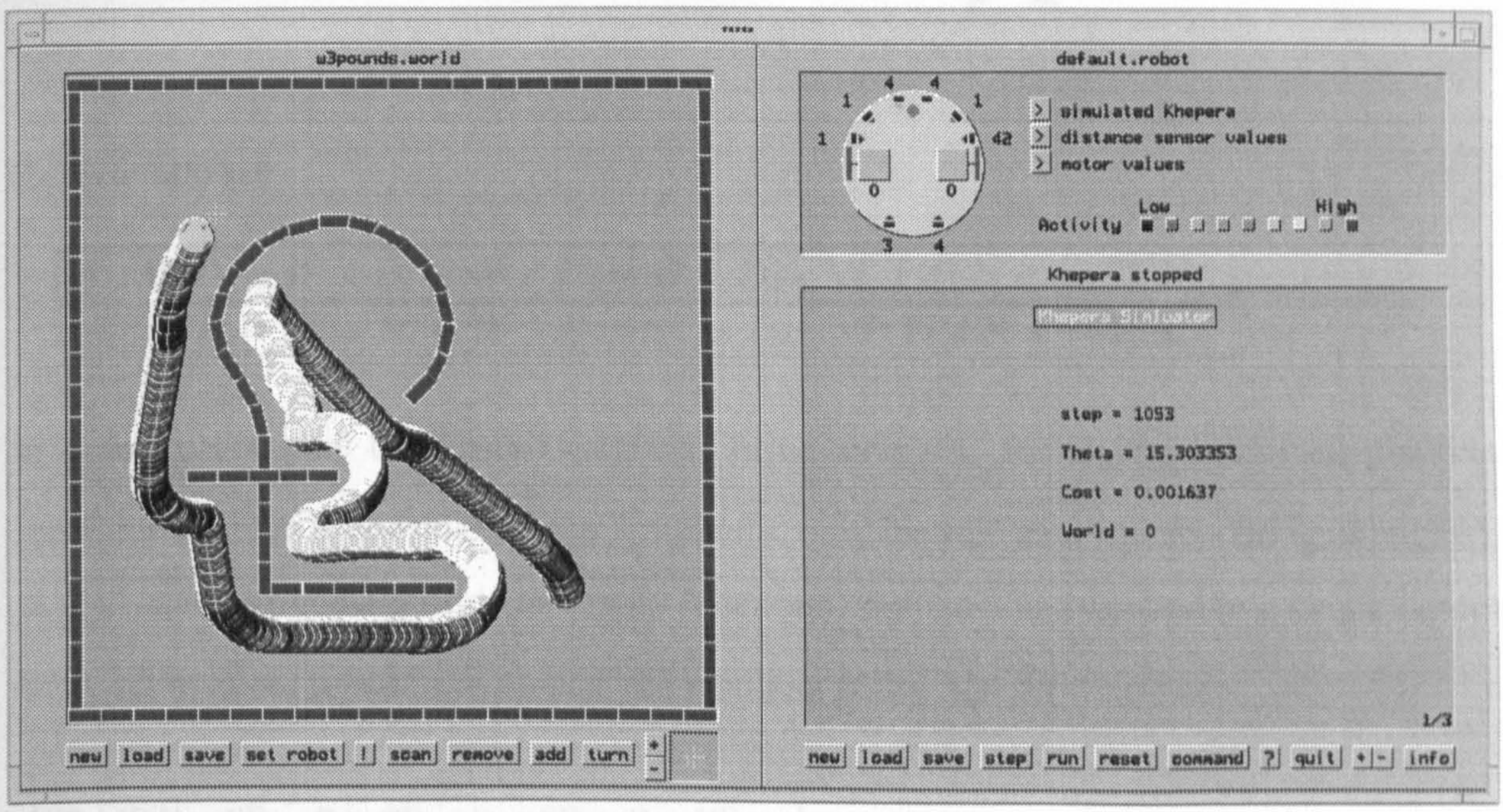


Figure 39 Demonstrating an intelligent behaviour in a previously unseen environment where all three behaviours (target-seeking, reactivity and wall-following) are activated using FDTs which exhibit a robust and coherent trajectory.

Appendix A

Generic rule set:

$\Re_1$: **IF** D_2 = SF $\wedge$ D_1 = SF $\wedge$ D_3 = SF $\wedge$ D_4 = VF **THEN** θ = TFW.
$\Re_2$: **IF** D_2 = SF $\wedge$ D_1 = SF $\wedge$ D_3 = SC $\wedge$ D_4 = VF **THEN** θ = TSR.

In order to compute the overall (highest) firing strength, $\mu_C(y)$, the list of all possible contributive rules is considered. For simplicity, only the rule antecedents (firing strengths), $\mu_{C_i}(y)$, are with the aid of membership functions computed and applied to a single output fuzzy set to avoid repetitions. Hence, the resultant fuzzy rule set would be:

$\Re_1$: IF D_2 = SF $\wedge$ D_1 = SF $\wedge$ D_3 = SF $\wedge$ D_4 = VF	$\mu_{C_1}(y) = min(1.0, 0.95, 0.8, 0.85) = 0.8$
$\Re_2$: IF D_2 = SF $\wedge$ D_1 = SF $\wedge$ D_3 = SF $\wedge$ D_4 = SF	$\mu_{C_2}(y) = min(1.0, 0.95, 0.8, 0.25) = 0.25$
$\Re_3$: IF D_2 = SF $\wedge$ D_1 = SF $\wedge$ D_3 = SC $\wedge$ D_4 = VF	$\mu_{C_3}(y) = min(1.0, 0.95, 0.1, 0.85) = 0.1$
$\Re_4$: IF D_2 = SF $\wedge$ D_1 = VF $\wedge$ D_3 = SF $\wedge$ D_4 = VF	$\mu_{C_4}(y) = min(1.0, 0.5, 0.8, 0.85) = 0.5$
$\Re_5$: IF D_2 = VF $\wedge$ D_1 = SF $\wedge$ D_3 = SF $\wedge$ D_4 = VF	$\mu_{C_5}(y) = min(0.3, 0.95, 0.8, 0.85) = 0.3$
$\Re_6$: IF D_2 = SF $\wedge$ D_1 = SF $\wedge$ D_3 = SC $\wedge$ D_4 = SF	$\mu_{C_6}(y) = min(1.0, 0.95, 0.1, 0.25) = 0.1$
$\Re_7$: IF D_2 = SF $\wedge$ D_1 = VF $\wedge$ D_3 = SF $\wedge$ D_4 = SF	$\mu_{C_7}(y) = min(1.0, 0.5, 0.8, 0.25) = 0.25$
$\Re_8$: IF D_2 = VF $\wedge$ D_1 = SF $\wedge$ D_3 = SF $\wedge$ D_4 = SF	$\mu_{C_8}(y) = min(0.3, 0.95, 0.8, 0.25) = 0.25$
$\Re_9$: IF D_2 = SF $\wedge$ D_1 = VF $\wedge$ D_3 = SC $\wedge$ D_4 = VF	$\mu_{C_9}(y) = min(1.0, 0.5, 0.1, 0.85) = 0.1$
$\Re_{10}$: IF D_2 = VF $\wedge$ D_1 = SF $\wedge$ D_3 = SC $\wedge$ D_4 = VF	$\mu_{C_{10}}(y) = min(0.3, 0.95, 0.1, 0.85) = 0.1$
$\Re_{11}$: IF D_2 = VF $\wedge$ D_1 = VF $\wedge$ D_3 = SF $\wedge$ D_4 = VF	$\mu_{C_{11}}(y) = min(0.3, 0.5, 0.8, 0.85) = 0.3$
$\Re_{12}$: IF D_2 = SF $\wedge$ D_1 = VF $\wedge$ D_3 = SC $\wedge$ D_4 = SF	$\mu_{C_{12}}(y) = min(1.0, 0.5, 0.1, 0.25) = 0.1$
$\Re_{13}$: IF D_2 = VF $\wedge$ D_1 = SF $\wedge$ D_3 = SC $\wedge$ D_4 = SF	$\mu_{C_{13}}(y) = min(0.3, 0.95, 0.1, 0.25) = 0.1$
$\Re_{14}$: IF D_2 = VF $\wedge$ D_1 = VF $\wedge$ D_3 = SF $\wedge$ D_4 = SF	$\mu_{C_{14}}(y) = min(0.3, 0.5, 0.8, 0.25) = 0.25$
$\Re_{15}$: IF D_2 = VF $\wedge$ D_1 = VF $\wedge$ D_3 = SC $\wedge$ D_4 = VF	$\mu_{C_{15}}(y) = min(0.3, 0.5, 0.1, 0.85) = 0.1$
$\Re_{16}$: IF D_2 = VF $\wedge$ D_1 = VF $\wedge$ D_3 = SC $\wedge$ D_4 = SF	$\mu_{C_{16}}(y) = min(0.3, 0.5, 0.1, 0.25) = 0.1$

$$\mu_C(y) =$$

$$max\left\{\begin{matrix}\mu_{C_1}(y), \mu_{C_2}(y), \mu_{C_3}(y), \mu_{C_4}(y), \mu_{C_5}(y), \mu_{C_6}(y), \mu_{C_7}(y), \mu_{C_8}(y), \mu_{C_9}(y), \mu_{C_{10}}(y), \\ \mu_{C_{11}}(y), \mu_{C_{12}}(y), \mu_{C_{13}}(y), \mu_{C_{14}}(y), \mu_{C_{15}}(y), \mu_{C_{16}}(y)\end{matrix}\right\} = \mathbf{0.8}$$

Appendix B

Performing 15-fold cross-validation:

Assigning instances to folds...
Running cross-validation:
.....

Run	Nodes	XTsts	Insts	Acc	
1:	43.00	5.29	18.00	83.33	
2:	43.00	5.47	18.00	88.89	
3:	43.00	5.51	18.00	72.22	
4:	43.00	5.17	18.00	66.67	
5:	43.00	5.53	18.00	100.00	
6:	41.00	5.51	18.00	61.11	
7:	43.00	5.47	18.00	88.89	
8:	43.00	5.54	18.00	88.89	
9:	39.00	5.15	17.00	88.24	
10:	45.00	5.60	17.00	82.35	
11:	37.00	5.38	17.00	70.59	
12:	43.00	5.11	17.00	76.47	
13:	43.00	5.37	17.00	82.35	
14:	43.00	5.51	17.00	82.35	
15:	43.00	5.44	17.00	94.12	
Avg	42.33	5.40	17.53	81.76	(iti-incremental)
SDv	1.95	0.16		10.63	(iti-incremental)

Performing 15-fold cross-validation:

Assigning instances to folds...
Running cross-validation:
.....

Run	Nodes	XTsts	Insts	Acc	
1:	43.00	5.48	18.00	88.89	
2:	43.00	5.33	18.00	83.33	
3:	41.00	5.50	18.00	77.78	
4:	43.00	5.48	18.00	77.78	
5:	35.00	5.53	18.00	77.78	
6:	43.00	5.25	18.00	88.89	
7:	43.00	5.54	18.00	61.11	
8:	43.00	5.22	18.00	83.33	
9:	41.00	5.44	17.00	64.71	
10:	43.00	5.40	17.00	88.24	
11:	43.00	5.35	17.00	82.35	
12:	43.00	5.51	17.00	94.12	
13:	43.00	5.15	17.00	88.24	
14:	43.00	5.35	17.00	88.24	
15:	43.00	4.96	17.00	88.24	
Avg	42.20	5.37	17.53	82.20	(iti-incremental)
SDv	2.11	0.16		9.22	(iti-incremental)

Table B.1 Results from two consecutive runs of 15-fold cross-validated unpruned tree shown in Figure 7.14

```
Pruning tree & turning on virtual pruning.

Performing 15-fold cross-validation:

Assigning instances to folds...
Running cross-validation:
.................
Run  Nodes  XTsts  Insts    Acc
-------------------------------
 1:  23.00   4.55  18.00  94.44
 2:  25.00   4.46  18.00  77.78
 3:  23.00   4.40  18.00  72.22
 4:  27.00   4.84  18.00  83.33
 5:  27.00   4.88  18.00  94.44
 6:  27.00   4.98  18.00  88.89
 7:  27.00   4.96  18.00  66.67
 8:  27.00   5.03  18.00  77.78
 9:  27.00   4.98  17.00  88.24
10:  27.00   5.02  17.00  70.59
11:  25.00   4.48  17.00  88.24
12:  27.00   5.02  17.00  76.47
13:  27.00   4.98  17.00  94.12
14:  25.00   4.49  17.00  88.24
15:  25.00   4.65  17.00  82.35
-------------------------------
Avg  25.93   4.78  17.53  82.92 (iti-incremental)
SDv   1.49   0.24         9.01 (iti-incremental)
```

```
Pruning tree & turning on virtual pruning.

Performing 15-fold cross-validation:

Assigning instances to folds...
Running cross-validation:
.................
Run  Nodes  XTsts  Insts    Acc
-------------------------------
 1:  25.00   4.95  18.00  66.67
 2:  27.00   5.06  18.00  77.78
 3:  27.00   4.89  18.00  94.44
 4:  23.00   4.49  18.00  66.67
 5:  33.00   4.91  18.00  72.22
 6:  25.00   4.74  18.00  94.44
 7:  27.00   4.81  18.00  94.44
 8:  23.00   4.22  18.00  88.89
 9:  23.00   4.54  17.00  76.47
10:  25.00   4.59  17.00  88.24
11:  27.00   4.99  17.00  88.24
12:  27.00   4.77  17.00  88.24
13:  23.00   4.41  17.00  82.35
14:  23.00   4.54  17.00  94.12
15:  27.00   4.86  17.00  82.35
-------------------------------
Avg  25.67   4.72  17.53  83.70 (iti-incremental)
SDv   2.69   0.24         9.81 (iti-incremental)
```

Table B.2 Results from two consecutive runs of 15-fold cross-validated pruned tree shown in Figure 7.14

In both tables, the column "Acc" specifies the prediction accuracy of each block, and the element (Avg, Acc) specifies the average prediction accuracy of each the run (averaged over 15 blocks).

Appendix C

Fuzzy training examples collected from the training stage describing (partially) w_2.

VF,VF,VF,VC,VC,CL,RI,TLE.
VF,VF,SF,SF,SF,VF,SR,TLB.
VF,VF,VC,VC,VF,SC,SR,TLB.
VF,VF,VF,SC,VC,VF,RI,TLE.
VF,VF,VF,SC,VC,VF,RI,TRB.
VF,CL,VC,VC,VF,VF,SR,TLB.
VF,VF,CL,VC,VC,SF,RI,TLB.
VF,VF,CL,VC,VC,VF,RI,TLB.
VF,SC,SC,VF,VF,VF,RI,TRB.
VF,SF,VC,VF,VF,VF,FR,TFW.
VF,CL,VC,SC,VF,VF,FR,TLB.
VF,VF,VC,VC,SC,VF,SR,TLB.
VF,VF,VF,SC,VC,VC,RI,TFW.
VF,VF,VF,VC,VC,VC,RI,TFW.
VF,VF,VF,CL,VC,VC,RB,TLB.
VF,SC,SC,VF,VF,VF,FR,TSR.
VF,VF,VC,VC,VC,VC,RB,TLB.
VF,VF,VF,CL,VC,VC,RB,TLB.
SF,VF,SF,VF,VF,VF,FR,TRI.
VC,VF,SF,VF,VF,VF,SL,TRI.
VC,SC,SF,SF,VF,VF,SL,TLB.
VF,VC,VF,VF,SF,VF,FR,TRB.
SF,VF,VF,SF,VF,VF,FR,TLE.
VF,VF,VC,VC,CL,VF,RB,TLB.
VF,VF,VF,CL,VC,CL,RB,TLE.
VF,SF,VF,VF,VC,CL,RI,TLE.
VF,VF,SF,SF,VF,VC,RI,TLB.
VF,VF,SF,SF,SF,VF,RB,TLB.
VF,VF,VF,SC,VC,VF,FR,TLE.
VF,VF,VF,SF,SF,VF,FR,TLB.
VC,SC,SF,VF,VF,VF,SL,TLB.
VF,VC,SC,VF,VF,VF,FR,TRB.
VF,VF,CL,VC,CL,SC,RI,TLB.
VF,VC,CL,SF,VF,VF,SL,TLB.
VF,VF,VC,VC,SF,VF,SR,TLB.
VF,VF,SF,CL,VC,CL,RI,TLB.
VF,VF,VF,SF,SC,VF,SR,TLE.
VF,SF,VC,SF,VF,VF,RI,TFW.
VF,CL,VC,VC,VF,VF,RI,TLB.
VF,VF,CL,VC,VC,VF,RI,TLB

Chapter 8

Conclusions and Further Work

There are no facts,
only interpretations.

F. Nietzsche
Nachlaβ

This chapter summarises the author's contribution of new knowledge which relates to the design and development of an intelligent control strategy for an autonomous robot. It also discusses the results obtained from the implementations of the learning systems described in chapters 4, 5 and 7, and compares the performances of the alternative methods. The potential applications in which the proposed techniques can be employed are discussed and areas of further research are identified.

8.1 Conclusion

The aim of this work was to establish a novel intelligent reactive strategy for the control and navigation of an autonomous robot. To achieve this, the work formulated the methodology of *Hierarchical Learning and Knowledge Decomposition* in the frame of a reactive architecture, in which intelligence is distributed in a hierarchy of behaviour complexities accessible to the intelligent agent. The methodology of concept learning was employed to establish these layers and they were implemented in decision trees (DTs). Learning was initiated in the lowest layer of the hierarchy (the world with no obstacles) in order to establish the survival instinct in which the aim is simply to find a set target (target-seeking behaviour). Knowledge propagated up the hierarchy during the training of the robot to enable it to learn complex behaviours in worlds containing obstacles. The novel contributions of the research carried out in this work are summarised below and are discussed in detail in the following sub-sections.

- Introduction of an off-line hierarchical learning approach employing knowledge decomposition.
- On-line implementation of the learning concept in realistic environments.
- On-line implementation of adaptive fuzzy DTs for behaviour fusion.
- Introduction and implementation of multi dimensional fuzzy associative memories (MDFAMs) to deal with highly non-linear fuzzy input spaces as well as to fuse these to lower dimensions.

Off-line hierarchical learning approach

The novel approach of hierarchical learning was first introduced and applied (see chapter 4) to a robot and environment about which a number of simplifying assumptions were made. This was used to demonstrate qualitatively the feasibility and robustness of the learning approach rather than to develop a navigation system with an optimal response in a representative range of perception examples. The training process was performed off-line, in that the robot was first trained in an environment by letting it explore its surroundings to collect training data. This process was used to grow DTs in an off-line manner which were then embedded in the control system and subsequently applied to navigation in unseen

situations. The performance of the robot in navigation tasks is hence reflected in the quality of its trajectories, and the robustness of the DT hierarchy.

Experimental results (presented in section 4.10) proved the validity of the approach, in that the robot was always able to find its target from any arbitrary position while avoiding obstacles. However, due to the assumptions made in the implementation, and in particular the low resolution of the working space, the quality of the trajectories was rather sub-optimal. However, due to the limited number of feature values which resulted, the trained DTs were small and could be built with a moderate number of training examples (the maximum required in the tests was 132).

On-line learning in realistic environments

The learning algorithm was then modified (see chapter 5) to operate in an on-line and in an incremental fashion in such a way that the DTs in the hierarchy were able to behave adaptively as new knowledge was gained. In this approach, ITI-2.8 was integrated into the control architecture to provide on-line learning, and the simplifying assumptions made previously were removed and a continuous environment was considered. Structurally, the increase in problem complexity was evident in the larger number of dimensions found in the trees following training. In general, an increase in the number of nodes in a DT is reflected in the greater specialisation of the DT, which in turn reduces its predictive power. To reduce the influence of this effect, post-pruning was performed when the total number of the training examples exceeded a set threshold. This kept the size of the DTs under control without losing classification accuracy, which was confirmed in the results by a 15-fold cross-validation following truncation.

Operationally, as the behaviours were being formed gradually, the adaptive behaviour showed an incremental improvement in the robot trajectories. In this approach, realistic and real-world suppositions were made in order to model the robot behaviour in the face of uncertainty and non-linearities. The experimental results (see section 5.10) confirmed that the learning system was robust and able to cope with non-linearities, both multiple-valued and multi-variant control parameters, and with a continuous environment. In contrast to the

off-line learning mode, the incremental improvement of the robot behaviour, even in the training phase, was evident in the generated trajectories.

However, one shortcoming (inherent in basic reactive strategies [1,2]) which was also observed in the operation of this approach was the problem of unsmooth or "zigzag" behaviour exhibited by the robot while following long walls. This arose due to the absolute behaviour arbitration within the hierarchy in scenarios where two behaviours activated each other alternatively. This problem was removed in the last stage of the development by introducing fuzzy logic and reasoning into the hierarchy, and this is discussed in the following section.

Adaptive fuzzy DTs for behaviour fusion

The outcome of introducing fuzziness into the hierarchy (see chapter 7) was a hybrid fuzzy-DT (FDT) learning system that combined symbolic learning and approximate reasoning for decision making in the face of partial information, uncertainty and noisy sensory data. The rationale of the hybrid system design was two-fold.

- To provide a more efficient noise rejection mechanism.
- To blend conflicting behaviours in order to suppress zigzag motions and to replace these with smooth trajectories [3].

In the vast majority of practical applications in which noise and non-linearity were present, implementations using fuzzy logic have been shown to outperform other non-conventional control strategies [4]. This effect has been demonstrated in the results achieved in the current work using the hybrid technique. FDTs have shown that they are able to blend conflicting behaviours by generating smooth trajectories and by being able to cope more efficiently with noisy data.

Although the number of dimensions of the total input space is now larger due to the introduction of a new and multiple-valued fuzzy variable (the distance values associated with each sensor), there is no significant increase in the size of FDTs compared with that of the pure DTs (chapter 5). This is largely the result of the production of the FDTs using ITI-2.8 which is able to generate relatively smaller trees. These results confirmed that DT-

based hierarchical learning orchestrated by fuzzy reasoning has the capability to deal with non-trivial real-world problems such as efficient noise cancellation, approximate reasoning and behaviour fusion. The results obtained were also very encouraging (as far as simulated environments are concerned), and of great value for further application in a real physical robot.

MDFAMs for the management of non-linear fuzzy input spaces

Another concept which is of a more fundamental significance in that it is applicable to a wide variety of fuzzy based intelligent systems, is the concept of multi dimensional fuzzy associative memory (MDFAM) which was also introduced and advocated in this work. As discussed in chapter 7, this approach provides an improved representation compared to that of linear nested FAMs [5]. MDFAMs are able to learn and to accommodate linguistic fuzzy rules whose antecedents are highly non-linear and comprise multi-variant parameters. Due to the characteristics of DT learning, these are also able to fuse highly non-linear input parameters to a number of representative parameters describing a certain concept. MDFAMs formed the underlying theory of fuzzy decision trees applied to robot control and navigation in this work.

8.2 Comparison of the alternative techniques

The experimental results reveal that the learning speed (number of training epochs) varies in the same proportion between layers in the hierarchy, regardless of the operation mode (on-line or off-line). In a given layer, the learning speed appears to depend on two factors, namely the number of input parameters to the learning algorithm and whether and to what extent meta-knowledge is available during training. This is most evident in the first layer, namely w_0, which represents the environment without obstacles. This layer appears to need the longest time of all layers to find almost all of the representative vectors, even though there is only one input parameter (the relative location of the target) to the learning algorithm and the associated DT is a linear tree. In general, the learning time decreases as one advances further up the hierarchy. The fact that longer training times are required in the first layer of the hierarchy and much shorter times (considering the non-linearity of the input space) in the higher layers can be attributed to the fact that the DT in the first layer is learned from inception, but as training proceeds to the higher layers, learning speed is aided

by the availability of an ever increasing wealth of meta-knowledge gained from the preceding layers.

Another aspect which is implicitly formulated into the structure of the generated DTs, and explicitly reflected in the coherence of the robot trajectories, is the quality of the learned knowledge. Capturing the most representative training vectors in the first layer of the hierarchy, w_0, is an important foundation in the production of coherent knowledge in the higher order worlds. This does not necessarily mean that the size of the DT for w_0 should be so large that it incorporates all the training vectors describing w_0, rather that it should be sufficiently general in structure to encode the necessary representative training vectors.

Comparing the hybrid technique (FDT) with the other two approaches (and particularly the on-line learning approach which is more comparable in its application), a significant improvement in the robot behaviour can be observed without introducing any major additional computational cost. FDTs can also use heuristics or common sense knowledge as the learning seed to reduce training time. Due to the linguistic nature of both the learning module and the fuzzy control, the generated FDTs are able to represent the control laws in the form of a set of intelligible rules which is important to users who require a transparent model of the control algorithm applied.

8.3 Potential application areas

The performance of the DT-based hierarchical learning, and the hybrid system in particular, proved to be robust and efficient in both control and navigation tasks. The next stage of the work would be to implement this learning mechanism in real-world navigation problems. The nature of this hybrid system suggests a number of areas in which its capabilities can be exploited, and put to the test, and these are as follows.

- Implementation in service robots.
- Implementation in manufacturing, intelligent transportation and factory automation.
- Application to intelligent control systems, in which either mode of operation can be employed (on-line or off-line learning), depending on the nature of the task.

In the above application areas, the second is perhaps the most suitable application for initial exploration as it can be investigated without the need to make major changes to the system architecture. One possibility is that the system could be applied as an intelligent alternative to "unintelligent" and pre-planned automatically guided vehicles (AGVs) which usually navigate inductively by following a fixed path of electrical wires and are typically used for the transportation of sub-assemblies in the manufacturing sector, and assume an unchanging environment. However, they are not flexible in their operation and can be expensive to install as they need to work in specially engineered environments.

The intelligent system introduced in this work can be used to replace the control architecture of AGVs to enable them to manoeuvre around obstacles and to make suitable decisions in highly dynamic environments such as manufacturing areas. As the intelligent algorithm does not require any pre-path planning or map making, no specific areas of the shop-floor would need to be allocated to the AGVs. Such an implementation would also save on hard wiring costs associated with current AGVs.

The concept of FDT learning has application to a large number of control systems, particularly those in which the plant dynamics are unknown, highly non-linear or difficult to model. In such a case, a set of heuristics (as an objective function) can be used to initiate the incremental learning of the controller. The controller developed in this way has a distinctive advantage compared with other model-based control systems such as those based on neural networks, in that the learned control rules are available and intelligible to the user. This would also aid the user in the identification of mis-classifications of control parameters which may have arisen as the result of firing control rules specified by a DT.

8.4 Suggestions for further research

Scientific research is an ongoing process in which existing knowledge is refined and to which new knowledge is contributed. In carrying out the work presented in this thesis, a number of research areas can be identified for further investigation.

The most immediate work would be to implement the hybrid system on a real robot and to evaluate its behaviour in structured (office areas and corridors) as well as in unstructured and dynamically changing environments. Given a reliable dead-reckoning system, the

results of self-learning in unstructured environments can be investigated, enabling a qualitative comparison to be made with the system performance when simulated learning results are transferred to the real robot. The comparison results would provide a suitable benchmark to indicate the robustness of the system in real-world scenarios.

A more challenging task would be to investigate the increase in the robot's degrees of freedom (using the same learning architecture) by adding a manipulative task to the repertoire of robot behaviour. This would also modify the current system's structure from a MISO system to a MIMO system, that is, the training examples would be identified with more than one class each of a different nature.

Since the majority of the currently available DT learning systems operate on single output training examples, this in itself would provide a suitable motivation for a theoretical investigation and implementation of MIMO-based DTs. The introduction of a MIMO-based hierarchical learning may suggest the introduction of neural networks (NNs) or genetic algorithm (GAs) into the system to replace DTs.

NNs and GAs both have the ability to perform learning and optimisation. By replacing the DT hierarchy by an array of NNs, the principal learning approach (knowledge decomposition) would remain intact. This means that a hierarchy of locally and behaviour-based NNs are trained and grown in an incremental manner, in which each NN would be used to provide meta-knowledge to train and to initiate the next and more complex NN in the hierarchy. The topology of the NNs in the hierarchy would depend on the dimensionality of the input-output space which in turn is determined by the number of the control parameters.

As discussed previously, the experimental results demonstrated that the learning convergence in the first layer of the hierarchy, w_0 (the environment without obstacles), is relatively slow compared with that of other DT networks and this is due largely to the absence of meta-knowledge. The introduction of a MIMO system may further increase the convergence time of this layer. Techniques for training NNs have been the subject of much research, and to suppress the slow convergence in the first NN layer of the hierarchy would be to use a learning algorithm such as conjugate gradients [6] with fast convergence

abilities. Although its implementation involves additional computational cost in that the Hessian of the function and its gradient need to be found, the algorithm may reduce convergence time in the first layer. The learning algorithms used in the higher NN layers of the hierarchy may not require application of the conjugate gradient algorithm, as the meta-knowledge gained in the previous NN layers would largely compensate the slow convergence of standard learning algorithms such as back-propagation.

The use of GAs in the hierarchical structure could also be investigated as an alternative to DTs or NNs. One distinct advantage of GAs in learning from a population of training examples is its ability to perform global search and, from this, to identify the global maximum or minimum of a desired function. They are also able to provide a population of potential solutions (which may not be necessarily optimum) to a given task. The disadvantage of the basic GA algorithms is their slow speed in finding the solution population if the dimensionality of search space is large, although this problem can be tackled by applying parallel GA algorithms and pre-partitioning the search space. In hierarchical structure, a hybrid version of GAs and DTs could be used (keeping the learning principle of knowledge decomposition intact) in order to generate layers of genetically learned knowledge. One advantage of this method would be rapid convergence and the identification of a solution population in each layer. DTs can be used to store, manage and fuse (keep the most representative chromosomes in each solution population and discard the rest) the solution populations. Training of the higher layers can be accomplished in the same manner as already described, by using the optimised chromosomes stored in the previous layer to form the initial population of the learning process for the next layer. A comparison with the purely DT approach could be carried out to assess its performance.

References

[1] P. Reignier, "Fuzzy Logic Techniques for Mobile Robot Obstacle Avoidance", *Robotics and Autonomous Systems*, 12 (1994), pp. 143-153.

[2] A. Dubrawski and J.L. Crowley, "Learning Locomotion Reflexes: A Self-supervised Neural System for a Mobile Robot", *Robotics and Autonomous Systems*, 12 (1994), pp. 133-142

[3] A. Saffiotti, E.H. Ruspini and K. Konolige, "Blending Reactivity and Goal-directedness in a Fuzzy Controller", *Proceedings of the Second IEEE Conference on Fuzzy Systems*, San Francisco, March 1993, pp. 134-139.

[4] R.M. Tong, "A Control Engineering Review of Fuzzy Systems", *Automatica*, Vol. 13, Pergamon Press, 1977, pp. 559-569.

[5] E.P. Dadios, "Non-Conventional Control of the Flexible Pole-Cart Balancing Problem", *PhD Thesis*, Department of Manufacturing Engineering, Loughborough University, 1996.

T. Masters, "Practical Neural Network Recipes in C++", Academic Press, Inc., 1993.

CONTAINS DISKETTE

UNABLE TO COPY

CONTACT UNIVERSITY

IF YOU WISH TO SEE

THIS MATERIAL