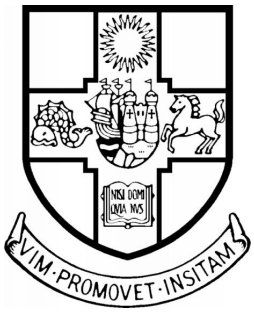
Behaviour-Based Learning

Evolution Inspired Development of Adaptive Robot Behaviours

Torbjørn Semb Dahl



A thesis submitted to the University of Bristol in accordance with the requirements of the degree of Doctor of Philosophy in the Faculty of Engineering, Department of Computer Science.

October 2002

Word count: 41477

Abstract

This dissertation presents Behaviour-Based Learning (BBL), a methodology, for developing rapidly adapting behaviours in Behaviour-Based (BB) robots. BBL deals with a set of current issues related to learning in robots, in particular: speed of adaptation, the use of domain knowledge, problem restriction, robustness and integration of programmability and adaptivity.

We also present an abstract model of behaviours called the Neural Circuit (NC) model. NCs are closely related to neural schema. The NC model was developed to facilitate the design of adaptive behaviour, in particular the integration of adaptive and non adaptive structures.

To support the implementation of robot controllers developed using the BBL methodology and the NC model, we present a class library of Programmable Learning Artificial Neural Circuits (PLANCS). The PLANCS class library provides a three-layered implementation of NCs. The first layer defines an abstract execution model that can be ported to any processor architecture. The second layer implements neuron inpired, sum-threshold, circuit activation model. The third layer implements a set of interefaces for passing structured data between NCs.

Three sets of experiments using the Webots Khepera Robot Simulator are presented, demonstrating rapidly adapting behaviours on Khepera robots. The behaviour are developed for three different problem domains: approach compensation, foraging/mapping and conflict resolution.

Lastly, we present recommendations for further work on developing intelligent behaviour on robots, in what we call a holistic approach to Artificial Intelligence.

For Jude, Francesca and Alida.

Acknowledgements

First I would like to thank my wife Jude for supporting me in doing my PhD. Jude has probably worked as hard for it as I have. I would also like to thank my daughters Francesca and Alida for putting up with Daddy working continuously for the last couple of years. Thanks also to my parents Inger Elisabeth Dahl and Knut Erling Semb and their partners Steinar Kubben Nilsen and Gerd Pettersen for their support, encouragement and love.

The most important support from outside my family has come from my supervisor Christophe Giraud-Carrier. I'd like to thank Christophe for his trust in my abilities, for giving me the chance to study for a PhD originally and for the flexibility to always follow the research direction that seemed most rewarding. David May and Peter Flach at the University of Bristol also deserve many thanks for allowing me to continue my studies by providing help and support. Thank you also to my previous colleagues Chris Preist and Ian Dickinson at Hewlett-Packard's Research Laboratories in Bristol. Their support was crucial in allowing me to complete my PhD studies. Cyberbotics very generously let me have a free student licence to run their Webots Khepera simulator. Thanks in particular to Olivier Michel for several licence extensions and much good advice. Keith Wiley also deserves praise for making the source code of his winning controller for the first Webots ALife Creators competition publicly available.

The work presented in this dissertation was influenced by Joanna Bryson's work at MIT and at the University of Edinburgh. She deserves special thanks for sharing her work and ideas openly. Thanks also to Maja Matarić and Gaurav

Sukhatme at the University of Southern California's Robotics Research Laboratory for their input and support.

Most of my co-students at Bristol University have also contributed helpful discussions and moral support. In particular Rene McKinney Romero, Rafal Bogacz, Anthony Bowers and Timothy Langford have been good friends throughout.

The work was funded by the Department of Computer Science at the University of Bristol, Hewlett-Packard's Research Laboratories in Bristol and the Norwegian State Educational Loan Fund. It was also made possible thanks to my inheritance from my grandparents Arve and Alida Johansen. I hope they would have approved and will always cherish their loving memory.

Author's Declaration

I declare that the work in this dissertation was carried out in accordance with the Regulations of the University of Bristol. The work is original except where indicated by special reference in the text and no part of the dissertation has been submitted for any other degree. Any views expressed in the dissertation are those of the author and in no way represent those of the University of Bristol. The dissertation has not been presented to any other University for examination either in the United Kingdom or overseas.

date:

T. S. Dahl

Short Table of Contents

Contents

List of Figures

Chapter 1

Introduction and Aims

Contents

1.1 Introduction

Adaptiveness has been identified as an important factor in intelligent behaviour [McFarland, 1999], though it is emphasised that the two are not the same. Reactive behaviour can be said to be intelligent if it is effective in the environment it is naturally expressed.

Cyberneticists study problems of control and adaptation, or *self-organisation* in their terms, in systems that sense and act in the real world. Their approach was originally inspired by the commonalities between engineered and biological solutions to dynamic system control [Weiner, 1948, Ashby, 1952, Grey, 1963]. It was computation however [Turing, 1937], a more anthropomorphic metaphor that became the dominant model of intelligence [Turing, 1950]. The practical and philosophical implications of using computation as a model of intelligence lead research away from embodiment and interaction. The computing machinery of the day was immobile with restricted interfaces to the external world. This lead research to focus on abstract problem representations and general problem solving techniques, based on symbol manipulation [Newell and Simon, 1963]. This approach to studying intelligence was named Artificial Intelligence (AI) and has later been refered to as Good Old-Fashioned AI (GOFAI) [Haugeland, 1997].

Controllers based on the computational metaphor were later used on embodied systems [Nilsson, 1984]. They were considered a success, but work on embodied systems in general had identified two problems with intelligence as computation: the problem of symbol grounding and the problem of identifying appropriate frames of reference [Brooks, 1991a]. These problems, along with smaller computers and improved sensory interfaces renewed the interest in embodied systems and in non-computational [Harvey, 1997], non-symbolic [Brooks, 1991b] and time-sensitive, metaphors for intelligence and control [Brooks, 1986, Arkin, 1989, Yamauchi and Beer, 1995]. They also revived the interest in animal intelligence [Wilson, 1991, Brooks, 1997]. The new wave of bottom-up approaches to AI are generally referred to as Behaviour-Based (BB) AI [Arkin, 1998].

As a reaction against the symbolic, state-based nature of the computational metaphor, the new metaphors minimised the use of internal representations of the external world and focused on minimal behaviours and the emergent properties of their interaction [Brooks, 1991b]. This minimalist approach restricts a system's adaptability. These restrictions can keep the such systems form scaling up to more complex tasks. This has been called the scaling problem of behaviourist approaches [Tsotsos, 1995].

In order to develop robots that are as adaptive as animals with respect to environmental changes and the behaviour of other entities, it is necessary to develop suitable adaptive mechanisms. Such mechanisms must be able to handle the time constraints of an embodied system, whilst providing the full range of adaptive capabilities found in animals, from simple forms of habituation and imprinting to complex cognitive abilities such as abstract reasoning. Hence, unifying the efficiency of cybernetics with the generality of computation is the challenge currently facing roboticists. The work presented in this thesis suggests solutions to this problem of integration.

1.2 Aims

The control metaphors for dynamic systems placed different demands on adaptive mechanisms [Matarić, 1994, Corbacho and Arbib, 1997] than the techniques based on the computational metaphor [Anthony and Biggs, 1997], particularly in terms of time-efficiency.

To support the development of adaptive robots, this thesis presents tools for designing and implementing rapidly adapting solutions to complex problems using neural schema-based mechanisms in an evolution-inspired manner. We present a methodology, a modelling framework and a class library that we have developed for this purpose. We also present experiments demonstrating rapidly adapting behaviours in robots for three problem domains: approach compensation, foraging

and conflict resolution.

Our aim was to provide tools that would facilitate incremental, evolution-based study of adaptation and learning. This approach to studying intelligence can help solve the problems of symbol grounding and reference frame related to the computational metaphor for intelligence and the problem of scaling related to the embodied approaches.

Chapter 2

Motivation and Issues in Adaptive Behaviours

Contents

2.1 Motivation

Consider an animal that competes with others members of its own species for a limited resource. Such an animal can have a number of increasingly sophisticated strategies for resolving conflicts. One very basic, perhaps implausible, strategy would be to fight until victory or death. Increasingly complex physiologies and cognitive abilities can support increasingly efficient conflict resolution strategies such as, yielding, adherence to pecking order, and coalition building. Each of these strategies, along with the physiological and cognitive features necessary to support them, will have evolved as a result of providing the animal with an increased chance of survival and reproduction. We will use this conflict resolution scenario as a motivating example recurring throughout this dissertation in the context of methodology, design, implementation and experimental evaluation.

Humans have evolved so far that our conflict resolution strategies use rules defined on a societal level, i.e. laws, and proxies for conflict participation, i.e. law-enforcers. These strategies need complex support machinery such as a an extensive long-term memory, an ability to reason logically and the ability to communicate using language. Each conflict resolution strategy can be seen from a learning point of view, as a solution to a problem of adaptation. As strategies become more sophisticated, so do the related learning problems. The more environmental, social and historical elements an entity can utilise in a strategy, the more successful it can be.

The goal of AI is to study and model intelligence as found in animals, including humans, and to use our understanding of intelligence to produce artifacts that embody it. One approach to studying intelligence is to study intelligent behaviour and the physiological mechanisms related to behaviour. Another approahch is to study mental processes through introspection. We can view modelling and implementing intelligent behaviours as an attempt to solve the problems of adaptation and learning similar to those that have faced animals through evolution.

The work presented here defines a behaviour-based methodology to the de-

velopment of adaptive robots that progresses by developing increasingly complex problem solving strategies. This incremental development allows the adaptive elements of new strategies to use existing cognitive machinery to restrict the new problem spaces. We call this approach to developing adaptive behaviours Behaviour Based Learning (BBL).

Our work contributes both to cognitive science and to intelligent systems engineering. Our contribution to cognitive science is a demonstration of how adaptive behaviours may have evolved or developed. We show that by incrementally adding specialised circuitry around general memories or traces of the activity of existing circuitry, it is possible to produce increasingly sophisticated adaptive strategies to cope with problems of survival and procreation. The process we describe also suggests mechanisms for the integration of behaviours, both adaptive and non-adaptive.

Our contribution to engineering is a general development methodology that produces adaptive algorithms for robot control. The methodology improves on existing methodologies in two ways. First by producing algorithms that employ multiple concurrent strategies of different sophistication to a given problem. In the case of limited hardware failures, such algorithms can retain a higher level of performance than algorithms that use a single solution. Secondly, the adaptivity in the algorithms developed using our methodology is highly restricted. Highly restricted adaptivity allows the algorithm to adapt quickly after minimal amounts of experience. The weakness of the algorithms our methodology produces is that the solutions are highly specialised compared to existing learning algorithms. Lastly, we provide a set of tools supporting our development methodology.

As a minor point, our methodology and tools have been developed to facilitate the implementation in particular, of strategies that have been recognised in animals. The evolutionary history of such strategies naturally provide solutions on different levels of sophistication. A such, our methodology and tools can also serve to develop and test theories about the evolutionary history of behaviours observed in

animals.

2.2 Definitions and Issues

We use the term *behaviour* to denote only externally observable displays of movements. We use the term *behavioural layer* to denote the internal structures that give rise to a specific behaviour. We call the solution to a problem provided by a behaviour a *behavioural strategy*.

A *behaviour* implements a partial mapping from a set of *stimuli* to a set of *responses* [Arkin, 1998]. *Adaptation* is a change in this mapping as a result of experience, i.e. previous stimuli and responses or *actions*. In general machine learning terms [Anthony and Biggs, 1997], a behaviour is a *hypothesis* that *classifies* different stimuli into a sets of actions. Adaptation is changes to the behavioural hypothesis based on the observed *examples*.

2.2.1 Speed of Adaptation

In order for adaptation to be effective in a situated system, it is necessary that the changes in the expressed actions are take place within a meaningful time-frame. Different strategies operate on different time-scales. In terms of conflict resolution, it is important in a fight to learn quickly whether you are stronger or weaker than your opponent in order to yield quickly if you are weaker. However, learning where to find a good solicitor to represent one in court can be a matter of gathering information over weeks, months, or even years.

There may be many factors that slow down the speed at which a robot adapts both with regards to *action selection* and *learning from experience*, the two subproblems of adaptation [Maes, 1994]:

- An unwillingness to explore new actions can reduce the available learning data/examples.

- Inappropriate action units, i.e. an inappropriate *alphabet* for describing the examples, can mean that the robot might have to explore a large number of actions before a useful combination can be found. The number of actions that must be explored grows exponentially with the number of actions that need to be combined. After many actions have been taken it is also difficult to identify which ones produced the observed result. This is the classic delayed reinforcement problem. Richer reward structures have been suggested as one way of overcoming this problem [Dorigo and Colombetti, 1993, Matarić, 1994].

- Unbiased selection of exploratory actions can mean that a robot must try a vast number of unhelpful actions before finding a useful one.

- If the representation of the mapping between the stimuli and the actions, the *hypothesis language* is complex, making changes or selecting actions can take a long time.

- If the changes that are made to the mapping are too small, i.e. the *learning rate* is too low, as can be the case with numerical utility estimation methods such as Q-learning [Sutton and Barto, 1998], it may take a lot of experience to produce a change in the expressed actions.

Matarić [Matarić, 1994] stated that a situated learning model should:

1. minimise the learner's state space

2. maximise learning at each trial

Another approach to speeding up learning is to provide more feedback. Extra feedback can help pruning the search space by indicating whether a learner is on the right or wrong track. When this is done in a social context during the development of a child, it is called *learning by scaffolding* [Breazeal and Scassellati, 1998]. When humans used this technique to teach animals novel behaviours, it was called *shaping* [Dorigo and Colombetti, 1998]. Finally when such mechanisms are de-

signed into a system they have been called *progress estimators* or *heterogeneous reward function* [Matarić, 1994].

2.2.2 Using Prior Knowledge

One way of using prior knowledge of a problem domain to increase the speed of adaptation is to express such knowledge in a machine readable formalism and let a learning algotihm use it automatically to improve the way the hypothesis is changed. This approach is common in Explanation Based Learning. Another way of using prior knowledge is to let it to guide the identification of an appropriate hypothesis space to search, i.e. let it influence the choice of exploratory actions. This kind of bias is called *declarative bias* [Russel and Norvig, 1995, Dahl, 1998].

2.2.3 Problem Restriction

There are several ways of reducing a learning problem. Reducing the number of stimuli or the number of actions reduces the dimensionality of the learning problem. Choosing actions that minimise the length of useful action sequences reduces the complexity of the hypotheses. Dividing a large learning problem into several smaller problems also helps to reduce the combinatorial complexity of a problem and hence restricts the total search space presented.

Nehmzow and Mitchell [Nehmzow and Mitchell, 1995] provided the most general definition of the Robot Learning Problem (RLP). They define the robot learning problem as learning the control function f, a mapping from the perceived state S to a set of actions A. The Behaviour-Based (BB) paradigm [Arkin, 1998] restricted RLP to only do learning related to a set of pre-specified behaviours [Mahadevan and Connell, 1991, Asada et al., 1995], learning in multiple isolated behaviours simultaneously [Stone and Veloso, 1999, Andre and Russel, 2001] or learning related to combining behaviours [Maes and Brooks, 1990, Matarić, 1997]. Problem restriction is an important topic for the whole field of ML. In robotics the real time constraints make problem restriction crucial. The layered restrictions in-

troduced by the BB paradigm have also been used by other pragmatic approaches [Stone, 1998]. Bryson [Bryson, 2001] proposes principles aligned with the BBL approach to learning, where the context of a behaviour restricts the learning problem considered.

2.2.4 Robustness of Solutions

There are generally two aspects to the robustness of a solution. First there is robustness with relation to failures in the computational hardware. Architectures that use distributed processing can still function after these 'lesions'.

Secondly there is robustness with respect to the environment. Robust solutions can function in spite of changes in the environment and sensor or actuator errors. The second kind is the kind of robustness that is increased by developing controllers for real robots [Maes and Brooks, 1990, Nehmzow and Mitchell, 1995] rather than in simulation [Christensen, 2000, Gerkey et al., 2001] or abstract problem representations [Fagg et al., 1998]. The first kind of robustness can be increased by using distributed processing architectures.

2.2.5 Integrating Programmability and Adaptation

The inherent plasticity of neural systems gives animals a pervasive adaptability that is not found in implementations based on high-level programming languages. On the other hand, the complexity of implementing specialised behaviours in terms of neural structures appears too great to be a reasonable approach.

Merging high-level behaviour specification and pervasive adaptivity is a fundamental problem in implementing adaptive behaviours. One possible solution to this problem is a high-level specification framework for neural structures.

Such a framework would have to allow different degrees of plasticity in different structures, to reflect the different degrees of adaptivity found in animal behaviours [Gould and Gould, 1999].

circuits access to the provided data $x\mathsf{p}$ for data type x.

The work presented in this dissertation deals with these issues in order to produce improved adaptive capabilities in robots. Adaptation, as stated above, is an integral part of developing sophisticated solutions to the problems facing an entity trying to optimise its performance in terms of survival or in other terms, as defined by human engineers.

The methodology and tools presented in this dissertation were developed to deal with the issues outlined above. The emphasis on design allows a progammer to use prior knowledge of the problem domain to develop efficient solutions. The incremental development methodology improves the speed of adaptiation by promoting highly specialised adaptive mechanisms. It also provides explicit guidelines on how to integrate pre-programmed and adaptive elements.

The use of multiple concurrent strategies of different sophistication, and the distributed nature of our behaviour model, both improve the robustness of the solutions produced. The quality of our example solutions is evaluated through experiments on simulated robots.

Chapter 3

The Neural Circuit Model of Behaviours

Contents

3.1 Schema Theory

Schema theory [Arbib, 1989] represents the structure of the brain as elements a network of distributed, communicating and concurrently executing nodes or *schemas*. A schema can represent a perceptual structure, a structure for motor control or a structure for any other activity in the brain in general, such as the recognition of different objects, planning or control. Schemas are ultimately defined in terms of interaction with an external environment rather than through reference to pre-set formalism.

Schemas have been successfully used as elements of behaviours, providing a finer grain of distributed computing than subsumption-style architectures [Arkin, 1989]. Figure 3.1 depicts a hierarchy of schemas.

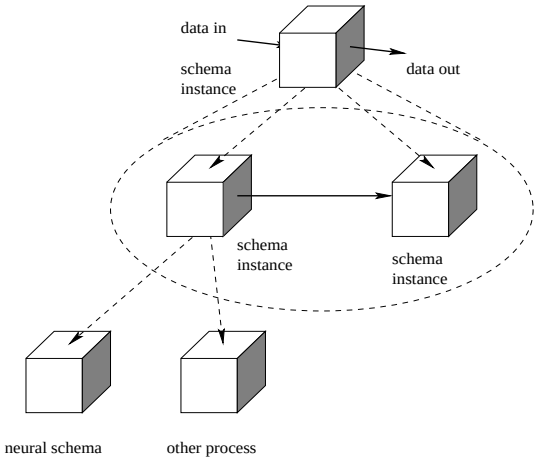


Figure 3.1: Schema Hierarchy, from [Weitzenfeld, 2000]

3.1.1 The Abstract Schema Language

The Abstract Schema Language (ASL) [Weitzenfeld, 2000] provides a formalism for describing schema based architectures. The ASL schema model addresses a number of design issues:

- **Encapsulation** - Schemas present a uniform interface independent of the internal structure or function of each schema instance.
- **Re-usability** - Output from a single schema can be used as input by many other schemas. This allows different behaviours to share sensor and motor schemas.
- **Heterogeneity** - Different schemas may be implemented with different programming paradigms as long as they support the required external interface.
- **Concurrency** - Schemas execute concurrently.
- **Hierarchy** - A schema may call on other schema to provide supporting functionality.
- **Abstraction** - A set of schemas can be treated as a single component called an *assemblage*.
- **Communication** - Message passing through uni-directional ports.

A schema is defined by specifying input and output ports, internal structures and an endless loop for processing the input and producing the output.

3.2 Neural Circuits

NCs correspond closely to neural schemas as defined by ASL, but with two main differences. NCs can pass structured data between each other, though still only uni-directionally. Neural circuits also keep excitation or control separate from data. The Neural Circuits were developed with a focus on programmability rather than biological credibility. It is our opinion that the generalised inter-circuit communication and the separation of the control- and data-streams makes the NCs easier to use for procedural implementation of pre-defined behaviours. However, a procedural implementation of the behaviour of a NC is not necessarily adaptive in the way an artificial neural network implementation would be. To emphasise, a NC does

not have to contain a neural network. In our examples the functionality of NCs are provided by a procedural implementation in C++.

'Motor program' is a term used in ethology when discussing innate behaviours. These behaviours used to be called fixed action patterns, but turned out to be very flexible in spite of being mainly reactive. The new designation means that the behaviour is decided by a specific set of pre-programmed neurons rather than learnt by a general learning structure [Gould and Gould, 1999]. The circuits we have presented here correspond to elements of reactive behaviour in animals and form a foundation for our study of adaptive behaviour.

Consider again our motivating example from Chapter 2, in which a small animal had to compete with other members of its species for a scarce resource. One of the simplest conflict resolution strategies one can imagine is to attack all competitors on sight, uncritically. While the evolutionary utility of this strategy is debatable, as an example it demonstrates clearly the use of neural circuits for behaviour design.

To implement this strategy, the animal would need to identify and attack its opponent probably using its eyes for identification and legs for propulsion. The simulated Khepera robots we used for our experiments had a K6300 colour camera and two controllable wheels providing corresponding physiological functionality. Our design of an *Uncritical Fighting* strategy started with abstracting away the application programming interfaces (APIs) of the camera and the wheels by encapsulating them in *Neural Circuits* (NCs).

NCs correspond closely to neural schema, being an abstraction for an arbitrary complex collection of neurons providing a restricted functionality. A neural circuit accepts inputs from a set of other circuits and provides one output when the input causes the circuit to activate or *fire*. However, while ASL only allowed boolean and numerical values to be passed between schema, NCs allow structured data to be passed between circuits, providing that the structure of the data is static. This reflects the fact that a set of axions leading into a collection of neurons can be inter-

preted to have any structure by the receiving neurons. However, the configuration of the axons connecting the neurons does not change.

The neural circuits of the *Uncritical Fighting* layer are shown as boxes in Figure 3.2. Boxes with rounded corners indicate encapsulations of external APIs. Boxes with dashed lines are circuits from other layers. The arrows connecting the NCs indicate excitation or control, whilst the labels on the arrows indicate the communication of percepts or data. The vertical lines ending in a circle indicate inhibition.

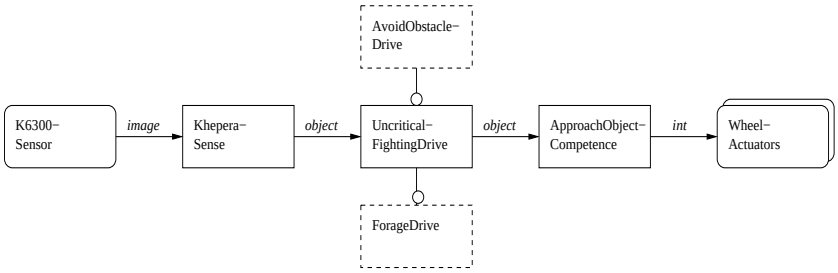


Figure 3.2: Circuits for Uncritical Fighting

The interface for camera control was encapsulated in a **K6300 Sensor** circuit which was always firing, providing an abstract data type, the *image* percept, which contained the image date provided by the camera. The interface for controlling the wheels was encapsulated in two **Wheel Actuator** circuits that accepted any number of signed integers. The circuits added these integers, the sum was bound by the maximum and minimum wheel speeds and used to set the speed of each wheel.

An animal following the *Uncritical Fighting* strategy described above would have to be able to use its visual and locomotional facilities to recognise and attack its competitors. These two skills are instances of more general capabilities: object recognition and approach. It can be useful to differentiate between approaching an immobile target and chasing a mobile target. Here , however, we ignore this difference for simplicity's sake. We imbued the general object recognition and ap-

proach capabilities in the simulated Khepera robot by means of a **Khepera Sense** circuit and an **Approach Competence** circuit. The **Khepera Sense** circuit recognised the colour of a Khepera robot in the provided *image* percept and used the hight and relative position of the robot in the image to estimate the other robot's range and bearing. The range and bearing were used to instantiate a general *object percept* which the **Khepera Sense** circuit provided as its output. The **Approach Object Competence** circuit accepted a general *object percept* and used the relative position of the object to servo the robot toward it by requesting the relevant wheel speeds from the **Wheel Actuator** circuits.

The reason for creating a dedicated neural circuit for a particular sense or capability was to remove functional duplicity. It was clear from our design that a Khepera recognition capability would also be needed for other more advanced conflict resolution strategies and an approach competence would also be needed by feeding related behaviours. Separating out these general capabilities and letting different behaviours share access to them prevents the reproduction of almost identical functions in different behaviours. We used the terminology **sensor** circuit and **actuator** circuit for circuits that encapsulate hardware. For derived capabilities we use the terms **sense** circuit and for general skills we use the term **competence** circuit.

The *Uncritical Fighting* behaviour sometimes had to be inhibited to allow the expression of other more pressing behaviours such as obstacle avoidance, and also has to inhibit less pressing behaviours such as foraging. If circuits that are shared between multiple behaviours are inhibited, this implies an inhibition of all the behaviours involved. In order to allow the inhibition affect only the *Uncritical Fighting* behaviour without inhibiting any other behaviours, we introduced a **Uncritical Fighting Drive** circuit. This circuit is unique to the *Uncritical Fighting* behaviour and is the recipient and origin of any inter-behavioural inhibitory connections.

3.2.1 General Circuits of Reactive Behaviour

The elements of reactive behaviours are presented in Figure 3.3. They were inspired by the BB *Edmund* architecture which had explicit elements for sensors, actuators and control [Bryson and McGonigle, 1997]. Concurrent and distributed execution of behavioural layers are two of the strengths of the BBAI paradigm that help improve efficiency and allow graceful degradation. Work by Nicolescu and Matarić [Nicolescu and Matarić, 2001] introduces another architecture that provides explicit access to elements of behaviour for learning purposes.

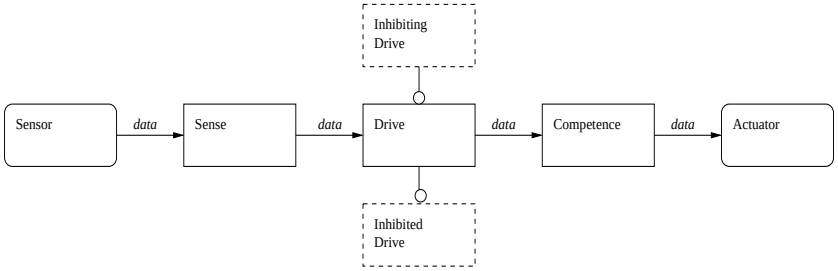


Figure 3.3: Circuits for Reactive Behaviour

Most physiological theories of behaviour concern different brain areas and their functionality [Carlson, 2000]. The neural circuit model allows us to take inspiration from these theories by allowing us to do cognitive modelling on not only a neural level but also on a super-neural or neural circuit level. This is useful when general effects and algorithms are suggested that can be implemented with a standard programming language although it is not clear how to implement anything equivalent using a low level neural model.

In the NC model, sensors and actuators encapsulate the lower-level hardware representations. senses and competences are high-level sensor and actuator interpretations respectively. Drives handle interactions between multiple senses and competences and interacts with other behavioural layers through inhibition. Each behavioural layer can contain several **Sensor**, **Sense**, **Competence** and **Actuator** circuits.

Sensors/Actuators As stated above, **Sensors** and **Actuators** are encapsulations of lower level hardware representations. **Sensors** are always firing, providing the raw sensor data for the senses to interpret. **Sensors** are commonly shared between the different behavioural layers needing the related data or actions.

Senses Senses or virtual sensors are interpretations of the raw data provided by the sensors. They are targeted to the need of the particular behavioural layer or layers they are part of. A sense only fires when it recognises one of the data configurations it has been programmed to detect, e.g. a food sense would only fire when the data from the relevant sensors indicate that food might be present. Some high-level senses are formed from combinations of lower-level senses. In these cases the senses provide several layers of activation filtering.

We interpret certain kinds of memories as types of senses, sensing the past effectively. The use of those kinds of memories corresponds to the use of senses in the activation filtering chain in that they are often used to create higher-level senses and to provide input to drives. The role of memory and learning circuits is expanded upon below.

Drives A drive is a merging point for activation data both from any senses that are part of the drive's behavioural layer and from drives in other layers. Drives are unique repsresentations of behavioural layers. They are the only NC that cannot be shared between layers. They typically handle evaluation of data from multiple sensors in order to determine its general excitatory state and then modify this state according to any inhibitions received from other behavioural layers. Firing drives may in turn inhibit other behavioural layers. In the NC model inhibitions are indicated by lines between circuits with a circle at one end indicating the inhibited circuit. This is the notation used in the original subsumption architecture [Brooks, 1986].

Some learning circuits, in particular **Associative Memory** circuits, can take the role of drives for certain learning behaviours. Any **Associative Memory** circuit

connects a set of senses and a set of competences, bypassing the hard-coded drives and taking over the role of a dsrive itself. This use of memory is discussed in further detail below.

Competences The competences are high-level abstractions of behavioural patterns. If activated they request actions from the actuators that correspond to the behavioural concept the competence encapsulates. The most commonly used competence in our work is the **Approach Object Competence** circuit. It requests the wheel speed modifications necessary to make the robot servo toward a given target.

Competences, like sensors and senses, can be shared by several behaviours. When this is the case, they act as additional activation filters following the senses and drives. If the competences implement selection or combination mechanisms that create one set of actuator requests from many competence activations, then they are a part of the activation filtering, filtering out a number of the participating behaviours.

3.2.2 Information Extraction

The NCs of a behavioural layer constitute a series of nodes for information extraction. All behavioural layers must convert the data provided by the **Sensor** circuits to the format accepted by the **Actuator** circuits. In the general behaviour model we presented in Figure 3.3 information extraction is done by the **Sense** and **Competence** circuits.

The **Sensor** circuits provide the raw data from the sensors. These data are received by **Sense** circuits that may also receive data from other **Sense** circuits. The **Sense** circuits process the received data, and the refined data are then passed on to the **Drive** circuits or other **Sense** circuits. In the example behaviour presented in Figure 3.2, the **Khepera Sense** circuit receives image data and processes this data into the internal concept of an object. The drives usually only do activation filtering, and pass the received data on without modifying it.

The **Competence** circuits further refine the data provided by the **Drive** circuits to the format needed by the **Actuator** circuits. In our example the **Approach Object Competence** circuit receives an **Object Percept**, the internal concept of an object, and produces wheel speeds.

3.2.3 Activation Filtering

The circuits of a behavioural layer also form a series of nodes for activation filtering. The activation of each circuit in the series depends on the activation of its predecessor and on the the data provided. The circuits in a behavioural layer constitute increasingly strong restrictions for activation. The **Sense** circuits implement data related restrictions on activation local to a behaviour, e.g. in the *Uncritical Fighting* behaviour presented in Figure 3.2, the absence of another Khepera robot will keep the **Khepera Sense** circuit from firing, and prevent the subsequent circuits from being activated by that behavioural layer. Shared **Competence** circuits, however, may be activated as a member of another behavioural layer. The **Drive** circuits implement global activation restrictions on a behavioural layer in terms of inhibition by other behaviours. Finally, the **Competence** circuits implement restrictions on activation related to actuator activation. We can imagine that the **Approach Object Competence** circuit in Figure 3.2 implemented a lazy approach strategy that stood still whenever the relevant object approached the robot of its own accord, such as when another robot was already approaching. In this case, the **Approach Object Competence** circuit would require the wheel actuators not to be activated.

3.2.4 Modularity and Extendability through Neural Circuits

The sharing of NCs between behavioural layers provides two advantages. It reduces code duplicity by having only one instantiation of certain cognitive functions. In more coarse models, it might have been necessary to re-implement some of this functionality for each behavioural layer.

The ability to share NCs also facilitates the implementation of many new cognitive functions. New cognitive functions are often relatively small variations on existing functions. In our model of behaviour, only the difference between the functions needs to be implemented, for example to produce a **Moving Khepera Sense** circuit we can use output from the existing **Khepera Sense** circuit and a buffer-type memory in order to analyse the changes in a Khepera's position over time.

3.3 Behaviour Design using Neural Circuits

The general design method we have used in the work presented in this dissertation is that of beginning with simple reactive solutions to a problem and from those, build up increasingly sophisticated solutions to the same problem by introducing increasingly adaptive behavioural layers until the target behaviour is reached.

Our work on designing behavioural layers using the NC model has raised a number of design issues that are more or less relevant to the thesis we defend; that BBL is an improvement on traditional ML techniques. In this Section we discuss those issues as a contribution to the BB paradigm in general rather than as direct support of our thesis.

3.3.1 Sense Extensions

Commonly, when adding a new behavioural layer to a controller, there is also a need to change the underlying sensory circuitry for example by defining a new perceptual attribute. In these cases we chose to modify existing sense circuits rather than to add new sensory circuits. The different feeder properties are based on attributes such as colour and relative size in the visual field. They are likely to be processed in parallel in biological vision systems [Bruce et al., 1997]. We implement the visual feeder related processing in one circuit not only for simplicity but also to avoid the computational overhead of creating two sensory circuits rather

than one. It is fair to argue that new sense circuits for empty feeders and feeder casing would be more in line with a cognitive model.

A good software engineering argument for adding a separate sense is that changing an underlying sense affects all the circuits dependent on that sense, whilst adding new senses does not affect other behavioural layers.

3.3.2 Competence Generalisation

During our work it was clear that the actuator manipulation that took place within several of the drives should be generalised and turned into an explicit competence. Simple behavioural layers, however, do not always need explicit competences but can leave the actuator manipulation to the drives.

Generalising existing competences is very intrusive in that it changes circuits in low level layers. This again affects all the behaviours relying on those circuits and introduces a lot of scope for bugs and unpredictable side-effects.

A good solution to this problem is to re-implement an existing specialised competence or drive in terms of two new competences, firstly, the new generalised version of the competence and secondly a new version of the old competence which keeps the interface to the dependent circuits and does the necessary translation in order to make use of the new more general competence for actuator manipulation. The general competence is subsequently available to new behavioural layers while there is no apparent change for the existing layers.

3.3.3 Design and Evolution

The design principle of non-intrusiveness is not necessarily in line with the way behaviours evolved in animals and humans. This presents a potential conflict between design issues and cognitive modelling issues.

Our work slants more toward cognitive science inspired robotics than cognitive modelling with robots, i.e. it is more focused on intelligent system engineering than on cognitive science.

We have done cognitive modelling to the level of replicating the functionality of vaguely defined brain areas, in particular in terms of visual and spatial senses. There is no neuro-scientific evidence defending our individual neural circuits or layers.

With a model so far abstracted from neural and evolutionary reality, it is premature to allow development to be dictated by adherence to the evolutionary process. We are currently concerned with relatively pragmatic issues such as the feasibility of expressing certain adaptive behaviours on a high level of abstraction. In studying these kinds of issues it is more beneficial to facilitate the programming of highly complex structures than to adhere strictly to the correctness of our cognitive models.

Following a set of design principles when doing cognitive modelling creates the potential for the kinds of conflicts described above. These conflicts may represent insurmountable differences between the two different methodologies. In such a conflict situation, cognitive models have the advantage that they represent the only existing solutions to the more complex behaviour expression problems. In general, the two should be kept as close as possible, and the possibility of conflicts should be kept in mind

Chapter 4

Behaviour-Based Learning

Contents

As a way of tackling the current issues in adaptive behaviour we suggest an approach to adaptivity that is inspired by the minimalism of BB systems. BB systems are able to provide efficient solutions to certain problems by abandoning generality and instead producing problem specific solutions. We suggest that a similar trade-off can be beneficial in the context of adaptive mechanisms.

However, a BB approach to adaptivity is likely to encounter similar problems of generalisation to those of the general BB approach [Tsotsos, 1995]. One of the main problems faced when developing a BB approach to adaptivity is to identify ways of re-introducing generality without re-introducing the issues the BB approach was originally developed to avoid.

We have developed a set of general guidelines for developing increasingly adaptive behavioural strategies based on neural schema. The guidelines constitute what we call Behaviour-Based Learning (BBL). They are inspired by mechanisms of the evolutionary process, and the evolutionary history of adaptive behaviour in animals.

4.1 Theories of the Evolution of Learning

In order to develop a model of BBL we studied a number of theories of the evolution of learning. Theories of the evolution of learning have been developed in several scientific fields. Here we review the four theories that have inspired our guidelines for developing adaptive behaviour. Two of these theories are from psychology, one is from neuro-science and one from from linguistics.

4.1.1 Moore's Theory of the Evolution of Imitation

The most detailed theory of how high-level learning capabilities have evolved from reactive behaviours is Moore's theory of the evolution of imitative learning [Moore, 1996]. Moore describes the evolutionary relationships between different forms of learning abilities starting with reflexive behaviour and ending with cross-modal movement imitation. The resulting tree structure is presented in Figure 4.1. Arrows rather than lines between forms of learning indicate that one process is a special case of another.

Moore's argument is that like other biological processes, forms of learning 'evolve from similar, simpler processes through series of small adaptive changes.' [Moore, 1996]. Moore also defends the biological plausibility of the hypothesised steps and argues their consistency with the comparative data.

Where not obvious, Moore discusses the changes that define the difference between the related forms of learning. The difference between alpha-conditioning and Garcia conditioning is that in alpha-conditioning the stimuli are all presented together, removing the need for memory structures to bridge the temporal gap between stimuli presentation. Garcia conditioning is also called selective conditioning because only a restricted set of stimuli and responses can be associated. Pavlovian conditioning generalises Garcia conditioning.

In Thorndikian learning a response or action must take place before a stimuli or reward is presented, but the action may be species-specific, i.e. a pre-programmed

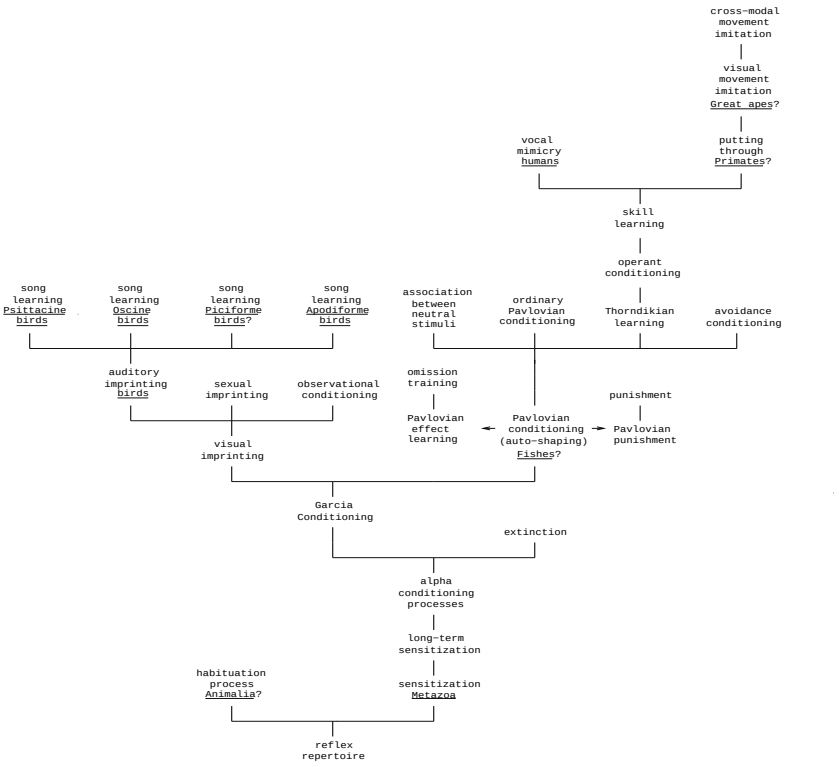


Figure 4.1: Relationships between Forms of Learning, from [Moore, 1996]

natural response to the situation. Skinner, on the other hand, demanded that for operant conditioning, the responses or operants can not be elicited, but have to be completely novel and arbitrary. Skill learning is a specialisation of conditioning in that it uses only implicit and not explicit reinforcement. Putting through or molding is guided skill learning where the skill must be learnt by reproducing a motion that has been passively experienced by having a teacher physically guide the student. This reduces the number of supporting stimuli by removing the kinaesthetic feedback uniquely associated with actively producing a response. Visual movement imitation relies on reproducing a motion from visual experience only as opposed to the visual and proprioceptive experiences available in putting through. Moore emphasises that this is a surprisingly small step when putting through is viewed as self-imitation.

The steps involved between Thorndikian learning and cross-modal movement imitation are summarised as: 'an expansion of repertoire, the dropping of explicit reinforcement, a loss of kinaesthetic feedback, a loss of other proprioceptive feedback and a product of programmed tactile exploration'.

4.1.2 The Mirror-System Hypothesis

Arbib [Arbib, 2000] presents a theory of how language learning can develop from imitation learning by use of a neural structure called the Mirror-System. The theory hypothesises five stages for the evolution of language.

1. grasping
2. a mirror system for grasping (i.e. a system that matches observation and execution)
3. an imitation system for grasping
4. a manual-based communication system
5. speech

Arbib presents data that shows how a region of the brain named the Mirror-System in the F5 area of a monkey's pre-motor cortex is active both when it grasps and when it observes others grasping. This kind of supportive structure suggests how it is possible to evolve self-imitation capabilities into general imitation capabilities as hypothesised by Moore in the theory we presented above.

The Mirror-System Hypothesis then argues that this structure is the homologue of Broca's area in humans, providing a neuro-biological missing link for the theory that sign language preceded speech. The parity requirement for language is met by the mirror-system for grasping because it provides the capability to produce and recognise a set of actions.

The step from recognising individual actions to recognising an action as an abstract representation of a class of actions is presented as a having taken place in five stages:

1. Pragmatic action directed toward a goal object.
2. Imitation of such actions.
3. Pantomime in which actions are produced away from the goal object.
4. Abstract gestures based on pantomime hat emphasise distinctions that are difficult to mime, such as the distinction between the verb 'grasping' and the adjective 'graspable'.
5. The use of abstract gestures for the formation of compounds which can be paired with meanings in a more or less arbitrary fashion.

Arbib argues that the transformation from sign-language to speech was a result of the manual-orofacial symbolic system 'recruiting' vocalisation to allow gestures to assume a more open referential character. Language syntax, Arbib argues, was learnt as an abstraction of regularities in many sentences. With syntactic and semantic knowledge, came the ability to extract the sequential or semantic structure of an utterance.

The Mirror-System Hypothesis adds to Moore's theory of the evolution of imitation a possible explanation of how speech evolved from imitation. The two theories together form a complete chain from reactive behaviour to natural language use. There are however, other aspects of learning that are not covered by the two theories reviewed above. Firstly, the theories do not discuss the physical structures and processes used to support the different forms of learning. Secondly, they do not discuss how to solve the problem of forming higher-order abstraction. Higher-order abstractions are necessary for reducing the complexity of high-level forms of learning. Below we present two other theories of the evolution of learning. One describes the evolution of a set of processes for establishing memories and also defines a number of properties for physical memory representation and the other describes the evolution of the ability to use increasingly abstract modes of reference.

4.1.3 The Multiple-Entry, Modular Memory Model

Psychologists have developed many theories of what structures and processes are involved in memory formation and usage [Baddeley, 1997, Collins et al., 1993, Conway et al., 1998].

The Multiple-Entry Modular (MEM) Memory Model developed by Marcia Johnson [Johnson, 1992, Johnson and Hirst, 1993] stands out by explicitly handling the evolutionary aspects of memory and learning, but also by being particularly clear and explicit about the processes and structures involved. This makes the MEM model well-suited as a basis for implementing adaptive capabilities in robots. Here we review the MEM model emphasising its contributions with regards to the evolution of memory and learning.

The MEM model describes a set of abstract processes that are involved in the establishment and use of memories. The processes are divided into four levels, two perceptual levels, P1 and P2, and two reflective levels, R1 and R2. The perceptual processes are activated by perceptual inputs while the reflective processes

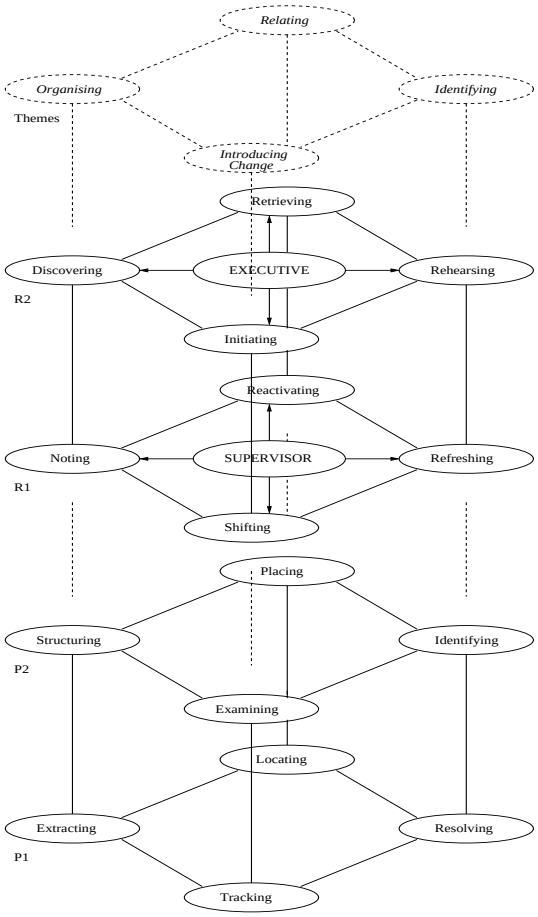


Figure 4.2: MEM Processes, from [Johnson and Multhaup, 1992]

are internally activated. The processes on each level describe increasingly complex instantiations of four different themes that run through all the levels. The processes of the different levels aligned according to their themes are presented in Figure 4.2.

The themes: identifying, relating, organising and introducing change, form a cycle of memory manipulation. The general task of the themes are:

- **Identifying:** Identifies and prolongs the activity of objects of perception and thought.
- **Relating:** Goes back to earlier objects of perception and thought. Also situates such objects in a context.

- **Organising:** Creates relations across time and/or events.
- **Introducing change:** Introduces changes in the systems activation patterns.

The task of the individual processes are as follows:

- P1:
  - Resolving - Resolves perceptual arrays into units, e.g. edge detection or *geons* [Biederman, 1987].
  - Locating - Locates e.g. abrupt changes in illumination.
  - Extracting - Extracts invariants from perceptual arrays, e.g. texture, gradients, flow patterns, horizon ratios, cues, or the rapid expansion of features in the visual field that indicates a stimulus coming toward you
  - Tracking - Tracks stimuli in motion.
- P2:
  - Identifying - Yields a sense of what something might be from combined perceptual primitives or integrated geons.
  - Placing - Places objects or events in spacial relation to each other.
  - Structuring - Structures abstract patterns of organisation across spatially and temporally extended stimuli, e.g. syntactic structure from a sentence.
  - Examining - Actively inspects aspects of a stimulus array, mainly perceptual investigation implying redirecting attention.
- R1:
  - Refreshing - Prolongs ongoing activation, extending the life of an already activated representation or pattern, allowing it to bridge the gap between activation patterns.
  - Reactivating - Reactivates currently inactive information.

  - Noting - Notes relations among activated stimuli.
  - Shifting - Shifts perspective in order to activate alternative aspects of stimuli. Including shifting working agenda to other (sub)agenda.

- R2:
  - Rehearsing - Cycles information in a self-generated format in order to remember or use it, bridging longer gaps than refreshing and giving rise to stronger phenomenal sense of keeping something in mind.
  - Retrieving - Retrieves memories by using internal memory cues, e.g. in trying to remember the name of a restaurant, you try to remember who might have told you about it.
  - Discovering - Discovers relations in a less direct manner than noting, e.g. relations that rely on other relations.
  - Initiating - Uses strategic ways of activating unactivated parts of information.

The different levels correspond to increasing cognitive complexity. A general intuitive presentation of the cognitive abilities on each level give the following abilities.

- **P1:** Allows reaching for an object.
- **P2:** Recognises what an object is.
- **R1:** Has goals and agendas for object manipulation.
- **R2:** Negotiates multiple goals and agendas.

Agendas Perceptive and reflective processes are related through abstract structures called agendas. Agendas are recipes for accomplishing cognitive actions

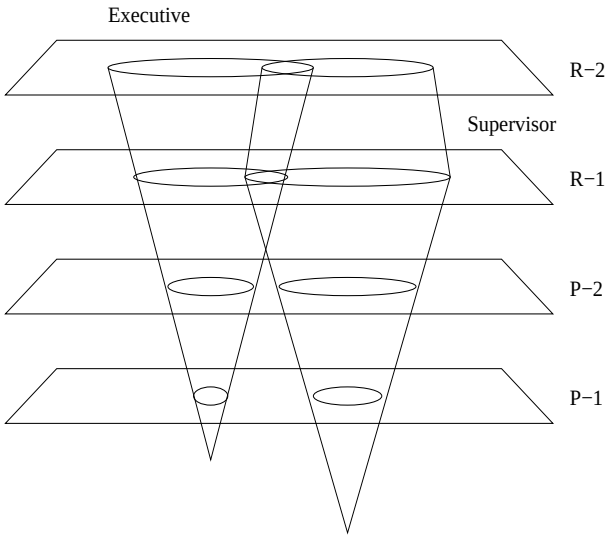


Figure 4.3: MEM Model Agendas, from [Johnson and Multhaup, 1992]

which can order, activate and monitor the cognitive processes of the different levels. They keep high-level goals and influence the activation of lower level memories. Routine goals can be schematised or compiled into cohesive agendas through practise.

The MEM model distinguishes between supervisory agendas that mainly relate to processes on the first reflective level R1, and executive agendas that mainly relate to R2 processes. The two different types of agendas and their influence are illustrated in Figure 4.3.

Supervisor processes can handle simple, well learnt regulation and monitoring tasks, e.g. setting simple criteria for old/new recognition judgements. Executive processes are necessary for more complex monitoring tasks involving multiple rules, testing imagined results against imagined consequences, e.g. the missionaries and cannibals problem, and embedded subgoals that are not routine.

Learning in MEM The MEM model sees a memory as a record of an operation performed by one or more of the processes described above, i.e. an echo or trace of

cognitive activity. Relating and structuring memories involves not only reactivating information, but also shifting from one aspect of meaning to another for the same item, noting relations or refreshing information.

Association between the content and the context plays an important role in remembering according to the MEM model. The context of a memory are remembered features incidental to the primary agenda, thus context is to some degree determined by the currently active agenda. Explicit memory for context is the heart of recollection.

The different layers of the MEM model support different forms of learning. P1 processes can for example allow adjusting to foreign accents or anticipate a baseball trajectory or locating more quickly over time an item that has a high probability of being in a particular location. P2 processes are involved in learning about the phenomenal perceptual world of objects such as chairs and balls and about events such as seeing a person sit down in a chair or catch a ball. R1 processes allow learning that relies on reactivation of memories such as temporal relation and complex world models. R2 processes support intentional learning where agendas are set up with the intention of learning something. This reflects learning types like skill learning and imitation.

MEM and the Evolution of learning The different layers of MEM correspond to different evolutionary stages. Each level describes processes that support an intact viable organism. A P1-only organism is able to engage in a variety of learning by connecting incoming stimuli to motor responses. An organism with P1 and an additional P2 system is able to attach responses differentially to recognised individuals, objects, and locations. Adding an R1 system permits comparing and connecting events across time and the expression of intention and control. Adding an R2 system allows the discovery of relations among many internally generated representations. P1 processes might be sufficient for acquiring skill at chasing prey, but P2 processes are necessary to recognise a familiar environment. R1 processes

are important for considering whether the current environment is preferable to yesterday's environment whilst R2 processes are needed to contrast one's own idea about relative desirability of two environments with someone else's idea.

P2 processes correspond to more complex forms of associative learning. The recognition aspect indicates that expectations are present on this level. R1 learning needs a complex world model. R2 learning needs hierarchical, high-level reference capabilities for executive control.

4.1.4 Deacon's Theory of the Evolution of Language

As a part of his research into the question 'Why do animals not use simple languages?', Terrence Deacon [Deacon, 1997] dismisses combinatorial complexity as the main difficulty in language acquisition. His argument is that simple languages can be constructed with small vocabularies and few grammatical rules that are combinatorially much simpler than the communication systems used by many species of animals. Such languages would clearly be a useful tool, so he asks: 'why have no other species than humans evolved this form of communication?'.

Deacon hypothesises that the answer is in the way language uses words as references to objects in the real world. He defines three modes of reference; iconic, indexical and symbolic and argues that all animal communication uses the two lower modes of reference while human language uses symbolic reference.

A mode of reference corresponds to the level at which a *sign* is interpreted as a reference to an idea, object or event called the *signified*. Iconic reference is the interpretation of a sign based on its similarity to the signified. Religious icons and portraits refer to religious entities and persons by means of their similarity in appearance. Indexical reference is a causal reference. A thermometer indicates the temperature by indexical reference. Deacon argues that most animal forms of communication uses indexical reference. Be it pheromone trails or alarm calls, there is a direct cause for the production of a sign. Lastly, a sign is a *symbol* used for symbolical reference when there is some social convention, agreement or

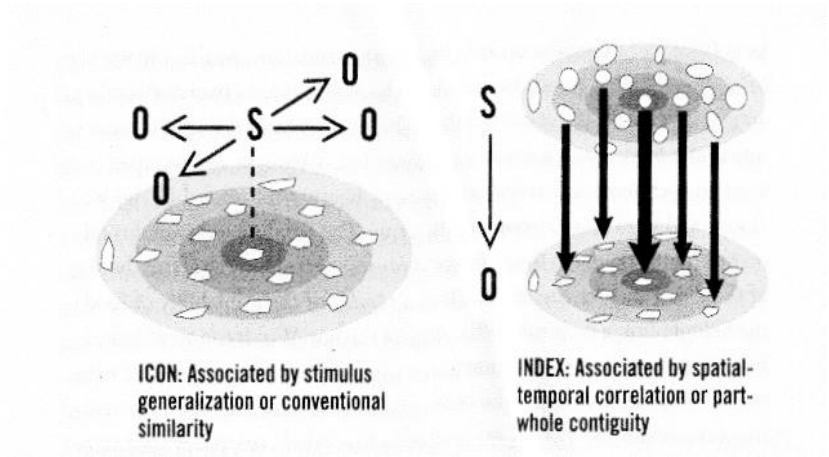


Figure 4.4: Iconic and Indexical Reference, from [Deacon, 1997]

explicit code which establishes the referential relationship. The difference between an icon and an index is illustrated in Figure 4.4.

Signs are not intrinsically icons, indexes or symbols, their mode of reference depends on the interpreter. Words as signs can be used for all the modes of reference, onomatopoeias such as 'splash' and 'bang' can reference events iconically by the similarity of sound. A word can also be used like a proper name, referencing indexically an entity through a unique link established through co-occurrence.

Symbolic reference is different in that there need not be any physical association between the sign and the signified. The meaning of a symbolic reference stems from its place in a system of such references. These systems build up an independence from physical co-occurrences and their meaning is decided by their relation to other symbols. The point when signs achieve this quality to an interpreter is by Deacon called the symbolic threshold.

This difference in mode of reference has implications for the kind of learning needed to develop these capabilities. Iconic reference does not entail any learning. Learning indexical reference is learning a deterministic relationship between a sign and a signified. Learning a symbolic reference however involves learning the relationship a sign has to other signs in the same symbolic system.

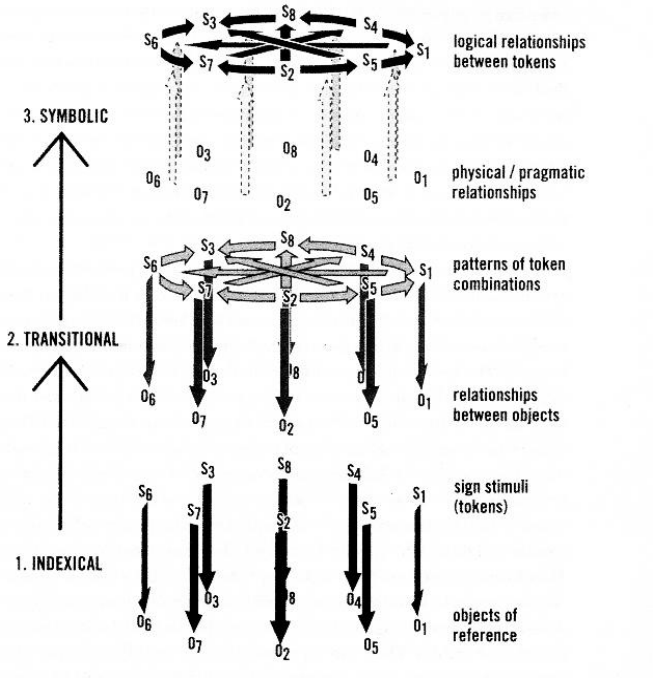


Figure 4.5: Constructing Symbolic Reference, from [Deacon, 1997]

Deacon shows how learning symbolic references depends on learning a large set of indexical references and reinterpreting the indexical representations internally to form the grammatical rules of the underlying symbolic system. This again depends on learning indexical references from a set of iconic references and reinterpreting these. Figure 4.5 illustrates the construction of the different modes of reference from reinterpretation of the underlying relations.

4.2 Modelling Animal and Human Learning

Many have seen the benefits of taking inspiration from biology when designing and implementing robot controllers [McFarland, 1999]. This merger of two scientific fields has created new areas of research such as the field of computational neuro-ethology [Cliff, 1991], the field of biorobotics [Beer et al., 1998] and the field of constructive biology [Nehaniv et al., 1999]. Taking inspiration from biology is particularly attractive as a way of restricting the search space for learning in autonomous agents and robots [Matarić, 2001a, Bryson and Stein, 2001].

Matarić praises the practicality of biology-inspired approaches over the traditional and mathematically pure approaches, and emphasises the multiple levels of bias present in biological learning systems [Matarić, 2001a]. Bryson and Stein point out the importance of the modularity found in the human brain as a way of reducing complexity [Bryson and Stein, 2001].

There is also biologically inspired work that models learning beyond the individual. Prescott and Ibbotson [Prescott and Ibbotson, 1997] look at the evolutionary history of the nervous system in particular with reference to spatial behaviour and trace fossils. Dautenhahn [Dautenhahn, 2000] presents a biological perspective on reverse engineering societies with respect to primates in particular.

In psychology, cognitive modelling is used to determine the computational mechanisms underlying cognitive control. When AI is inspired by psychological models, it is necessary to make those models implementable and efficient. Here we present work on biologically and psychologically inspired models for robot controllers. We also discuss how the controllers relate to the original theories and what benefits and/or constraints the theories provide.

Cognitive theories can describe both structural and functional aspects of cognitive machinery. Below we present robot controllers based on both aspects.

4.2.1 Modelling Structures

Different structures in the brain are attributed specialised functionality according to recent psychological and neuro-scientific theories [Carter, 1998, Carlson, 2000]. Work on modelling the mirror-system [Arbib, 2000] links the perception of particular action with the circuitry for performing those actions. Here we review models of two other brain areas, the hippocampus and the basal ganglia.

The Hippocampus Burgess, Donnet and O'Keefe [Burgess et al., 1994] present a neural simulation of hippocampal navigation in rats that models the place cells and 'head-direction' cells found in regions neighbouring the rat hippocampus as well as population vectors found in the monkey motor cortex. Population vectors encode a preferred action-related direction.

Region CA1 of the rat hippocampus provides many place fields covering an environment. Each place field consists of a collection of place cells. The firing phases of the different neurons related to a place field indicate the rat's placement within that field. Burgess *et al.* reproduce this firing pattern in a neural model and add goal cells encoding the position of goals. The goal cells also receive the reward that allows the system to learn maps by changing the strengths of the synapses of their incoming connections.

Redish and Touretzky [Redish and Touretzky, 1998] have further developed the neural model of the interaction of the rat's place and head direction systems presented above. In the extended model, the two main functions of the hippocampus are self-location and route replay. Route replay models neural activity during periods of sleep and goes beyond current evidence from experiments with animals by suggesting that rats with hippocampal lesions should be able to perform tasks that do not involve switching environments. This hypothesis has not yet been tested through experiments.

The hippocampus has traditionally been related to the ability to form long term memories. This work argues that the routes stored in the hippocampus are written

out to the cortex through route replay so that they can later be associated with the local views presented by switching environments.

The work presented by Redish and Touretzky models and contributes to the understanding of neural structures and also provides working implementations of specific behaviours. The low level at which this model is implemented and its high neural complexity, however, makes it difficult to incorporate as a part of a more general architecture wherein navigation is only one of multiple abilities.

The Basal Ganglia Gurney, Prescott and Redgrave model another brain area called the basal ganglia [Gurney et al., 1998, Prescott et al., 1999].

Gurney *et al.* argue that the connections and dynamics of these ganglia and their surrounding areas allow them to arbitrate between functional modules of the brain that compete for the same actuators, hence acting as an action selection device.

The model of the basal ganglia is on a circuit level showing how the different areas can interact to provide an action selection mechanism. Their model describes two main modules cooperating, the selector module and the adaptive controller module, both consisting of different parts of the basal ganglia. The selector module ensures the inhibition of behaviours that were not selected while the adaptive controller has two theorised functions: to make the selection circuit robust to variations in the number of competitors and to allow faster switching between functional modules.

Their simulation results demonstrate clean switching between competing modules. Their switching of their lesioned models is compromised in a manner that can be compared to some effects of Parkinson's disease, thus providing suggestions for future research on that disease.

As with the models of the hippocampus, this model only looks at an isolated area of the brain. By suggesting centralised action selection mechanisms for complex controllers, however, it provides valuable guidance for future models of com-

plex controllers.

4.2.2 Modelling Functionality

Models of capabilities do not embody the closeness to biological structures that is prevalent in the models presented above. Capability-centred models are free to use more abstract solutions and, as such, can more easily model higher level cognitive properties. Our approach to cognitive modelling is of this more pragmatic flavour. On a higher level of abstraction, it is also possible to take inspiration from psychological as well as from physiological theories.

Minimalism and Stigmergy One area of robotics tries to minimise the cognitive abilities needed in a system by using many interacting creatures of minimal complexity. Holland and Melhuish [Holland and Melhuish, 1996] have used this approach to model large groups of robots with the minimal cognitive machinery necessary for tasks such as organisation, sorting [Holland and Melhuish, 1999] and construction [Melhuish et al., 1999].

This work relies to a great extent on stigmergy, i.e. it relies on the state of the world rather than on internal state. In other words, the behaviour displayed by these models is mainly reactive. Dennet [Dennet, 1991] argues that using the external world explicitly as an aid in cognition is common even in humans.

Neural Control of Chemotaxis and Phonotaxis Barbra Webb [Webb, 1998] presents work on modelling the cognitive machinery necessary to control two low level insect behaviours; ant chemotaxis or pheromone trail following and cricket phonotaxis or sound following. This work models closely the specific chemical and auditory sensors found on ants and crickets in order to provide similar data to the cognitive machinery and avoid sensor abstraction.

Several neuron-level controllers are demonstrated, modelling closely the neural architectures of the corresponding insects. The controllers provide the kind of behaviour found in the real animals with realistic side effects and limitations to

sensory input. Webb concludes that the experiments contribute to a clarification and greater understanding of the different forms of taxis and how their apparent similarity conceals fundamentally different requirements for control. Webb also underlines the danger of abstract sensors which would not have revealed the different control requirements.

While this work demonstrates how hardware and control models can work together to provide highly realistic models of insect behaviour, it does not address issues of learning or behaviour integration. The main reason for the clarity of this work is the choice of behaviour to model. Webb is explicit about choosing behaviours that are well studied in ethology and neuro-ethology.

This approach is not available for the study of learning as the neural circuitry involved is highly complex and poorly understood. As a result, this work uses behavioural theories on a more abstract level.

Models of Animal Learning Finally, work that is very close to ours in that it presents models of different forms of animal learning, is presented by Balkenius [Balkenius, 1995].

This work suggests that learning in BB systems may take place in four different layers [Balkenius, 1993]; reactive adaptation, reinforcement learning, expectation-based learning and planning. It also emphasises the need to avoid general learning mechanisms as biological evidence suggests that learning is highly modular and specialised. Balkenius' work goes into more detail concerning different types of associative learning in animals than ours does. It differentiates between associations containing stimulus, response, approach and place. Like in our models, chain forming and expectation forming are handled separately. More general versions of all the types of learning in Balkenius' work are covered by our models.

Balkenius presents implementable models of different forms of animal learning in the form of neural network abstractions. He also discusses how the different models relate to each other. One of the fundamental units in Balkenius' work is

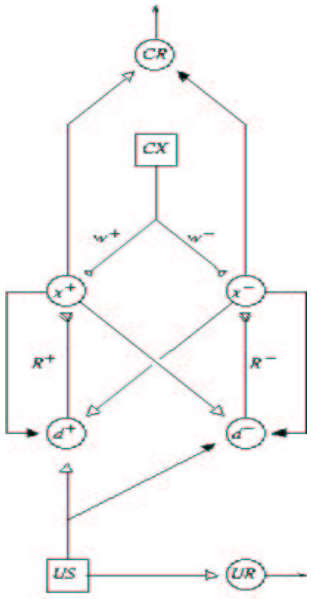


Figure 4.6: Network for Classical Conditioning, from [Balkenius, 1995]

presented in Figure 4.6. It shows the neurons necessary to associate an unconditioned stimulus with a conditioned response.

The models Balkenius presents suggest specific solutions for implementing adaptive mechanisms. He describes how similar adaptive mechanisms can be used to support different forms of learning, but does not go into the evolutionary relationship between these mechanisms.

Balkenius also presents work on more abstract computational models relating to some of the cognitive structures that support learning, such as emotions and the endocrine system [Morén and Balkenius, 2000], structures for supporting attention [Balkenius, 2000], structures for supporting context [Balkenius and Morén, 2000] and finally support structures for communication using a symbolic mode of reference [Balkenius et al., 2000]. The work on symbolic reference adds to Deacon's theory of symbolic reference presented in Section 4.1, suggesting a number of prerequisites for the different referential capabilities. This work also takes steps toward the integration of a number of these capabilities in unifying models

[Balkenius, 2000]. Balkenius *et al.* are not, however, explicit about an evolutionary approach and as a result do not cover some forms of learning that seem fundamental when the problem is studied from an evolutionary viewpoint, in particular skill learning, model learning and imitation. However, their abstract computational approach could easily be extended to take these forms of learning into account.

4.3 Guidelines for Adaptive Behaviours

Adaptation and learning are mechanisms that have allowed animals and humans to evolve sophisticated behaviours far more efficient than their reactive alternatives.

Being able to learn, however, also comes with the evolutionary costs of learning the wrong thing. Learning only gives an evolutionary advantage when it is correctly biased. The Garcia effect in rats is an example of an evolved natural bias [Garcia and Koelling, 1966].

Based on these two arguments, it is important to consider the balance between the advantages and costs of learning in a BBL solution rather than seeing learning as purely beneficial. Unrelated attributes of the environment should not necessarily be available to the same learning behaviour.

During the design and implementation of different behavioural strategies we noted common processes that were repeated for each strategy. We have generalised these processes to form a methodology for developing adaptive behaviours.

4.3.1 Hypothetical Evolutionary Paths

Our BBL methodology starts with a top-level behavioural strategy that solves a problem faced by the robot. In our motivating example, the problem is conflict resolution and the top-level behavioural strategy was the *Stylised Hierarchy Formation* strategy, where robots used stylised courtship displays rather than fighting to establish their positions in a hierarchy.

The BBL methodology works by reducing a behavioural strategy to a set of simpler, underlying strategies from which the top-level strategy could have evolved by the principle of duplication and specialisation. Each of the supporting strategies implement a simpler, but complete solution to a problem facing the robot, the underlying problems are often, but not necessarily similar to or related to the problem solved by the top-level strategy. This process is then repeated on each of the supporting strategies until an the supporting strategies have been reduced to simple

reactive behaviours that can be implemented directly.

The supporting strategies are hypothetical evolutionary stages, elements in a hypothetical evolutionary chain, connected by hypothetical evolutionary steps leading from simple strategies and simple problems to the top-level strategy and problem. Certain top-level strategies might need multiple supporting strategies, in which case the hypothetical evolutionary path will have one branch for each of the supporting strategies.

The evolutionary path we hypothesised for the *Stylised Hierarchy Formation* strategy is given in Figure 4.7.

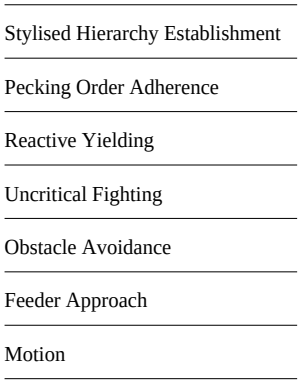


Figure 4.7: Hypothetical Evolutionary Path for Stylised Hierarchy Formation

4.3.2 Sense and Skill Delegation

The first step in identifying supporting strategies for a given top-level strategy is to identify the sense and skill requirements of the top-level strategy. The sense requirements are the features of the environment that must be recognised by the robot to allow it to follow the top-level strategy. Likewise, the skill requirements are the motory skills the robot must have.

The second step is to try to design strategies less complex than the top-level strategy that make use of the same or simpler versions of the senses or skills needed for the top-level strategy.

By delegating as many as possible of the required sense and skill capabilities to supporting layers we minimise the evolutionary step up to the top-level and simplify the incremental development.

In the case of the *Stylised Hierarchy Establishment* strategy, we needed to recognise another robot and be able to tell when it was putting on a display. We also needed the ability to behave according to a place in a hierarchy. We dedicated the robot recognition and hierarchy dependant behaviour to an underlying *Pecking Order Adherence* strategy which used a simulating fighting behaviour to establish a hierarchy and which implemented the hierarchy dependant behaviour. This left only the display recognition to be implemented by the *Stylised Hierarchy Establishment* layer.

4.3.3 Memories as Traces

In order to facilitate an evolution based development style we used a model of memories as traces or echoes of existing senses. This often allowed the **Sense** circuit to be delegated to an underlying layer, leaving only the handling of the memory to the top-level layer.

The *Stylised Hierarchy Establishment* strategy needed to analyse the behaviour of the other robot over time in order to tell whether it was displaying or not. To do this, the *Stylised Hierarchy Establishment* layer kept a copy of a **Khepera Sense** circuit which was delegated down to a *Uncritical Fighting* layer. When the observing robot stood still, this memory was compared to the current output from the **Khepera Sense** circuit and consistent discrepancies were interpreted as motion. The stylised display behaviour was to stand still, and successive identical observations of an opponent was interpreted as a display.

The basic setup for the *Stylised Hierarchy Establishment* layer is shown in Figure 4.8. The **Memory** circuit is presented using an octagon. The dashed arrow indicates that the activation of the **Yield Drive** circuit is not direct, but goes through a number of other circuits omitted here for clarity.

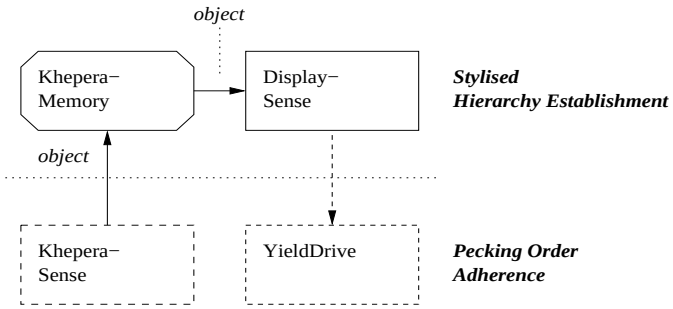


Figure 4.8: Circuits for Adaptive Behaviour

4.3.4 Duplication and Specialisation

When adding new behaviours to a controller it can be clarifying for behavioural layer design to think in terms of evolutionary stages. Firstly, we imagine that a general, relatively uniform layer of neurons evolves which can store memories of the underlying sense- and action-related neural activity. Supported by these memories, more specialised neural structures can evolve connecting the memories to existing senses and competences. Finally, the circuits involved can be adapted and dedicated circuits added to support the new behaviours.

This way of thinking forces developers to consider what underlying structures must be in place to support new behavioural layers. This ordering of circuit addition is not to be taken as a literal recommendation to implement the intermediate stages when adding a new adaptive behavioural layer, but only as an aid during design.

The motivation to do ordered addition of new circuits is that evolution is known to work at least in part, through duplication and specialisation [Allman, 1999]. When biologists talk about highly evolved structures they refer to the level to which those structured have been specialised for particular purposes compared to similar structures in other species [Strickberger, 1995]. A general indication of the specificity of an structure is its complexity, so it is common to say that the human eye is more ‘highly evolved’ than the light sensitive spots on certain species of fish. Likewise the human brain is more highly evolved than the brain of most other species.

This use of the expression ‘highly evolved’ must however, not be confused with closeness to human form. A cat’s whiskers are more highly evolved than human facial hair and the eye of the eagle is more highly evolved than the human eye. The idea of adding a general, uniform memory layer and then adding specialised structures to support new capabilities is inspired directly by these two aspects of evolution, replication and specialisation.

During development and experimentation, we found that we were often missing the underlying structures suggested by the ordered circuit addition approach. In this case it is necessary to develop the supporting structures in more basic layers that have their own evolutionary grounding in that they implement improvements on the existing solutions.

Using ordered circuit addition to develop underlying structures generally improves the robustness of suggested solutions and makes them more plausible in an evolutionary cognitive modelling context.

4.3.5 Integrating Learning Behaviours

As a general rule learnt behaviours must override reactive behaviours in order for the most sophisticated and efficient strategies to be expressed. However, survival critical behaviours must always take precedence over less immediate behaviours. Expressing a less ideal but less computationally expensive behaviour is usually preferable to doing nothing while waiting for a more sophisticated computation-heavy behaviour to be expressed.

A result of BBL is that behavioural layers form ’columns’ of increasingly sophisticated solutions to particular problems, such as conflict resolution and foraging. The integration of behaviours from different dimensions is non-trivial, and, in addition, some behaviours span multiple dimensions. Exploring e.g. serves both feeding and mating related behaviours. An action selection mechanism integrating behaviours from these different problem domains must reflect the relative importance of solving one problem compared to solving any others. A general

behaviour integration mechanism should model such animal behaviour integration mechanisms as hormones.

4.4 Neural Schema Models of Animal Learning

Our guidelines for developing adaptive behaviours only consider learning that takes place within a single problem domain. Animals are able to do more than this. They are able to associate senses and competences from behavioural layers that relate to different problems and more impressively, they are able to think laterally, i.e. to use competences developed for similar but unrelated sense patterns to solve problems‘.

In this Section we present schema based models of a selection of evolutionary significant learning capabilities. These models suggest ways in which our development methodology can be extended to produce algorithms that do these, more complex forms of learning while retaining the strengths of the algorithms currently produced: rapid adaptation, and robust solutions. Such extensions would provide solutions to the problem of merging the efficiency and robustness of BBAI with the powerful generality of GOFAI.

We have not developed models for the simplest or most complex forms of learning as they fall outside the scope of our thesis and are too poorly understood respectively. The intermediate forms of learning are divided into associative learning and discrimination learning, a division that is common in animal learning literature. For simplicity, we generalise **Sensor** and **Sense** circuits to **Sense** circuits and likewise **Competence** and **Actuator** circuits to **Competence** circuits.

4.4.1 Precursors to Learning

Certain adaptive mechanisms and and uses of memory do not have a clear place in our evolutionary inspired behaviour-based learning model. Here we discuss those mechanisms and the reason why they are not included in our framework.

Attention and Expectations Attention and hard-coded expectations are cognitive features that do not fit naturally under the umbrella term ’learning’, but which are closely related to many learning phenomena.

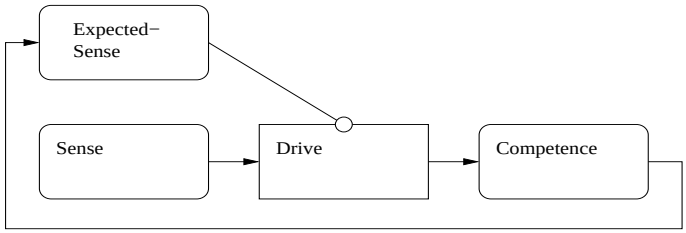


Figure 4.9: A Circuit-Based Model of Hard-Coded Expectations

Attention uses the memory of previous sensory experiences to guide the following use of the involved senses [Foner and Maes, 1994]. Hard-coded expectations are used to estimate progress or the lack thereof in order for adaptation to take place [Corbacho and Arbib, 1997].

In Figure 4.9 we present circuits describing a system where hard-coded expectations are used to inhibit the original behaviour and activate a second behaviour if the hard-coded expectations are not met. The only thing remembered in the hard-coded expectation circuit is the fact that the first drive fired.

An example of a hard coded expectation is the expectation of finding open space after having turned to avoid an obstacle. This typically fails if there are several obstacles around and a more drastic avoidance behaviour than usual is then necessary.

4.4.2 Basic Memories

The simplest forms of learning found in animals serve only to register that a cognitive event has taken place or to register specific cognitive data. These kinds of learning contain no elements of association or discrimination but are purely memory functions available for specific use by dedicated behavioural circuitry. Figure 4.10 presents a circuit-based model of this kind of learning.

A cognitive event will activate one of the circuits, something that will lead to the establishment of a corresponding memory. This memory can then be used as input to other circuits.

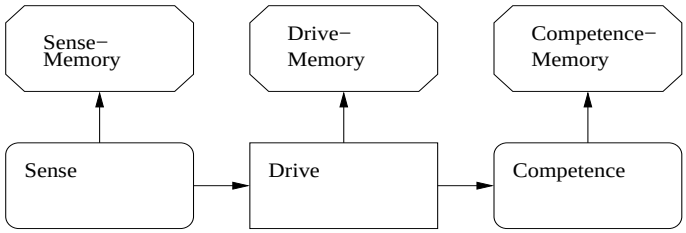


Figure 4.10: A Circuit-Based Model of Memory-Only Learning

Habituation/Excitation In habituation and excitation the memory circuit need not contain any data but just an indication whether a circuit has been recently activated or not. The memories established in habituation/excitation are typically relatively short term and so decay rapidly after having been established. The strength of the memory is also commonly accumulative up to a limit on repeated presentations of the relevant stimuli.

Imprinting During imprinting a memory of a particular feature in the environment is established and later this memory is used by dedicated behaviours to recognise those features. In chick imprinting, strategic features of the mother are remembered in order to tell her from other birds later and in male chicks to chose good mates later in life. Imprinting memory is typically a stable long term memory.

4.4.3 Discrimination Learning

The senses are the default mechanism for discriminating the state of the world into meaningful states. In addition to this fundamental discrimination, most animals have the capability to further divide and combine sensory values to form complex classifications and concepts. In this Section we describe a hierarchy of discrimination capabilities inspired by the hierarchy of associative learning used above. The hierarchy is based on the complexity of discrimination capabilities found in animals and humans.

Here we present a series of steps in which high level discrimination learning

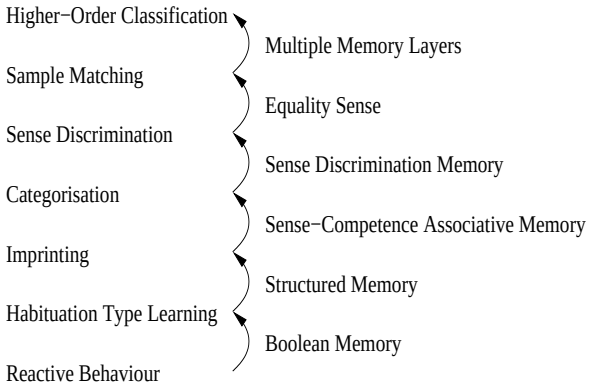


Figure 4.11: Steps in Discrimination Learning

can be developed from basic memory capabilities. The aim of a step-wise development is to preserve efficiency and robustness between each layer. These properties were lost in the direct implementation of high level forms of learning that lead to traditional ML classification methods. We have summarised our suggested steps in the development of discrimination learning and the new capabilities they introduce in Figure 4.11.

The highest forms of discrimination learning are only found in primates and humans. They need support by multiple layers of adaptive neural circuitry.

Categorisation There are currently many theories of how animals discriminate world states and whether they have or form concepts [Pearce, 1997], but from the existing evidence it is clear that most animals show the ability to solve categorisation problems.

A typical test for categorisation capabilities used for pigeons is to have them learn to discriminate two sets of slides containing leaf silhouettes. By rewarding the pigeons for pecking at slides containing oak leaf silhouettes, but not for silhouettes of other leaves, the pigeons were taught to discriminate the two sets to near perfection after 24 trials.

We suggest that categorisation on the most basic level can be modelled as as-

sociations between existing senses. A concept that corresponds to a combination of senses can be learnt by simple association as described above. In this case, the language formed by the senses correspond exactly to the language available to the learning mechanism.

Sense and Motor Discrimination and Generalisation It is often desirable to categorise within the high level abstractions presented by senses or to extend a sense to be sensitive to a wider range of input. To accomplish these changes, we suggest a model that uses discrimination memory circuits which are targeted at the underlying senses and are able to classify the abstract representations used in the sense percept.

Figure 4.12 presents the circuits used for forming two categories from a sense and associating these with different competences. In practise the associative memory circuit keeps a number of the associations that could be kept by the association circuit. The locality of a sense related to a behaviour provides a natural abstraction for a set of associations and also a bias on the kind of associations that can be made in related learning problems. The possibility to include other associations in learning problems where this is necessary is available through the association circuits.

In addition to the basic circuitry used for alpha-conditioning, Figure 4.12 adds a discrimination memory circuit customised to handle Sense1 percepts. This memory circuit can divide the perceptual data into several categories which can again be associated with different competences through the association circuit we introduced earlier, or it can add activation states that are not part of the existing sense in order to generalise the concept described by the sense.

A corresponding memory can be used for competences in order to modify the range of values they can produce or to divide a competence into sub-competences.

Matching to Sample The ability to classify relationships between stimuli is present in humans, primates and some highly developed mammals [Pearce, 1997].

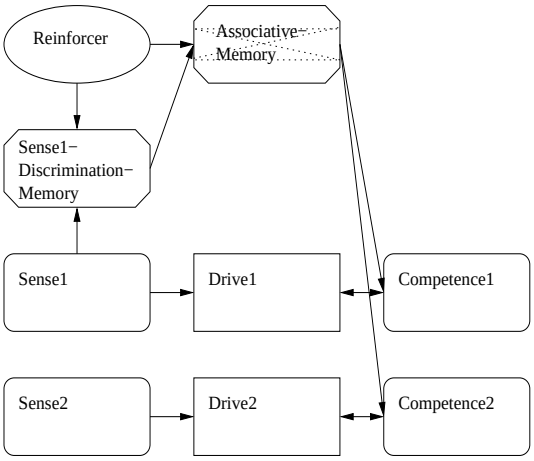


Figure 4.12: A Circuit-Based Model of Categorisation

These animals have all displayed the ability to form concepts of sameness and difference between two stimuli.

We suggest that associative learning can be modelled using a mechanism for recognising incongruence between expectations and senses in connection with reinforcement prediction. Rewarding a response only after two similar stimuli have been presented can create an expectation of a second matching stimulus. In such a situation, the incongruence sense works as a sameness sense that can be associated with a response or competence in the NC terminology.

Higher-Order Relationships Humans are able to appreciate second-order relationships between stimuli. An example test used for this capability is the presentation of two pairs of stimuli, e.g. AA and BB or AX and BY. In both examples the relationship is the same between the stimuli in each pair, i.e. same and different. In the pair AA and BX the internal relationship in the given pairs would be different.

Like second-order learning problems, problems of analogy also relate to the relationships between pair of stimuli, but the involved relationships go beyond simple sameness and difference. A typical test is for example *'A dog is to a puppy what a cow is to a ...?'*. Chimpanzees and children under six years of age are generally

incapable of solving these kind of learning problems.

We suggest that model of this kind of learning needs to build up new representations of the entities and relationships involved and associate the right ones with a response. This indicates a need for a second layer of memory circuits on top of the discrimination circuits used to classify percepts according to their attributes. A second layer of discrimination memory circuits can take multiple percepts as input and classify the relation between them. Another layer of discrimination circuits is then needed for second-order classifications.

4.4.4 Associative Learning

Associative learning takes place when 'there is a change in an animal's behaviour as a result of one event being paired with another.' [Pearce, 1997]. Internally this indicates an association of a sensory event, the conditioned stimulus (CS), with a particular behaviour, the conditioned response (CR), as a result of a reinforcing sensory event, the unconditioned stimulus (US).

The fundamental concept in our model of associative learning is the associative memory circuit. An basic associative memory circuit relates all firing inputs whenever a reinforcer fires. Later activation of any of the inputs leads to an activation of all the related circuits.

The development steps for associative learning serve the same purpose as the steps for discrimination learning presented above. We have summarised the suggested steps in the development of associative learning in Figure 4.13.

From Imprinting to Alpha-Conditioning In the form of learning called alpha-conditioning the unconditioned and conditioned stimuli are presented together. This allows immediate associations to be made directly from the sensory circuitry to the circuitry representing the conditioned behaviour.

Figure 4.14 shows how an association circuit takes three kinds of inputs, senses, competences and reinforcers. The association circuits also have the competences

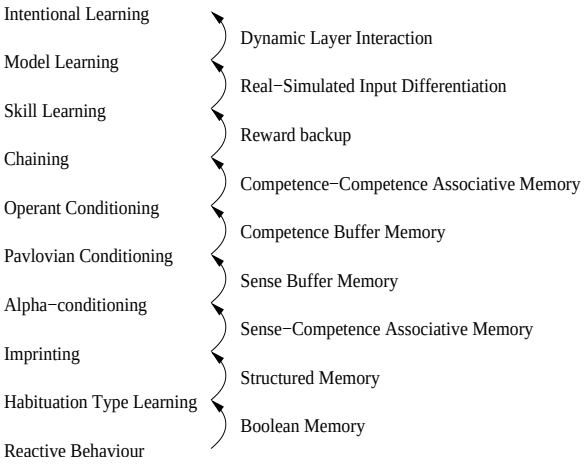


Figure 4.13: Steps in Associative Learning

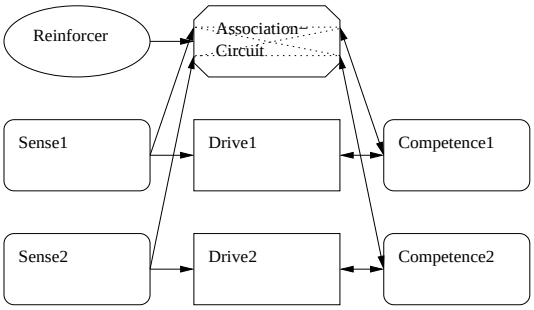


Figure 4.14: A Circuit-Based Model of Alpha-Conditioning

as outputs.

Whenever a reinforcer fires, the associative memory circuit associates the currently firing senses with the currently firing competences. The resulting association later affects the related competences, activating them if the reinforcement was positive, or inhibiting them if the reinforcement was negative. Negative associations need not inhibit underlying drives.

In the example of learning to blink (CR) by associating a certain tone (CS) with a puff of air to the eye (US), the tone and air-puff senses are connected to the association circuit as well as a blinking competence. When both stimuli are active,

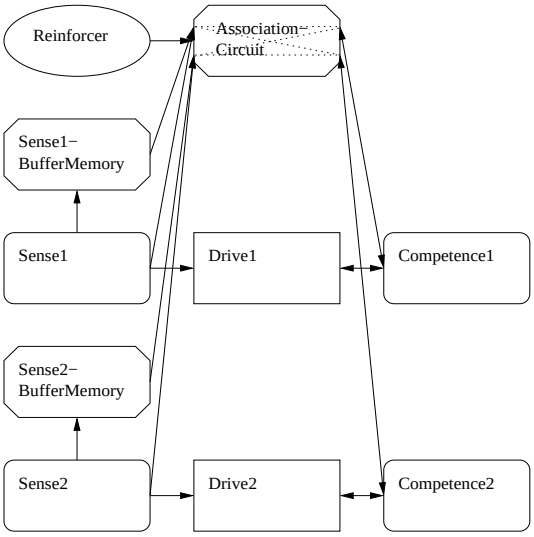


Figure 4.15: A Circuit-Based Model of Pavlovian Conditioning

the tone is associated with the blink competence and further presentations of that tone will lead to activation of the blinking behaviour.

From Alpha- to Pavlovian Conditioning In the traditional form of conditioning, Pavlovian conditioning, the CS immediately precedes the US. In this case some form of memory is needed in order to relate the CS to the UR.

The solution we present in Figure 4.15 uses a dedicated memory circuit for each sense circuit. We call this memory buffer memory because its only function is to store the output produced by the sense circuit until the next time the sense circuit fires. It does not rely on a reinforcer to establish a memory. An associative memory circuit is then used as in alpha-conditioning with the buffer memory as an added input so that associations can be made from the buffer memory as well as from the sense directly.

The buffer memory bridges the gap between alpha-conditioning and Pavlovian conditioning.

Conditioning and Auto-shaping The most basic form of *operant conditioning* is what is called *auto-shaping*. Auto-shaping is a form of conditioning where a UR is associated with a CS through repeatedly presenting it together with a US, e.g. a pigeon will start pecking at a key if the key is repeatedly presented together with food.

Our NC model of auto-shaping is alpha-conditioning from a response point of view. An unconditioned response (UR) is related to a CS by that stimulus being repeatedly presented together with an US. The neural circuitry we presented in Figure 4.14 which provided alpha-conditioning capabilities also provides auto-shaping capabilities.

From Auto-shaping to Operant Conditioning Operant conditioning, is also called instrumental conditioning, and takes place when a randomly produced behaviour, also called operant behaviour, is related to a stimulus by an US. The typical example is a rat that is taught to push a lever by being rewarded every time its frantic behaviour leads to the lever being pushed.

This type of learning requires two new elements in our model: the display of *random* or *operant* behaviour or at least a random selection of pre-programmed behaviours, and an association between the randomly produced UR and the CS, the lever, thus making the successful behaviour, the lever pushing, a conditioned response (CR).

When none of the life-critical behaviours of an animal are activated there is often a bottom level exploring, experimenting or playing behaviour that applies different competences to novel and sometimes unlikely objects in the world. Such an exploring capability could form the basis for operant behaviour in our suggested model of operant conditioning. Having failed expectations inhibit otherwise active behaviours could also produce similar effects.

To make the association between the UR and the CS, buffer memory circuits are needed for both the stimulus and the response, or in circuit terms, both the senses

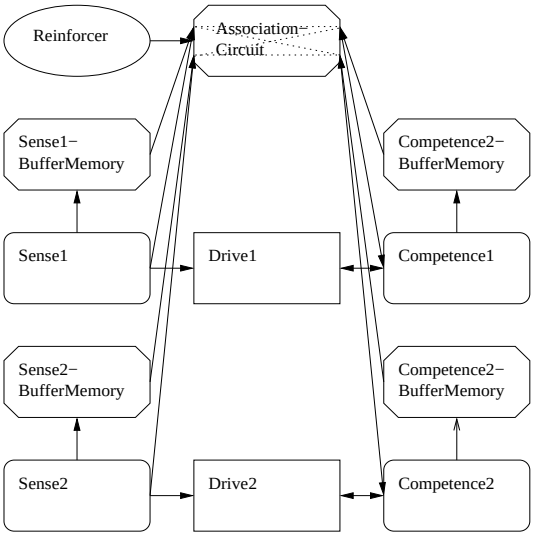


Figure 4.16: A Circuit-Based Model of Operant Conditioning

and the competences. The robot also needs to be able to connect these through an associative memory circuit to activate the appropriate behaviour when the stimulus is presented again later. We do not explore the issue of random behaviours further as it is not immediately relevant to the learning problem. The circuits that provide operant conditioning capabilities are presented in Figure 4.16.

4.4.5 Reward Prediction

Our models of basic forms of associative learning presented above are straightforward enough to be implemented directly.

Here we discuss the formation of more complex associative structures. These structures support the representation chains of action, the estimation of future reward and the modelling of the environment. By increasing the sophistication of the associative structures and their supporting circuitry, we describe a series of steps from simple chaining to internal world model revision.

We do not provide NC models or implementations of learning forms that imply reward prediction. There are several issues that must be tackled in order to develop

the general NC modelling framework to be able to describe such capabilities in a way that can substantially facilitate implementation. Below we discuss these issues and provide suggestions for extending the NC modelling framework to provide models and implementations of high level learning.

The reward prediction and model learning are also the pinnacles of animal and human learning. By extending our chain of development from advanced conditioning to model learning, we cover all the types of learning that are found in nature.

Human language and symbolic reference abilities allow for an extreme form of model learning that is qualitatively different from other model learning in that a symbolic model is less tightly connected to real world experiences. This makes it able to learn faster but also makes it more vulnerable to mis-learning. We do not discuss symbolic model learning in particular as our models are currently too basic to cover so complex a topic, but below we discuss external model integration aspects of learning. The Symbolic Species theory argues that symbolic reference capabilities and natural language capabilities must have co-evolved. The most complex form of learning related to external model integration is symbolic model revision from natural language based interaction.

Chaining Experiments on operant conditioning often create complex chains of CRs that have the previous CR as part of their CS, e.g. a pigeon may learn to push a box to a certain place, then jump onto it and then start pecking.

This kind of learning capability demands a new role from the association memory in Figure 4.16 in that it now also needs to be able to associate competences to competences.

Psychologists currently debate what kind of mechanisms produce the observed chaining [Pearce, 1997]. The stimuli for each response might be external as in the box position rather than internal as in the pushing behaviour. It might also be the case that a higher level learning mechanism produces the observed behaviour. There might also be a specialised mechanism for learning sequences of actions

rather than immediate action to action associations. Our model shows that chaining can be modelled as a small extension of operant conditioning.

The last thing that happens in the activation sequence of a behavioural layer is the firing of competences and actuators. In order to associate the activation of a competence with the activation of another competence, the associative memory needs to retain the activation energy implied by the associations from one activation pass to the next. This use of retained association activation allows the bridging of the temporal gap between activation passes and allows the learning of action sequences [Frezza-Buet et al., 2001].

Skill Learning Skill learning is different from operant conditioning in that it creates new behaviour patterns without explicit or direct reinforcement. Instead it learns partial goals or estimators that recognise progress toward an explicit reinforcer. When progress is recognised using dedicated senses, such as having a light seeking behaviour as an initial part of a heat seeking behaviour, this is not generally called skill learning. Learning to predict reinforcement is the defining feature in skill learning.

There are a number of issues related to a model of reinforcement prediction. Firstly, a mechanism for establishing expectations or predictions is necessary. Secondly, a mechanism is needed to detect incongruence between the expected and the received reinforcement and to update the expectations accordingly.

A model that provides the necessary mechanisms for skill learning must let rewards spread through the aaaociation structures to achieve the kind of backup effects found in reinforcement learning.

Model Learning Models as found in ML improve learning by allowing simulated experience. Models are however an engineering construct and not generally a part of psychological models. In psychology, models are referred to as declarative knowledge representations, as opposed to procedural representations.

An explicit association structure similar to the one we suggested for associative

learning, chaining and skill learning can also be used as a model if the right modifications are made. The structure must be able to differentiate real from simulated input in order to suppress acting on simulated inputs. Associations made from real input can have a double role as simulated input for using the association structure as a model.

This view of model learning provides a suggested implementation that would be a relatively simple extension of the underlying neural circuitry. It also presents new ideas to classic problems with separate models, such as how to integrate them with existing procedural structures.

The MEM model is based on results that show how internal reactivation of a memory strengthens it. The MEM model suggests reactivation as a process important for learning. The reactivation of an existing memory as a part of learning fits our model of model learning in that a reactivation would effect the model directly.

Intentional Learning An element central to the MEM memory model is the *agenda*. Agendas are in short high level behavioural structures that can use and manipulate underlying behavioural structures to achieve general goals. They can set sub-goals and monitor the success of the underlying behaviours.

Agendas represent two important aspects of learning. Firstly there are supervisory agendas. These are general mechanisms for regulating and monitoring concurrently active behaviours. Learning matching to sample as described above involves using an agenda to set up the criteria for similarity evaluation. Secondly there are executive agendas that in addition to monitoring use their own goal to activate underlying behaviours. This allows for a new form of learning where a high level learning agenda activates underlying behaviours in order to produce relevant information.

Agendas describe dynamic layer interaction where high level layers typically use the results from underlying layers as inputs. They use generalised behaviour interaction patterns in the same way that competences are generalised patterns for

interacting with the world. Agenda-style behaviour interaction can be programmed using the NBC model of behaviours without any fundamental changes to allow high level forms of learning such as matching to sample and putting through. On a higher level, similar interaction patterns can provide intentional learning as described by the MEM memory model.

Agenda-like monitoring and activation of behaviours provide top-down biases on learning produced by the current dynamic context of the system. This is a crucial bias for general learning methods.

4.4.6 Integrating External Behaviour Models

Some forms of learning are able to translate between observed behaviours, internal behaviour representations and the use of observations to update internal behaviour models. This kind of learning, like discrimination and associative learning, has a number of increasingly complex manifestations. Below we discuss a number of these types of learning in increasing order of complexity.

Putting Through In putting through, a teacher physically guides the motions of the pupil. From the motion memories, the pupil is able to reproduce the behaviour. An example is Pig-tailed macaques who are taught to harvest coconuts by twisting the coconut to tighten the stem before they can bite through it.

To reproduce the motion pattern, the pupil must build up a new motion pattern from raw sensor data. If the motion can be reproduced by an existing competence, the learning is called pseudo-putting through, a form of Thorndikian learning or simple conditioning.

To reproduce the taught behaviour pattern it is necessary to relate the individual stimuli to form a chain of stimuli that can later be reproduced. It is also necessary to have some monitoring circuitry that can start and stop behaviours internally in order to create chains of actions to match the chains of remembered stimuli. Above we discussed monitoring as a part of supervisory behaviours and gave suggestions

for implementing such mechanisms.

The ability to produce new motion patterns and new competences has important implications for search space of a learning capability and hence its efficiency.

Imitation Putting through can be seen as a form of self imitation. An attempt to produce behaviours that result in sensory data that approximate the data remembered from the learning experience. Reformulated in the terms of visual simulation, this amounts to producing behaviours that will result in sensory data that approximate the learning experience.

Communication Animals communicate many different aspects of the world, the state of the environment, their own internal state and in some cases, their own models of the environment.

Here we only look at the communication of internal behaviour structures. The teacher in the case of putting-through expresses a model of the desired behaviour. In imitation, a teacher with a model of the behaviour in question can demonstrate the behaviour to facilitate the learning.

Natural language provides an efficient way of communicating models, but it requires a vast amount of supporting circuitry to translate between sounds and internal behaviour representations.

Chapter 5

Programming Adaptive Behaviours with PLANCS

Contents

5.1 Implementing Behaviours

A number of behaviour representations have been used through the history of BBAI [Arkin, 1998]. For adaptation and learning it is necessary that some aspects of behaviours are made explicit and available to the relevant adaptive mechanisms. Existing behaviour representations do not provide such explicit access to particular elements of behaviours. To support the kind of behaviour modularisation needed for learning we developed the Neural Circuit (NC) model of behaviours and learning. This representation of behaviours provides the structures necessary to support learning by dividing reactive behavioural layers into five different types of elements or circuits and by adding explicit memory circuits for behaviour layers that do learning.

5.1.1 The Neural Simulation Language

The Neural Simulation Language (NSL) [Arbib et al., 2001] is an object-oriented system, developed as a general-purpose neuro-simulator. It provides an interface to a set of predefined artificial and biological neural models. NSL has two levels of abstraction, *modules* and *neural networks*. The modules implement the functionality of ASL using unidirectional ports to connect modules. The neural networks can be specified in terms of arrays of homogeneous neurons and *connection masks*, representing synaptic weights. A schematic of an NSL module is presented in Figure 5.1.

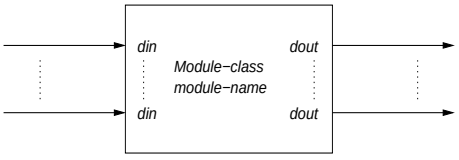


Figure 5.1: An NSL Module, from [Weitzenfeld94 and Arbib, 1994]

The content of a module need not be a neural network. NSL also allows procedural programming of the contents of a module in either C++ or Java. NSL also

provides visualisation tools for the activity in the neural networks.

5.1.2 The Behaviour Language

Brooks developed a lisp-based language, called the Behaviour Language (BL) for programming distributed subsumption style control architectures [Brooks, 1990]. BL was developed for multi-processor systems and allows a program to specify different target processor architectures for the programs. The top level unit in the BL is a *behaviour* consisting of a set of *rules*. Each behaviour is first compiled down to a set of lisp representations of Augmented Finite State Machines (AFSMs). The AFSMs again are compiled down to machine code for the specified processor architecture. The rules contain an activation condition and a body describing the computation to be done on activation.

Each AFSM can be unidirectionally connected to the ports of other AFSMs. The connections can also *inhibit* o r*suppress* other ports. A subsumption module is presented in Figure 5.2.

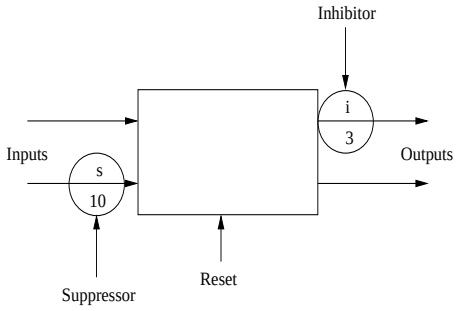


Figure 5.2: Subsumption Architecture, from [Brooks, 1986]

5.1.3 Port Arbitrated Behaviours

The Ayllu language generalises port arbitrated behaviour architectures (PABs) such as NSL and BL in a platform independent C implementation [Werger, 2000]. Ayllu also adds three extensions to the common PAB framework: connections over net-

works, flexible port structures that allows arbitrary many behaviours to communicate over the same port without overwriting messages, and a new write-inhibition connection type that inhibits outgoing messages on specified ports.

Ayllu allows arbitration between behaviours on different robots through a process called cross-subsumption. Each robot broadcasts locally estimated eligibility to a dedicated port on another robot. The robot that has a local estimate equal to the highest estimate broadcast can then express the related behaviour by inhibiting that behaviour on all the other robots. The use of inter-robot communication is illustrated in Figure 5.3.

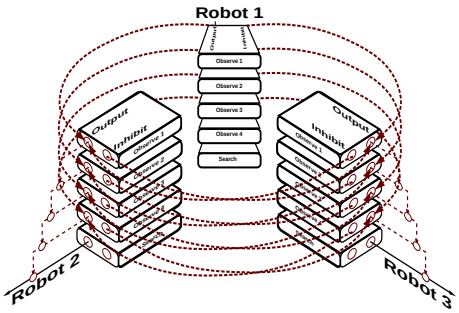


Figure 5.3: Cross Subsumption, from [Werger, 2000]

5.2 The PLANCS Library

In order to facilitate the implementation of behaviours designed using NCs, we developed a library of classes providing the basic NC functionality on single processor machines. We called the classes programmable, learning, artificial neural circuits (PLANCS) [Dahl and Giraud-Carrier, 2001b] reflecting the unification of programmability and adaptivity they embody as well as our inspiration from cognitive modelling. The classes were implemented in C++ and Java. Unlike BL, PLANCS does not support multi-processor systems.

The PLANCS library has a layered architecture presented in Figure 5.4.

The *Network* layer emulates a multi-processor, distributed processor architec-

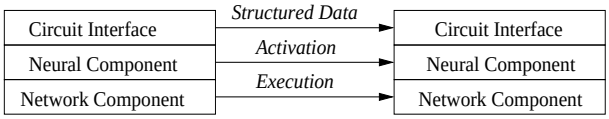


Figure 5.4: The PLANCS Classes

ture. This layer implements the **Network Component** class. Each instance of this class is conceptually an independent computational unit.

The *Neural* layer implements a neuron-based connection and activation model on top of the **Network Component** class. It implements the **Neural Component** class which has connections to a set of input components and a single connection to an output component. If the sum of the excitation levels of the input components exceed a given threshold, the **Neural Component** is activated, setting an excitation level that is propagated along the output connections.

The *Circuit* level implements more complex data communication structures on top of the inputs and outputs implemented by the *Neuron* layer. This level implements a set of **Interface** classes that allow structured data to be passed between circuits.

NCs are conceptually collections of neurons. Such collections can have a complex, though static interface to other NCs and can implement complex computations. Overloading the evaluation function of the **Neural Component** class allows a programmer to implements an arbitrary complex input-output function.

5.2.1 The Computational Model

The fundamental class of the PLANCS architecture, the **Network Component** class, provides methods for connecting and coordinating computational components in a network structure, ensuring that all the components are executed at appropriate times. This class abstracts away the underlying processor structure and facilitates porting programs written using PLANCS to distributed architectures.

The original version of the subsumption architecture was implemented in Lisp

on a network of off-board processors that communicated with a uni-processor robot over a radio link [Brooks, 1995]. One of the main motivations behind the subsumption architecture was the additivity of processing power that was achieved through concurrent behavioural layers with low bandwidth communication.

The PLANCS emulates a network of processors on single-processor architectures. In the area of Artificial Neural Networks (ANNs) [Hertz et al., 1991] this has been done either *synchronously*, by updating all elements at the same time, or *asynchronously* by using a random updating order or random updating times. Our implementation uses a static, programmer defined execution order.

5.2.2 Control- and Data-Flow

BB systems decompose controllers by activity rather than by functionality. This means that every layer must contain complete control- and data-flows from sensors to actuators.

The immediate subclass of the *Network Component* class is the *Neural Component* class. This class provides methods for handling neuron level interaction and implements an excitatory state. Every **Neural Component** object has a set of inputs and a set of outputs and continuously keeps track of the sum of the inputs and activates if this sum exceeds the activation threshold.

On activation, The default behaviour of this class models a sum-threshold neuron which fires whenever the input threshold has been exceeded. Thus, **Neural Component** objects can be used to implement neuron level models. However, neuron level models of natural algorithms are rare, and such implementations would have impractically large computational overheads. Our classes are meant primarily for circuit level cognitive modelling. The spreading of activation through the **Neural Component** objects constitutes the flow of control for the controller. Many theories about the functionality of different areas of the brain do not include neuron level detail. It is common to attribute certain processing or semantic properties to areas of the brain [Carlson, 2000, Carter, 1998]. In particular, the visual path-

ways are well studied [Bruce et al., 1997] and their functional modularity provides many opportunities for concurrent modular solutions on a circuit level. We wanted to model these theories in a BB framework and a concurrent neural circuit framework was the natural synthesis.

In addition to providing activation, input objects can also provide data. An object can be an instance of a number of *Receiver* and *Provider* interfaces, such as the *Integer Provider* interface or *Object Receiver* interface. An object that is a subclass of both the *Neural Component* class and a *Receiver* interface, can on activation, collect the data provided by the input objects. The input objects must be instances of a *Provider* interface that corresponds to the specified *Receiver* interface, i.e. a **Float Receiver** object must have **Float Provider** objects as inputs.

The divide between control- and data-flow in PLANCS abstracts away the details of neuron-level interactions, thus reducing the complexity of programming the object interactions. In the PLANCS architecture the control or activation flow is described by the *Neural Layer* while the data flow is described by the interfaces on the circuit level.

The guidelines for BB robotics state that the communication between the computational nodes should be specified down to the wire that interconnects them [Brooks, 1991a]. The *Neural Circuit* abstraction sticks to this rule in a cognitive modelling context, but in a programming context it allows structured data to be passed between nodes along unspecified communication channels. The type of the data passed between circuits is an abstraction of the set of interconnections between the neurons in the circuits and the way the receiving circuit interprets these, e.g. the **Uncritical Fighting Drive** circuit takes as input a feeder percept object containing information about a feeders height and horizontal placement in the visual field. This information could be transmitted by connecting all the binary green sensitive pixels in a vision circuit (retina) to the approach feeder sense and letting the approach feeder sense calculate the data. However, in nature a lot of intermediary processing is done in the visual cortex [Bruce et al., 1997]. We reflect this in

our implementation by having a feeder sense circuit that constructs a feeder percept object from the raw data. The passing of a percept object between the feeder sense and the approach drive reflects an underlying collection of axons from neurons in the feeder sense to neurons in the approach drive. The approach drive then interprets these connections as a feeder.

The ability to accept and provide percepts and the corresponding neural connection model are part of a circuit's class and as such they are a static property of the circuit. Making the communication structure a static property reflects the fixed nature of the underlying neural connections and this static property respects the principles of BB programming. Making the connection properties dynamic would not have allowed the cognitive interpretation of object passing as axons.

5.2.3 The Main PLANCS Classes

Some of the classes implemented by the PLANCS class library are essential for implementing a complete robot controller. We describe these briefly below. Many other classes implement variations on the essential classes and additional functionalty. Such classes are not discussed.

The WebotsController: The **WebotsController** class encapsulates the Webots API for yielding control to, and receiving control from, the simulator. In Webots this is done through calling the procedure:

- `int khepera_step(int step);`

When the call returns, the **WebotsController** activates all the **Sensor** circuits. The **Sensor** ciruits then activate the **Sense** circuits and so on. Lastly, the **Actuator** circuits activate the **WebotsController** which then yields control to the simulator again.

The PLANCS classes also contain a **PlayerController** class that provides similar functionality for the Player [Gerkey et al., 2001] robot device server developed at the University of Southern California to control ActivMedia's Pioneer robots.

The NetworkComponent: All classes implementing a NC inherit from the **NetworkComponent** class, including the **Controller** classes. The **NetworkComponent** class implements the neural activation model, i.e. the first level of the PLANCS architecture, on single processor machines.

The C++ specification of the **NeuralComponent** class is given in Figure 5.5.

```
//=========================================================
// NetworkComponent.hh
// g++ 3.0
//=========================================================

/**  Implements the Neural Activation Model on single
     processor architectures.

     @author Torbjorn Semb Dahl, University of Bristol
*/
class NetworkComponent
{
public:
  void addInput(NetworkComponent* netc);

  void notify(void);

protected:
  vector<NetworkComponent*>* outputs;

  void notified(void);

}; // Ends class NetworkComponent
```

Figure 5.5: NetworkComponent Class Specification

Calling the *addInput* method will register the **NetworkComponent** called as an output with the **NetworkComponent** given as an argument.

A **NetworkComponent** is given control by calling its *notify* method. This method calls its own *notified* method and then goes on to call the *notify* method on all the **NetworkComponent** it has registered as *outputs*.

When implementing a network of **NetworkComponents**, some care must be taken so that a sensible control structure is specified so that all components are executed regularily. In particular, loops and omissions must be avoided.

The NeuralComponent: All classes implementing a NC also inherit from the **NeuralComponent** class, including the **Controller** classes. Our implementation makes the **NeuralComponent** class a sub-class of the **NetworkComponent** class. The **NeuralComponent** class implements the neural activation functionality, i.e. the second level of the PLANCS architecture.

The C++ specification of the **NeuralComponent** class is given in Figure 5.6.

```
//=========================================================
// NeuralComponent.hh
// g++ 3.0
//=========================================================

/**  Fires with a strength equal to its excitation when
     the total excitation is above the excitation threshold
     and the total inhibition is below the inhibition
     threshold.

     @author Torbjorn Semb Dahl, University of Bristol
*/
class NetworkComponent
{
public:
  // Default threshold values
  static const int ET_DEF = 20;
  static const int IT_DEF = 20;

  NeuralComponent(int et=ET_DEF,int it=IT_DEF);

  void addExcitor(NeuralComponent* neuc);
  void addInhibitor(NeuralComponent* neuc);

  void notified(void);
  void fired(void);

protected:
  int et; // Excitation threshold;
  int it; // Inhibition threshold;

  int exctitation;
  int inhibition;
  int fstr;

  vector<NeuralComponent*>* excitors;
  vector<NeuralComponent*>* inhibitors;

}; // Ends class NeuralComponent
```

Figure 5.6: NeuralComponent Class Specification

The constructor allows the exctitation threshold and the inhibition threshold to be set to other values than the class defaults.

Calling the *addExcitor* method adds the **NeuralComponent** specified by the argument to the set of excitors kept by the **NeuralComponent** that is being called and allows it to contribute to its activation.

Calling the *addInhibitor* method adds the **NeuralComponent** specified by the argument to the set of inhibitors and allows it to contribute to its inhibition.

When the *notify* method is called from the underlyinmg **NetworkComponent** object, the **NeuralComponent** calculates the total excitation and inhibition. If the total excitation is greater than the excitation threshold and the total inhibition is less that the inhibition threshold, the *fired* method is called. The default behaviour of the *fired* method is to set the firing strength of the component equal to the total excitation. New input-output functions can be defined by overloading the *fired* method.

The IntegerProviderInterface: **ProviderInterface** classes in general implement a protected variable $x\mathsf{p}$ for the provided data type x. They also implement the public method *get$X$Provided* which allows other circuits access to the provided data $x\mathsf{p}$ for data type x.

The C++ specification of the **IntegerProviderInterface** class is given in Figure 5.7.

The IntegerReceiverInterface: **ReceiverInterface** classes keep a set of **ProviderInterfaces** of the same type. New **ProviderInterfaces** are added using the *addX-Provider* methods.

The C++ specification of the **IntegerReceiverInterface** class is given in Figure 5.8.

Together a **Provider** class and a **Receiver** class relating to the same data structure, are able to pass instances of that data structure from the **Provider** object to the **Receiver** object.

```
//============================================================
// IntegerProviderInterface.hh
// g++ 3.0
//============================================================

/**  Allows an object to pass an integer to an
     IntegerReceiverInterface object.

     @author Torbjorn Semb Dahl, University of Bristol
*/
class NetworkComponent
{
public:
  static const int INTP_DEF = 20;

  NeuralComponent(int intp=INTP_DEF);

  int getIntegerProvided(void);

protected:
  int intp;

}; // Ends class IntegerProviderInterface
```

Figure 5.7: NeuralComponent Class Specification

```
//============================================================
// IntegerReceiverInterface.hh
// g++ 3.0
//============================================================

/**  Allows an object to receive integers from any number of
     other IntegerReceiverInterfaces.

     @author Torbjorn Semb Dahl, University of Bristol
*/
class IntegerReceiverInterfaces
{
public:
  void addIntegerProvider(IntegerProviderInterface* ip);

protected:
  vector<IntegerProviderInterface*>* ipis;

}; // Ends class IntegerReceiverInterfaces
```

Figure 5.8: NeuralComponent Class Specification

5.3 Implementing a Behavioural Layer

Each Behavioural Layer is implemented incrementally as a sub-class of the class for the previous layer. The first layer is a sub-class of the **WebotsController** class. The **WebotsController** class encapsulates the Khepera controller API and is again a sub-class of the **NeuralController** and **NetworkController** classes. These classes implement the control-flow features necessary to include instances of the **Controller** classes in networks of **NetworkComponent** objects.

5.3.1 The Uncritical Fighting Layer

As an example of an implementation of a behavioural layer, we present the C++ specification and implementation of the **UncriticalFighting** layer in Figures 5.9 and 5.10 respectively. The *Uncritical Fighting* strategy is one of the strategies employed by The NC design of the *Uncritical Fighting* layer was discussed in detail in Chapter 3 and was presented graphically in Figure 3.2. The controller implementations have two main methods: the *constructor* deals with the instantiation of circuit objects needed in that and previous layers. The second method, *init*, connects the circuit objects after they have been instantiated.

The *addInput* method includes the circuit in PLANCS's computation schedule, ensuring that the circuit's *fired* method will be called when appropriate. This connects the two circuits on the *Network* level as presented in Figure 5.4. The *addExcitor* method adds a **NeuralComponent** to the set of inputs that may excite the circuit on the *Neural* level. This sets up the *Neural* level control flow. Finally the *addImageProvider*, *addProjectionProvider* and *addObjectPerceptProvider* methods connect circuits on a *Circuit* level by providing **Receiver** interfaces with appropriate **Provider** interfaces. This establishes the system's data-flow.

```
//============================================================
// UnriticalFightingController.hh
// g++ 3.0
//============================================================

#include<ColourSense.hh>
#include<ObjectSense.hh>
#include<NeuralComponent.hh>
#include"MapController.hh"

/**  Attacks other robots on sight.
     @author Torbjorn Semb Dahl, University of Bristol
*/
class UncriticalFightingController:public MapController
{
public:
  static const uint8 KHEPERA_GREEN = 40;
  static const uint8 KHEPERA_RED   = 200;
  static const uint8 KHEPERA_BLUE  = 40;

  UncriticalFightingController(void); // Constructor

  void init(int argc,char* argv[]);

protected:
  ColorSense* khepcolsense;
  ObjectSense* khepsense;
  NeuralComponent* ufightdrive;

}; // Ends class UncriticalFightingController
```

Figure 5.9: Uncritical Fighting Layer, Class Specification

```
//=============================================================
// UnriticalFightingController.cc
// g++ 3.0
//=============================================================

UnriticalFightingController(void)
{
  khepcolsense=
    new ColorSense(KHEPERA_GREEN,KHEPERA_RED,KHEPERA_BLUE);
  khepsense=new ObjectSense();
  ufightdrive=new NeuralComponent();
}; // Ends UnriticalFightingController

void UncriticalFightingController::
       init(int argc,char* argv[])
{
  // k6300sensor inherited from ApproachFullFeederContr.
  khepcolsense->addInput(k6300sensor);
  khepcolsense->addExcitor(k6300sensor);
  khepcolsense->addImageProvider(k6300sensor);
  khepsense->addInput(khepcolsense);
  khepsense->addExcitor(khepcolsense);
  khepsense->addProjectionProvider(khepcolsense);
  ufightdrive->addInput(khepsense);
  ufightdrive->addExcitor(khepsense);
  ufightdrive->addObjectPerceptProvider(khepsense);
  // approbjcomp inherited from ExploreController
  approbjcomp->addInput(ufightdrive);
  approbjcomp->addExcitor(ufightdrive);
  approbjcomp->addObjectPerceptProvider(ufightdrive);
}; // End init
```

Figure 5.10: Uncritical Fighting Layer, Class Implementation

5.4 Implementing a Neural Circuit

Neural circuits are in general developed by deriving new classes from either the network component class, the neural component class or the memory component class and overriding the default method for creating output from the inputs.

When developing control algorithms for specialised circuits, we kept the analogy to the massively parallel algorithms of the brain as strong as possible. Thinking about the input, output and control algorithm as an abstraction of a collection of neurons helps produce a natural division of tasks between circuits as well as practical objects for inter-circuit communication.

The implementation of a new NC is a sub-class of the **NeuralComponent** class from which it inherits its control flow related features. According to the inputs and outputs specified by the NC design, the new class must inherit from the relevant **Interface** classes. The input-output function is specified by overloading the *firing* method from the **NeuralComponent** class.

5.4.1 The Approach Object Competence Circuit

As an example we present the implementation of the **Approach Object Competence** class. This class is part of the *Uncritical Fighting*. The NC level design for that layer was presented in Figure 3.2. The C++ specification and implementation of this class are given in Figure 5.11 and 5.12 respectively.

The *excitors* vector is implemented by the **NeuralComponent** class, the *oops* vector is implemented by the **ObjectPerceptReceiverInterface** class and the *provided_integer* variable by the **IntegerProviderInterface** class. The **ObjectPercept** class defines many features of a perceived object, including the *horizontal_centre* variable.

The PLANCS classes implement a number of default algorithms for producing output from the inputs for different interfaces, like the **Adder** class, which receives integers, adds them up and provides the integer sum as an output. Examples of

```
//============================================================
// ApproachObjectCompetence.hh
// g++ 3.0
//============================================================

#include<NeuralComponent.hh>
#include<ObjectPerceptReceiverInterface.hh>
#include<IntegerProviderInterface.hh>

/**  Provides the horizontal centre of the first excited
     object percept provider.
     @author Torbjorn Semb Dahl, University of Bristol
*/
class ApproachObjectCompetence:
  public NeuralComponent,
  public ObjectPerceptReceiverInterface,
  public IntegerProviderInterface
{

public:
  ApproachObjectCompetence(void); // Constructor

protected:
  void fired(void); // Overloading NeuralComponent

}; // Ends class ApproachObjectCompetence
```

Figure 5.11: Approach Object Competence Class Specification

```
//============================================================
// ApproachObjectCompetence.cc
// g++ 3.0
//============================================================

void ApproachObjectCompetence::firing(void)
{
  vector<NeuralComponent*>::iterator ei;
  vector<ObjectPerceptProvider*>::iterator oi=opps->begin();
  for(ei=excitors->begin();ei!=excitors->end();ei++)
  {
    if((*ei)->isFiring())
    {
      ObjectPercept* recobj;
      recobj=(*oi)->getObjectPercept();
      provint=recobj->horizcent;
      break;
    }
    oi++;
  }
}; // End void firing(void)
```

Figure 5.12: Approach Object Competence Class Implementation

other default classes are the: **Inverter**, **Equality** and **RangeTest** classes.

5.5 Testing and Debugging PLANCS

The NC model of behaviours and learning and the supporting PLANCS classes facilitate incremental development of behavioural layers as suggested in the original work on the subsumption architecture. When using the PLANCS classes, adding cross layer interaction from the topmost layer involved allows the underlying layers to compile and execute independently.

The PLANCS classes also allow individual behavioural layers to be incrementally developed by having circuits as independent objects. This means that the initial circuits of a behavioural layer can be tested and debugged before the following dependant circuits are added. This greatly facilitates development.

Chapter 6

Implementations and Experiments

Contents

In order to demonstrate that our BBL approach can produce rapidly adapting solutions to different problems we have carried out three sets of experiments. All the experiments use a reactive base-controller developed for submission to the Second ALife Creators Contest. This base-controller is described in Section 6.2.

The first experiment shows how habituation learning can be used for approach compensation. The second experiment demonstrates four increasingly sophisticated solutions to the problem of foraging, ranging from Random Wandering to Mapping.

The third and final experiment revisits our recurring example and implements the five strategies for conflict resolution we used as an example for describing BBL in Chapter 4.

All our experiments were done using Cyberbotics' Webots Simulator. The simulated Khepera robots all had an array of Infra-Red proximity detectors and a K6300 colour camera. The real Khepera robots are small differential drive robots developed by the K-Team company. They have been extensively used in Robotics research [Jakobi et al., 1995, Nolfi et al., 1994].

6.1 Simulators versus Real Robots

The only way to show that a robot controller produces the desired behaviour is to use it to control a robot in the real world. Using simulators, however, has a number of advantages, in particular concerning cost and speed of development. An area of research that has been dependent on some of these advantages is ALife. The need for fast and automated evaluation of robot behaviours has forced ALife researchers to look at how to create high level robot simulators and how to avoid their inherent weaknesses [Jakobi et al., 1995].

In general, the danger of using a simulator rather than a real robot is that we might study problems that do not exist in the real world and ignore problems that do [Mahadevan and Connell, 1991]. The measure of a good simulator is that the behaviour produced by a controller in the simulator corresponds closely to the behaviour produced when the controller is down-loaded onto a real robot.

The Webots Khepera simulator used in our work is based on the major recommendation for building high quality simulators presented below:

- It is built on empirical data obtained from experiments.
- It is based on a spatially continuous model of real world physics.
- It takes account of noise.

Another recommendation which concerns the robot controllers rather than the simulators is the usage of noise-tolerant methods for controller implementation such as ANNs. The conclusion from experiments using the Khepera simulator is that it is good enough to expect equivalent behaviours in real robots for simple robot environment interactions [Jakobi et al., 1995].

On a cautionary note, it is important to emphasise that it is relatively easy to build a good simulator for Khepera-like robots. Simulators for more complex robots in a three-dimensional space are currently not up to the standards of two-dimensional simulators. The increased computational complexity that comes

with modelling more complex robots in three-dimensional space makes it uncertain whether it will ever be possible to implement high quality simulators of this kind [Husbands and Harvey, 1992].

As long as the restrictions on simulators are kept in mind, it is possible to gain the advantages they present in simple learning scenarios without degrading the quality of the work done.

Advantages to Using Simple Robots In addition to cost and speed related issues of using a robot simulator, the simplicity of controlling a simple robot with a clear programming interface has facilitated our work by hiding much of the complexity that would have been needed to control a real robot.

By allowing us to focus on the abstract control structures, the simulator allowed us to develop our memory circuit model of learning and our PLANCS framework for controller implementation. The flip-side of this clarity is the fact that it remains to see whether models as abstract as ours will be useful in implementing controllers for more complex real robots.

6.2 The ALife Creators Contest

The purpose in developing a reactive controller for the ALife Creators Contest [Christensen, 2000] was twofold. We wanted to develop a set of reactive base behaviours that we could use as a platform for experiments on adaptive behaviours. We also wanted to test and develop the ideas in the NC behaviour model and the PLANCS class library.

The base behaviours implemented emulated only non-adaptive neural structures or *motor programs*, they did not make use of any memory circuits. We later used the reactive behaviours presented here as a foundation for implementing adaptive capabilities, mirroring the way that rigid motor programs are fundamental to the evolution of adaptive animal behaviours [Gould and Gould, 1999].

6.2.1 The Contest Format

The ALife Creators Contest is arranged annually by the Webots company both online and as part of the *European Conference on Artificial Life (ECAL)*. The contest uses the Webots company's commercial Khepera robot simulator which is designed to be as close a simulation as possible of real Khepera robots.

In the ALife Creators Contest, two simulated Khepera robots with an array of eight Infra Red proximity sensors and a K6300 colour camera, compete for decreasing amounts of food in a complex simulated environment. Food takes the form of energy dispensers called feeders. The feeders provide the robots with energy on physical contact, but take an increasingly long time to refill. A full feeder is coloured green, while an empty feeder is red. Each of the robots have a continuously decreasing energy level. When the energy level reaches zero, the robot is removed. The longest remaining robot is the winner.

An example of a simulated environment that the two robots had to negotiate is presented in Figure 6.1, where two feeders can be found, one on the very left and one in the top right corner of the simulator display. The competition used several

different environments for the individual *matches*.

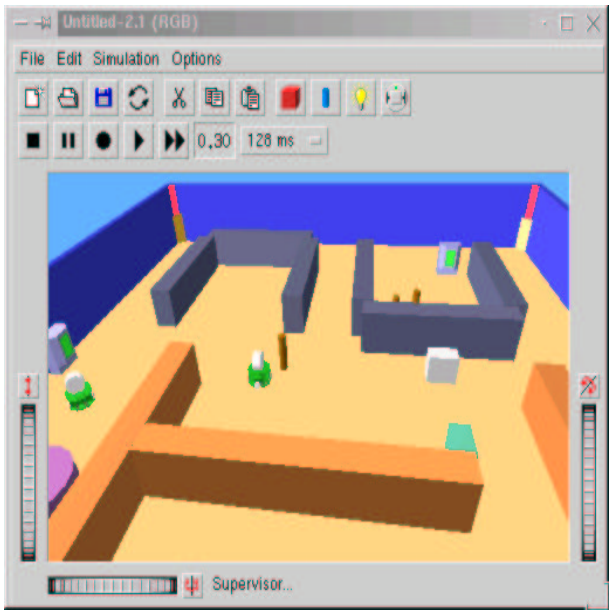


Figure 6.1: The Webots ALife Creators Contest Environment

In order to rank the participants, they were placed in an arbitrary order and adjacent robots then competed for the higher ranking over a number of rounds until no more changes in ranking took place.

Our reactive controller ended as number four out of eight with a late burst up the ranking due to the upgrading of the Java virtual machine used to run the matches, from a version that contained a crippling thread-related bug.

6.2.2 A Reactive Controller

We designed, implemented, and submitted to the ALife Creators Contest, a basic, reactive robot controller that moved around a world and approached visible feeders. The submitted controller was in large parts a simplified re-design and re-implementation and of the winner of the 1999 contest, the Toxic Oreo, by Keith Wiley from the Institute of Genomic Research in Maryland. The Toxic Oreo was far superior to our controller and contained several adaptive behaviours which we

did not copy. As a token of our respect and in gratitude to Keith Wiley's publication of the Toxic Oreo source code, the biscuit reference, which originated as a comment on the Khepera robot's general appearance, was kept but Anglicized in our controller, the Jammy Dodger.

Our controller consisted of five simple behavioural layers. These are presented in Figure 6.2. Each of the layers are discussed in detail below.

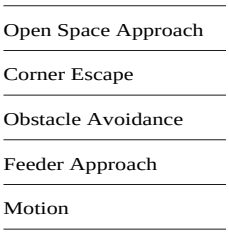


Figure 6.2: The Behavioural Layers of the ALife Contest Controller

6.2.3 Motion

This layer drives the robot constantly forward. This is done through the presence of a **Motion Drive** circuit that continuously provides the two **Wheel Actuator** circuits with a constant integer. The two **Wheel Actuator** circuits use the sum of their provided integers to set the speed of the respective wheels. The **Motion Drive** circuit and the **Wheel Actuator** circuits are presented in Figure 6.3.

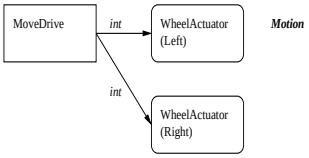


Figure 6.3: Circuits for Motion

6.2.4 Full Feeder Approach

This layer added the ability to approach full feeders. The circuits of the feeder approach layer are presented in Figure 6.4.

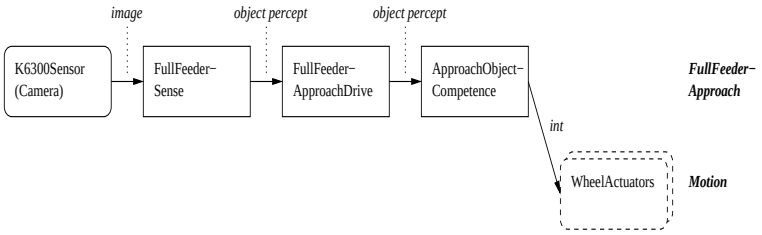


Figure 6.4: Circuits for Feeder Approach

First we added a **K6300 Sensor** circuit that encapsulated the Khepera API with the simulated K6300 Colour Camera. The **Full Feeder Sense Circuit** used the image provided by the **K6300 Sensor** circuit to detect the presence of a full feeder based on their unique range of greens. The **Approach Object Competence** circuit used the relative position data in the provided **Object Percept** to servo the robot toward that object.

6.2.5 Obstacle Avoidance

This layer added the ability to steer away from obstacles and hence keep moving until the robot ran out of energy. The circuits that make up the avoid obstacle layer are presented in Figure 6.5.

To avoid crashing into objects in the world we added two **Proximity Sensor** circuits, one for the left and one for the right side of the robot. Each **Proximity Sensor** circuit encapsulated three simulated IR and provided a single integer representing the proximity of objects on the respective side. The integers from the **Proximity Sensor** circuits were used by the two lateral *Avoid Proximity Drive* circuits that again provided negative integers to the opposite **Wheel Actuator** circuits in order to steer away from any obstacles.

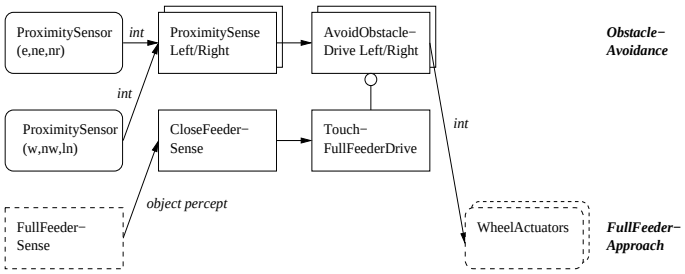


Figure 6.5: Circuits for Obstacle Avoidance

The introduction of the avoidance behaviour introduced a problem in that the robot also avoided the feeders that it needed to touch in order to feed. To overcome this problem we introduced the **Close Feeder Sense** circuit which takes an **Object Percept** and fires if the height of the feeder is above than a set threshold, indicating that if it is a feeder, it is close to the robot. The **Close Feeder Sense** again activated a **Touch Full Feeder Drive** circuit which inhibited the **Avoid Obstacle Drive** circuit. Inhibiting the feeder whenever a feeder was visible lead to the robot crashing into obstacles while approaching the feeder. By only inhibiting the **Avoid Obstacle Drive** circuit when the feeder was close, we avoided this.

6.2.6 Corner Escape

The lateral obstacle avoidance drives regularly got into a competitive state whenever the robot got into a corner. This lead to a floundering motion which denied any of the **Obstacle Avoidance Drive** circuits the consistent initiative necessary to take the robot out of the corner. The circuits that form the escape corner layer are presented in Figure 6.6.

To avoid this state of flux, we added an **Escape Corner Drive** circuit that looked at the data from both the **Proximity Sensor** circuits and inhibited the right **Avoidance Obstacle** circuit if both the proximity values were high, thus making the robot turn to the right.

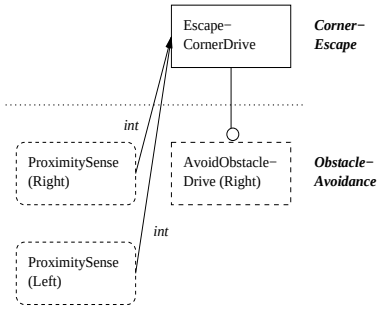


Figure 6.6: Circuits for Corner Escape

6.2.7 Open Space Approach

Lastly, our reactive controller was given a behaviour that made it approach the greatest visible area of open space. This had proved a good general exploring behaviour with the Toxic Oreo controller and was chosen above other common reactive exploring strategies such as wall-following. The circuits of the approach open space layer are presented in Figure 6.7.

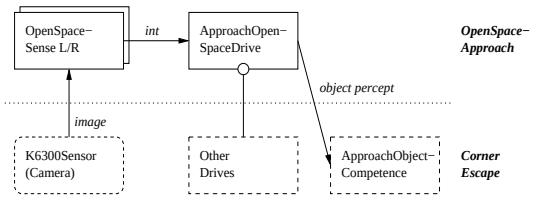


Figure 6.7: Circuits for Open Space Approach

We implemented the **Open Space Sense** circuits which scanned the left and right side of the image provided by the **K6300 Sensor** circuit for floor colour and provided an integer describing how many pixels were of this colour. This integer was used by the **Approach Open Space Drive** circuit to construct an **Object Percept** reflecting the relative position of the largest open area. This **Object Percept** was then sent to the underlying **Approach Object Competence**. The **Approach Open Space Drive** circuit was only active when no other items of interest were

present and hence was inhibited by the other drives.

6.2.8 Results

Our final ranking in the ALife Creators Contest was fourth place out of eight contestants. This ranking shows that our controller, though completely reactive, compares favourable with other designs. We emphasise that the value of the development of Jammy Dodger was not mainly in the achieved ranking, but in the evaluation and development of the NC behaviour model and the PLANCS class library. Even though no adaptation or learning was present in Jammy Dodger, the NC/PLANCS framework showed itself to be a simple and intuitive way of expressing behaviours.

6.3 Excitation for Approach Compensation

Habituation type learning takes place when repeated applications of a stimulus leads to temporarily decreased responsiveness. For example, the escape response of the guppy to a shadow passing overhead decreases when such stimulus is repeatedly presented. The opposite, sensitisation, occurs when responsiveness is temporarily increased as a result of the presentation of a stimulus, for example, a common octopus is increasingly likely to emerge from its home to attack a neutral stimulus after it has been fed.

We use the term 'habituation type learning' to describe all forms of learning that depend on a simple but quickly degrading memory of an event. This kind of learning is here demonstrated in a robot that remembers if it has just changed course in order to avoid an obstacle that stands between it and a source of food. Using the habituation learning terminology, a compensation behaviour is sensitised by avoiding an obstacle in the context of food approach.

6.3.1 Experimental Setup

Figure 6.8 shows the simulated environment used for our experiments with approach compensation. It contains a feeder and a simulated Khepera Robot identical to the ones used in the ALife Creators Contest. It also contains two basic obstacles.

As in the contest scenario, the robots are given an initial energy level which decreases over time. If a robot encounters a feeder, its energy level is increased.

The purpose of this particular setup is to force the robot to abort a feeder approach behaviour by forcing it to avoid the strategically placed obstacle. The initial energy level was set to 1.0, and the energy decrease rate was set to 0.0025 units per 64 milliseconds. In the given environment these values implied that, unless the robot could compensate for the obstacle avoidance, it would run out of energy before it could find the feeder a second time.

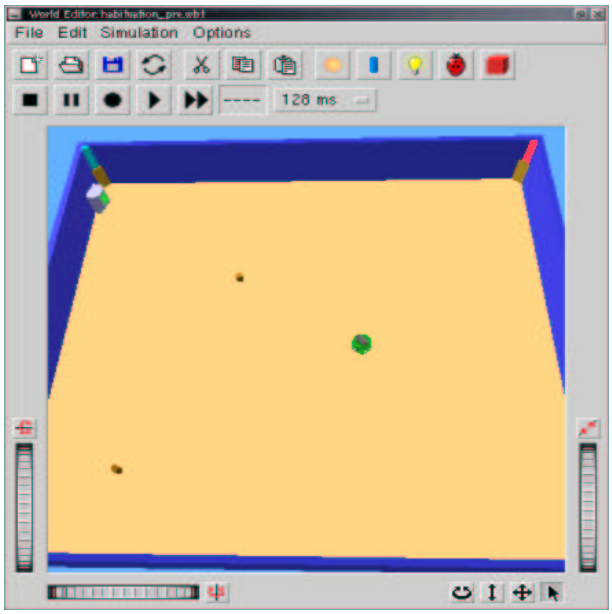


Figure 6.8: Webots Environment for Approach Compensation

Controllers To show evaluate the performance of Approach Compensation we carried out experiments with two controllers.

The first controller was a simplified version of the reactive controller we developed for the ALife Creators Contest. The simplified base controller did Random Wandering, Feeder Approach and Obstacle Avoidance.

The second controller added the *Approach Compensation* layer to the reactive base. This resulted in a controller containing the behavioural layers shown in Figure 6.9.

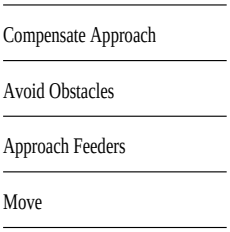


Figure 6.9: The Layers of the Approach Compensation Controller

6.3.2 The Base Controller

For the initial experiment we had to transfer the reactive controller used for the ALife Creators Contest from Java to C++. The contest framework was based on Java while the general Webots Khepera Simulator API is in C++. For simplicity we did not include the *Approach Open Space* and *Escape Corner* layers.

Base Controller Performance We ran twenty trials using the reactive base controller. In twenty out of twenty trials, the robot controlled by the reactive base controller would turn to avoid the obstacle. This would make it loose sight of the feeder, forget about it, and continue with the default wandering behaviour implemented by the other layers.

Figure 6.10 is a schematic representation of the simulated environment shown in Figure 6.8, showing the typical paths taken by the base controller during a trial. As a result of imperfect actuators, the obstacle might come up on the left of the right side of the robot forcing it to turn right or left respectively. The random controller turned left eighteen times and right twice. The base controller turned left nineteen times and right once. The dashed lines in Figure 6.15 indicates the alternative path when the robot turned right.

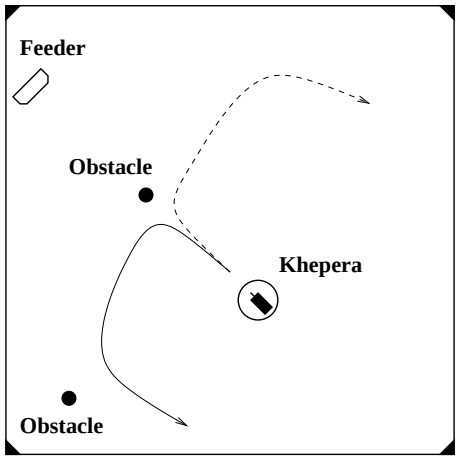


Figure 6.10: Obstacle Avoidance Path

Typical position and orientation plots for the uncompensated approach are presented in Figures 6.11, 6.12 and 6.13, with the solid graphs representing a left turn and the dashed graph representing a right turn. The positions is given in meters from the centre line with the x values describing the horizontal dimension in Figure 6.15 and the z value the vertical dimension. Orientation is given in radians with 0 being 'down' or 'south' in Figure 6.15.

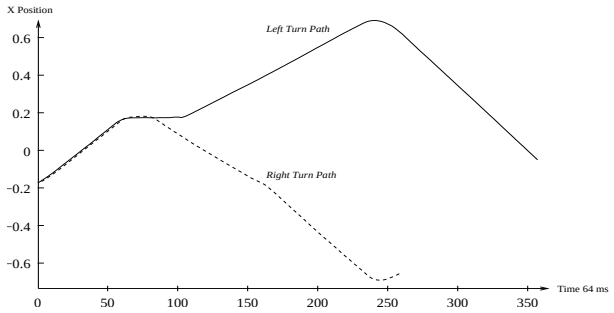


Figure 6.11: X-Position over Time for Uncompensated Approach

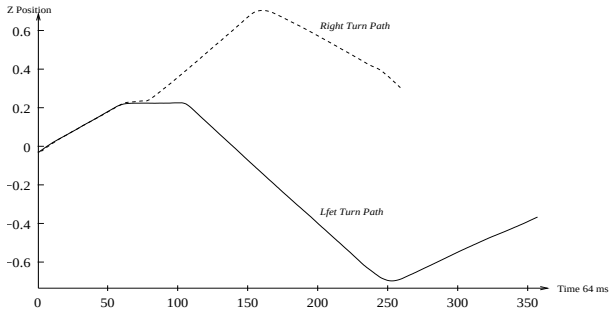


Figure 6.12: Z-Position over Time for Uncompensated Approach

6.3.3 Approach Compensation

Figure 6.14 shows the circuits added to the reactive controller in order to produce a sensitisation behaviour.

Firstly we added the **Interrupted Approach Memory** circuit to remember when the environmental context for approach compensation had been observed.

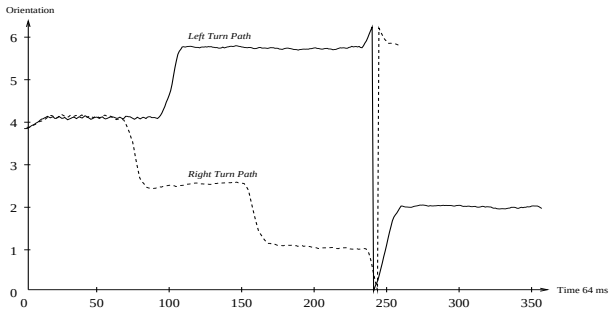


Figure 6.13: Orientation over Time for Uncompensated Approach

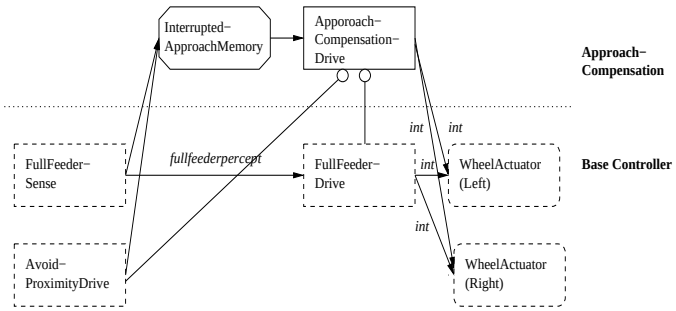


Figure 6.14: Circuits for Approach Compensation

This memory is established when both the **Feeder Sense** circuit introduced in Figure 6.4 and the **Avoid Proximity Drive** circuit described in Figure 6.14 are firing simultaneously. This indicated that the robot had been approaching a feeder, but was currently forced to turn away from it to avoid an obstacle. This memory decayed rapidly over a few seconds.

Secondly, we added the **Approach Compensation Drive** circuit which was excited by the **Interrupted Approach Memory** circuit, but inhibited by the **Avoid Proximity Drive** circuit. This circuit would turn the robot in the direction opposite to that which the **Avoid Proximity Drive** circuit had turned it.

A memory of an approach interrupted would hence exist for a short time after the event. After the obstacle avoidance was done, if that memory had still not decayed completely, it would excite the compensation drive which would turn the robot back toward the feeder. When the feeder came within sight again, it would again be picked up by the **Feeder Sense** circuit which would excite the **Feeder Approach Drive** circuit and as a result the robot would head toward the feeder.

Approach Compensation Performance We ran twenty experiments using the Approach Compensation controller. In nineteen out of twenty cases the robot reached the feeder. On the failing trial, the obstacle got positioned between the two front IR-sensors. As a result, the obstacle avoidance behaviour got stuck, floundering from left to right according to the readings of these two sensors. This floundering was usually overcome by the random variation of the robot's movements produced by the obstacle avoidance behaviour, but on this trial the energy ran out before the robot managed to free itself.

Figure 6.15 is a schematic representation of the simulated environment shown in Figure 6.8, showing the typical paths taken by the Approach Compensation during a trial. The approach compensation controller turned left eighteen times, got stuck once and turned right once. The dashed line in Figure 6.15 indicates the alternative path when the robot turned right.

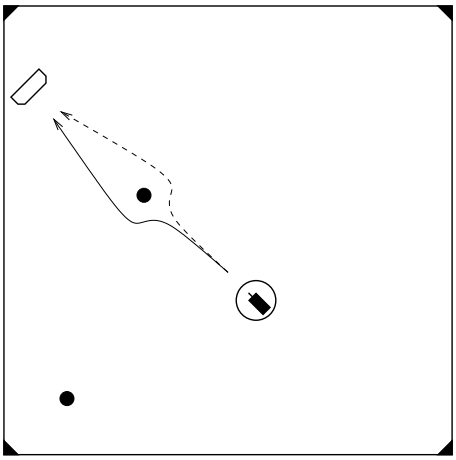


Figure 6.15: Compensation Dependant Feeder Approach Paths

The position and orientation plots for the compensated approaches are presented in Figures 6.16, 6.16 and 6.18.

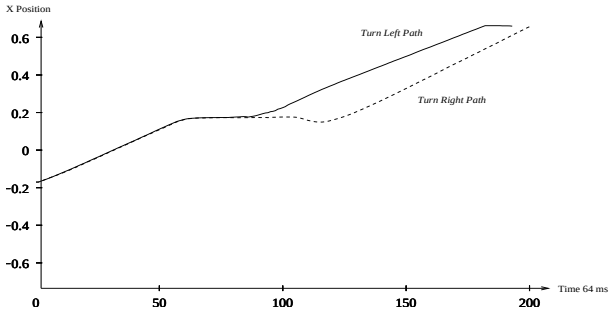


Figure 6.16: X-Position Over Time for Compensated Approach

These results demonstrate that the approach competence behaviour increases the performance of the robot on the given task. It also shows a very high degree of robustness, the failed trial being due to the obstacle avoidance behaviour and not the approach compensation behaviour.

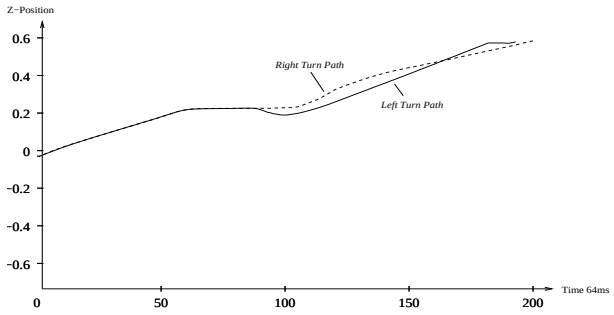


Figure 6.17: Z-Position over Time for Compensated Approach

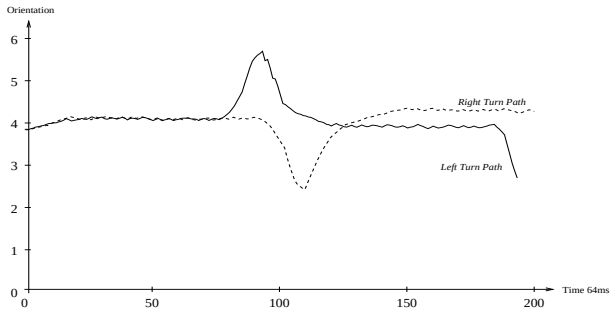


Figure 6.18: Orientation over Time for Compensated Approach

6.4 From Reactive Foraging to Mapping

Foraging, or locating and gathering resources is a fundamental problem that all animals are faced with and which has attracted a lot of attention from the robotics community [Balch, 1999, Goldberg and Matarić, 2001].

When the solution to a foraging problem includes remembering spatial structures and the positions of the resources that are being gathered, the problem is more accurately described as mapping. The discovery of *place cells* in rats, gave birth to a novel solution to the problem of spatial learning or mapping in robots based on associations between places and food [Fuhs et al., 1998, Mataric, 1990].

The different strategies for spatial learning in animals have been shown to follow a path of increasing sophistication from dead reckoning through navigation based on landmarks to navigation based on cognitive maps [Gallistel, 1990, Gould and Gould, 1999]. Here we present an experiment with four increasingly sophisticated foraging strategies, starting with a Random Walk and culminating in a strategy that does associative mapping similar to the place cell inspired work mentioned above.

6.4.1 Experimental Setup

The Environment For the experiments on foraging and mapping we used the simulated environment presented in Figure 6.19.

As with the *Approach Compensation* experiment, the simulated environment contained one feeder and two obstacles, but this time in a different configuration. Each corner of the environment was also marked with a unique colour. The new configuration was chosen to maximise the effects of the experiments. The feeder was turned away from the centre so that the robot was forced to explore the environment in order to find it.

At the beginning of each experiment the robot had an energy level of 1.0. Each 64 ms the energy level decreased by 0.02. When the robot was in contact with a

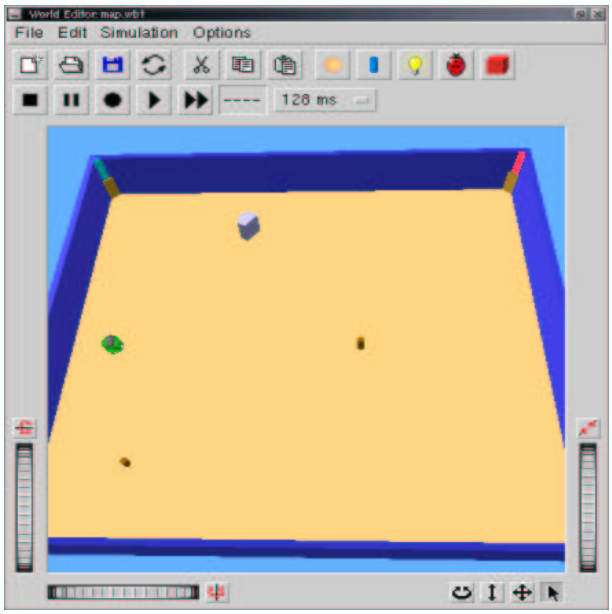


Figure 6.19: Webots Environment for Foraging and Mapping

feeder, the energy level rose to 2.0. If the energy level fell to zero, the robot was considered dead and removed from the environment.

The Strategies Our mapping implementation contained four layers of increasingly sophisticated foraging strategies, each implemented in a behavioural layer.

On top of a base controller that did not have any foraging strategies, but which could do *Random Wandering*, we implemented three layers of increasingly sophisticated foraging strategies: *Structured Exploring* and *Feeding Position Approach*. The layers of the complete solution are presented in Figure 6.20.

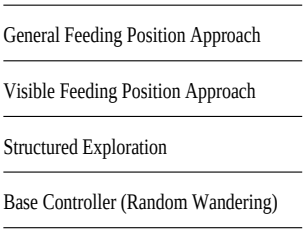


Figure 6.20: The Layers of the Mapping Controller

The layer for *Structured Exploration*, implements a reactive strategy for exploration that make us of four **Corner Sense** circuits. The layer implementing the *Visible Feeder Position Approach* strategy, uses the **Corner Sense** circuits to support a **Position Sense** which againis used as a basis for remebering where feeding has taken place. This layer also provides an energy level sensor and a competence for approaching positions recognised as previous feeding positions when energy levels are low and when such a position is visible. Lastly, the *General Feeder Position Approach* implementation provides a competence for turning the robot toward a remembered feeding position when the robot's energy level is low, hence making the feeder position visible and activating the *Visible Feeder Position Approach* strategy.

6.4.2 The Base Controller (Random Wandering)

As a base for development we used the controller developed for the experiments on Approach Compensation. This controller did random wandering, avoiding obstacles and approaching feeders when visible.

Random Wandering Performance The Random Wandering behaviour implemented by the Base Controller relied on chance to reach different areas of the environment. This could be very inefficient.

Depending on the differences produced by imperfect sensors and actuators, two runs following the *Random Wandering* strategy could have wildly varying paths. There were however similarities between many of the paths e.g. one group reached the north wall in Figure 6.21 with an orientation less π, hence turning right, another group reached the north wall with an orientation greater than π, , hence turning left. A typical path taken by the base controller in eight of the 20 trials is provided in Figure 6.21.

The position and orientation plots for one particular trial from the group of trials that produced paths like the one in Figure 6.21 are presented in Figures 6.22

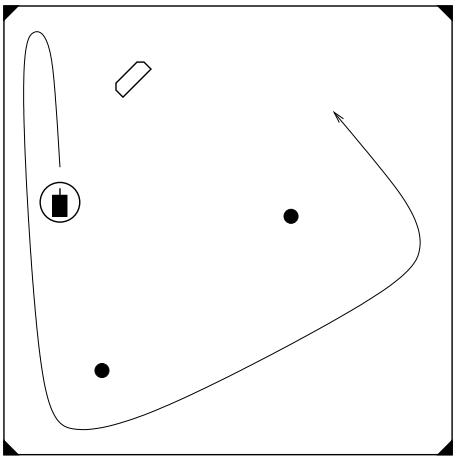


Figure 6.21: Random Wandering Path

and 6.23.

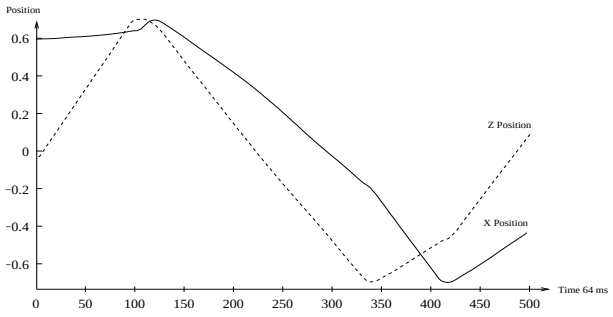


Figure 6.22: Position Over Time for Random Wandering

In order to compare the quality of the different solutions to the foraging problem, we did 20 experiments with each new layer and measured the average *feeding rate*, i.e. the number of times the robot reached the feeder during two minutes.

For the Base Controller, doing Random Wandering, the mean feeding rate was 1.5/2 minutes with a standard deviation of 0.95.

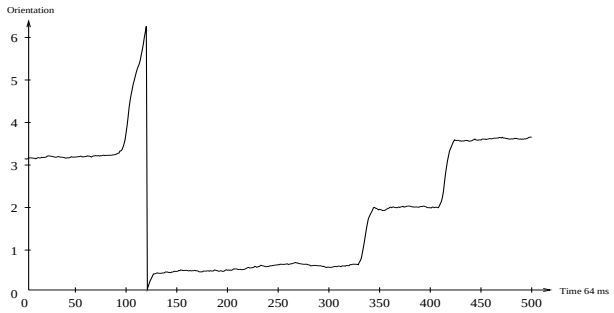


Figure 6.23: Orientation over Time for Random Wandering

6.4.3 Structured Exploration

As an improvement on the *Random Wandering* strategy, this layer implemented an exploring behaviour which surveyed an area by following a hard-coded pattern through the environment, turning and accelerating according to the perceived position.

Layer Architecture The circuits added to the base controller in order to implement *Structured Exploration* are presented in Figure 6.24.

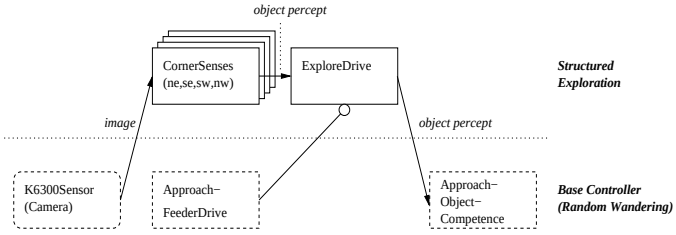


Figure 6.24: Circuits for Structured Exploration

We implemented a set of four **Corner Sense** circuits which relied on he information in the image provided by the **K6300 Camera** circuit and the unique markings of each corner in the environment. The **Corner Sense** circuits passed **Object Percepts** describing the relative position and height of the visible corners to the **Explore Drive** circuit. The **Explore Drive** circuit implemented a pre-defined

pattern of exploration by choosing one of the **Object Percepts** and passing it on to the **Approach Object Competence** circuit described in Section 6.2.

Structured Exploration Performance A typical path taken by the base controller is provided in Figure 6.25. As can be seen from this Figure, the pre-defined exploration pattern was a square along the boundaries of the environments. This path brings the robot back to the feeder at regular intervals.

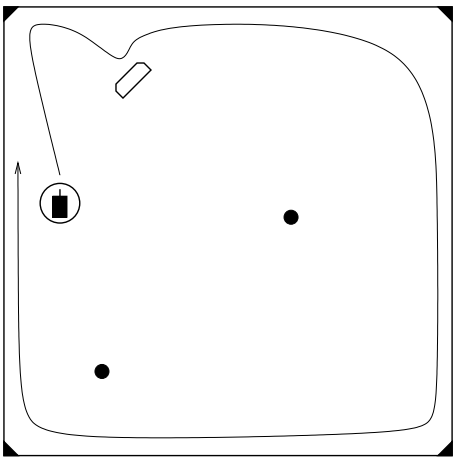


Figure 6.25: Structured Exploration Path

The position and orientation plots for one path typically taken by the base controller are presented in Figures 6.26 and 6.27.

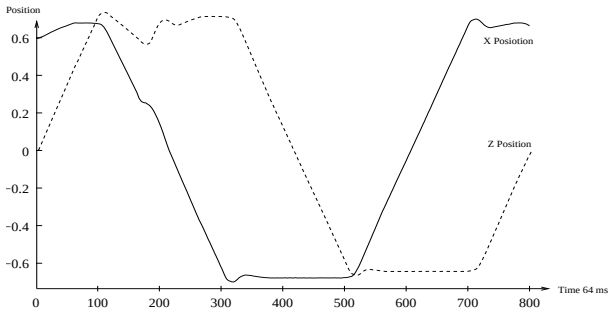


Figure 6.26: Position Over Time for Structured Exploration

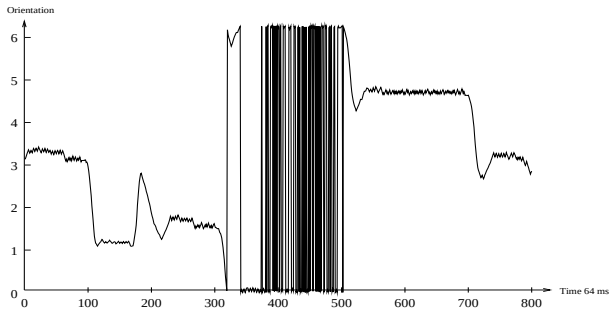


Figure 6.27: Orientation over Time for Structured Exploration

The mean feeding rate for robots following the *Structured Exploration* strategy was 2.7/2 minutes with a standard deviation of 0.5. This feeding rate is significantly higher than the feeding rate for the *Random Wandering* strategy on a 99% confidence level. This shows that in the given environment, the *Structured Exploration* strategy performs consistently better than the *Random Wandering* strategy.

6.4.4 Visible Feeding Position Approach

Supported by the corner sensor circuits, this layer implemented a foraging strategy with a sense of position, a memory for feeding positions, and a drive that could approach positions where feeding had previously taken place.

Layer Architecture The circuits that make up the Visible Feeding Position Approach layer are presented in Figure 6.28.

To give the robot the basic ability to return to a position where it had previously fed, we first implemented a **Feeding Sense** circuit. This circuit took two excitors, the **Empty Feeder Sense** circuit and the **Full Feeder Memory** circuit. The **Full Feeder Memory Circuit** decayed rapidly, so that together these two circuits indicated whether the robot had just emptied the feeder.

Based on the **Corner Sense** circuits from the previous layer, we implemented a **Position Sense** circuit that indicated roughly in which of the four corners the robot was currently situated. Whenever the **Feeding Sense** circuit fired, it would

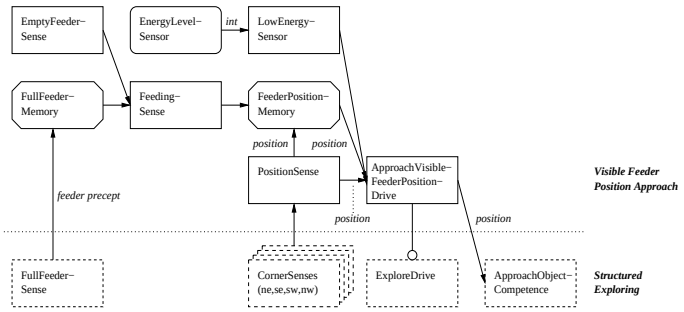


Figure 6.28: Circuits for Visible Feeding Position Approach

establish a memory in the **Feeding Position Memory** circuit containing the current position and hence indicating to the robot in the future, where feeding had taken place.

We also implemented an **Energy Level Sensor** circuit which provided an integer reflecting the robot's current energy level. This we used to implement a **Low Energy Sense** circuit that fired when the energy level fell below a threshold.

Lastly we implemented an **Approach Visible Feeding Position Drive** circuit. When the **Low Energy Sense** circuit was fired, the **Approach Visible Feeding Position** circuit compared the **Object Percepts** provided by the **Corner Sense** circuits with the position provided by the **Feeding Position Memory** circuit. If a match was found, the **Approach Visible Feeding Position Drive** circuit passed the relevant **Object Percept** on to the underlying **Approach Object Competence** circuit and inhibited the **Explore Drive** circuit. This lead to the robot abandoning exploration and heading straight for the corner in which it had fed whenever that corner became visible and the robot's energy was low.

Visible Feeding Position Approach Performance A typical path taken by a robot using the *Visible Feeding Position Approach* strategy is provided in Figure 6.29. It shows that the robot is now able to head for the position where the feeding took place as soon as it came into view. By setting the right threshold for the low energy level sensor circuit, these circuits allow the length of the path the

robot must follow to return to the feeding position to be shortened. As a result the feeding rate increases. Here we present experimental data demonstrating this improvement in performance.

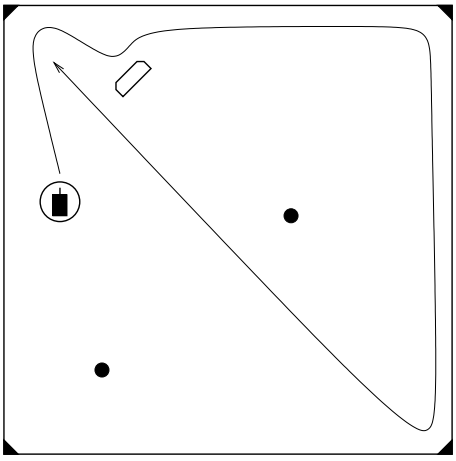


Figure 6.29: Visible Feeding Position Approach Path

The position and orientation plots for the Visible Feeding Position Approach controller are presented in Figures 6.30 and 6.31.

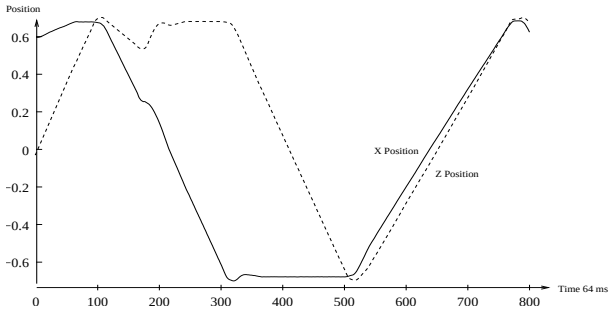


Figure 6.30: Position over Time for Visible Feeding Position Approach

The mean feeding rate for robots following the *Visible Feeding Position Approach* strategy was 3.0/2 minutes with a standard deviation of 0.0. This feeding rate is significantly higher than the feeding rate for the *Structured Exploration* strategy on a 99% confidence level. This shows that in the given environment, the

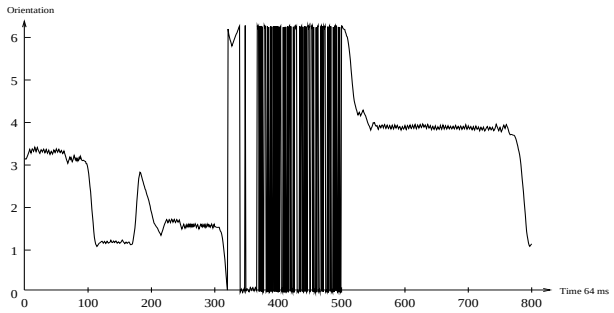


Figure 6.31: Orientation over Time for Visible Feeding Position Approach

Visible Feeding Position Approach strategy performs consistently better than the *Structured Exploration* strategy.

6.4.5 General Feeding Position Approach

This layer generalises the ability to approach places where feeding has previously taken place. The *Visible Feeding Position Approach* strategy could only approach such places when they were in sight. The *General Feeding Position Approach* strategy adds the ability to turn toward these positions when they are not visible.

Layer Architecture The circuits that implemented the *General Feeding Position Approach* strategy are presented in Figure 6.32.

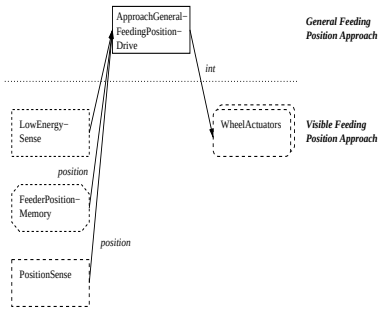


Figure 6.32: The Circuits of the General Feeding Position Approach

The only circuit we added to generalise the *Visible Feeding Position Approach*

strategy to a *General Feeding Position Approach* strategy, was the **Approach General Feeding Position Drive** circuit. This circuit was excited by the **Low Energy Sense** circuit. It got one **Position** percept from the **Position Sense** circuit and the one **Position** percept from the **Feeding Memory** circuit. From this data it calculated where to turn to in order to bring the corner where the feeding took place back into view. The **Approach General Feeding Position Drive** circuit was inhibited by the **Approach Visible Feeding Position Drive** circuit so that when the corner where the feeding took place came into view, the robot would use the underlying *Approach Visible Feeding Position* strategy to take it to the feeder.

General Feeding Position Approach Performance Our final strategy for foraging was the *Approach General Feeding Position* which allowed the robot to head for the position where the feeding took place any time the **Low Energy Sense** circuit was firing. By adjusting the definition of low energy in that circuit, the *Approach General Feeding Position* strategy could increase the feeding rate of the *Approach Visible Feeding Position* strategy. Here we present experimental data demonstrating this improvement.

A typical path is provided in Figure 6.33, showing that the *Approach General Feeding Position* strategy cuts short the path produced by the *Approach Visible Feeding Position* strategy presented in Figure 6.29.

The position and orientation plots for the base controller are presented in Figures 6.34 and 6.35.

The mean feeding rate for robots following the *General Feeding Position Approach* strategy was 3.6/2 minutes, with a standard deviation of 0.5 seconds. This feeding rate is significantly higher than the feeding rate for the *Structured Exploration* strategy on a 99% confidence level. This shows that in the given environment, the *General Feeding Position* strategy performs consistently better than the *Visible Feeding Position Approach* strategy.

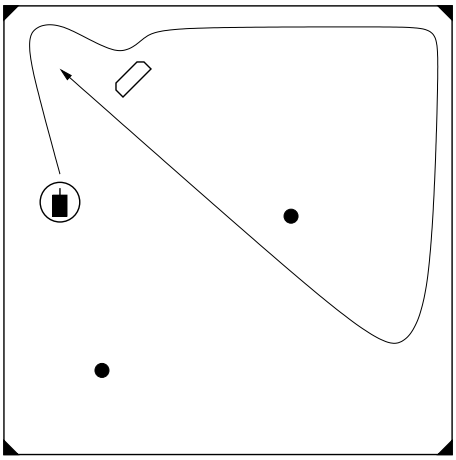


Figure 6.33: Visible Feeding Position Approach Path

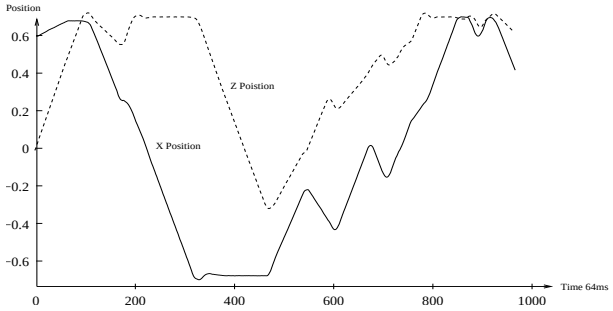


Figure 6.34: Position Over Time for General Feeding Position Approach

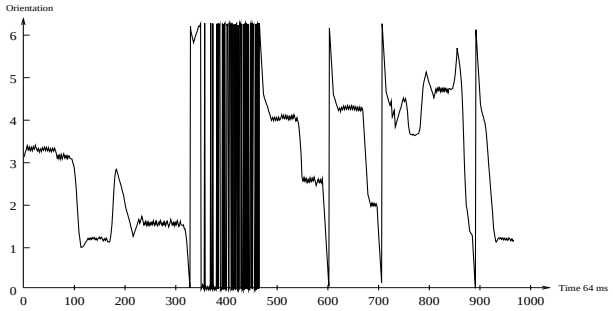


Figure 6.35: Orientation over Time for General Feeding Position Approach

6.5 Multiple Adaptive Layers for Conflict Resolution

One important form of learning in humans and animals is social learning which includes learning from interaction with others. As a final demonstration of BBL, we look at our recurrent example of conflict resolution example. When two entities compete for a limited resource, there are a number of increasingly sophisticated strategies available to the participants. Here we present implementations of four such strategies and demonstrate through experiments, the consistent improvement in performance produced by each new layer.

6.5.1 Experimental Setup

The simulated environment in which our experiment on conflict resolution took place is presented in Figure 6.36. It is identical to the environment used for the experiment on foraging but has an additional Khepera robot.

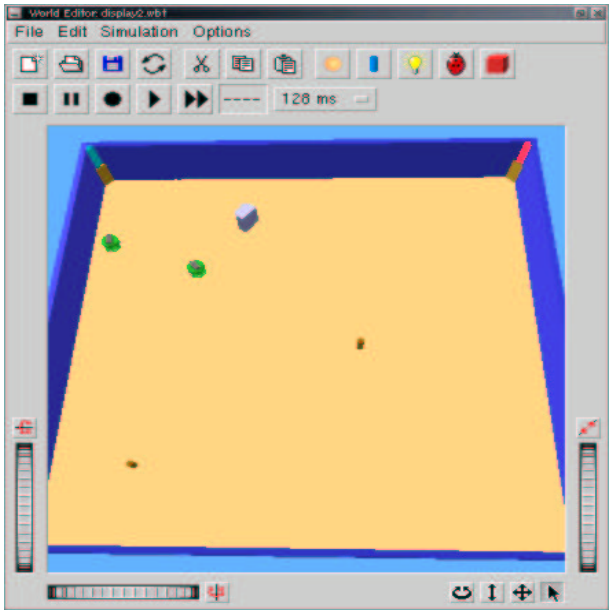


Figure 6.36: Simulated Environment for Conflict Resolution

In addition to the features described in Section 6.4, we added a *strength* value and an *injury* value to Khepera robots in this experiment. The two robots were

given different and unchangeable strength values from the start. The strength values were 0.005 and 0.001. We refer to the robot with the highest strength value as the strong robot and the robot with the lowest strength value as the weak robot.

Injury was accrued whenever the two robots were within a given proximity of each other. As long as the robots remained within close proximity of each other, more injury was received every 64 ms. The added injury was identical to the strength of the other robot. The initial injury level was 0.0 and if a robot's injury exceeded 2.0, the robot was considered dead and removed from the environment.

The measures used to evaluate the performance of the conflict resolution strategies below are the number of times each robot fed from the feeder and the final injury levels. The energy reduction level was set to 0.0, implying that none of the robots would die from low energy levels in these experiments.

6.5.2 The Strategies

We designed four increasingly sophisticated strategies for conflict resolution, presented in Figure 6.37.

Stylised Hierarchy Establishment
Pecking Order Adherence
Reactive Yielding
Uncritical Fighting
Base (Avoidance)

Figure 6.37: Behavioural Layers for Conflict Resolution

In the *Avoidance* strategy, the robots were oblivious of each other's existence and only perceived the other robot as an obstacle to be avoided. The *Uncritical Fighting* strategy introduces minimal conflict resolution capabilities that let the robots attack each other until one is dead. This strategy is completely one sided in that the strong robot quickly disposes of the weak robot and gets exclusive access to the feeder. The following strategies let the weak robot survive for an increasingly

long time and hence increase the average number of times it accesses the feeder. In the *Reactive Yielding* strategy the weak robot does not attack when it can sense that it is the weakest. The *Pecking Order Adherence* strategy remembers whether the robot is weaker than the other robot and if it is, it never attacks the other robot. In the *Stylised Hierarchy Establishment* strategy the robots avoid the fight necessary to establish the strength relation by using a display behaviour. Displays are common in animals and are one of the simplest forms of animal communication [Hauser, 1996].

6.5.3 The Base Controller (Avoidance)

As a base for the conflict resolution experiments we used the foraging controller we presented in Section 6.4. This controller generally brought the two robots back to the same place and created ample conflict situations.

The plots for the X position, the Z position and the orientation of the two robots using the *Avoidance* strategy are shown in Figures 6.38, 6.39 and 6.40 respectively.

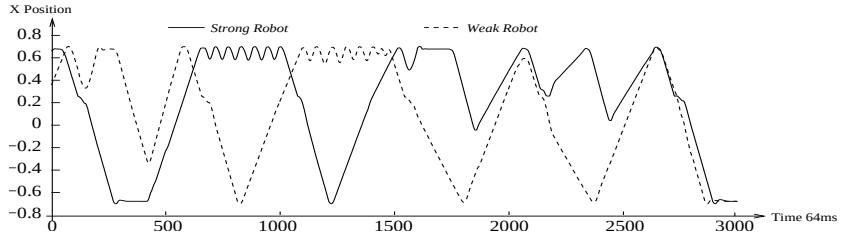


Figure 6.38: X Positions over Time for Avoidance

6.5.4 Uncritical Fighting

The *Uncritical Fighting* strategy is a simple, reactive strategy for interaction. It has the robots attacking each other on sight without regard for any other factors. The circuits that implement the *Uncritical Fighting* strategy are presented in Figure 6.41.

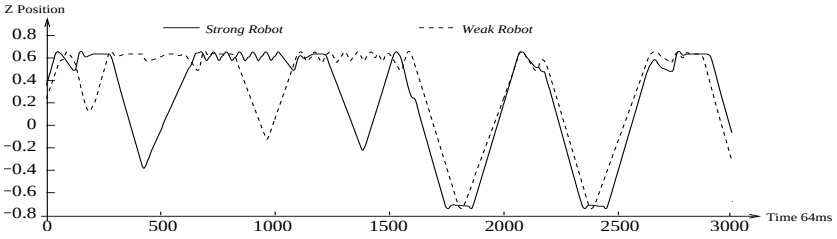


Figure 6.39: Z Positions over Time for Avoidance

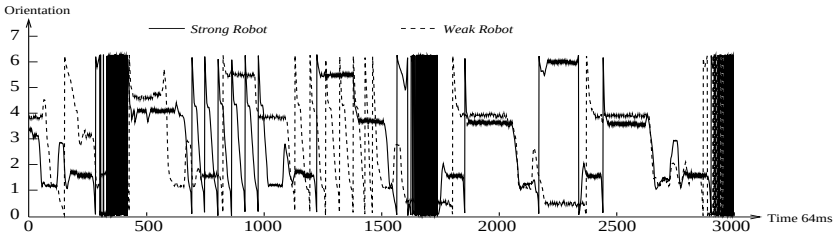


Figure 6.40: Orientation over Time for Avoidance

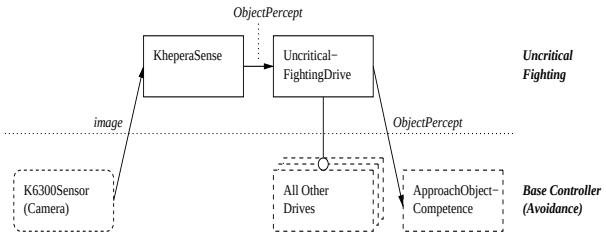


Figure 6.41: Circuits for Uncritical Fighting

A **Khepera Sense** circuit was added which recognised the colour of a simulated Khepera robot in an image from the **K6300 Camera Sensor** circuit and produce an **Object Percept** describing the size and relative position of the observed Khepera. The presence of a Khepera Robot excited an **Uncritical Fighting Drive** circuit which inhibited all the underlying drives and passed the **Object Percept** describing the Khepera Robot to the **Approach Object Competence** circuit introduced in Figure 6.24.

Uncritical Fighting Performance This strategy allowed the strong robot to kill off the weak one and significantly increase the number of times it fed when compared to the *Avoidance* strategy. The plots for the X position, the Z position and the orientation of the two robots using the *Uncritical Fighting* strategy are shown in Figures 6.42, 6.43 and 6.44 respectively.

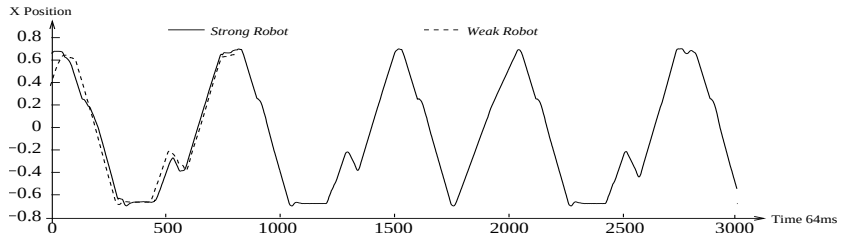


Figure 6.42: X Positions over Time for Reactive Fighting

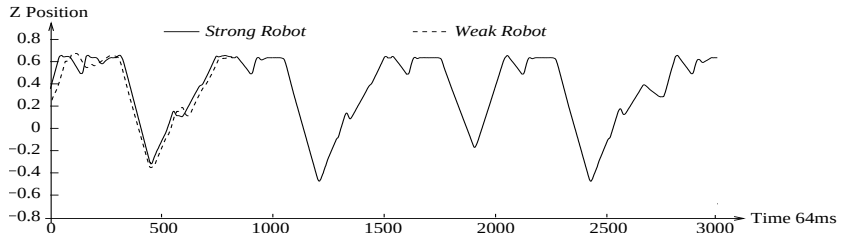


Figure 6.43: Z Positions over Time for Reactive Fighting

The average survival time for the weak robot during these experiments was 63.4 seconds, with a standard deviation of 25.0 seconds.

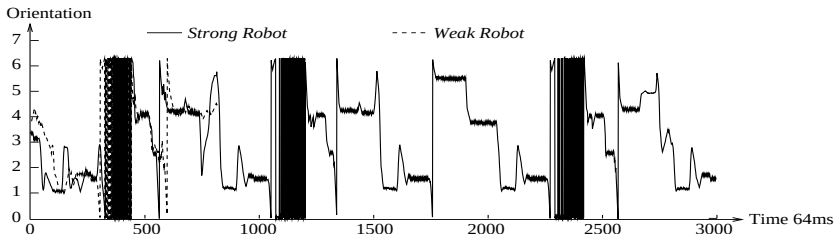


Figure 6.44: Orientation over Time for Reactive Fighting

6.5.5 Reactive Yielding

This strategy added the ability to yield, i.e. to stop fighting, whenever a robot sensed that it was the weaker of the two robots. The circuits that implement the *Uncritical Fighting* strategy are presented in Figure 6.45.

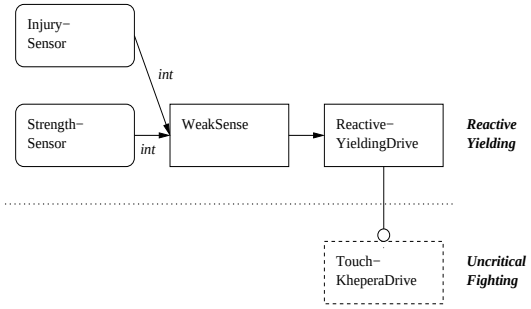


Figure 6.45: Circuits for Reactive Yielding

We implemented a **Strength Sensor** circuit to give the robot access to its own strength and an **Injury Sensor** circuit that reflected the strength of the opponent indirectly though the amount of injury received. We used the information from these two sensors to implement a **Weak Sense** circuit which fired only if the injury received was greater than the robot's strength, i.e. when the robot was weaker than its opponent. Finally we implemented a **Reactive Yielding Drive** circuit which was activated by the **Weak Sense** circuit. The **Reactive Yielding Drive** circuit inhibited the **Uncritical Fighting Drive** circuit presented above, letting the weak robot steer away from the strong one.

Reactive Yielding Performance The *Reactive Yielding* reduced the rate at which the weaker robot sustained injury by unilaterally aborting its fighting behaviour. The strong robot on the other hand, would keep fighting and chasing the weak robot. The plots for the X position, the Z position and the orientation of the two robots using the *Reactive Yielding* strategy are shown in Figures 6.46, 6.47 and 6.48 respectively.

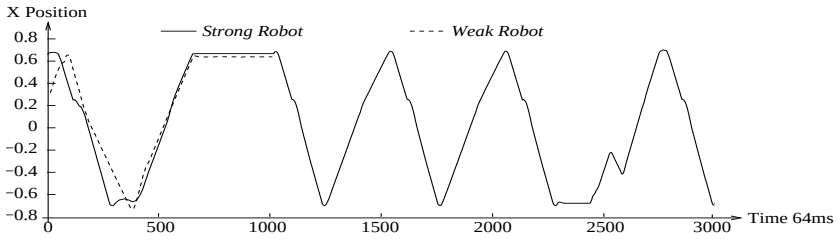


Figure 6.46: X Positions over Time for Reactive Yielding

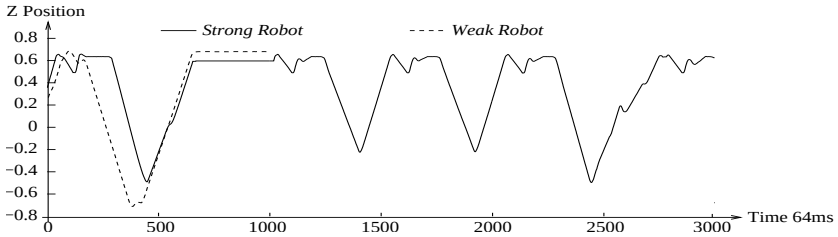


Figure 6.47: Z Positions over Time for Reactive Yielding

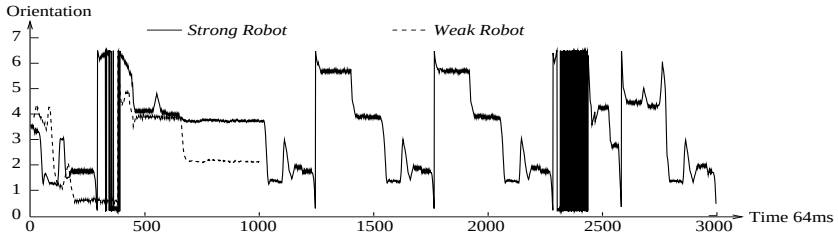


Figure 6.48: Orientation over Time for Reactive Yielding

The average survival time for the weak robot following the *Reactive Yielding*

strategy was 73.9 seconds with a standard deviation of 37.1 seconds. The student t-test indicates that on a 75% confidence level, the survival time is consistently longer for robots that follow the *Reactive Yielding* strategy than it is for robots that follow the *Uncritical Fighting* strategy.

6.5.6 Pecking Order Adherence

Building on the reactive *Uncritical Fighting* strategy we introduced the circuitry necessary to establish a pecking order that would stop the weak robot from attacking the strong one. The circuits that provided the robots with this capability are presented in Figure 6.49

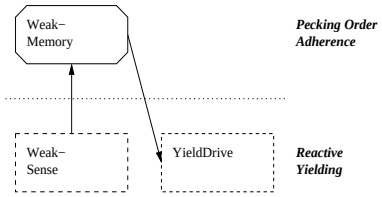


Figure 6.49: Circuits for Pecking Order Adherence

The only added circuit was the **Weak Memory** circuit. This memory was established whenever the **Weak Sense** circuit introduced for *Reactive Yielding*, fired. This memory circuit inhibited the **Uncritical Fighting Drive** circuit so that the weak robot would always avoid the strong robot after an initial encounter.

Pecking Order Adherence Performance By learning its place in the pecking order and using this information to avoid the strong robot, the weak robot significantly increases its lifespan, represented by the number of times it is able to feed. The plots for the X position, the Z position and the orientation of the two robots using the *Pecking Order Adherence* strategy are shown in Figures 6.50, 6.51 and 6.52 respectively.

The average survival time for the weak robot following the *Pecking Order Adherence* strategy was 95.4 seconds with a standard deviation of 32.95 seconds. The

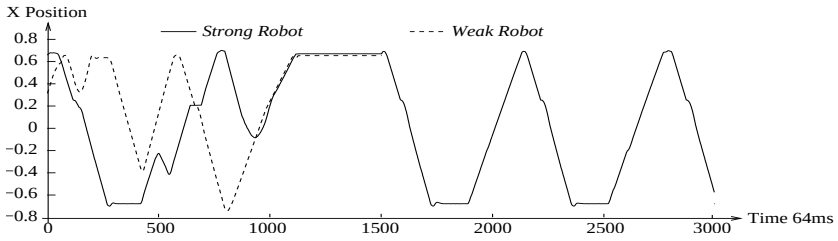


Figure 6.50: X Positions over Time for Pecking Order Adherence

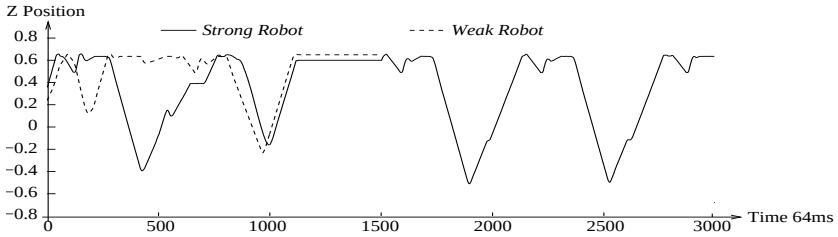


Figure 6.51: Z Positions over Time for Pecking Order Adherence

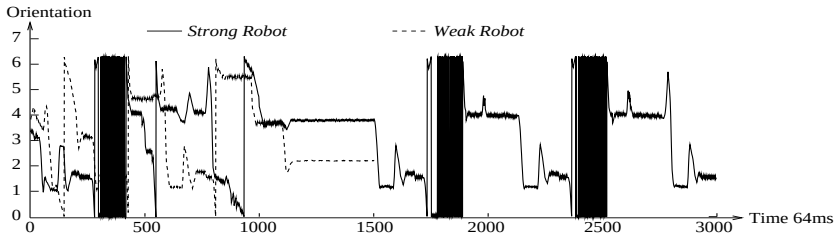


Figure 6.52: Orientation over Time for Pecking Order Adherence

student's t-test indicates that on a 95% confidence leveli, the survival time is consistently longer for robots that follow the *Pecking Order Adherence* strategy than it is for robots that follow the *Reactive Yielding* strategy.

6.5.7 Stylised Hierarchy Establishment

As our ultimate strategy for conflict resolution, we implemented a *Stylised Hierarchy Establishment* strategy that uses a display behaviour instead of direct interaction to establish the relationship of strength between the two robots.

The circuits implementing the *Stylised Hierarchy Establishment* strategy are presented in Figure 6.53.

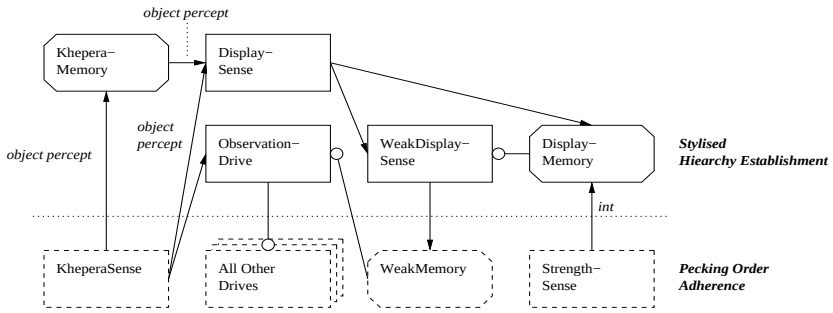


Figure 6.53: Circuits for Stylised Hierarchy Establishment

We implemented standing still as the display behaviour. In order to see whether the other robot was standing still we implemented an **Observe Drive** circuit. This inhibited all underlying drives when the other robot was in sight. To see whether the other robot was displaying, we implemented a **Khepera Memory** circuit that remembered the last **Object Percept** provided by the **Khepera Sense** circuit and provided the stored **Object Percept** next time it was triggered. By comparing the **Khepera Memory** circuit and the **Khepera Sense** circuit we could find out whether the other robot was moving or standing still. If the two were the same, the other robot was standing still. If they were different, the other robot was moving. We added the **Display Sense** circuit to do this comparison.

To keep a display behaviour active for an amount of time that corresponded to the robot's strength, we added a **Display Memory** circuit. This memory could only be established once within a given time period and had a strength that corresponded to the integer provided by the **Strength Sensor** circuit and it decayed slowly so that the time it would take to reach zero, i.e. to be forgotten, corresponded directly to the robot's strength.

To sense whether the robot was strong or weak, we implemented the **Weak Display Sense** circuit. If the **Display Sense** circuit was still active when the display period, as decided by the **Display Memory** circuit was over, this would indicate that the other robot was displaying for a longer time and hence was stronger. Having the **Weak Display Sense** circuit being excited by the **Display Sense** circuit, but inhibited by the **Display Memory** circuit mirrored these circumstances. The excitation of the **Weak Display Sense** then established a memory in the **Weak Memory** circuit which again was made to inhibit further displays.

Stylised Hierarchy Establishment Performance By learning its place in the pecking order this way, the robots significantly increases their lifespans.

The plots for the X position, the Z position and the orientation of the two robots using the *Stylised Hierarchy Establishment* strategy are shown in Figures 6.54, 6.55 and 6.56 respectively.

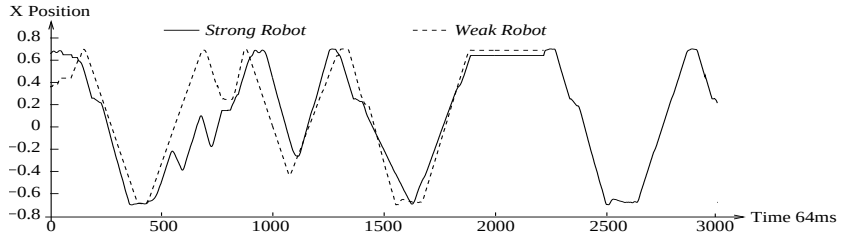


Figure 6.54: X Positions over Time for Stylised Hierarchy Establishment

The average survival time for the weak robot using the *Stylised Hierarchy Establishment* strategy was 104.6 seconds with a standard deviation of 54.7 seconds.

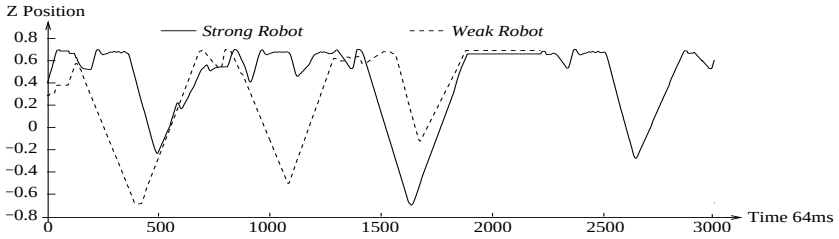


Figure 6.55: Z Positions over Time for Stylised Hierarchy Establishment

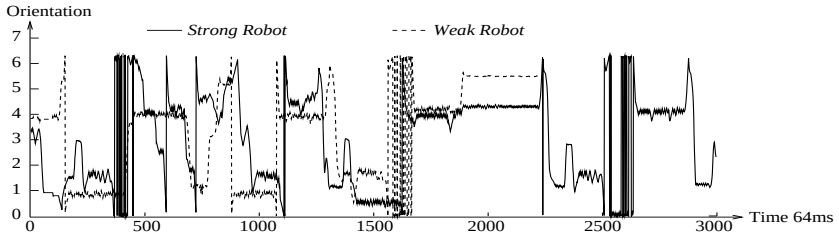


Figure 6.56: Orientation over Time for Stylised Hierarchy Establishment

The student's t-test indicates that, on a 60% confidence level, the survival time is consistently longer for robots that follow the *Stylised Hierarchy Establishment* strategy than it is for robots that follow the *Pecking Order Adherence* strategy.

6.6 Conclusions

We have presented three experiments where BBL was used to produce rapidly adaptive solutions to diverse problems. In the approach compensation problem we demonstrated how BBL produced a consistent improvement in the ability of a robot to reach an observed target. The foraging experiment demonstrated increasingly sophisticated BBL solutions to the foraging problem, each producing a consistent improvement in the feeding rate of the robot. Our final experiment on conflict resolution demonstrated how BBL was used to produce consistent increases in the survival time for robots competing for a scarce resource.

Overall, our three experiments demonstrate consistent improvement in performance as a result of rapid adaptation using BBL. The diversity of the problem domains and the complexity of the solutions we have demonstrated suggest that BBL can be used as a general design paradigm to produce solutions that are quick to adapt and robust with respect to partial failures.

Chapter 7

Holistic Artificial Intelligence

Contents

7.1 A Holistic Approach to Artificial Intelligence

Our BBL methodology describes a way of implementing efficient and robust animal learning. The aim is to extend the BBL methodology to include guidelines for developing high-level learning behaviours such as skill learning and reasoning in a step-wise manner from simpler underlying adaptive capabilities, taking inspiration from psychological and physiological theories of animal and human intelligence.

In this Section we present some of the perceived shortcomings of current AI research based on our experiences with transfer of psychological and traditional physiological theories of learning to robotics. We present the shortcomings as a set of high level recommendations. There are many pragmatic reasons not to follow these recommendations, but we think there is value in presenting them as an ideal for research in AI [Dahl and Giraud-Carrier, 2001a].

We have called this approach behaviourally holistic because it recommends integrating solutions from all dimensions of behaviour as a way of providing the necessary biases for high level learning and hence avoiding the brittleness of one-dimensional solutions.

7.1.1 Holistic Recommendations

Our holistic approach to robotics is most simply described by the following three recommendations:

1. Retrace evolutionary levels.
2. Include as many dimensions of behaviour as possible on each level.
3. Provide biology-inspired problems, hardware and environments of suitable complexity for each level.

Retracing Evolution The first recommendation is a modification of a the original recommendation for BBAI to take inspiration from evolution [Brooks, 1991b].

Our recommendation emphasises the need to look at robots on a number of different evolutionary stages rather than just considering evolution in general. The relation to BB robotics is discussed further below.

Behavioural Holism The second recommendation reflects the realisation that complex behaviours cannot realistically be explored without being immersed in a rich behavioural context.

The motivation for retracing evolution is that many of the behaviours found in animals and humans are so complicated and poorly understood that robust and efficient direct implementations are impossible. In these cases, retracing evolution forces an investigation of the evolutionary history of the behaviour in question. The evolutionary histories of behaviours are highly interrelated, and looking at a limited number of behaviours cannot reveal all the details necessary for a comprehensive understanding. When considering general AI and robotics, undertaking a complete behavioural investigations can be preferable to implementing complex behaviours directly.

One of the main sources of inspiration for our holistic development is the growing amount of knowledge of the physiological and evolutionary history of biological systems in areas such as ethology, neuro-science, and the cognitive sciences. There is currently a wide scope for using this knowledge in AI implementations.

Designing Fitting Environments As important as defining sensible steps for robot development in terms of coherent cognitive structures is the development of fitting environments for such robots to inhabit. The complexity of the habitat, including the social environment, presents many of the different problems that drives evolution.

A noticeable gap exists between the environments commonly inhabited by existing research robots and the habitats of animals that are relatively simple physically and cognitively speaking. The complexity of natural environments presents many problems that are generally ignored by the robotics community, such as deal-

ing with natural visual scenes and propulsion across natural surfaces. There are of course practical limitations to the environments robots can inhabit, but by viewing these low level problems as less important than the problems on the higher levels, we are in danger of placing the cart before the horse.

The more complex learning gets, the more likely it is to need support from other behaviours in different behavioural dimensions as well as from a suitable environmental complexity.

7.1.2 The Need for Holistic Solutions

Conclusions from Our Own Work As we analysed increasingly complex forms of animal learning, it became difficult to design natural learning problems to test the different learning types due to the poverty of the underlying controllers. In designing a particular controller not included in this dissertation, for demonstrating alpha-conditioning, we needed the robot to recognise that a certain stimulus would regularly occur together with food. The underlying controller could only recognise other robots and food, so we developed an artificial pink box sense.

This was a poor solution. In nature an animal would most likely have developed the sensory machinery to recognise a number of different objects as part of a rich set of basic behaviours. The associative abilities are then likely to have evolved on top of these rather than the associative abilities and novel sensory machinery having evolved together.

If a holistic approach had been taken, we would have had a larger number of basic behaviours to choose from so that our alpha-conditioning experiment would have been more natural and perhaps would have brought up issues of behaviour integration that were missed because of the artificial nature of our pink box sense.

From our analysis of animal learning it also seems that the more complex a form of learning is, the more underlying behaviours it needs to support it. This makes sense in an ML framework when considering that the underlying behaviours provide the learning biases for the adaptive mechanisms. The more general the

learning mechanism becomes, the more bias it will need to reduce the search space to a practical size. The most general learning mechanisms in animals and humans, such as skill learning and symbolic reasoning are likely to use biases from a large number of underlying behaviours in a number of different ways. In trying to implement high level learning bottom-up it is dangerous to leave out basic behaviours as these might provide necessary biases later. We call a lack of supporting behaviours in learning *behavioural starvation*.

Arguments from Cognitive Robotics Brooks criticised work in BBAI before 1997 for not having a wide enough *behavioural repertoire* [Brooks, 1997] and also listed seven other issues that must be dealt with when considering *cognitive robotics*, which he saw as the next step on from BB robotics. The issues were:

- bodily form
- motivation
- coherence
- self-adaptation
- development
- historical contingencies
- inspiration from the brain

What Brooks was doing in this argument was to insist that the robot be placed in a behavioural space. This criticism corresponds to his 1991 criticism of symbolic AI where he insisted that AI be placed in physical space. What our holistic approach does is to define more closely the dimensions of the behavioural space, while keeping the evolutionary dimension of behaviour and tracing a sensible and practical development path up through the evolutionary levels of complexity.

In our analysis in Figure 7.1 *bodily form* is presented as the sensor and actuator dimensions. *Motivation* is represented by the traditional four F's of animal behaviour, feeding, fighting, fleeing, and procreation. Finally, we merged *self-adaptation* and the learning aspects of *development* into one adaptability dimension, while the interaction aspects of development are present in our procreation and social context dimensions.

Brooks' issues of *historical contingencies* and *inspiration from the brain* are methodological issues rather than technological issues. These issues point out the main aspect in which a holistic approach diverges from cognitive robotics, the holistic approach following evolution more closely. In general the cognitive robotics approach talks about the behavioural space of humans, the most complex manifestations of the elements along the horizontal axis in Figure 7.1. The holistic approach on the other hand, keeps the evolutionary dimension of the behavioural space according to Brooks' original suggestions and uses it to facilitate robotics and AI research by traversing it from the bottom-up. The evolutionary dimension of behavioural space is represented as the vertical axis in Figure 7.1.

Brooks' point about avoiding the implementation of things that are merely *historical contingencies* best demonstrates the difference between holistic and Cognitive Robotics. Finding and exploring historical contingencies is a necessary part of our approach and one that cannot and should not be avoided. Likewise, *coherence* or behaviour integration is less of a problem with an evolutionary approach. On a human level, the question of coherence is overwhelmingly complex, while extending simple coherent robots in small steps breaks the problem down.

On the issue of taking *inspiration from the brain* we wholeheartedly agree with the cognitive robotics approach. This is openly reflected in the cognitive modelling features of our PLANCS framework.

Arguments from Evolution Zoologists have provided one of the strongest arguments for a holistic approach to AI:

> No single characteristic could evolve very far toward the mammalian condition unless it was accompanied by appropriate progression of all the other characteristics. However, the likelihood of simultaneous change in all the systems is infinitesimally small. Therefore only a small advance in any one system could occur, after which that system would have to await the accumulation of small changes in all the other systems, before evolving a further step toward the mammalian condition.
>
> *T.S. Kemp [Kemp, 1982]*

This quote was also used in [Allman, 1999], which in addition presents the following example. In order to maintain a constant body temperature and extend their periods of activity, warm blooded animals need to consume an order of magnitude more food than cold-blooded animals. As a result, they have changed the way they chew food, their breathing, their locomotion, their parenting behaviour, their senses, their memory capacity, and their brain size.

In cognitive modelling, we can make simultaneous changes, but we cannot make large changes along one behavioural dimension without appropriately advancing others.

7.1.3 Mapping Behavioural Evolution

In order to develop a sense of the different cognitive, physical and environmental levels of complexity that provide sensible combinations for artificial systems, we have taken a first step to producing a road-map for an evolutionary and holistic traversal of behavioural space. We have done an initial analysis of the evolution of animal and human behaviour. It is meant as a rough map for choosing appropriate levels for different interdependent features of artificial systems when designing holistic solutions.

Tyrrell [Tyrrell, 1993] reviews several taxonomies of behaviours and survival problems in his work on action selection. Action selection is closely related to be-

	Physical Environment	Social Environment	Sensors	Actuators	Fleeing	Fighting	Procreation	Feeding	Adaptability
6				Speech organ					Symbolic learning
5	Settlements		Colour vision	Hands	Nesting	Armed fighting		Agriculture	Insight
								Armed hunting	
4		Hierarchical Group							Temporal association
							Raise young		Imitation
3		Uniform Group		Vocal tract	Group protection	Group fighting			
2	Land	Family	Stereo vision	Legs	Hiding	Displaying	Mating	Hunters	
			Stereo hearing		Fleeing pursuer		Release		
			Motion compensation	Head					
			Directed vision						
1	Water	Solitary	Exterio−receptors	Body	Moving	Physical fighting	Division	Grazers	Association
			Interio−receptors						Habituation

Figure 7.1: Behavioural Evolution

haviour integration in that they both study mechanisms for expressing a beneficial behaviour from many available alternatives.

Our rough analysis of evolution presents six important evolutionary stages. Between the stages are behaviours and features that are likely to have coexisted during evolution. These behaviours and features are likely to have influenced and supported each other and artificial solutions that include all these behaviours can draw on this support to facilitate development. By including all the behaviours and features between stages we also reduce the risk of brittle solutions and behavioural starvation.

By plotting other AI and robotic systems on the map, we can evaluate their holistic quality. The greater the vertical gap between the most and least complex instantiations of the behavioural dimensions, the greater the danger of brittleness in the complete solution. Robots for particular problems in restricted environments perform well with highly specialised solutions where a small number of behavioural dimensions are developed to a high level of complexity. The holistic

approach, however, aims to develop efficient and robust general capabilities for a wide range of environments and as a result needs to consider all the dimensions of behaviour.

For further research in robotics to be successful, it will be necessary to develop the map to a higher level of detail and realism by undertaking a thorough study of evolution.

Limitations on Progress In robotics, research is currently taking place on issues on all the different levels of complexity presented in Figure 7.1 and very little work has been done on integrating behaviours in holistic frameworks. The behaviours on the lower levels might be studied in isolation with some credibility, while further up, one-dimensional research gets less and less useful in a general AI and robotics context as the solutions get increasingly impractical to port to situated real-time systems.

Existing research in areas such as planning, reasoning, natural language and interaction protocols has produced important scientific results and impressive software engineering tools, but it is not obvious that any such results or tools have an immediate place in BB robotics and situated agents [Agre and Chapman, 1990].

Chapter 8

Related Work

Contents

8.1 Early Approaches to Machine Intelligence

8.1.1 Cybernetics

The recognition of commonalities between biological and artificial control systems [Weiner, 1948] gave rise to the field of Cybernetics. This field produced the world's first autonomous robotic animals, at the Burden Neurological Institute in Bristol, UK [Grey, 1950, Grey, 1963], and initiated the connectionist paradigm [McCulloch and Pitts, 1943]. Cybernetics focuses on control and communication and has traditionally been concerned with control-related researcg topics such as self-organisation, feedback-cycles and communcation.

With the advent of computers, the sciences that focused on computerised control took over many of the issues traditionally belonging in the field of Cybernetics and rephrased them in their own terms [Heylighen and Joslyn, 2001]. However, the field of Cybernetics has retained some influence as inspiration for the areas such as ANNs and BBAI.

8.1.2 Artificial Intelligence

The electronic computer is, so far, the only artifact that has ever even been suspected, by the scientific community, to be able to embody the same mental faculties as the evolved brain.

The idea of general computing machinery was initially conceived of in terms of Babbage's Analytical Engine [Bromley, 1998]. However, the first serious consideration of implementing human level intelligence was published by Turing in 1950 [Turing, 1950]. The science that researches this problem has since the Dartmouth Computer Conference at Dartmouth College, New Hampshire, in 1956 been called AI.

Since its infancy, AI has focused on high-level mental capacities. Turing defined a test of intelligence, later known as the Turing Test. The test constitutes the requirements for a computer to pass for a human in the eyes of another human.

This view of intelligence as high-level mental and communicative processes was reflected in the topics researched in early AI; natural language processing, automated reasoning, expert systems and machine learning.

8.1.3 Agent-Oriented Systems

The general trend in software engineering tools is toward higher levels of abstraction and a data and control representation increasingly tailored to human comprehension. Software engineering paradigms have followed a natural progression from assembly language programming through high-level languages and procedural languages to object-oriented and Agent-Oriented languages [Shoham, 1993]. In Agent and Multi-Agent systems, program structures are often defined using high-level cognitive terms such as beliefs, desires, intentions and goals, and functionality is described as the interaction between these features [Rao and Georgeff, 1995].

The co-existence of rationality and reactivity within an entity has been held up as one of the defining properties of agent-hood [Wooldridge and Jennings, 1995]. In recognising this duality, Agent-Oriented systems have arrived at the same problem: that of combining high-level and low-level cognitive processes. This recognition has also fed back into classical rational paradigms such as logic programming [Kowalski, 1979], making them recognise that high-level reasoning and logic also contain elements of reactivity [Kowalski and Sadri, 1996].

8.2 Behaviour-Based Artificial Intelligence

A shift in the philosophy of intelligence which emphasised embodiment and situatedness [Lakoff and Johnson, 1999] provided inspiration for a new approach to AI [Brooks, 1991b]. This resulted in a new wave of research, studying embodied, low level systems and their interaction with the world.

Whereas GOFAI performed certain high-level mental tasks, such as playing chess [Hsu et al., 1990], with unrivalled efficiency, it fell short when it came to real

world interaction. The highly complex, noisy and dynamic state of the real world was too much to handle by brute force reasoning. The most publicised success of the GOFAI approach to adaptive robotics was Shakey the robot [Nilsson, 1984] who used symbolic planning to interact with the world. The problem with this work was that Shakey navigated a very clean, simple, and well lit environment. Brooks' argument [Brooks, 1991a] is that: '...complete objective models of reality are unrealistic-and hence the methods of Artificial Intelligence that rely on such models are unrealistic.'.

This new, pragmatic paradigm excelled at solving simple problems of real world interaction by minimising the control and information paths from sensors to actuators and the use of explicit internal world representation. From the new paradigm's focus on tasks rather than internal processes was derived the name BB systems. This new difference in division of control is also referred to as a horizontal division where each layer forms a complete path from inputs to actions as opposed to the traditional vertical division where control is divided sequentially into sensing, planning and acting.

BBAI solutions are robust in that they are tolerant to noise as a result of having been developed in noisy environments. They are also tolerant to limited internal failures due to their distributed, concurrent execution model. The BB approach to AI also facilitates development by dividing the monolithic problem of intelligent behaviour into separate sub-problems or layers.

8.2.1 States, Concepts and Representations

BBAI has been criticised as being 'computational behaviourism' [Tsotsos, 1995], because he denounces any use of internal representations. Brooks also suggested increasing the reactivity of robots by avoiding the use of internal state. This has in extreme cases, been interpreted as a claim that intelligent behaviour is 'concept free'[Kirsch, 1991].

This criticism is partly a result of using different semantics for the term 'repre-

sentation'. Brooks sees it as necessary to pass numbers between processes. These numbers can be given an objective interpretation in terms of the states of the communicating processes. It is also clear that such a number might represent a highly synthesised part of a distributed percept, e.g. it could be objectively interpreted as the height of a persons ear. However, Brooks does not agree to calling such a number a representation, explicit or implicit, and states that one should '...never use tokens that have any semantics that can be attached to them'.

This interpretation of 'representation' must be seen in the light of the planning systems Brooks differentiated himself from. These systems tried to force raw input into pre-labelled categories that could be used in high-level abstract planning systems. An interpretation of 'representation' similar to Brooks' was made by Evolutionary Roboticist, Harvey [Harvey, 1996] who states that 'A symbol P is used by a person Q to represent, or refer to, an object R to a person S.', i.e. there must be two people present for a process of representation to take place. In order to clarify, we the use the term *internal concept*, to refer to internal or implicit representations that do not refer, by name, to any world feature but must be interpreted in terms of system state to be given semantics. We use the term 'representation' to mean the use of named variables where the name refers to the variable's semantic meaning.

Tsotsos's criticism of BB was based on showing that the minimisation of internal concepts made a number of problems computationally harder rather than simpler. Another argument against concept-free intelligence is evidence from biology and neuro-science which shows that animals and humans do have internal concepts according to the definition above: for example cells in the visual cortex that fire when a slanted line appears in a region of the visual field [Bruce et al., 1997]. Maes also made it clear that BB systems could have goals [Maes and Brooks, 1990]. Bryson later presented the case for hierarchical rather than fully parallel structures between behaviours [Bryson, 2000].

Another problem with the initial BB philosophy was behaviour integration. Subsumption of low level behaviours by high-level ones is not always appropri-

ate [Gat, 1998]. Sometimes a critical, low level behaviour should be expressed and the high-level alternative should be subsumed. The subsumption architecture initially only allowed top-down subsumption. More liberal approaches within the BB paradigm have later introduced more flexible abstractions and techniques for behaviour interaction [Brooks, 1990] as well as internal state with rich use of internal concepts [Matarić, 1992]. This greatly increased the applicability of the BB paradigm with a minimal loss in performance. The more flexible approach to behaviour integration has later been called the Port-Arbitrated Behaviour Paradigm (PAB) [Werger, 2000].

8.2.2 Hybrid Systems

A class of controllers that accepts the duality between rational and reactive processes and which uses these processes in separate, but interacting, layers, are *hybrid systems* [Lynch and Krogh, 2000]. Some hybrid systems differentiate between three types of processes: reactive processes, sequencing processes, which keeps memories of the past, and deliberative processes, which make predictions about the future[Gat, 1998]. A second tripartite classification of processes in hybrid systems differentiates between a servo layer, a subsumption layer, and a symbolic layer [Connell, 1992].

We believe that it is more enlightening to look at behaviours as forming a continuum of increasingly memory dependent processes from completely reactive to heavily dependent on internal representations. The decoupling of processes on different levels that define hybrid systems reflects the old philosophical duality between mind and body.

Our work provides a unifying framework that can erase the implementational differences between the sub-parts of a hybrid system by expressing all these functions in a schema theoretic framework. This difference between a hybrid and schema based approach, however, would be purely implementational. A reimplementation would not provide increased functionality, but a unified paradigm can be

advantageous for avoiding the brittleness of human translations between low level internal concepts and high-level representation.

8.2.3 Evolutionary Robotics

Evolutionary Roboticists assume that the cognitive machinery necessary for intelligence is too complex for humans to design. Instead they try to use ML as an aid in the construction of intelligent systems and not as a part of the intelligent system itself.

The layer-wise approach lends itself intuitively to evolutionary approaches. The area known as Evolutionary Robotics uses evolution-based search techniques to develop BB controllers [Cliff et al., 1993, Nolfi and Floreano, 2000].

The problem of behaviour integration still persists within this field however. When traditional ML methods based on evolution [Goldberg, 1989] have been used to develop a controller that solves a given problem, developing the controller further can cause serious degradation of the controller's performance on the initial problem [Nolfi and Floreano, 2000].

Evolutionary Roboticists [Harvey et al., 1997] have pointed out that classical models that see evolution as optimisation and intelligence as computation are insufficient for designing intelligent behaviour. As a response they have developed new models of cognitive machinery such as Dynamic Recurrent Neural Networks (DRNNs) and new methods for artificial evolution, including competitive co-evolution.

The problems considered by Evolutionary Roboticists however, are problems of perception [Harvey et al., 1994], of navigation [Yamauchi and Beer, 1995] and of propulsion [Ijspeert, 98]. These problems are no more complex that the problems engineered solutions handle, such as topological mapping [Matarić, 1992] and simulated soccer [Balch, 1999]. The predicted disadvantages of a design approach [Nolfi and Floreano, 2000] are still unverified.

8.3 Adaptation and Learning

The area of AI that concerns itself with learning has traditionally been called Machine Learning (ML) [Mitchell, 1997]. The classic definition of ML theory [Anthony and Biggs, 1997] is the creation of a hypothesis from a set of training examples and the classification of new examples according to that hypothesis. Later alternative views on learning distanced themselves from this traditional view, e.g. Artificial Neural Networks and Evolutionary Robotics.

There are two general ways of using ML which are not always clearly separated. One is the scientific study of learning as a phenomenon. The other is the use of learning as an engineering tool. Learning is a capability found in animals, including humans. Reproducing this capability is a part of the AI effort and thus, models of learning are models of biological intelligence. ML can, however, also be used to produce or synthesise programs automatically. This use of ML is software engineering (SE). Science and Engineering merge when ML is used to synthesise models of learning.

8.3.1 Machine Learning in Robotics

ML can be applied to RLP in two different ways. One way is to cyclically develop new controllers from old ones based on the performance of the old controllers on a target application. Later, the new controllers are tested and their performance provides data for further development. The controllers developed can be adaptive or not. This form of learning is called off-line learning.

The second way of applying ML to RLP is to embed adaptive mechanisms into controllers to allow them to learn from experiences and adapt their behaviour during their lifetime. This form of learning is called on-line learning.

8.3.2 Off-Line Learning

Many of the algorithms that are used to develop BB controllers are off-line. These are generally Genetic Algorithms [Goldberg, 1989]. Genetic Algorithms have been used to evolve the connection weights of an ANN that was controlling a robotic hexapod [Lewis et al., 1992] and for visually guided orientation, object discrimination and pointing [Gallagher and Beer, 1999].

This dissertation describes work on adaptive robot behaviours. As such we are not primarily interested in off-line learning methods unless these methods produce adaptive controllers, as was the case with some of the work in Evolutionary Robotics. Some genetic algorithms, however, have been developed for on-line learning [Steels, 1994]. Meme-theory [Dawkins, 1976] makes a strong case for the existence of evolutionary processes supporting cognitive development on a certain level of abstraction. The level of the problem considered by the work on on-line Genetic Algorithms however is much lower than the one used in relation to memetics.

8.3.3 On-line Learning

The most common ML technologies in robotic behaviour learning are Reinforcement Learning [Kaelbling et al., 1996, Sutton and Barto, 1998] and Artificial Neural Networks [Hertz et al., 1991]. These somewhat overlapping technologies propose solutions to intrinsic problems in learning from a robot point of view such as problem representation and credit assignment. Learning the value of possible actions in different states or Q-learning is by far the most common problem representation and Temporal Difference Learning is a common way of solving the credit assignment problem [Russel and Norvig, 1995].

8.4 Learning in Behaviour-Based Systems

The BB paradigm facilitates learning for robots by providing behavioural layers which naturally restrict the learning problems. It also allows layers to use abstracted data from underlying layers as input to their learning algorithms. Correspondingly top-down learning biases are available from higher layers.

In this Section we review and evaluate work that has been carried out on learning in BB systems and discuss its relevance to and implications for our work.

Arkin [Arkin, 1998] dedicates a chapter of his book on BB robotics to learning. In addition, Matarić has recently done a review of the current state of behaviour-based learning [Matarić, 2001b]. A review of connectionist learning in BB mobile robots has also been done by Rylatt, Czarnecki and Routen [Rylatt et al., 1998].

8.4.1 The Scope of Learning in Behaviour-Based Systems

BB systems define a particular environment for learning, with a number of given biases on the possible solutions. Here we look at the scope for learning in BB systems and at different elements that have been learnt in previous work.

Formal Analysis of Behaviours and Learning Arkin's [Arkin, 1998] analysis of BB systems defines a behaviour as a mapping β, between a stimulus domain, $\mathbf{s}$, and a range of responses, $\mathbf{r}$ /proviexpressed in Equation 8.1.

$$\beta(\mathbf{s}) \mapsto \mathbf{r} \tag{8.1}$$

Generalised for a behaviour, a particular response $\mathbf{r_i}$ is defined by the behavioural mapping, β_i of the relevant states $\mathbf{s_i}$ multiplied by the related gain values, g_i, an activation strength measure. This definition is presented in Equation 8.2.

$$\mathbf{r_i} = g_i * \beta_i(\mathbf{s}_i) \tag{8.2}$$

In this framework Arkin suggests that a number of relations can be learnt:

1. The stimulus, $\mathbf{s}$, for a given response, $\mathbf{r}$.

2. The response, $\mathbf{r}$, for a given stimulus, $\mathbf{s}$.

3. The mapping β_i for the given stimulus domain and range of responses.

4. The magnitude of the response, g_i.

5. New behaviours, new stimuli or responses.

For a set of behaviours Arkin defines the global response ρ in terms of a coordination function $\mathbf{C}$ that arbitrates a set of behaviours $\mathbf{B}$ working on a set of stimuli $\mathbf{S}$ where the behaviours have a set of related gains $\mathbf{G}$. The relation is expressed in Equation 8.3.

$$\rho = \mathbf{C}(\mathbf{G} * \mathbf{B}(\mathbf{S})) \tag{8.3}$$

In this context Arkin suggests learning the following:

1. The subset of β_i to be included in $\mathbf{B}$.

2. The relative strengths in $\mathbf{G}$.

3. The coordination function $\mathbf{C}$.

Matarić [Matarić, 2001b] emphasises that, rather than trying to attain asymptotic optimality over a system's lifetime, situated learning, in particular multi-robot learning, should focus on improved efficiency on a shorter time scale. Matarić also suggests other ways of learning in BB systems, in particular learning behaviour policies, learning models of the environment, learning models from the behaviour activation history and learning from humans and other agents/robots.

Below we review work done on learning in BB systems and relate this work to the two taxonomies discussed above.

8.4.2 State Discrimination and State-Action Mappings

Some of the founding work on using learning in BB systems [Asada et al., 1995, Mahadevan and Connell, 1991, Maes and Brooks, 1990] applied tabula rasa learning or learning from scratch to BB architectures. Later work on learning state discrimination and state-action mappings goes beyond the initial tabula rasa learning [Millan, 1994] and describes methods for integrating pre-programmed behaviours and adaptive mechanisms.

In a general robot learning context work has studied learning hand-eye coordination in reaching [Cooperstock and Milios, 1993, Li and Ogmen, 1994]. Below we review work by Millán that touches on one of the precursors of such learning in BB systems by producing actions with stochastic variations.

Formally, learning state-action mappings generalises to cover all other forms of robot learning, but in the BB sense of the term, this kind of learning implies state-action mapping within a behaviour. The strict use of the word behaviour as an encapsulated structure that is a member of a hierarchy of behaviours is not reflected in neurological and biological texts on learning. There are however biological mechanisms for course-grained high-level behaviour control such as the hormones of the endocrine system [Carlson, 2000] and the related emotional states [LeDoux, 1998]. There is also evidence for neural structures that recognise high-level intentional behaviours in other animals.

We discuss the learning of intra-behavioural mappings here and discuss higher level control structures below as a part of learning behaviour coordination.

Learning to Push Boxes In experiments performed by Mahadevan and Connell [Mahadevan and Connell, 1991] on learning box-pushing, a robot learns a mappings, β_i from a set of stimuli represented as an eighteen bit string encoding sonar and infrared data to five different responses; go forward, turn left, turn hard left, turn right and turn hard right. Three different mappings are learnt for three independent behaviours, finding a box, pushing a box and getting un-wedged. There

is a pre-programmed order between the behaviours involved which solves the integration issues. In Arkin's framework this corresponds to pre-setting the relative activation strengths in $\mathbf{G}$.

Mahadevan and Connell use Temporal Difference learning for temporal credit assignment and demonstrate two different methods for structural credit assignment; weighted hamming-distance and statistical clustering. Weighted hamming distance generalises the input states using the hamming distance between the input bit strings as a measure of similarity. The statistical clustering method improves on the weighted hamming distance method by keeping only statistical data about observed input states to define generalised input states. These statistically defined areas of the input space are called statistical clusters.

Mahadevan and Connel's work introduces a clean efficient model of structural credit assignment which is still influential on models of learning in BB systems. As one of the first papers on learning in BB systems, its influence has been significant. It does however ignore many important aspects of learning in a BB setting that has later been handled by work building on these results.

An integrated framework for learning in BB systems must deal with dynamic input generalisation and Mahadevan and Connel's work presents one of the cleanest algorithms for achieving this. An alternative dynamic algorithm is presented below in the review of work done by Millán.

Learning to Shoot a Football at Goal Asada, Noda, Tawaratsumida and Hosoda [Asada et al., 1995] present work on applying reinforcement learning to vision based systems. Vision sensors are more complex and computationally expensive than the sonars used by Mahadevan. Asada *et al.* present a robot that learns to shoot a ball into a goal using Q-learning and discuss the related problem of 'state-action deviation'. The ball image is divided into nine sub-states by design and the goal image into 27 sub states, giving a total of 319 states. Due to this coarse externally-defined state space, moving in the environment can cause a change in

state or not. This deviation causes problems for the convergence of the learning. The solution presented is to redefine an action as a sequence of identical actions that lead to a state change, thus de-facto defining a behaviour. The action set used is forward, stop and backward for the two wheels independently, creating an action set of size 9.

However, the problem considered in Asada *et al*'s work does not contain conflicting goals and hence does not give rise to the action selection problems that can cause disastrous interference for learning. The relative triviality of this aspect of the learning problem belittles the claims of the authors that their work goes beyond approaches that divide the learning problem into pre-ranked sub-problems such as the work by Mahadevan *et al.*.

Learning Instinct-Rules for Navigation Nehmzow, Smithers and McGonigle also present work that demonstrates learning of an increasing behavioural repertoire [Nehmzow et al., 1993]. However, this work learns instinct-rules rather than behaviours. An instinct-rule associates a dedicated sensor with an action that forces that sensor to keep a certain state, and is expressed by imperatives such as 'Keep forward motion sensor on!' and 'Keep whiskers straight!". A single associative memory structure in the form of a two-layered feed forward artificial neural network is used to create the instinct-rules.

Their input vector contains seven bits describing the state of three whiskers and a move-forward sensor. Their action set has four actions; forward, backward, left and right. Using this structure, the robot learns incrementally the instinct-rules to move forward, to avoid obstacles, to follow walls and to follow corridors.

The obvious limitation to this kind of learning is the number of sensors the robot has. Abstract behaviours can encapsulate overlapping combinations of inputs and actions, something that allows for a much greater number of learnable combinations. Above we discussed learning of state-action mappings without regard to behaviour interaction as intra-behavioural learning. The simplicity of the

learning of instinct-rules places it alongside intra-behavioural learning.

Learning Avoidance Gaussier and Zrehen [Gaussier and Zrehen, 1994] present a robot that learns to avoid walls using two neural networks. One is used to topologically map the input state and another is used to associate the input states with actions. The work was carried out on a Khepera robot.

Data from eight distance sensors was pre-processed into three binary maps representing contours, walls and holes. The inspiration for this mapping was the contour extraction capabilities of the mammalian retinas as well as the off-centre and off-surround cells also found in mammalian retinas.

This work has a strong cognitive modelling slant in that it implements models of several attributes of mammalian retinas. These models are beautifully implemented and helpful to solving the learning problem. The complexity of the learning done however is restricted by the simplicity of the wall avoidance problem and the lack of a behavioural context.

Learning Modular Neural Networks for Navigation Millàn has presented a robot which learns to navigate corridors by using an initial set of basic reflexes and which improves its performance by dynamically adding nodes to its two layer fully connected artificial neural network architecture [Millan, 1994]. The nodes in the first layer of the network are modules containing localised receptive fields or exemplars corresponding to fields in the input space similar to Mahadevan's statistical clusters. The modules produce one out of 16 directions or prototypical actions.

The second layer of the architecture is a stochastic action unit that produces actions according to the direction given by the first level module that best describes the current input state. The direction produced by the action unit has a given variation for experimental purposes. The module chosen is the one with the highest expected reward. The expected reward is reflected in the weights of the neural network. These are updated using a modified form of temporal difference learning.

Reinforcement is received after every action.

New exemplars are added when no existing module covers the perceived situation. If the exemplars receptive field overlaps the receptive field of an existing exemplar and that exemplar's module has the same prototypical action as the basic reflex that is activated by the current input state, the new exemplar is added to that module. If no such module exists, the new exemplar is added to a new module with the relevant prototypical action.

Millàn's work extends initial work on learning reactive behaviours using reinforcement learning by allowing pre-programmed default behaviour and using the default behaviour to bias the learning of new reactive behaviours. Like Mahadevan's work it implements sense generalisation and also goes some way toward adaptive actions by using a probability distribution rather than a fixed action value. The probability distributions are not adaptable however, and learning does not influence the action values distribution in any other way than by a general increase of the variance. Only the state-action value mappings are changed through learning.

A major limitation of all of the work presented above is that it demonstrates learning with relation to small specific problems only, although the explicit aim is to use learning as a SE tool and a replacement for programming.

8.4.3 Learning Behaviour Coordination

The interaction between different behaviours that form a hierarchy of sense-action mappings is less well studied than the basic intra-behavioural sense-action association learning described above. The subsumption architecture suggested inhibition as one of the main methods of behaviour interaction, but it is also necessary for high-level behaviours to be able to activate lower level behaviours and a certain amount of information must also be passed through a reasonable behavioural hierarchy vertically from layer to layer. This was recognised in the early days of BB programming and tools such as the Behaviour Language [Brooks, 1990] contains

mechanisms for implementing such inter-behavioural control and data flow.

The inter-behavioural control and data flow vastly complicates the problem of behaviour coordination learning. In animals and humans the forms of learning that are able to produce novel behavioural patterns such as operant conditioning, skill learning and imitation, mostly take place as part of high-level goal oriented behaviours that provide the necessary behaviour integration structures and learning biases. Our thesis is that it is not feasible to learn novel behaviours without first providing the supporting biases of such a behavioural context. Of the work presented below, only Nicolescu goes some way toward implementing a high-level contextual framework for behaviour learning rather than general methods of behaviour adjustment.

The work presented below is far removed from current cognitive models and relies on complex algorithms expressed in the form of conventional programming constructs and abstract data structures. A generalisation and reformulation of the main ideas in these learning models in more connectionist terms would force the creation of new such models describing interacting networks and the interaction between, control flow, data flow and memory. Such models would provide important additions to current cognitive models.

Learning to Walk Maes and Brooks' robot [Maes and Brooks, 1990] learns to coordinate the swing forward behaviours of six legs by learning activation with reference to the binary 'up or down' state of the other legs. In Arkin's framework this amounts to learning $\mathbf{C}$ as a property of the activation of the individual β_i. To do this, the behaviour activation information must be included in the set of stimuli $\mathbf{s}$. The learning algorithm used in this work is distributed over the different behaviours.

The algorithm Maes uses calculates simple correlations between the elements of the input state and the reinforcement for each behaviour in order to decide which elements are relevant locally to achieve the overall reward.

Maes provides an early method for learning behaviour integration with clear connectionist parallels. This mode of expression is facilitated by a very low level concept of a behaviour, the movement of a leg up and forward, and an absence of a wider behaviour context or hierarchy.

Learning Foraging by Modelling Behaviour Activation Michaud and Matarić [Michaud and Matarić, 1999] describe work that uses the history of executed actions to improve the performance of a robot on a general foraging task. The motivation for modelling internal activity through behaviour activation rather than external data through sensors and actuators is that highly dynamic environments containing multiple robots are significantly more complex to model than static environments due to the fact that topological and metric models do not directly apply in such environments. This affects not only the robot's chances of learning useful models, but also a designers ability to express useful a priori models.

By keeping traces of the behaviours activated every time a task is performed, their robots build up a tree-structure which models the different behavioural paths taken to achieve the task. Michaud and Matarić divide their behaviours into three categories, task-behaviours, maintenance-behaviours and alternative-behaviours. The task-behaviours are pre-programmed solutions to achieve the foraging task. The maintenance-behaviours are task independent behaviours that deal with harmful situations and interference e.g. obstacle avoidance. Alternative-behaviours are a set of general behaviours with no a priori activation conditions. These are used experimentally by the adaptive algorithm to improve the overall performance. An overview of the architecture with the different behaviours is reproduced in Figure 8.1.

The adaptive algorithm expresses previous trials as different branches of a tree. It compares the average task completion time from previous trials, with the estimated completion time of the branch of the tree it currently places itself in according to its active behaviours and recent execution history. If the comparison is

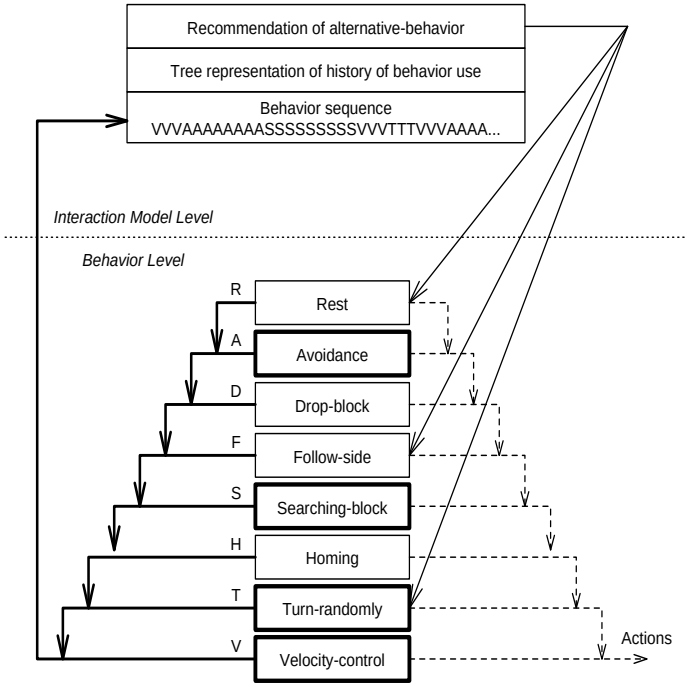


Figure 8.1: Learning Foraging, from [Michaud and Matarić, 1999]

favourable it does not recommend any changes in behaviour. If not, it recommends trying a better behaviour from either the tree model or a global options table or it experiments by recommending an untried behaviour. This system is able to learn beneficial action sequences relatively independent of external stimuli.

This work also integrates pre-programmed and adaptive behaviours. There is however no attempt to try to express the tripartite model of behaviour in terms of a general model. The specificity of the tripartite model limits the general application of the algorithm and its integration with existing models of behaviours by the design constraints the tripartition imposes.

The work presented by Michaud and Matarić offers a unique model of chaining, a pivotal form of animal learning called chaining. Chaining occurs when animals learn *sequences* of actions [Pearce, 1997]. This work includes recent actions as a part of the state space to be mapped to behaviours in learning behaviour coordination. The ability to include recent actions in state space seems to be present in most animals and is an important part of a general model of behaviour learning and integration.

Learning Efficient Mine Clearance by Tracking Behaviour Activation Times Another technique for behaviour structuring based on models of internal behaviour activation is presented by Goldberg and Matarić [Goldberg and Matarić, 2000]. This system keeps statistical information about the duration of the active periods of existing behaviours in an extended behaviour-based architecture called an Augmented Markov Model.

This architecture is demonstrated on a mine clearing problem where there are two kinds of mines, small and large, with different reward values. By comparing the times spent searching before finding different mines, this architecture allows the robot to estimate the time it will have to spend to find a large mine and compare the cost of this time with loss in reward associated with picking up a small mine. The robot thus learns whether to pick up small mines or not.

This work addresses an important general question about dynamically assigning quality measures to behaviours.

8.4.4 Learning World Models

There is neurological evidence for the presence of explicit world models in animals, such as the dedicated place cells in rats [O'Keefe and Conway, 1978]. There is also evidence from animal studies [Gillan, 1981] showing learning of preference relations between different types of food, and use of transitive inference over these structures.

A model of the world is a set of assumptions. However, assumptions held by animals or robots can be so general that the term 'model' is misleading. All reactive behaviours are based on implicit assumptions about the world, and all learning makes assumptions in the form of generalisations. We call such non-explicit assumptions, *implicit world models*.

Below we present two different approaches to learning an explicit model of the world. The first is highly specialised while the second is general.

Learning Maps and Paths Matarić's mapping robot Toto [Mataric, 1990] does spatial learning by embedding a spatial representation and related mapping and planning algorithms in a behavioural context. The adaptive mechanism constructs a graph dynamically. Each node in the graph is a landmark implemented by a dynamically created dedicated behaviour. The landmarks are connected by motion directives corresponding to the motion the robot executed to reach the particular landmark.

Each landmark behaviour continuously broadcasts distance data to its neighbouring landmarks. The receiving landmarks add to these distance messages and hence create path descriptions. The currently active landmark behaviour, representing the current position of the robot, receives messages originating from all the other landmark behaviours on every cycle. The sum of distances in the differ-

ent messages indicate possible path-lengths to every other landmark, allowing the robot to always pick the shortest path without having to re-plan if the robot strays from the desired path.

This kind of targeted, optimised learning is, according to our thesis, the only way of doing complex learning efficiently.

Learning Homing Chesters and Hayes [Chesters and Hayes, 1994] have presented work on a Lego robot that learns to avoid collision and find a given location by using two neural networks, one called the model network which learns a state transformation model of the world and one called the heuristic network which learns state values.

A framework called the policy uses the two networks to implement an adaptive controller. The model network is trained on each step to learn to associate the current state with the previous state. The heuristic network is trained to value a state at the value of the best state the model can reach in one step from the current state with the value discounted at a constant rate. The model network effectively provides pseudo-experiences. The work experiments with three different networks for modelling state transformation: one is a three layer ANN with fourteen input nodes and fourteen hidden nodes, the second is a three layer ANN with 28 input nodes and 28 hidden nodes that takes both the current state and the previous state as input, thus being able to learn significant relationships over sequences of two states. The third network is a recurrent ANN with fourteen input nodes, fourteen hidden nodes and fourteen context nodes. The recurrent ANN is shown to be the most efficient for learning the state transition model due to its ability to remember significant relationships over longer sequences of events.

Though this work uses a biologically plausible, connectionist learning model, it also uses a general, unstructured approach to model learning; something that is unlikely to be found in natural systems due to the complexity issues that are raised by the lack of bias in such systems.

The problem considered is a very low-level collision avoidance problem without a behavioural context. Complexity-wise, this learning problem is similar to the intra-behaviour learning problems we discussed above in the beginning of this Section. It is not made clear why a complex learning approach like model learning is necessary for such a problem.

8.4.5 Learning from Observation and Communication

There are many unresolved issues concerning what supporting circuitry is necessary to learn behaviour patterns from communication. Here we present work on implementing this kind of learning in robots.

Learning Task Distribution Learning to optimise task distribution in general has been studied by Parker [Parker, 1997]. This work uses expectations of task completion times to rearrange tasks in heterogeneous teams by having robots taking over tasks or acquiesce according to estimates of their own and others' efficiency with regards to specific tasks.

This work trivialises the learning part of the task distribution problem by providing overly specific support structures. Only the time taken for another robot to achieve a task is learnt, everything else is pre-programmed. This is a highly optimal approach, but it says little about the unsolved issues of behaviour integration and learning. Though efficiency and optimality is important, the problem considered in this work is so specialised that the solution has little applicability to other problems.

Learning from Imitation Nicolescu and Matarić [Nicolescu and Matarić, 2001] use the BB framework to implement learning by observation and imitation. This is made possible by a behaviour representation that includes pre- and post-conditions for abstract behaviour construction. This method has also been extended to humanoid imitation [Jenkins and Matarić, 2000].

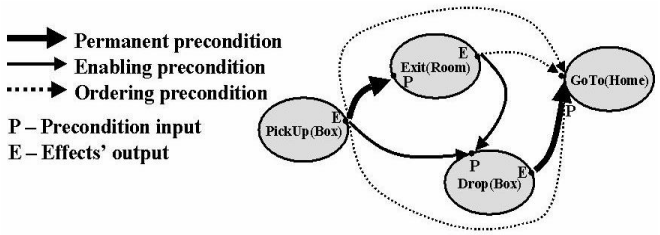


Figure 8.2: Behaviour Network, from [Nicolescu and Matarić, 2001]

This work extends common behaviour representations in order to handle re-use of behaviours and structuring of primitive behaviours into abstract behaviours. The abstract behaviours in Nicolescu's work, called behaviour networks, order primitive behaviours into novel configurations that allow the robots to solve new tasks, relying on the primitive behaviours to implement the actions taken in the real world.

The abstract behaviours are concerned with traditional issues of symbolic planning such as matching pre- and post-conditions and variable binding. Each behaviour has pre-defined activation conditions or pre-conditions and effects or post-conditions. The pre- and post-conditions are defined on the input state. The pre-conditions are divided into permanent pre-conditions, enabling pre-conditions and ordering pre-conditions. The permanent pre-conditions must be true during the entire activation of a behaviour. Enabling pre-conditions must be true at the moment of activation and ordering pre-conditions must have been true at some point before behaviour activation.

A new behaviour network is constructed by following a demonstrator and remembering the intervals for which the effects or post-conditions of the basic behaviours were satisfied. For each interval in this list, an instance of the related behaviour is added to the new behaviour network and the different behaviour instances are then related to each other by means of the different types of pre-conditions according to the temporal relations between the recorded intervals. An example behavioural network is reproduced in Figure 8.2.

Nicolescu suggests a way of using the post conditions for doing backward plan-

ning from a goal. The abstract behaviours have also been used for imitation learning where the goal is to copy observed behaviour. The post-conditions are abstractions of world states and as such are used as a mechanism for recognising world states demonstrated by a teacher. The recognised states are merged into a task representation that can later be used to execute the demonstrated task.

8.5 Problem Division

There have been several attempt at reducing the complexity of learning in robotic systems by dividing the problem space into sub-problems. Some of this work has attempted to find solutions to all the sub-problems concurrently, while some has tried an incremental approach.

8.5.1 Integrated Learning

Stone and Veloso [Stone, 2000, Stone and Veloso, 1999] have developed a set of principles for dividing complex learning problems into multiple smaller interacting problems. Starting with low-level sub-tasks they build up ML new layers of sub-tasks until they reach the high-level task that deals with the full domain complexity. Different, suitable ML techniques are then used on the different sub-tasks individually. Stone and Veloso identify three ways that sub-tasks can affect each other:

1. By providing training examples.
2. By providing features of examples.
3. By pruning the output set.

This approach is applied to the domain of simulated robot football where they identify three different sub-tasks: ball interception, pass evaluation and pass selection. These sub-tasks relate to three different contexts: single agent behaviour, multi-agent behaviour, and team behaviour.

8.5.2 Incremental Learning

Shaping One form of incremental learning which relies heavily on a teacher or supervisor is *shaping*. During shaping, an individual is rewarded for solving increasingly larger parts of a problem. This allows the entity to prune the search space by ignoring alternatives to the actions that have been rewarded and focus experimentation on actions following the last action that produced a reward.

Asada *et al.* [Asada et al., 1995] describe a methodology based on shaping called learning from easy missions (LEM). LEM is used as a way to overcome the delayed reinforcement problem rather than the subtask decomposition used by Mahadevan. This takes the work by Asada *et al* beyond tabula rasa learning.

Dorigo and Columbetti [Dorigo and Colombetti, 1998] have performed a series of experiments on shaping in a simulated predator/prey scenario and on a real robot learning to approach a light source. They use an automated reinforcement program to provide feedback to the learning system and argue that it is often easier to implement such a program than to implement the desired behaviour directly. The experiments evaluated different controller architectures, different shaping policies and different memory structures. On the real robot they also studied the effects of degraded capabilities such as a blind eye or inverted eyes, or incorrectly calibrated or inverted motors.

Dorigo and Columbetti's also study the effect of short term *sensor memory* and the learning of sequential behaviours using a *state word*, i.e a memory of what action was last taken.

The general conclusion from Dorigo and Columbetti's work is that the optimal system used a hierarchical control architecture where each low level behaviour was trained independently and a coordinating behaviour was then trained on top of these. Dorigo and Columbetti called this kind of shaping *modular shaping*.

Scaffolding Breazeal and Scasselati [Breazeal and Scassellati, 1998] have developed a learning strategy called learning by *scaffolding* that is dependant on the

presence of a human 'caregiver'. This learning strategy is inspired by the way infants learn from interaction with their parents.

Breazil and Scasselati present a controller that has as its goal to maintain a set of drives such as *socialness* and *curiosity* within given homoeostatic bounds. If the drive values go outside the given bounds, they potentiate related emotions. If the *socialness* drive falls below the lower bound, it potentiates the *lonely* emotion. These emotions are called *primary emotions*. The combination of active emotions activates motors to produce an *expressive state*. Each drive also has a related *consummatory* behaviour. In the case of the *socialness* drive, the consummatory behaviour is *socialise*, in the case of the *curiosity* drive, it is *play*. The consummatory behaviours are also potentiated by the drives, and when they are, they can be activated by an environmental stimulus such as a face for the *socialise* behaviour and a gently moving non-face object for the *play* behaviour.

Breazil and Scasselati have demonstrated that the drives can be kept within the homoeostatic bounds by the correct interaction between the robot and a caretaker. Breazeal and Scasselati have also demonstrate formation of *emotional memories* or *secondary emotions*, i.e. associations between environmental stimuli and the primary emotions. These associations allow environmental stimuli to activate the emotions and their related behaviours.

The intended use of this ambitious framework is to demonstrate incremental learning as observed in infants that interact with their caregivers. By recognising the robots emotional state, the caregiver can reward the robot for certain actions. In this context reward implies activating the consummatory behaviour of a drive outside its homoeostatic bounds as indicated by the expressive state, e.g. waving a toy in the robot's visual field when it looks bored. To the robot, the rewarded actions become alternative behaviours for maintaining the drives within the homoeostatic bounds. Such alternative behaviours are called *appetitive behaviours*. By creating appetitive behaviours, the caretaker can teach the robot new consummatory behaviours, i.e. teaching it to take actions that will cause the environment to acti-

vate the existing consummatory behaviours. So far however, only the learning of secondary emotions has been reported.

8.6 Integration of Programmability and Learning

Much of the early work on learning in BB systems focused on learning a 'substrate from which genuinely complex forms of behaviour can ultimately emerge.'. This was effectively learning from scratch, and as such the issue of incremental development was ignored. The behaviour integration mechanisms were emphasised by Brooks as a strength of the subsumption architecture [Brooks, 1986]. The non-incremental approach was defended by the conclusion that subsumption-style architectures: '...will fail to scale up in the longer term because they are not truly adaptive' [Rylatt et al., 1998].

The BB paradigm has been extended since Brooks introduced it and now includes systems with more general use of state and more complex behaviour interaction models [Matarić, 2001b]. As a result, BB systems can be truly adaptive. Stateless systems are generally referred to as reactive. The inclusion of state in behaviours was the first step in integrating pre-programmed and adaptive elements of behaviour.

Work on programmable reinforcement learning units [Parr and Russel, 1998, Andre and Russel, 2001] and on hierarchical memory-based reinforcement learning [Hernandez-Gardiol and Mahadevan, 2000] provides interesting results on a different approach to the question of merging programmability and adaptability in BB systems. That work makes the learning mechanisms programmable rather than making the programs adaptable as we do in our approach. In connectionist models these distinctions are less distinct as both programming and learning are cases of adjusting weights on connections.

Chapter 9

Discussion, Conclusions, Future Work and Summary

Contents

9.1 Discussion

9.1.1 The Problem Complexity Level

We see our work as a first step in looking at implementing complex forms of learning through a process of small additions to simpler forms of adaptation and learning. In order to ground the development process we had to first implement the simplest forms of learning. This led to a relatively low level of complexity in the learning problems implemented and kept us from implementing higher forms of BBL. Implementations of robust and efficient high level learning would provide crucial evidence in support of our thesis.

The problem with focusing too strongly on relatively simple learning problem is that they say little about the main question in this thesis, whether robust and efficient high level learning can be step-wise implemented. In order to answer this question it is necessary to look at forms of learning that don't have immediate, obvious solutions. The problems we looked at in Section 6 did not have obvious solutions, but also only used relatively simple forms of learning. Looking at problems that are closer to the human level forms of learning traditionally considered by traditional ML approaches would have provided stronger support for our thesis.

Some learning problems that would have provided such support, e.g. operant conditioning, skill learning or imitation are discussed as suggestions for future experiments on higher forms of learning in Section 9.3.

9.1.2 The Missing Behavioural Dimension of Procreation

The learning problems chosen for our implementations and designs do not touch on procreative behaviour. Feeding, fighting, and fleeing are all covered, but neither the implementations nor our designs touch on the fundamental behaviours related to reproduction.

According to our holistic approach presented in Section 7, high level learning facilities are used to solve problems from all the fundamental behavioural dimen-

sions. It is therefore important to look at the basic adaptive functions in all these dimensions to study how complex learning has evolved and to ensure that all the supporting behaviours needed for complex learning are present.

Even though the exclusion of the procreation dimension has no immediate effect on the results of the experiments or the analysis, it must be added later to complete the foundations for the further work suggested in Section 9.3.

9.2 Conclusions

In this Section we review our contribution in a larger context, evaluate its implications for the greater context of the related research areas and its significance and limitations.

Current Weaknesses For the approach described in this dissertation to make an impact on the general approach to learning in BB systems, it is necessary to demonstrate high level learning capabilities in robots that are clearly more robust and efficient than what can be achieved using existing methods. This would indicate a major step along the suggested path to BBL.

The current tools are promising but immature. The suggested models are also suffering from an incomplete exposure to existing theories of animal behaviour.

Our work on learning is a part of a greater move away from the top-down thinking of GOFAI and toward the bottom up gospel of BBAI. As such it is the result of a trend in robotics. By addressing issues in the field of ML as it relates to robotics and autonomous agents our work sets the agenda for still unconvinced researchers in this field to follow the general trend in AI.

9.2.1 The BBL Models Provides a Feasible Approach

The BBL models of animal and human learning presented in Section 4.4 provide a possible if incomplete approach to robust and efficient implementations of high level learning.

The step-wise approach of BBL mirrors the incremental development approach that is one of the strengths of BBAI. The major challenge is whether efficiency and robustness can be carried over between the increasingly complex forms of learning. Our implementations show that this is possible for simple forms of learning.

9.2.2 The NC Model Simplifies Behaviour Design

Our implementations of BB controllers show that the NC model of behaviours and learning is a good tool for designing easily implementable BB solutions. Its modularity is particularly well suited for designing behavioural layers incrementally as suggested originally in relation to the subsumption architecture.

Our work also demonstrated that the NC model encapsulates the right entities for simple BBL by presenting the two high level concepts of senses and competences to the related learning circuitry.

9.2.3 The PLANCS Classes Facilitate Implementation

The PLANCS classes have been successful in facilitating the implementation of NC models. Most circuits are implemented with a minimal addition to either network or neural component classes.

By supporting independent compilation and execution of both layers and circuits the PLANCS classes allows incremental development of behavioural layers, but also goes further in supporting incremental development of each layer by allowing incremental development and testing of each circuit of a behavioural layer.

9.3 Future Work

This Section describes possible directions for future research and discusses the implication of the suggested work to our thesis; that robust and efficient high level learning can be incrementally developed in BB systems by taking inspiration from evolution.

9.3.1 Implementing Complex Forms of Learning

One of the weak points of our work is that we could not develop and implement more complex forms of BBL. Because of the need to ground our development chain argument we were forced to implement simple forms of BBL. We also followed a BB design approach where, in order to maximise learning biases for all elements of learning, we restricted the learning problems maximally and used the simplest form of learning that could possibly solve that problem. This had the unfortunate effect that solutions to complex problems such as the mapping problem and the conflict resolution problem presented in Section 6 could be implemented using only simple forms of learning. By restricting ourselves to look at only simple forms of BBL we also denied ourselves the opportunity to demonstrate higher forms of BBL.

In Section 4.4 we developed BBL models of complex forms of animal and human learning such as operant conditioning, skill learning, model learning and imitation. The most important work that needs to be done to prove our thesis conclusively is to implement more complex forms of BBL and show that these can keep the robustness and efficiency of the simpler forms.

The next step from the simple forms of association we have implemented is according to our development chain, operant conditioning, chaining and skill learning. Work on developing BBL versions of these learning forms will rid our work of its weak point of apparent triviality compared to traditional ML methods. For our integrative approach, work on unifying other forms of learning is particularly interesting. Related work on integrating many forms of associative learning

in the reinforcement learning paradigm has been presented by Sutton and Barto [Sutton and Barto, 1998].

9.3.2 Repeating Experiments in Real Robots

In Chapter 6 we discussed the problems with using robot simulators rather than real robots. A demonstration of efficient and robust learning would not be complete without being demonstrated on a real robot in a natural environment.

Although simulations go a long way toward showing that the learning methods will perform adequately there is still a difference between strong implications and proof by demonstration.

9.3.3 Expectations

In Section 4.4 we discussed a mechanism for setting up expectations and evaluating whether expectations were met. This is a central and important capability both as a hardwired solution to simple problems but also for providing negative feedback and progress estimation. Matarić [Matarić, 1994] used heterogeneous reward functions that included progress estimators. This kind of reward appears crucial for speeding up convergence in learning.

As an important part of learning and behaviour, it would be wise to implement this functionality into any agent that will be used to model high level learning.

9.3.4 Hormonal Control

The hormone-based, emotion related endocrine control system found in animals has a fundamental effect on behaviours and provides an additional bias on animal learning. A hormone-based system running in parallel with our neuron based control system would provide more general, long term influences on behaviours and activate different sets of behaviours according to which emotional state the agent was in.

Our thesis states that by taking inspiration from evolution and the cognitive sciences we can provide the biases necessary for efficient learning. The endocrine system is central to all established theories of biological behaviour control. Physiological theories differentiate between Hebbian learning and learning with a general hormone based reinforcement system. Instrumental learning requires a general reinforcement system while classical conditioning does not.

9.3.5 A Thorough Analysis of Evolution

In Section 7 we argued that it is important to develop complete sets of behaviours to make the behaviours robust enough to function in a wider behavioural space. This argument mirrors Brooks' argument for the importance of looking at situated agents to make behaviours robust enough to handle real world interaction.

As a way of defining helpful constraints on the physical and cognitive complexity of a robot, we suggested an evolutionary analysis, where co-existing entities from different behavioural dimensions were mapped out as a guide to what combinations of anatomy and cognitive ability would form appropriate architectures for implementation and research.

Our initial sketch of such a map needs refinement and exposure to criticism from existing evolutionary theories. It would benefit research by providing a focus on problems and solutions related to general intelligence. Such a focus would be beneficial to limit overspecialising technologies with respect to increasingly artificial problem domains.

9.3.6 Duplication Experiments

As discussed in Section 4, evolution works through duplication and specialisation. Duplication happens on many different levels and copies all parts of the physical body including neural circuits.

In Section 4 we presented a behaviour design method called ordered circuit addition based on copying general memory circuits on top of existing circuitry and

subsequently refining the behaviour of those circuits.

There is no reason for restricting the use of duplication in layer design to any particular type of memory circuit. As far as our knowledge of evolution goes, it would be equally valid to duplicate and specialise any other circuit, collection of circuits, layer or collection of layers to implement new behaviours if this makes sense. We have not tried this approach but we find it a very interesting avenue for exploration.

9.4 Summary

In this dissertation we presented an evolution-inspired methodology for developing adaptive behaviours in robots. We followed the motivating example of an animal that competes with other members of its species and looked at how to solve the problem of conflict resolution that occurs when two entities want the same resource. We demonstrated how to use our methodology and supporting tools to develop increasingly sophisticated strategies for conflict resolution from problem specification to implementation on the Webots Khepera robot simulator.

To create a recurrent theme we introduced our motivating example and discussed the main issues related to implementing adaptive strategies for conflict resolution on robots: speed of adaptation, use of prior knowledge, problem restriction, robustness of solutions and integrating programmability and adaptation.

We first presented our Neural Circuit (NC) Model of behaviour. The NC model is based on schema theory and breaks a behavioural layer into a series of circuits for activation filtering and information extraction. A NC belongs in one of six general classes: sensor, sense, drive, competence, actuator and memory circuits. NCs also have a restricted interface for communicating with other circuits. We presented how to design a behavioural layer to support a conflict resolution strategy called *Uncritical Fighting* from these standard circuits.

Next we presented the guidelines that make up the Behaviour-Based Learning (BBL) methodology for developing adaptive behaviours. Again, using conflict resolution as an example, we showed how the BBL guidelines produce algorithms that use multiple solutions to the same problem to allow graceful degradation on limited hardware failure and to optimise learning speed by maximally restricting the search space of the adaptive mechanisms used. The price these algorithms pay for their efficiency is generality. The solutions produced are specific to the given problem domain and do not provide general learning capabilities.

To support implementation of BBL algorithms we presented a library of classes for Object Oriented Programming called Programmable Learning Artificial Neural

Circuits (PLANCS). The PLANCS classes provide a general implementation of NCs and allows the programmer to specify the communication interface used by a NC and also to overload the base functionality and give it an arbitrary input-output function. The PLANCS classes also abstract away the hardware model the program is executed on and as such facilitates the migration of an implemented controller do different processor architectures.

To demonstrate that our methodology and our supporting tools can produce rapidly adapting behaviours in robots we presented three sets of experiments in the problem domains of Approach Compensation, Foraging/Mapping and Conflict Resolution. The results showed that the robots consistently improved their performance when they adopted the increasingly sophisticated strategies developed using the BBL methodology, designed and implemented using the NC model and the PLANCS class library.

From our experiences with developing multiple adaptive behaviours we extracted a number of general recommendations for research in AI. We called the approach to AI described by these recommendations *Holistic* AI.

Lastly we presented work related to hours and discussed differences and similarities. We also discussed the value and consequences of our research, drew a set of conclusions, and set out possible future work.

Bibliography

[Agre and Chapman, 1990] Agre, P. E. and Chapman, D. (1990). What are Plans for? *IEEE Robotics and Autonomous Systems*, (6):17–34.

[Allman, 1999] Allman, J. M. (1999). *Evolving Brains*. Scientific American Library.

[Andre and Russel, 2001] Andre, D. and Russel, S. J. (2001). Programmable Reinforcement Learning Agents. In *Advances in Neural Information Processing Systems 13*. MIT Press.

[Anthony and Biggs, 1997] Anthony, M. and Biggs, N. (1997). *Computational Learning Theory*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

[Arbib, 1989] Arbib, M. A. (1989). *The Metaphorical Brain 2*. Wiley.

[Arbib, 2000] Arbib, M. A. (2000). The Mirror System, Imitation, and the Evolution of Language. In Nehaniv, C. and Dautenhahn, K., editors, *Imitation in Animals and Artifacts*. MIT Press.

[Arbib et al., 2001] Arbib, M. A., Alexander, A., and Weitzenfeld, W. (2001). Nsl neural simulation language. In Arbib, M. A. and Grethe, J. S., editors, *Computing the Brain: A Guide to Neuroinformatics*, pages 43–69. Academic Press.

[Arkin, 1989] Arkin, R. C. (1989). Motor Schema-Based Mobile Robot Navigation. *International Journal of Robotics Research*, 8(4):92–112.

[Arkin, 1998] Arkin, R. C. (1998). *Behaviour Based Robotics*. MIT Press.

[Asada et al., 1995] Asada, M., Noda, S., Tawaratsumida, S., and Hosoda, K. (1995). Vision-Based Reinforcement Learning for Purposive Behavior acquisition. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 146–153.

[Ashby, 1952] Ashby, W. R. (1952). *Design for a brain*. John Wiley and Sons.

[Baddeley, 1997] Baddeley, A. (1997). *Human Memory, Theory and Practice*. Psychlogy Press, revised edition.

[Balch, 1999] Balch, T. R. (1999). The impact of diversity on performance in multi-robot foraging. In Etzioni, O., Müller, J. P., and Bradshaw, J. M., editors, *The proceedings of the Third International Conference on Autonomous Agents*, pages 92–99, Seattle, WA. ACM Press.

[Balkenius, 1993] Balkenius, C. (1993). Natural Intelligence for Autonomous Agents. In *Proceedings of the International Workshop on Mechatronical Computer Systems for Perception and Action*, pages 181–189.

[Balkenius, 1995] Balkenius, C. (1995). *Natural Intelligence for Artificial Creatures*. PhD thesis, Lund University Cognitive Science.

[Balkenius, 2000] Balkenius, C. (2000). Attention, Habituation and Conditioning: Toward a Computational Model. *Cognitive Science Quarterly - CSQ*, 2(1).

[Balkenius et al., 2000] Balkenius, C., Gärdenfors, P., and Hall, L. (2000). The Origin of Symbols in the Brain. In *Proceedings of the Third Conference on the Evolution of Language*.

[Balkenius and Morén, 2000] Balkenius, C. and Morén, J. (2000). A Computational Model of Context Processing. In Meyer, J.-A., Berthoz, A., Floreano, D.,

Roitblat, H. L., and Wilson, S. W., editors, *From Animals to Animats 6: Proceedings of the Sixth International Conference on the Simulation of Adaptive Behaviour*. MIT Press.

[Beer et al., 1998] Beer, R. D., Chiel, H. J., Quinn, R. D., and Ritzmann, R. E. (1998). Biorobotic Approaches to the Study of Motor Systems. *Current Opinion in Neurobiology*, (8):777–782.

[Biederman, 1987] Biederman, I. (1987). Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147.

[Breazeal and Scassellati, 1998] Breazeal, C. and Scassellati, B. (1998). Infant-like social interactions between a robot and a human caregiver. *Adaptive Behavior*, 8(1):49–74.

[Bromley, 1998] Bromley, A. G. (1998). Charles babbage's analytical engine, 1838. *IEEE Annals of the History of Computing*, 20(2).

[Brooks, 1986] Brooks, R. A. (1986). A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, (RA-2).

[Brooks, 1990] Brooks, R. A. (1990). The Behavior Language; User's Guide. Technical Report 1227, MIT A.I. Memo.

[Brooks, 1991a] Brooks, R. A. (1991a). Intelligence without reason. In *Proceedings of IJCAI 91*, pages 569–595. Morgan Kaufmann.

[Brooks, 1991b] Brooks, R. A. (1991b). Intelligence without Representation. *Artificial Intelligence*, (47):139–159.

[Brooks, 1995] Brooks, R. A. (1995). A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, (RA-2):14–23.

[Brooks, 1997] Brooks, R. A. (1997). From Earwigs to Humans. *Robotics and Autonomous Systems*, 20(2–4):291–304.

[Bruce et al., 1997] Bruce, V., Green, P. R., and Georgeson, M. A. (1997). *Visual Perception: Physiology, Psychology, and Ecology*. Psychology Press, Third edition.

[Bryson, 2000] Bryson, J. J. (2000). Hierarchy and Sequence vs. Full Parallelism in Reactive Action Selection. In Meyer, J.-A., Berthoz, A., Floreano, D., Roitblat, H. L., and Wilson, S. W., editors, *Proceedings of the Sixth International Conference on the Simulation of Adaptive Behaviour (SAB'2000)*, pages 147–156.

[Bryson, 2001] Bryson, J. J. (2001). *Intelligence by Design: Principles of Modularity and Coordination for Engineering Complex Adaptive Agents*. PhD thesis, Massachusets Institute of Technology, Department of Electrical Engineering and Computer Science.

[Bryson and McGonigle, 1997] Bryson, J. J. and McGonigle, B. (1997). Agent Architecture as Object Oriented Design. In *Intelligent Agents IV, Proceedings of the Fourth International Workshop on Agent Theories, Architectures, and Languages (ATAL'97)*, number 1365 in Lecture Notes in Artificial Intelligence, pages 15–30. Springer Verlag.

[Bryson and Stein, 2001] Bryson, J. J. and Stein, L. A. (2001). Modularity and Specialized Learning: Mapping Between Agent Architectures and Brain Organization. In Wermter, S., Austin, J., and D.Willshaw, editors, *Emergent Neural Computational Architectures Based on Neuroscience*, pages 98–113. Springer Verlag.

[Burgess et al., 1994] Burgess, N., Recce, M., and O'Keefe, J. (1994). A model of hippocampal function. *Neural Networks*, 7(6/7):1065–1081.

[Carlson, 2000] Carlson, N. R. (2000). *Physiology of Behaviour*. Allyn and Bacon, seventh edition.

[Carter, 1998] Carter, R. (1998). *Mapping the Mind*. Weidenfeld and Nicolson.

[Chesters and Hayes, 1994] Chesters, W. and Hayes, G. (1994). Connectionist Environment Modelling. In Cliff, D., Husbands, P., Meyer, J.-A., and Wilson, S. W., editors, *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB'94)*, pages 189–197. MIT Press.

[Christensen, 2000] Christensen, H. I. (2000). The WEBOTS Competition. *Robots and Autonomous Systems Journal*, 31(4):351–353.

[Cliff, 1991] Cliff, D. T. (1991). Computational Neuroethology: A Provisional Manifesto. In Meyer, J. A. and Wilson, S. W., editors, *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior (SAB'91)*. MIT Press.

[Cliff et al., 1993] Cliff, D. T., Harvey, I., and Husbands, P. (1993). Explorations in Evolutionary Robotics. *Adaptive Behaviour*, 2:73–110.

[Collins et al., 1993] Collins, A. F., Gathercole, S. E., Conway, M. A., and Morris, P. E., editors (1993). *Theories Of Memory*. Psychology Press.

[Connell, 1992] Connell, J. H. (1992). SSS: A Hybrid Architecture Applied to Robot Navigation. In *Proceedings of the 1992 IEEE Conference on Robotics and Automation (ICRA'92)*, pages 2719–2724.

[Conway et al., 1998] Conway, M. A., Gathercole, S. E., and Cornoldi, C., editors (1998). *Theories of Memory II*. Psychology Press.

[Cooperstock and Milios, 1993] Cooperstock, J. R. and Milios, E. E. (1993). Self-supervised learning for docking and target reaching. *Robotics and Autonomous Systems*, (11):243–260.

[Corbacho and Arbib, 1997] Corbacho, F. and Arbib, M. A. (1997). Schema-Based Learning: Towards a Theory of Organization for Adaptive Autonomous

Agents. In *Proceedings of the 1st International Conference on Autonomous Agents*, Marina del Rey, California.

[Dahl, 1998] Dahl, T. S. (1998). Background Knowledge in the Tertius First Order Knowledge Discovery tool. Technical Report CSTR-99-006, Department of Computer Science, Bristol University.

[Dahl and Giraud-Carrier, 2001a] Dahl, T. S. and Giraud-Carrier, C. (2001a). Evolution, Adaption and Behavioural Holism in Artificial Intelligence. In *Proceedings of the 6th European Conference on Artificial Life (ECAL'01)*, pages 499–508.

[Dahl and Giraud-Carrier, 2001b] Dahl, T. S. and Giraud-Carrier, C. (2001b). PLANCS: Classes for Programming Adaptive Behaviour Based Robots. In *Proceedings of the 2001 Convention on Artificial Intelligence and the Study of Simulated Behaviour (AISB'01), Symposium on Nonconscious Intelligence: From Natural to Artificial*, pages 9–20.

[Dautenhahn, 2000] Dautenhahn, K. (2000). Reverse engineering of societies - a biological perspective. In *Proceedings of the AISB Symposium, Starting from Society - the application of social analogies to computational systems*.

[Dawkins, 1976] Dawkins, R. (1976). *The Selfish Gene*. Oxford University Press.

[Deacon, 1997] Deacon, T. (1997). *The Symbolic Species - The Co-Evolution of Language and the Human Brain*. Penguin.

[Dennet, 1991] Dennet, D. C. (1991). *Consciousness Explained*. Penguin Books.

[Dorigo and Colombetti, 1993] Dorigo, M. and Colombetti, M. (1993). Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence*, 71(2):321–370.

[Dorigo and Colombetti, 1998] Dorigo, M. and Colombetti, M. (1998). *Robot Shaping*. MIT Press.

[Fagg et al., 1998] Fagg, A. H., Barto, A. G., and Houk, J. C. (1998). Learning to reach via corrective movements. In *Proceedings of the Tenth Yale Workshop on Adaptive and Learning Systems*, pages 179–185, New Haven, CT.

[Foner and Maes, 1994] Foner, L. and Maes, P. (1994). Paying attention to what's important: Using focus of attention to improve unsupervised learning. In Cliff, D., Husbands, P., Meyer, J.-A., and Wilson, S. W., editors, *From Animals to Animats: Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB'94)*, pages 256–265, Brighton, UK. MIT Press.

[Frezza-Buet et al., 2001] Frezza-Buet, H., Rougier, N., and Alexandre, F. (2001). Integration of biologically inspired temporal mechanisms into a cortical framework for sequence processing. In Sun, R. and Giles, C. L., editors, *Sequence Learning: Paradigms, Algorithms, and Applications*, LNAI 1828, pages 321–348. Springer.

[Fuhs et al., 1998] Fuhs, M. C., Redish, A. D., and Touretzky, D. S. (1998). A Visually Driven Hippocampal Place Cell Model. In Bower, J., editor, *Computational Neuroscience: Trends in Research*, pages 101–106. Plenum Publishing.

[Gallagher and Beer, 1999] Gallagher, J. C. and Beer, R. D. (1999). Evolution and Analysis of Dynamical Neural Networks for Agents Integrating Vision, Locomotion and Short-Term Memory. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 1273–1280.

[Gallistel, 1990] Gallistel, C. R. (1990). *The Organization of Learning*. MIT Press.

[Garcia and Koelling, 1966] Garcia, J. and Koelling, R. A. (1966). Relation of cue to consequence in avoidance learning. *Psychonomic Science*, (4):123–124.

[Gat, 1998] Gat, E. (1998). On three-layer architectures. In Kortenkamp, D., Bonasso, R. P., and Murphy, R., editors, *Artificial Intelligence and Mobile*

Robots: Case Studies of Successful Robot Systems, pages 195–210. AAAI Press/The MIT Press.

[Gaussier and Zrehen, 1994] Gaussier, P. and Zrehen, S. (1994). A Topological Neural Map for On-line Learning: Emergence of Obstacle Avoidance in a Mobile Robot. In Cliff, D., Husbands, P., Meyer, J.-A., and Wilson, S. W., editors, *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour (SAB'94)*, pages 282–290. MIT Press.

[Gerkey et al., 2001] Gerkey, B. P., Vaughan, R. T., Støy, K., Howard, A., Sukhatme, G. S., and Matarić, M. J. (2001). Most Valuable Player: A Robot Device Server for Distributed Control. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*, pages 1226–1231, Wailea, Hawaii.

[Gillan, 1981] Gillan, D. J. (1981). Reasoning in the chimpanzee: Ii. transitive inference. *Journal of Experimental Psychology: Animal Behavior Processes*, (7):1–17.

[Goldberg and Matarić, 2000] Goldberg, D. and Matarić, M. J. (2000). Learning Multiple Models for Reward Maximization. In *Proceedings from the Seventeenth International Conference on Machine Learning (ICML'00)*.

[Goldberg and Matarić, 2001] Goldberg, D. and Matarić, M. J. (2001). Design and Evaluation of Robust Behavior-Based Controllers for Distributed Multi-Robot Collection Tasks. In Balch, T. and Parker, L. E., editors, *Robot Teams: From Diversity to Polymorphism*, pages 315–244. A K Peters Ltd.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms Search, Optimization, and Machine Learning*. Addison-Wesley.

[Gould and Gould, 1999] Gould, J. L. and Gould, C. G. (1999). *The Animal Mind*. Scientific American Library.

[Grey, 1950] Grey, W. W. (1950). An imitation of life. *Scientific American*, pages 42–45.

[Grey, 1963] Grey, W. W. (1963). *The Living Brain*. W. W. Norton, New York.

[Gurney et al., 1998] Gurney, K. N., Prescott, T. J., and Redgrave, P. (1998). The Basal Ganglia viewed as an Action Selection Device. In *Proceedings of the Eighth International conference on Artificial Neural Networks*.

[Harvey, 1996] Harvey, I. (1996). Untimed and misrepresented: Connectionism and the computer metaphor. *Newsletter of the Society for the Study of Artificial Intelligence and Simulation of Behaviour (AISB Quarterly)*, (96):20–27.

[Harvey, 1997] Harvey, I. (1997). Cognition is not Computation: Evolution is not Optimisation. In Gerstner, W., Germond, A., Hasler, M., and Nicoud, J.-D., editors, *Artificial Neural Networks - ICANN97, Proceedings of the 7th International Conference on Artificial Neural Networks (ICANN'97), LNCS 1327*, pages 685–690, Lausanne, Switzerland. Springer-Verlag.

[Harvey et al., 1994] Harvey, I., Husbands, P., and Cliff, D. (1994). Seeing the light: artificial evolution, real vision. In Cliff, D., Husbands, P., Meyer, J.-A., and Wilson, S., editors, *From Animals to Animats 3: Proceedings of the Third International Coference on SImulation of Adaptive Behavior*, Cambridge, MA. MIT Press.

[Harvey et al., 1997] Harvey, I., Husbands, P., and Cliff, D. (1997). Evolutionary Robotics: the Sussex Approach. *Robotics and Autonomous Systems*, (20):205–224.

[Haugeland, 1997] Haugeland, J. (1997). What is Mind Design. In Haugeland, J., editor, *Mind Design II*. MIT Press.

[Hauser, 1996] Hauser, M. D. (1996). *The Evolution of Communication*. MIT Press.

[Hernandez-Gardiol and Mahadevan, 2000] Hernandez-Gardiol, N. and Mahadevan, S. (2000). Hierarchical Memory-based Reinforcement Learning. In *Proceedings from the Fifteenth International Conference on Neural Information Processing Systems (NIPS'200)*.

[Hertz et al., 1991] Hertz, J. A., Krogh, A., and Palmer, R. G. (1991). *Introduction to the theory of neural computation*. Addison Wesley.

[Heylighen and Joslyn, 2001] Heylighen, F. and Joslyn, C. (2001). Cybernetics and second order cybernetics. In Meyers, R., editor, *Encyclopedia of Physical Science and Technology*, volume 4, pages 155–170. Academic Press, 3rd edition.

[Holland and Melhuish, 1996] Holland, O. and Melhuish, C. (1996). Getting the most from the least: lessons for the nanoscale from minimal mobile agents. In *Proceedings of the 5th International Conference on Artificial Life*.

[Holland and Melhuish, 1999] Holland, O. and Melhuish, C. (1999). Stigmergy, Self-Organization, and Sorting in Collective Robotics. *Artificial Life*, 5(2).

[Hsu et al., 1990] Hsu, F., Anantharaman, T., Campbell, M., and Nowatzyk, A. (1990). A Grandmaster Chess Machine. *Scientific American*, 263(4):44–50.

[Husbands and Harvey, 1992] Husbands, P. and Harvey, I. (1992). Evolutions versus Design: Controlling Autonomous Robots. In *Integrating Perception, Planning and Action, Proceedings of the Third Annual Conference on Artificial Intelligence, Simulation and Planning*, pages 139–146. IEEE Press.

[Ijspeert, 98] Ijspeert, A. (98). From lampreys to salamanders: Evolving neural controllers for swimming and walking. In Pfeifer, R., Blumberg, B., Meyer, J.-A., and Wilson, S. W., editors, *From Animals to Animats 5, Proceedings of the Fifth International Conference on the Simulation of Adaptive Behavior*, pages 390–399, Cambridge, MA. MIT Press.

[Jakobi et al., 1995] Jakobi, N., Husbands, P., and Harvey, I. (1995). Noise and The Reality Gap: The Use of Simulation in Evolutionary Robotics. In Moran, F., Moreno, A., and Chacon, P., editors, *Advances in Artificial Life: Proceedings of the Third European Conference on Artificial Life*, number 929 in Lecture Notes in Artificial Intelligence, pages 704–720. Springer-Verlag.

[Jenkins and Matarić, 2000] Jenkins, O. C. and Matarić, M. J. (2000). Primitive-Based Movement Classification for Humanoid Imitation. In *Proceedings of the First IEEE-RAS International Conference on Humanoid Robotics*. MIT Press.

[Johnson, 1992] Johnson, M. K. (1992). MEM: Mechanisms of Recollection. *Journal of Cognitive Neuroscience*, (4):268–280.

[Johnson and Hirst, 1993] Johnson, M. K. and Hirst, W. (1993). MEM: Memory Subsystems as Processes. In Collins, A. F., Gathercole, S. E., Conway, M. E., and Morris, P. E., editors, *Theories of Memory*, pages 241–286. Lawrence Erlbaum Associates.

[Johnson and Multhaup, 1992] Johnson, M. K. and Multhaup, K. S. (1992). Emotion and MEM. In Christianson, S.-A., editor, *The handbook of emotion and memory: Current research and theory*, pages 33–66. Hilldale NJ: Lawrence Erlbaum Associates Inc.

[Kaelbling et al., 1996] Kaelbling, L. P., Littman, K. L., and Moore, A. W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, (4):237–285.

[Kemp, 1982] Kemp, T. S. (1982). *Mammal-like Reptiles and the Origin of Mammals*. Academic Press.

[Kirsch, 1991] Kirsch, D. (1991). Today the earwig, tomorrow man? *Artificial Intelligence*, (47):161–184.

[Kowalski, 1979] Kowalski, R. A. (1979). Programming = logic + control. *Communications of the ACM*, 22(7):424–436.

[Kowalski and Sadri, 1996] Kowalski, R. A. and Sadri, F. (1996). Towards a unified agent architecture that combines rationality with reactivity. In *Proceedings of International Workshop on Logic in Databases*. Springer-Verlag.

[Lakoff and Johnson, 1999] Lakoff, G. and Johnson, M. (1999). *Philosophy in the flesh: the embodied mind and its challenge to western thought*. Basic Books.

[LeDoux, 1998] LeDoux, J. (1998). *The Emotional Brain*. Weidenfeld and Nicolson General.

[Lewis et al., 1992] Lewis, M. A., Fagg, A. H., Solidum, A., and Bekey, G. (1992). A genetic Programming Approach to the Construction of a Neural Network for Control of a Walking Robot. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 2618–2623.

[Li and Ogmen, 1994] Li, L. and Ogmen, H. (1994). Visually guided motor control: Adaptive sensorimotor mapping with on-line visual-error correction. In *Proceedings of the World Congress on Neural Networks*, pages 1127–1134.

[Lynch and Krogh, 2000] Lynch, N. and Krogh, B. H., editors (2000). *Hybrid Systems: Computation and Control, Proceedings of the Third International Workshop (HSCC'2000)*. Springer.

[Maes, 1994] Maes, P. (1994). Modeling adaptive autonomous agents. *Artificial Life Journal*, 1(1–2):135–162.

[Maes and Brooks, 1990] Maes, P. and Brooks, R. A. (1990). Learning to Coordinate Behaviours. In *Proceedings of the Eight National Conference on Artificial Intelligence (AAAI'90)*.

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

[Mataric, 1990] Mataric, M. J. (1990). Navigating With a Rat Brain: A Neurobiologically-Inspired Model for Robot Spatial Representation. In Meyer, J. A. and Wilson, S., editors, *From Animals to Animats: International Conference on Simulation of Adaptive Behavior*, pages 169–175. MIT Press.

[Matarić, 1992] Matarić, M. J. (1992). Integration of Representation Into Goal-Driven Behavior-Based Robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312.

[Matarić, 1994] Matarić, M. J. (1994). *Interaction and Intelligent Behavior*. PhD thesis, Massachusets Institute of Technology.

[Matarić, 1994] Matarić, M. J. (1994). Reward functions for accelerated learning. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML'94)*.

[Matarić, 1997] Matarić, M. J. (1997). Using communication to reduce locality in distributed multi-agent learning. In *Proceedings of the (AAAI'97)*, pages 643–648, Providence, Rhode Island.

[Matarić, 2001a] Matarić, M. J. (2001a). Great Expectations: Scaling Up Learning by Embracing Biology and Complexity. *Machine Learning*, 1.

[Matarić, 2001b] Matarić, M. J. (2001b). Learning in Behavior-Based Multi-Robot Systems: Policies, Models, and Other Agents. *Cognitive Systems Research, special issue on Multi-disciplinary studies of multi-agent learning*, 2(1):81–93.

[McCulloch and Pitts, 1943] McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, (5):115–133.

[McFarland, 1999] McFarland, D. (1999). *Animal Behaviour, Psychobiology, ethology and evolution*. Addison Wesley Longman Ltd, fourth edition.

[Melhuish et al., 1999] Melhuish, C., Welsby, J., and Edwards, C. (1999). Using Templates for Defensive Wall Building with Autonomous Mobile Ant-Like Robots. In *Proceeding of Towards Intelligent Mobile Robots (TIMR'99)*.

[Michaud and Matarić, 1999] Michaud, F. and Matarić, M. J. (1999). Representation of Behavioral History for Learning in Nonstationary Conditions. *Robotics and Autonomous Systems*, (29):187–200.

[Millan, 1994] Millan, R. (1994). Learning Efficient Reactive Behavioural Sequences from Basic Reflexes in a Goal-Directed Autonomous Robot. In Cliff, D., Husbands, P., Meyer, J.-A., and Wilson, S. W., editors, *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behaviour (SAB'94)*, pages 266–274. MIT Press.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Series in Computer Science (Artificial Intelligence). McGraw-Hill.

[Moore, 1996] Moore, B. R. (1996). The Evolution of Imitative Learning. In Heyes, C. M. and Galef, B. G., editors, *Social Learning in Animals: The Roots of Culture*, pages 245–265. Academic Press.

[Morén and Balkenius, 2000] Morén, J. and Balkenius, C. (2000). A Computational Model of Emotional Learning in the Amygdala. In Meyer, J.-A., Berthoz, A., Floreano, D., Roitblat, H. L., and Wilson, S. W., editors, *From Animals to Animats 6: Proceedings of the 6th International Conference on the Simulation of Adaptive Behavior (SAB'2000)*, Paris, France. MIT Press.

[Nehaniv et al., 1999] Nehaniv, C. L., Dautenhahn, K., and Loomes, M. J. (1999). Constructive Biology and Approaches to Temporal Grounding in Post-Reactive Robotics. In *Sensor Fusion and Decentralized Control in Robotics Systems II, Proceedings of the International Society for Optical Engineering (SPIE)*, number 3839, pages 156–167.

[Nehmzow and Mitchell, 1995] Nehmzow, U. and Mitchell, T. (1995). The Prospective Student's Introduction to the Robot Learning Problem. Technical Report UMCS-95-12-6, University of Manchester, Department of Computer Science.

[Nehmzow et al., 1993] Nehmzow, U., Smithers, T., and McGonigle, B. (1993). Increasing behavioural repertoire in a Mobile Robot. In *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 291–297. MIT Press.

[Newell and Simon, 1963] Newell, A. and Simon, H. (1963). GPS - A program that simmulates human thought. In Feigenbaum, E. A. and Feldman, J., editors, *Computers and Thought*, pages 279–296. McGraw-Hill.

[Nicolescu and Matarić, 2001] Nicolescu, M. and Matarić, M. J. (2001). Experience-based representation construction: learning from human and robot teachers. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[Nilsson, 1984] Nilsson, N. J. (1984). Shakey the Robot. Technical Report 323.

[Nolfi and Floreano, 2000] Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press.

[Nolfi et al., 1994] Nolfi, S., Floreano, D., Miglino, O., and Mondada, F. (1994). How to evolve autonomous robots: Different approaches in evolutionary

robotics. In Brooks, R. and Maes, P., editors, *Proceedings of Artificial Life IV*, pages 190–197, Boston, Massachusets. MIT Press.

[O'Keefe and Conway, 1978] O'Keefe, J. and Conway, D. H. (1978). Hippocampal place units in the freely moving rat: Why they fire when they fire. *Experimental Brain Research*, (31):573–590.

[Parker, 1997] Parker, L. E. (1997). L-ALLIANCE: Task-Oriented Multi-Robot Learning in Behaviour-Based Systems. *Advanced Robotics, Special Issue on Selected Papers from IROS'96*, 11(4):305–322.

[Parr and Russel, 1998] Parr, R. and Russel, S. (1998). Reinforcement Learning with Hierarchies of Machines. In *Advances in Neural Information Processing Systems 10*. MIT Press.

[Pearce, 1997] Pearce, J. M. (1997). *Animal Learning and Cognition*. Psychology Press, 2nd edition.

[Prescott and Ibbotson, 1997] Prescott, T. J. and Ibbotson, C. (1997). A robot trace-maker: modelling the fossil evidence of early invertebrate behavior. *Artificial Life*, 3(4):289–306.

[Prescott et al., 1999] Prescott, T. J., Redgrave, T., and Gurney, K. (1999). Layered Control Architectures in Robots and Vertebrates. *Adaptive Behavior*, (7):99–127.

[Rao and Georgeff, 1995] Rao, A. S. and Georgeff, M. P. (1995). BDI Agents: From Theory to Practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 312–319. AAI Press.

[Redish and Touretzky, 1998] Redish, A. D. and Touretzky, D. S. (1998). The Role of the Hippocampus in Solving the Morris Water Maze. *Neural Computation*, (10):73–111.

[Russel and Norvig, 1995] Russel, S. and Norvig, P. (1995). *Artificial Intelligence, A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall.

[Rylatt et al., 1998] Rylatt, M., Czarnecki, C., and Routen, T. (1998). Connectionist Learning in Behaviour-Based Mobile Robots: A Survey. *Artificial Intelligence Review*, 12:445–468.

[Shoham, 1993] Shoham, Y. (1993). Agent-oriented programming. *AI Journal*, 60(1):51–92.

[Steels, 1994] Steels, L. (1994). Emergent Functionality in Robotic Agents through On-Line Evolution. In *Proceedings of ALife IV*. MIT Press.

[Stone, 1998] Stone, P. (1998). *Layered Learning in Multi-agent Systems*. PhD thesis, Computer Science Department, Carnegie Mellon University.

[Stone, 2000] Stone, P. (2000). *Layered Learning in Multi-Agent Systems: A Winning Approach to Robotic Soccer*. MIT Press.

[Stone and Veloso, 1999] Stone, P. and Veloso, M. (1999). Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork. *Artificial Intelligence*, 110(2):241–273.

[Strickberger, 1995] Strickberger, M. W. (1995). *Evolution*. The Jones and Bartlett Series in Biology. Jones and Bartlett Publishers, Second edition.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press.

[Tsotsos, 1995] Tsotsos, J. K. (1995). Behaviourist inteligence and the scaling problem. *Artificial Intelligence*, (75):135–160.

[Turing, 1937] Turing, A. M. (1937). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Second Series*, 42:230–265.

[Turing, 1950] Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59:433–460.

[Tyrrell, 1993] Tyrrell, T. (1993). *Computational Mechanisms for Action Selection*. PhD thesis, Centre for Cognitive Science, University of Edinburgh.

[Webb, 1998] Webb, B. (1998). Robots crickets and ants: models of neural control of chemotaxis and phonotaxis. *Neural Networks*, 11(7/8):1479–1496.

[Weiner, 1948] Weiner, N. (1948). *CYBERNETICS or Control and Communication in the Animal and the Machine*. MIT Press.

[Weitzenfeld, 2000] Weitzenfeld, A. (2000). ASL/NSL: A Multi-level Computational Model for Distributed Neural Simulation. In *Proceedings of the International Conference on Artificial Intelligence (ICAI'00)*, Las Vegas, Nevada.

[Weitzenfeld94 and Arbib, 1994] Weitzenfeld94, A. and Arbib, M. A. (1994). NSL - Neural Simulation Language. In Skrzypek, J., editor, *Networks Simulation Environment*. Kluwer.

[Werger, 2000] Werger, B. B. (2000). Ayllu: Distributed Port-Arbitrated Behavior-Based Control. In Parker, L. E., Bekey, G., and Barhen, J., editors, *Distributed Autonomous Robotic Systems 4*, pages 25–34. Springer.

[Wilson, 1991] Wilson, S. W. (1991). The Animat Path to AI. In ans S. W. Wilson, J.-A. M., editor, *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior (SAB'91)*, Cambridge, Massachusets.

[Wooldridge and Jennings, 1995] Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2):115–152.

[Yamauchi and Beer, 1995] Yamauchi, Y. and Beer, R. (1995). Sequential behavior and learning in evolved dynamical neural networks. *Adaptive Behavior*, 2(3):219–246.

To facilitate academic research we have referenced papers that are not restricted by copyrights whenever these are of an equal or better quality than restricted alternatives.