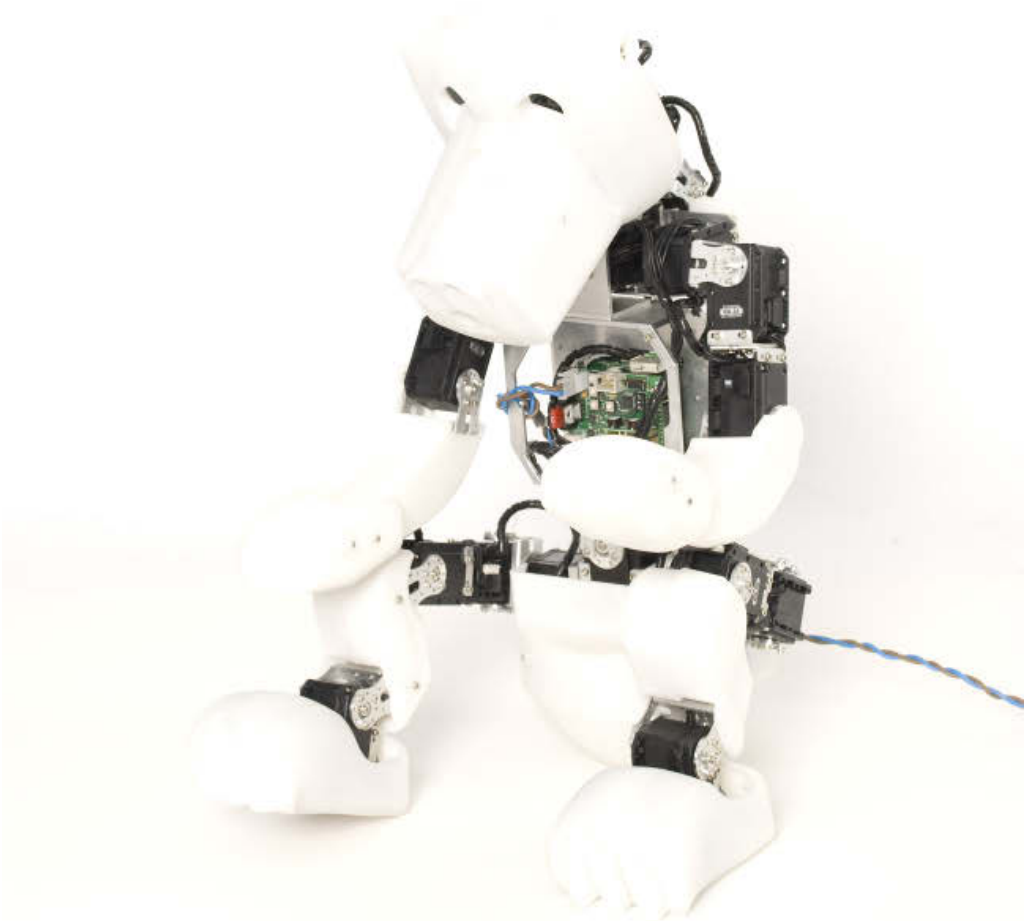


Flexible Attention-based Cognitive Architecture for Robots

Rony Novianto

Submitted to the
Faculty of Engineering and Information Technology
University of Technology, Sydney
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy (Computer Science)
2014

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as a part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Rony Novianto

Acknowledgement

This research would not have been possible without the support of following people and organisations who I would also like to thank:

- **Professor Mary-Anne Williams** (University of Technology Sydney, Australia): my research supervisor, who provided me with countless support, opportunities, feedback, suggestions and advice throughout my candidature.
- **Doctor Benjamin Johnston** (University of Technology Sydney, Australia): who kindly spent the time to provide me with enormous feedback and suggestions, as well as to review this dissertation.
- **My fellow students** in the Innovation and Enterprise Research Laboratory: who made the laboratory a friendly and productive environment for working on my research.
- **Professor Xiaoping Chen（陈小平教授）** (University of Science and Technology of China (USTC), China): who generously hosted me for collaborating with his research laboratory.
- **Professor Peter Gärdenfors** (Lund University, Sweden): who generously hosted me for my Australian Endeavour Award.
- **Professor Christian Balkenius** (Lund University, Sweden): who generously hosted me for my Australian Endeavour Award.
- **The students** in Professor Xiaoping Chen's Multi-Agent Systems Laboratory, Professor Peter Gärdenfors' laboratory and Professor Christian Balkenius' laboratory: who warmly welcomed me into their research group.
- **Glenn Wightwick** (previously at International Business Machines (IBM), Australia, and currently at University of Technology Sydney, Australia): who

continuously supported my work.

- **Professor Michael Genesereth** (Stanford University, USA): my examiner for this dissertation who provided me with invaluable and kind feedback, suggestions and advice.

- **Doctor Paul Robertson** (Massachusetts Institute of Technology, USA): my examiner for this dissertation who provided me with invaluable and kind feedback, suggestions and advice.

- **Professor Reid Simmons** (Carnegie Mellon University, USA): my examiner for this dissertation who provided me with invaluable and kind feedback, suggestions and advice.

- **Australian Government and IBM**: who supported me under Australian Research Council (ARC) Discovery Project Grant and provided me with Australian Endeavour Fellowship Award 2011 and IBM Ph.D. Fellowship Award 2011.

- **My parents, family, partner and friends**: who continuously and patiently supported me throughout my difficulties.

- **My sister and first nephew**: who patiently waited and endured pain when I was busy working.

Abstract

Robots have been working in factories to achieve tasks autonomously with little human intervention for some time. Even though robots are commonly found as vacuum cleaners in homes and assistants in hospitals, by comparison with factory robots, service robots have not been widely deployed in society because there remains several challenges deploying robots to achieve complex tasks in open, unstructured, uncontrolled and complex environments.

Critical research gaps arise from the lack of cognitive architectures that support robots to undertake tasks in open and complex environments. Throughout the history of AI, researchers have developed various algorithms, representations and mechanisms, to solve specific tasks. However, each of these techniques has different strengths and weaknesses when applied to particular problems. A cognitive architecture provides a unifying infrastructure that can integrate various techniques to solve open and complex tasks.

However, four important issues become apparent when current cognitive architectures are applied to service robotic tasks. First, they are not capable of managing robot resources and as a result robotic developers must take responsibility for managing the resources manually. Second, they are not capable of integrating independently developed techniques, which are often needed to solve problems. Third, they are inflexible, unable to adapt to design changes and require considerable time and effort to modify. Fourth, they are inadequate for supporting the necessary capabilities required by robots such as multiple goals, reliability and maintainability. These issues are confirmed when cognitive architectures are applied to a standard benchmark problem in AI: the autonomous robot soccer problem.

The purpose of this dissertation is to address these significant gaps so as to accelerate the development, deployment and adoption of service robots undertaking tasks in open and complex environments. This dissertation develops a novel bio-

inspired cognitive architecture (called ASMO) that has been designed and developed to address all four identified shortcomings of current cognitive architectures.

In ASMO, intelligent behaviours to solve open and complex tasks is a result of the emergence of constituent processes, rather than from careful top-down control engineering. Minsky has argued in his Society of Mind that intelligent behaviours can emerge from the interaction of many simple processes, even though each process may lack 'intelligence' in isolation. In addition, Anderson argued that an emergent system produces more complex behaviours and properties that cannot be reduced to the sum of its components.

ASMO has attention, emotion and learning mechanisms that are inspired by human intelligence. It treats each action as a concurrent, independent and self-governed black box process that competes for the robot's attention to perform actions. The attention mechanism is used to mediate the competition among processes, which correspond to the set of potential actions. The emotion mechanism is used to bias the attention demanded by the processes. The learning mechanisms are used to modify the attention in order to improve robots' performances.

Combining concurrent, independent and self-governed black-box processes with attention and emergent approaches allows ASMO to address the four shortcomings of current cognitive architectures. First, the attention mechanism manages resources explicitly. Second, the black-box design allows any kind of independently developed technique to be integrated without the need to know its internal algorithm, representation or mechanism. Third, attention weighted values enables various techniques to be (re)integrated or (re)structured on the fly with considerably less time and effort. Fourth, the concurrent, independent and self-governed designs support the capabilities required by robots by allowing processes to (i) achieve multiple goals concurrently, (ii) fail without causing the whole system to fail and (iii) be maintained in isolation.

ASMO is evaluated using two robotic problems: (i) the RoboCup soccer standard benchmark problem is used to demonstrate proof-of-concept that a team of robots can be supported by ASMO. In particular, a real robot can be governed by ASMO's attention mechanism to undertake complex tasks. (ii) a companion robot problem is used to demonstrate that ASMO's attention, emotion and learning mechanisms overcome the four identified shortcomings of current state-of-the-art cognitive architectures.

This dissertation presents ASMO, an innovative cognitive architecture that addresses the four shortcomings of current state-of-the-art cognitive architectures, and that can also accelerate the development, deployment and adoption of service robots. ASMO provides a more natural and easier approach to programming robots based on a novel bio-inspired attention management system. ASMO allows researchers and robot system developers to focus on developing new capabilities as processes rather than having to be concerned about integrating new capabilities into a cognitive architecture.

Contents

1 Introduction **1**

1.1 Scientific Challenge and Motivation 3

1.2 Aims and Objectives 6

1.3 Significance and Contributions 6

1.4 Scope 8

1.5 Dissertation Organisation 9

2 Robot Cognitive Architecture **11**

2.1 Definition of Robot 11

2.2 Robot Environment 14

2.3 Definition of Intelligence 16

2.4 Definitions of Cognitive Architecture 19

2.4.1 Comparison with Other Types of Software 21

2.5 Evaluation Criteria for Cognitive Architectures 24

3 Existing Cognitive Architectures **27**

3.1 Approaches based on Knowledge Design 27

3.2 Approaches based on Knowledge Utilisation 29

3.2.1 Lookup-based Approach 31

3.2.2 Case-Based Finite State Machine Approach 32

3.2.3 Priority-based or Hierarchical-based Approach 34
3.2.4 Goal-based Approach 35
3.2.5 Connection-based Approach 37
3.2.6 Utility-based Approach 39
3.3 Current State of The Art 40
3.3.1 Soar 42
3.3.2 Adaptive Control of Thought – Rational (ACT-R) 45
3.3.3 Subsumption 49
3.3.4 Situated Automata 51
3.3.5 Procedural Reasoning System (PRS) 53
3.3.6 Agent Network Architecture (ANA) 55
3.3.7 ICARUS 59
3.3.8 Three Tier / Three Layers Architecture (3T) 62
3.3.9 Polyscheme 64
3.4 Related Architectures 67
3.5 Summary and Gaps 70

4 ASMO Cognitive Architecture 73

4.1 Overall Design 73
4.1.1 Memory 75
4.1.2 Attention Mechanism 77
4.1.3 Emotion Mechanism 78
4.1.4 Learning Mechanisms 78
4.2 Share Management of Resources 80
4.3 Cognitive Capabilities 81
4.3.1 Outer World, Inner World, Sensation and Actuation 82

4.3.2 Perception, Conception, Simulation and Planning 83

5 ASMO's Attention Mechanism 85

5.1 Need for Attention in Decision Making 85

5.2 Theories of Attention 87

5.2.1 Selective or Focused Attention 87

5.2.2 Divided Attention 89

5.2.3 Automaticity 91

5.3 Computational Design and Model 93

5.4 Implementation 97

5.4.1 Modules 99

5.4.2 Attention Value 103

5.4.3 Boost Value and Reflex Priority 106

5.4.4 Attention Competition 107

5.5 Other Work 112

6 ASMO's Emotion Mechanism 119

6.1 Needs for Emotion and Subjective Bias 119

6.2 Theories of Emotion 121

6.2.1 Representation 121

6.2.2 Causality 123

6.2.3 Evaluation: Innate or Learned 126

6.3 Computational Design and Model 127

6.4 Implementation 131

6.4.1 Dimension and Label Nodes 132

6.4.2 Biological Factor and Cognitive Factor Nodes 133

6.4.3 Subjective Weight 134

6.5 Other Work 138

7 ASMO's Learning Mechanisms 140

7.1 Needs for Learning 140

7.2 Theories of Learning 141

7.2.1 Habituation and Sensitisation 141

7.2.2 Operant Conditioning 143

7.2.3 Classical Conditioning 144

7.2.4 Observational Learning 145

7.3 Computational Design and Model 146

7.4 Implementation 147

7.4.1 Habituation and Sensitisation 148

7.4.2 Operant Conditioning 150

7.4.3 Classical Conditioning 152

7.4.4 Observational Learning 153

7.5 Other Work 156

7.5.1 Habituation and Sensitisation 156

7.5.2 Operant Conditioning 159

7.5.3 Classical Conditioning 160

7.5.4 Observational Learning 162

8 Evaluation 165

8.1 RoboCup Soccer SPL Standard Benchmark Problem 165

8.1.1 ASMO's Attention Mechanism 170

8.1.2 ASMO's Emotion and Learning Mechanisms 175

8.2 Smokey Robot Companion Problem 177

8.2.1 ASMO's Attention Mechanism 181

8.2.2 ASMO's Emotion Mechanism 186
8.2.3 ASMO's Habituation and Sensitisation Mechanisms 189
8.2.4 ASMO's Operant Conditioning Mechanism 194
8.2.5 ASMO's Classical Conditioning Mechanism 196
8.2.6 ASMO's Observational Learning Mechanism 200
8.3 Analysis and Discussion 201

9 Conclusion **208**
9.1 Significance and Contributions 208
9.2 Limitation and Future Work 210
9.3 Final Thoughts 212

A Non-Cognitive Architectures **214**
A.1 Autonomous Robot Architecture (AuRA) 214
A.2 Distributed Architecture for Mobile Navigation (DAMN) 216
A.3 Emotionally Grounded (EGO) Architecture 219

List of Figures

1.1 Assistance Needed for People with Disability by Tasks 2
1.2 The Current Standardised Hardware Platform for 2014 4
2.1 Classification of Agents 12
2.2 Cognitive Architecture and Other Software 22
3.1 Decision Making Categories 29
3.2 Competitive vs. Collective Decision Making 30
3.3 Priority Scenario 35
3.4 Neural Network Example 38
3.5 Robots Governed by Soar 42
3.6 Soar Architecture [120] 44
3.7 Robots Governed by ACT-R 46
3.8 ACT-R Architecture 47
3.9 Subsumption Layer [42] 49
3.10 Subsumption Module [42] 50
3.11 Circuit Generated by Gapps 52
3.12 Procedural Reasoning System 54
3.13 Agent Network Architecture 56
3.14 Robots Governed by ICARUS 59

3.15 ICARUS Architecture 61

3.16 Three Tier / Three Layers Architecture [34] 62

3.17 Robots Governed by Polyscheme 65

4.1 Robots Governed by ASMO 74

4.2 ASMO Cognitive Architecture 74

4.3 Cognitive Capabilities proposed by Gärdenfors and Williams 82

4.4 ASMO Body 83

5.1 Attention Competition or Election 98

6.1 Theories of Emotional Causality 124

6.2 Design of ASMO's Emotion Mechanism 128

6.3 Types of Nodes Used by ASMO's Emotion Mechanism 132

7.1 Neural Networks for Observational Learning 154

8.1 RoboCup Soccer Scenario 167

8.2 Pseudocode of Some Modules Used in RoboCup Soccer System 171

8.3 Attentional Decision Making in The RoboCup Soccer SPL Competition 2010 176

8.5 Neural Network Used by The attend_motion Module 182

8.6 Attentional Decision Making in Smokey 186

8.7 Causal Bayesian Network Used in Smokey 187

8.8 Experiments of Emotional Subjective Bias in Smokey 190

8.9 Habituation and Sensitisation Learning Experiment in Smokey 192

8.9 Experiments of Habituation and Sensitisation Learning in Smokey . . 193

8.10 Experiments of Operant Conditioning Learning in Smokey 197

8.11 Experiments of Classical Conditioning Learning in Smokey 199

8.12 The Neural Network of The Observational Learning in Smokey Robot Companion 200

8.13 Experiments of Observational Learning in Smokey 202

A.1 Robots Governed by AuRA 215

A.2 DAMN Constraint Arbitration [186] 217

A.3 DAMN Actuation Arbitration [186] 217

A.4 DAMN Effect Arbitration [186] 218

A.5 Robots Governed by EGO 220

A.6 EGO Architecture 222

A.7 An Example of A Module Tree Structure in EGO Architecture 222

List of Tables

3.1 Lookup Table Example 32
3.2 Priority Table Example 34
3.3 Action Description Table Example 36
3.4 Comparison to Bryson's Trend of Cognitive Architectures 41
3.5 Comparison of Approaches in Cognitive Architecture based on Knowledge Design 70
3.6 Comparison of Approaches in Cognitive Architecture based on Knowledge Utilisation 71
3.7 Comparison of Current State of The Art of Cognitive Architectures . 72

5.1 Stroop Effect 91
5.2 Effects for Different Rates in Attention Value 106

8.1 Users' Preferences of Playing Ball and Drums 183
8.2 The Subjective Weights of the play_ball and play_drums Modules . . 188
8.3 Users' Requests 198
8.4 Probability of Requests Asked by Users 198

Chapter 1

Introduction

Service robots have the potential to dramatically transform our lives by achieving tasks more autonomously with less of intervention. They are currently available for a variety of applications, such as self-driving cars to transport passengers safely, health-care robots to assist patients, companion robots to improve mental health of the elderly, entertainment robots to educate children and vacuum cleaner robots to clean rooms. They help to improve our quality of life enormously in many dimensions, including safety, time and cost, to make our lives more rewarding, productive, convenient and enjoyable. The McKinsey Global Institute [138] has identified robotics as one of the twelve disruptive technologies that will transform life, business and the global economy.

However, by comparison with industrial robots, service robots have not been widely deployed or adopted in society. Unlike industrial robots, these service robots face complex tasks in open, unstructured, uncontrolled and complex environments. They are often required to work in environments that are difficult to engineer. They are required to perform more general tasks than industrial robots.

This problem remains significant as many people are still desperately looking for physical assistance in their everyday lives. In Australia, for example, 60% of the four million people with a disability in 2009 are living without adequate support and assistance [2]. They still need assistance for basic living tasks, including property maintenance, health care and household chores (see Figure 1.1).

In order to solve this problem, there have been enormous ongoing effort in Artificial Intelligence (AI) to produce general intelligence by creating *cognitive architec-*

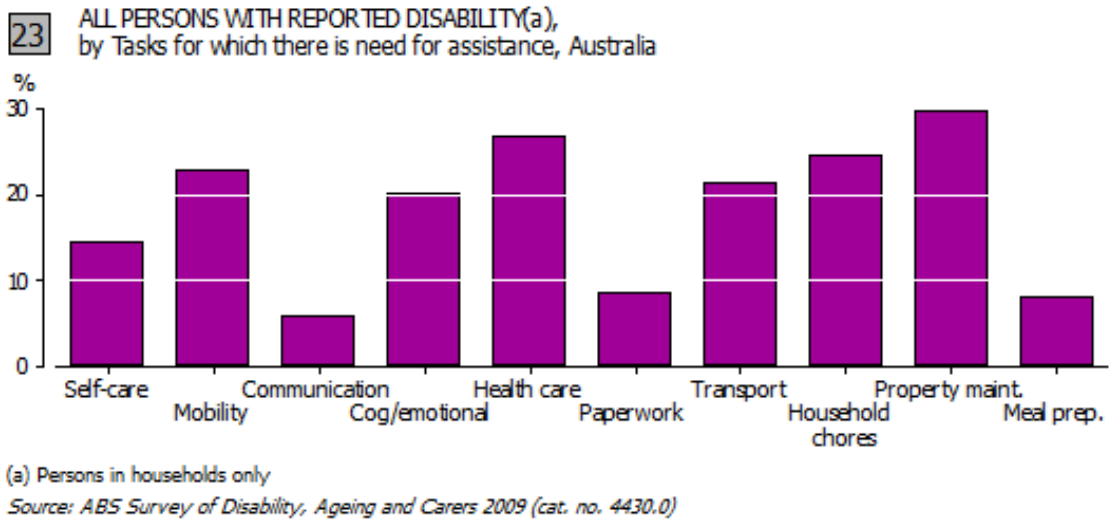


Figure 1.1: Assistance Needed for People with Disability by Tasks

tures. Throughout the history of AI, researchers have discovered various techniques, such as algorithms, representations and mechanisms, to solve specific tasks. Each technique has different strengths and capabilities to solve particular problems. Cognitive architectures aim to provide a unifying coherent approach to solve more complex and general tasks in open and complex environments by integrating capabilities of various techniques [120, p. 1] [120, p. 11].

However, four important issues become apparent when current cognitive architectures are applied to service robotic tasks in open and complex environments: (i) incapable of managing resources, (ii) incapable of integrating independently developed techniques, (iii) inflexible when adapting to design changes and (iv) inadequate in supporting capabilities required by robots.

These issues are confirmed when cognitive architectures are applied to a standard benchmark problem in AI: the RoboCup Soccer problem [114]. Many robot soccer systems rely on developers for managing resources manually. These systems involve carefully crafted and integrated techniques that cannot be developed independently. They are precisely engineered and finely tuned for a particular environment (e.g. RoboCup soccer field). As a result, they are inflexible when adapting to design changes.

The purpose of this dissertation is to address these gaps so as to accelerate the development, deployment and adoption of service robots. The following research question is raised in this dissertation: how can a flexible cognitive architecture

be created to handle limited resources and integrate any kind of independently developed techniques in order to support robotic tasks? My hypothesis to this question is that attention and emergence approaches can be used to create such cognitive architecture.

Attention and emergence have been long observed in biological agents like humans. People pay attention to salient stimuli and perform tasks that they are consciously or unconsciously (i.e. automatically) attending to. In addition, their behaviours present as a result of the emergence of independent techniques and capabilities that handle resources and flexibly adapt to changes. However, there is no cognitive architecture that is based on both attention and emergence of techniques for robots.

Inspired by attention and emergence in humans, I have studied and developed a novel cognitive architecture called ASMO (Attentive and Self-MOdifying) to address all of the four issues identified above. I view intelligence for solving open, complex and general tasks as the result of the emergence of constituent processes. An emergent system consisting of simpler or smaller processes can give rise to intelligence although the processes themselves may lack intelligence [144]. It produces more complex behaviours and properties that are different to simply the sum of its parts [15].

1.1 Scientific Challenge and Motivation

Intelligence for solving open, complex and general tasks have been widely studied across many disciplines, including computer science, cognitive science, cognitive psychology, neuroscience and robotics. Within the disciplines of computer science and robotics, the RoboCup initiative [114] provides a standard benchmark problem for comparing intelligence in machines (i.e. intelligent agents). This problem has been viewed as a new grand challenge in artificial intelligence after computer chess. It is evaluated through international competitions where a wide range of technologies, algorithms and architectures can be integrated, examined and shared.

The RoboCup competitions are held on an annual basis. Each year, their rules are refined to advance the state of the art of artificial intelligence. Today, they are divided into several domains and leagues, one of which is the RoboCup Soccer domain Standard Platform League (SPL). The aim of the SPL is to build a team of

fully autonomous robot soccer players using a standardised hardware platform (see Figure 1.2).

Figure 1.2: The Current Standardised Hardware Platform for 2014

RoboCup Soccer SPL is a domain characterised by Russell and Norvig [192, p. 44] as having 'difficult' properties; it is a known environment that is partially observable, non-deterministic, sequential, dynamic and continuous (see Section 2.2). Hence, it is an excellent problem for evaluating ASMO cognitive architecture.

The vast majority of participants in RoboCup Soccer SPL competitions, including our RoboCup 2008 runner-up team *WrightEagleUnleashed!*, developed a cognitive architecture for playing soccer based on the case-based finite state machine approach [194, 22, 5, 247, 231, 193]. Tasks are divided into a number of situations encoded as states, where specific technique and behaviour are created to handle each state. Formation of states is manually designed and coordinated to create the central representation that covers all possible situations.

In this kind of architecture, different configurations are tested, results are analysed and changes are made to the system to suit the competition's needs. This process is repeated until the system is highly tuned for optimal performance. While this architecture is easy to understand since each state is easy to examine, it is limited and suffers from several weaknesses. It creates serious engineering problems:

1. **Manual management of resources**

   Soccer robots have limited physical resources (e.g. legs, arms, CPU, RAM,

etc). This architecture does not manage resources used by the robots. It requires developers to manually handle the resources when designing the techniques and behaviours.

2. **Incapable to integrate independently developed techniques**

   This architecture requires states to be defined in advance, which makes the integration of various independently developed techniques difficult or not possible. For example, it is difficult to integrate all classical planning, neural networks and utility theory together into a system's behaviour using this traditional approach.

3. **Inflexible to adapt to changes**

   Designing and structuring the states for an entire system can take a considerable amount of time. Often a small change to one state requires the entire formation of states to be reconsidered and redesigned.

4. **Less robust and difficult to extend**

   A small error in one state can cause the whole robot to *crash* and stop operating because the tasks are centrally represented by the formation of states that are connected to each other. A parallel system with a central representation will also suffer a similar problem because the central representation serves as a single bottleneck. Another consequence of having coordinated states (i.e. central representation) is that a behaviour in a state cannot be modified or fixed in isolation without requiring the robot to stop operating.

While our team was able to work around these limitations and win second place in the 2004 and 2008 (Standard Platform League AIBO robot division) competitions through careful consideration and precise system engineering, it comes as little surprise that robots based on a case-based finite state machine approach fail to work in more open, general and complex environments, such as our homes.

It would be unreasonable to expect predefined states and fixed set of mechanisms and representations to work well for every home. It would also be impractical for developers to fine-tune each design for each home. Unlike controlled environments such as factories and warehouses, the home environment is open, uncontrolled, unstructured, unregulated and complex. Consequently, home robots need to be governed by a flexible cognitive architecture that can accommodate the necessary mechanisms and representations required for a robot to meet its design goals.

These issues further motivate the work in this dissertation to develop a flexible cognitive architecture that addresses the fundamental design problem, rather than working around with the problem and focusing on *short-cuts* aimed only at winning the RoboCup competition.

1.2 Aims and Objectives

The main aim of this dissertation is to develop a novel cognitive architecture that addresses the four issues (i.e. the gaps) identified in the research literature (see Section 3.5). This dissertation also aims to validate the hypothesis that robots are capable of achieving complex tasks based on attention and the emergence of techniques. The aims involve following three major objectives:

1. **Classify and review existing cognitive architectures**

   This dissertation shall classify and review existing cognitive architectures. It will demonstrate the research issues and significant gaps in the scientific literature on cognitive architectures when applied to robots in open and complex environments.

2. **Develop a novel cognitive architecture**

   This dissertation shall present a novel cognitive architecture that addresses the gaps identified in the research literature.

3. **Evaluate the cognitive architecture**

   This dissertation shall evaluate the cognitive architecture in real-life robotic domains or applications. It shall show that the novel cognitive architecture could address the gaps identified in the research literature.

1.3 Significance and Contributions

Studying and developing a cognitive architecture that addresses the gaps identified in the research literature is significant and provides important contributions:

1. **Accelerate the development, deployment and adoption of service robots**

Service robots have not been as widely deployed and adopted as industrial robots. In this dissertation, a novel cognitive architecture is proposed to address the gaps identified in the research literature. This cognitive architecture can help to accelerate the development, deployment and adoption of service robots.

2. **Develop a more intuitive approach to programming robots**

   Conventional robots are programmed in an unnatural manner and one that is completely different to the way humans are instructed to perform tasks. This dissertation develops a more intuitive approach to programming robots based on managing their attention. This approach is more natural and easier to understand by programmers because they themselves are attention-driven agents.

3. **Test theories of attention and emergence**

   Theories of attention and emergence are traditionally difficult to test because they involve the human mind that cannot be easily and fully inspected. This dissertation implements theories of attention and emergence and tests them using robots.

4. **Advance the study in interdisciplinary fields**

   Many robotics studies focus on a single discipline. Studies in interdisciplinary fields are more complicated than studies in a single field. This dissertation advances the study in interdisciplinary fields pertinent to robots by taking theories from cognitive science and psychology, and implementing the theories into a system in engineering and computer science.

5. **Put cognitive science into practice**

   Cognitive science is a new emerging field. Its problem is not as well-defined as in other fields, such as engineering. It involves vague terms and definitions and also philosophical issues that are not practical. This dissertation studies and designs a cognitive architecture based on the cognitive science theories and implements the architecture into a concrete working system in robots.

6. **Provide a better understanding of cognitive architectures**

   Previous work classifies cognitive architectures into three general categories, namely top-down, bottom-up and a hybrid approach. This dissertation pro-

vides a more specific classification based on knowledge utilisation, which are lookup-based, cased-based finite state machine, priority-based or hierarchical-based, goal-based, connection-based and utility-based approach.

7. **Motivate the use of a standard benchmark problem for evaluating and sharing significant work**

   Existing cognitive architectures are difficult to compare, because they are evaluated in a variety of different problems. This dissertation motivates the use of a standard benchmark problem to compare or evaluate cognitive architectures, share the work and measure progress. The proposed cognitive architecture will be evaluated in the RoboCup Soccer SPL standard benchmark problem, where progress is evaluated each year. In addition, a new robot companion problem will be introduced as a basis for benchmarking in the future.

1.4 Scope

The scope of this research is necessarily limited by the time and resources of a doctoral dissertation. In particular, while this work is intended to be biologically inspired, it does not attempt to include comprehensive models of human cognition and biological processes, nor does it seek to mimic the mechanisms of human cognition.

- **Computational Agent**

  The work in this dissertation is focused on computational agents, especially robots. Non-computational agents such as biological agents are out of the scope of this dissertation. However, this dissertation will still draw significant *inspiration* from biological agents.

- **Thought *and* Action**

  The work in this dissertation is concerned with systems that produce intelligent *behaviours*, not just intelligent thoughts. Intelligent robots do not only think, but must act to undertake physical tasks.

- **Bio-inspired System**

  The work in this dissertation is inspired by biological systems (i.e. bio-inspired), especially humans. It is not created with the intent to imitate them

(i.e. biomimetic): a robot can think and act differently to the way that people might.

Ideas from the study of biological systems in multiple disciplines, including cognitive science and cognitive psychology, are translated into an innovative practical cognitive architecture. An aircraft is an excellent example of a bioinspired solution to flight but it is not a biomimetic machine. Its design is inspired by birds, but it does not flap its wings to fly. Biomimetic architectures are outside the scope of this dissertation.

- **Single Agent**

  The work in this dissertation studies single agent systems and architectures. Diversity of mechanisms and representations may seem to be the work of a multi-agent system and architecture, however, they are used within a single agent context in this dissertation. The proposed new cognitive architecture may actually be used in multi-agent systems, but its application in multi-agent systems is outside the scope of this dissertation.

1.5 Dissertation Organisation

This dissertation is organised as a typical computer science dissertation, which involves the identification of major gaps in the research literature, the construction of software that addresses the gaps and a proof of concept of the solution. This chapter (i.e. Chapter 1) introduced the motivation, purpose, significance and scope of this dissertation. The next seven chapters describe the main contributions of this dissertation. The final chapter concludes the dissertation. The outline of this dissertation is as follows:

1. **Chapter 1: Introduction**

   Chapter 1 (i.e. this chapter) introduces the scientific challenge and motivation, aims, objectives, significance, contributions and scope of this dissertation.

2. **Chapter 2: Robot Cognitive Architecture**

   Chapter 2 explores a definition of an agent including a robot, properties of an agent environment, key definitions of intelligence and definitions of cognitive architecture. It provides a working definition of a cognitive architecture. It also

describes the relationship between a cognitive architecture and other related software.

3. **Chapter 3: Existing Cognitive Architectures**

   Chapter 3 discusses the trends in cognitive architectures and related issues. It identifies the current gaps in the research literature. It also provides a novel classification of cognitive architectures based on knowledge design and knowledge utilisation.

4. **Chapter 4: ASMO Cognitive Architecture**

   Chapter 4 describes the overall design of the proposed innovative cognitive architecture called ASMO (Attentive and Self-MOdifying), its mechanisms, its functional compositions and its integration issues.

5. **Chapter 5: ASMO's Attention Mechanism**

   Chapter 5 describes and defines decision making that can be used in robots, explores attention theories, translates the theories into a computational mechanism and describes the design and implementation of the mechanism.

6. **Chapter 6: ASMO's Emotion Mechanism**

   Chapter 6 describes and defines subjective bias, explores emotion theories, translates the emotion theories into a computational mechanism and describes the design and implementation of the mechanism.

7. **Chapter 7: ASMO's Learning Mechanisms**

   Chapter 7 describes and defines learning, explores learning theories, translates the theories into a computational mechanism and describes the design and implementation of the mechanism.

8. **Chapter 8: Evaluation**

   Chapter 8 evaluates ASMO cognitive architecture using (i) the RoboCup Soccer SPL standard benchmark problem and (ii) a robot companion problem. It also analyses the evaluations and shows that the novel ASMO cognitive architecture addresses the gaps identified in the research literature.

9. **Chapter 9: Conclusion**

   Chapter 9 provides a summary of this dissertation and describes the future work.

Chapter 2

Robot Cognitive Architecture

In the previous chapter, I introduced the main purpose of this dissertation which is to develop ASMO cognitive architecture to address critical gaps in the research literature. In this chapter, I review definitions of robots and cognitive architectures. I start by defining a robot for the purposes of this dissertation in Section 2.1. I follow by listing different properties of a robot environment in Section 2.2. I then define intelligence for cognitive architecture in Section 2.3. Finally, I discuss the definition of a cognitive architecture, its role and how it is related to other terminologies in Section 2.4.

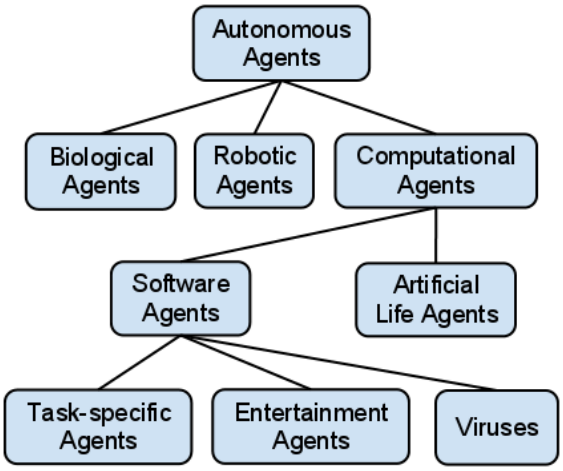
2.1 Definition of Robot

In order to understand what a robot is, I start with the definition of an agent. In 1997, Franklin and Graesser [72, p. 25] have analysed various definitions of an agent and have concluded with the following definition: "a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future". In other words, an agent is anything that, during its operation, autonomously perceives inputs from the environment and produces outputs that affect its task performance and next inputs. Outputs of an agent can be texts, images, motor movements or actions. Inputs of an agent can be texts, images, sounds or sensor data.

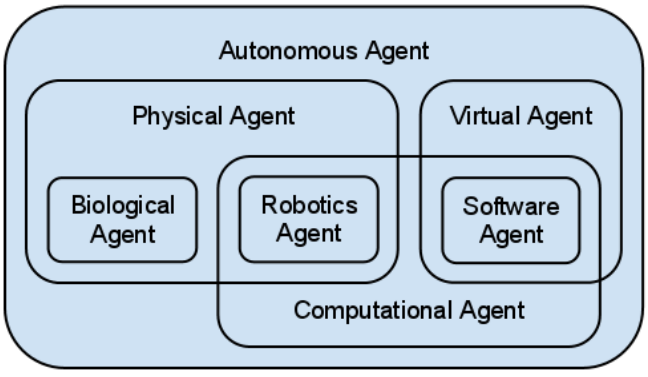
Franklin and Graesser classify agents into three types, namely biological agents, robotic agents and computational agents (see Figure 2.1a) [72, p. 31]. They differ-

entiate robotic agents from computational agents. However, I argue that a robotic agent should be regarded as a computational agent, because it is built on computing platform(s) and its behaviours are performed as a result of computation. I refer to a *computational agent* as an agent that runs on computing platform(s) where its behaviours or the way it perceives inputs and produces outputs can be computed.

Therefore, unlike Franklin and Graesser, I propose two main types of agents, namely *physical agents* and *virtual agents* (see Figure 2.1b). Physical agents perceive inputs and produce outputs in physical environments (e.g. biological and robotic agents). In contrast, virtual agents perceive inputs and produce outputs in non-physical or virtual environments (e.g. software agents). Robotic and software agents fall into the category of computational agents.



(a) Franklin and Graesser's Classification of Agents [72]



(b) Classification of Agents Used in This Dissertation

Figure 2.1: Classification of Agents

Franklin and Graesser distinguish an agent from just any program. An agent is a subclass of ordinary programs. It produces outputs that affect its task performance and next inputs whereas an ordinary program does not necessarily have that effect.

Similarly, a robotic system should be distinguished from an ordinary mechatronic system. A robotic system is a subclass of ordinary mechatronic systems. A robotic system produces outputs that affect its task performance and next inputs whereas an ordinary mechatronic system does not necessarily have that effect.

For example, an automated teller machine (ATM) is an ordinary mechatronic system because its action of ejecting money does not determine what inputs it will receive in the future. In contrast, a self-driving car is a robotic system because its action to turn or go straight determines what its camera will capture in the future (i.e. next inputs). In addition, a pedestrian traffic light is an ordinary mechatronic system. However, it can be modified into a robotic system if it is equipped with a camera to recognise pedestrians and it moves its camera based on the pedestrians' activities (i.e. future images will be determined by the pedestrians' activities in the current image).

In summary, a robot is a subclass of an agent and its system is a subclass of an ordinary mechatronic system. The operation of a robotic system or an agent is said to be *autonomous* whereas the operation of an ordinary mechatronic system or an ordinary program is said to be *automatic*.

In this dissertation, I focus on the work of cognitive architectures for governing computational agents, particularly robots (i.e. autonomous physical computational agents). Non-computational agents are out of the scope of this dissertation. Some may disagree with my definition of a robot. However, the purpose of defining a robot (this section) in this dissertation is to clarify the kind of agents (hence their cognitive architectures) that I cover in this dissertation. In this case, I distinguish between a robotic system and an ordinary mechatronic system. When referring to a robotic system, I mean an autonomous physical agent system where its outputs affect its task performance and next inputs, rather than an automatic program.

2.2 Robot Environment

In order to understand what a robot environment is, I start with the description of an agent environment. According to Russell and Norvig [192, pp. 42–44], agents live in environments that can be characterized by seven dimensions:

1. **Fully Observable $\longleftrightarrow$ Partially Observable**

   An environment is *fully observable* if all information relevant to the choice of action in the environment can be obtained or known at each point in time. The less information can be obtained or known to make a decision at each point in time, the more *partially observable* the environment. In partially observable environments, agents have to remember and keep track of the environment as well as to infer information from previous observations in order to make a good decision. For example, at a given point in time, a RoboCup Soccer SPL agent can only see some parts of the field, so it needs to use its past observations to predict its location on the field.

2. **Deterministic $\longleftrightarrow$ Non-deterministic**

   An environment is *deterministic* if the next state or outcome of an action can always be determined by the current state and action. The more uncertain the next state or outcome, the more *non-deterministic* the environment. In non-deterministic environments, the same action performed in the same state can cause different outcomes. For example, the same strength of kick performed by a RoboCup Soccer SPL agent at the same distance to the ball does not always move the ball the same distance. This non-deterministic characteristic may be caused by several factors, such as the ball's wear and tear, the ball's friction and the precise dynamics of the kick.

   A non-deterministic environment can become deterministic if all factors that govern the outcome are known to the designers. However in practice, the factors are unknown and can be difficult to determine or specify. In this case, the environment will be treated as a non-deterministic environment.

3. **Episodic $\longleftrightarrow$ Sequential**

   An environment is *episodic* if current situations do not depend on past actions. In other words, an environment is episodic if future situations are not affected by current actions. The more past actions the current situations depend on,

the more *sequential* the environment. In sequential environments, the current situation depends on past actions, and future situations are affected by current actions. For example, a RoboCup Soccer SPL striker can fool the opponent's defender by passing the ball to its teammate and back to the striker. The success of fooling the defender (i.e. future situation) is determined by the success of sequential passes (i.e. current actions).

4. **Static $\longleftrightarrow$ Dynamic**

   An environment is *static* if it only changes due to an agent's actions. The more factors the environment can change irrespective to the agent's actions, the more *dynamic* the environment. In a dynamic environment, changes can occur independently from agents' actions. For example, the ball in a RoboCup Soccer SPL game can be intercepted by an opponent or taken by referees while the agent is aligning its position to the ball.

5. **Discrete $\longleftrightarrow$ Continuous**

   An environment is *discrete* if actions and events occur at defined time or intervals. The finer the time or intervals, the more *continuous* or *real-time* the environment. In a continuous or real-time environment, actions and events occur continuously in time, thus they can occur concurrently or overlapping. For example, a RoboCup Soccer SPL agent does not need to wait for its turn to chase the ball. It can run to chase the ball at any time while moving its head to search for the goalposts.

6. **Single Agent $\longleftrightarrow$ Multi-agent**

   A *single agent* environment is affected only by a single agent. The more agents the environment is affected by, the more *multi-agent* the environment. A multi-agent environment is affected by multiple agents that are themselves affected by each other's behaviours. For example, scoring a goal in a RoboCup Soccer SPL game is typically affected by the actions of several agents.

7. **Known $\longleftrightarrow$ Unknown**

   An environment is *known* if agents or designers have knowledge (or rules) about the environment and know how the environment responses. The less knowledge (or rules) the agents or designers have about the environment or the less they know how the environment responses, the more *unknown* the environment. A known environment can be partially observable. For example,

while the position of the ball, goalposts and other agents cannot be observed in a RoboCup Soccer SPL game when they are not within the view of an agent, the designers know the rules of the game and can design an effective algorithm to search for these objects. In contrast, an unknown environment can be fully observable. For example, the states and actions of a character in a new computer game can be fully observed from the screen even though new users may not know the function of each button and how to play the game.

Each of the seven dimensions ranges from a simple property to a complex property. The most difficult properties are partially observable, non-deterministic, sequential, dynamic, continuous, multi-agent and unknown [192, p. 44]. An agent that can deal with a complex property can also deal with a simple property. For example, along the dimension of observability that ranges from fully observable to partially observable, an agent that can manage a partially observable environment can also manage a fully observable environment.

Since a robot is a subclass of an agent, its environment is also characterised by the seven dimensions above. However, since a robot is a physical agent (i.e. live in a physical world), its environment cannot be deterministic, static and discrete. Instead, its environment is non-deterministic, dynamic and continuous.

This dissertation is motivated by the issues discovered in the research literature as well as when applying cognitive architectures to the RoboCup soccer standard benchmark problem. This problem provides an excellent standard benchmark for evaluating cognitive architecture, because all of the most difficult properties are observed in this problem except the unknown property. The unknown property is not observed because this problem has well-defined rules. Strictly speaking, the unknown property does not really refer to the environment itself, instead, it refers to the robot's or designer's knowledge about the environment (e.g. rules and physical laws) [192, p. 44]. Designers are often required to follow rules and specifications in developing robots.

2.3 Definition of Intelligence

As described in Chapter 1, service robots in society require general intelligence to solve general tasks in open and complex environments. However, the definition of

intelligence itself has not been widely accepted. In artificial intelligence, there are four categories of definition of intelligence [192, p. 2]:

1. **Human-like Thought**

   The *human-like thought* category defines intelligence as the capability to think like human beings. The goal is not to create machines that always make correct decisions, but to imitate the way humans think when making the same decisions. It is fine for such machines to make the same errors as what humans make.

   This category aligns with the cognitive modelling approach and it has been studied intensively in cognitive science. The main aim of the cognitive modelling approach and cognitive science is to study intelligence by creating models of human cognition that match the psychological data of human performance.

   The IBM deep blue computer chess system [99] that defeated the human world champion chess player, Gary Kasparov, in 1997 is not considered intelligent in this category, because it thinks and plans its moves using a brute force search, which is different to how humans would think and plan their moves. A brute force search evaluates all possible moves and continuously evaluates all possible moves within each possible move (i.e. to a few degree of depth) until a satisfactory move is found or it reaches the end of the game. In contrast, humans concentrate on a few quality moves instead of all possible moves.

2. **Human-like Act**

   The *human-like act* category defines intelligence as the capability to act like humans. The goal is to create machines that imitate the way humans act. Intelligence is viewed in the behaviour of the machine instead of only in the thinking. Unlike the *human-like thought* category, a machine can think differently to humans and still be considered as intelligent if it acts like humans.

   This category aligns with the Turing Test approach [230]. The Turing Test proposes an imitation game as a benchmark of machine intelligence. The game is played between a person, a machine and an investigator. The person and the machine are invisible to the investigator. They are located in a separate room. The investigator can ask any question that does not require a physical answer. The investigator has to identify the person and the machine correctly.

A machine is said to be intelligent and pass the Turing Test if the investigator cannot distinguish the machine from the person.

3. **Rational Thought**

   The *rational thought* category defines intelligence as the capability to think rationally. The goal is to create machines that produce correct and consistent inferences or conclusions given a set of premises. Unlike the *human-like thought* category, a machine can think differently to humans and still be considered intelligent if it makes logical inferences.

   This category corresponds with the law of thought approach and has been studied extensively in the field of logic [192, p. 4]. There are two main obstacles to this approach [192, p. 4]. First, it is difficult to formalise informal knowledge into logical notation, especially when the knowledge is uncertain. Second, there is a huge difference between solving problems in principle verses in practice.

4. **Rational Act**

   The *rational act* category defines intelligence as the capability to act rationally. The goal is to create machines that act correctly to achieve the best performance given what they know. Intelligence is viewed in the behaviour of the machine instead of only in the thinking. Unlike the *human-like act* category, a machine can act differently to humans and still be considered as intelligent if they act to achieve the best performance.

   This category corresponds to the rational agent approach. A rational agent acts to achieve the best outcomes or expected outcomes. A RoboCup soccer robot is considered intelligent in this category because it plays (e.g. searches and chases the ball) based on algorithms that maximise its performance despite that humans would play differently.

In this dissertation, intelligence is mainly viewed from the *rational act* perspective. Robots should not only think, but also undertake physical actions to help humans, especially in situations where humans are deficient. For example, a robotic car should apply the brakes to avoid hitting a pedestrian when moving too fast due to human error. Robots can use any technique different to the way humans think and act, in order to make rational decisions and achieve the best performance.

In addition, in this dissertation, intelligence is also viewed from the *human-like thought* perspective. Since service robots in society often interact with humans, they have to understand how humans think in order to help and interact with humans better. Their rational decisions can be biased by their understanding about humans.

2.4 Definitions of Cognitive Architecture

There are various definitions of a cognitive architecture across the disciplines, such as computer science, cognitive science, and cognitive psychology. The study of a cognitive architecture is perhaps originated from cognitive psychology where its focus is to model and imitate the human mind. In computer science and in this dissertation, the focus of the cognitive architecture study is to build a system with general intelligence like the human mind. Thus, a cognitive architecture is a kind of software.

The term *cognitive architecture* has been used interchangeably with the term *agent architecture*. Currently, there is no clear and widely accepted definition of these terms. Some of their definitions are as listed as follows:

- In computer science:
  - Russell and Norvig define a cognitive architecture as "models of human reasoning" [192, p. 336].
  - Shaw and Garlan describe an agent architecture as follows: "agent architectures, like software architectures, are formally a description of the elements from which a system is built and the manner in which they communicate. Further, these elements can be defined from patterns with specific constraints [201]" [104, p. 367].
  - Laird defines a cognitive architecture as "the fixed computational structures that form the building blocks for creating generally intelligent systems" [120, p. 1].
- In cognitive science:
  - Wilson and Keil define a cognitive architecture as "the design and organization of the mind" [243, p. 124].

  - Wilson and Keil define an intelligent agent architecture as "a model of an intelligent information processing system defining its major sub-systems, their functional roles, and the flow of information and control among them" [243, p. 411].
  - Friedenberg and Silverman define a cognitive architecture as "the structure and function of many different cognitive systems and how the structure and function interact" [74, p. 139].
  - Bermúdez defines an agent architecture as "a blueprint that shows the different components that make up an agent and how those components are organized" [30, p. 288].

- In cognitive psychology:

  - Newell defines a cognitive architecture as "the fixed structure that realises a symbol system" [153, p. 80].
  - Braisby and Gellatly define a cognitive architecture as "an overarching framework that can account for a wide range of phenomena using a fixed set of mechanisms" [36, p. 610].
  - Kellogg defines a cognitive architecture as "the design or organization of the mind's information-processing components and systems" [107, p. 9].

There are two main opinions of a cognitive architecture reflected in the definitions above. While some have argued that a cognitive architecture is the model, design, structure and organisation of the human mind, others have argued that a cognitive architecture is the structure and framework of a system, i.e. not necessarily only the human mind. In this dissertation, the latter view is adopted, that is, a cognitive architecture is not only restricted to the human mind. This dissertation will use *human cognitive architecture* when referring to the architecture of the human mind.

In summary, I define *cognitive architecture* to be a structure that supports a range of techniques to make decisions in order to produce general intelligent behaviours. There are three key aspects that define a cognitive architecture:

- **General Intelligence**

  A cognitive architecture is a structure that aims to produce general intelligence capabilities, instead of specific capabilities (e.g. a vision system to recognise balls and cars).

- **Integrate Techniques**

  A cognitive architecture is a structure, instead of a technique such as an algorithm, a representation or a mechanism. It integrates multiple techniques, as well as programmability and flexibility [120, p. 11].

- **Decision Making and Interaction**

  A cognitive architecture must support decision making (or action selection) and interaction within an environment [120, p. 6]. Other systems may not make decisions and interact with an environment (e.g. knowledge classification system, such as Cyc [132]).

2.4.1 Comparison with Other Types of Software

In the research literature, not only the term cognitive architecture is used interchangeably with the term agent architecture, but work in cognitive architectures are sometimes evaluated and compared with other work in agent frameworks, control architectures and software architectures. These other work may or may not describe a cognitive architecture depending on whether or not they involve the three key aspects listed previously: general intelligence, multiple techniques and decision making and interaction.

Laird [120, p. 6] proposes to distinguish a cognitive architecture from knowledge classification system (e.g. Cyc [132]), AI languages (e.g. Lisp [141]), toolkits and frameworks (e.g. blackboard system [157]).

Similarly, I propose that to distinguish a cognitive architecture from the following software (see Figure 2.2):

1. **Software Application / Program**

   A software application encapsulates models, instructions and some knowledge needed to perform a particular task. Examples of software applications are a robot soccer application, a robot companion application and a self-driving robotic car application.

   A cognitive architecture does not contain models, instructions and knowledge to perform the tasks. Instead, it is a structure of various techniques that govern the models, instructions and knowledge irrespective to the tasks. When

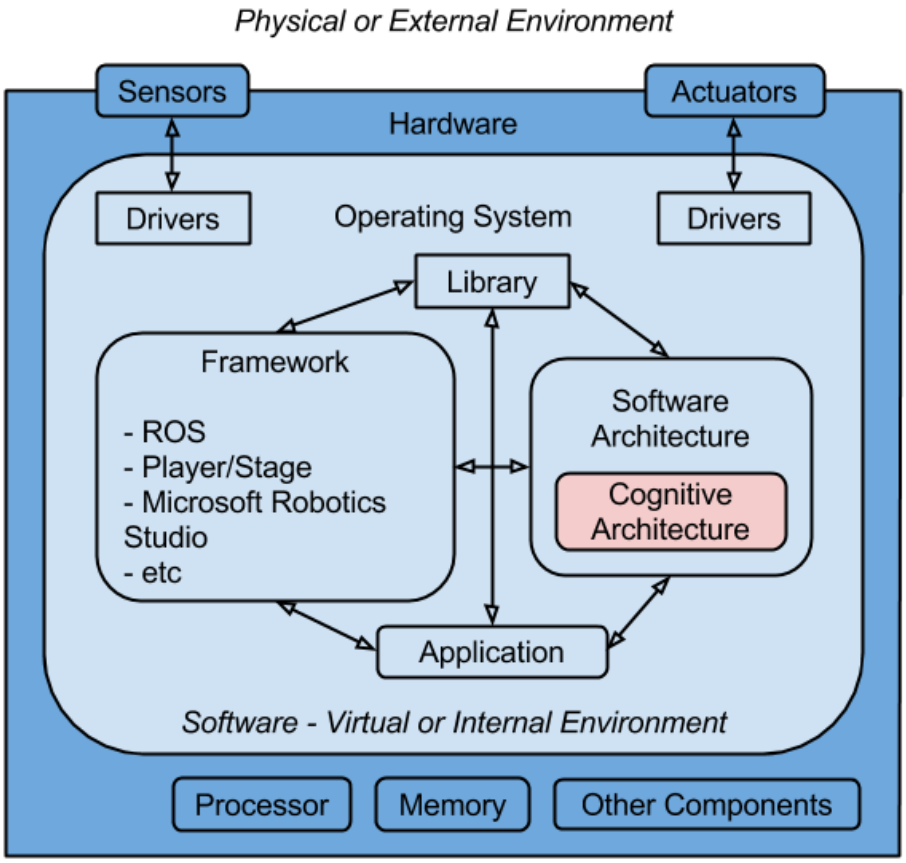


Figure 2.2: Cognitive Architecture and Other Software

running by itself, it cannot achieve a task without the appropriate software application.

As an analogy, a cognitive architecture is like a building, a software application or a program is like a collection of furnitures inside the building. Russell and Norvig formulates the relationship between agents, cognitive architecture and program as the following simple equation [192, p. 36]: agent = architecture + program.

2. **Software Libraries**

A software library is a collection of common functions, subroutines and data structures that is called or managed by a software application. Its purpose is to allow software codes to be shared and reused by different software applications without much need to redevelop the codes. Examples of libraries are the standard C programming math library, a http library and an image processing library.

A cognitive architecture is not called or managed by a software application. Instead, it manages the software application. It governs the models, instructions and knowledge in the software application.

3. **Operating system**

   An operating system manages and provides an interface between hardware and software applications, such that the same hardware can be shared by different software applications and the same software application can be used on different hardware. Its purpose is to allow hardware to be replaced or upgraded without much need to redevelop the software applications. Examples of operating systems are Microsoft Windows, Ubuntu Linux and Google Android.

   A cognitive architecture does not provide an interface between hardware and software applications. Instead, it provides a decision making or action selection mechanism in order to produce intelligent behaviours.

4. **Software Framework**

   A software framework provides common features more than a library, such as runtime environment, programming paradigms, coding objects and templates, in order to facilitate the development of software applications. *Framework* is defined by the Oxford dictionary of computing as "a template for the development of software applications or components" [58]. Examples of software frameworks are .NET application framework, Django web framework and ROS robotic framework.

   A cognitive architecture is similar to a software framework, but its focus is to provide a decision making or action selection mechanism in order to produce intelligent behaviours.

5. **Software Architecture**

   A software architecture is defined by ISO/IEC as "fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution" [100]. It is also defined by Microsoft computer dictionary as "the design of application software incorporating protocols and the means for expansion and interfacing with other programs" [181]. Examples of software architectures are blackboard software architecture [79, 97], client-server software architecture [79] and publish-subscribe software architecture [67].

A cognitive architecture is a subset of software architectures that focuses on general intelligence, multiple techniques and decision making. It should be distinguished from a software architecture when referring to the specific focus.

6. **Agent Framework**

   An agent framework is a subset of software frameworks. Examples of agent frameworks are JADE [29], NetLogo [241], Cougaar [98]. As discussed above, a cognitive architecture is not a software framework, hence it is also not an agent framework.

7. **Control Architecture**

   A control architecture is a subset of software architectures. Depending on the context in which a control architecture is used, it can be as same as a cognitive architecture if it involves general intelligence, multiple techniques and decision making.

In practice, a cognitive architecture may include the role of other software, such as an operating system, a software framework or a software architecture. In addition, it may be developed using a software framework or a software architecture. For example, A recent version of ASMO is developed using *Robot Operating System* (ROS) framework [183], which itself is based on a publish-subscribe architecture [67].

Note that the name of the software can sometimes be misleading. For example, Robot Operating System (ROS) is not an actual operating system despite its name 'operating system'. Instead, it is a robotic framework that runs on a Linux operating system.

2.5 Evaluation Criteria for Cognitive Architectures

Various sets of requirements or criteria have been proposed in the research literature for designing and evaluating a cognitive architecture [120, 127, 216, 10, 6, 42]. Currently, there is no widely accepted set of criteria, because they have been developed with a specific goal in mind. In this dissertation, I identify four criteria *common* to those sets of criteria that are important for service robotic tasks:

1. **Versatility**

   Versatility can be defined in terms of the difficulty encountered in developing intelligent systems across a given set of tasks and environments [127]. The less effort and time it takes to get an architecture to produce intelligent behaviour in those environments, the greater its versatility. This criterion is related to the programmability criterion proposed by Alami et al. [6]. The goal is that a robot should be easily programmed to achieve multiple tasks. Its functions should be easily combined based on the task to be executed.

2. **Reactivity**

   Reactivity can be defined in terms of the speed the robot responds to situations or events, or in terms of the probability that the robot will respond to a given situation [127]. The more rapidly an architecture responds, or the greater its chances of responding, the greater its reactivity. Robots have to take into account events with time bounds compatible with the correct and efficient achievement of their goals (including their own safety) [6]. This criterion is related to the multiple goals criterion proposed by Brooks [42]. Robots often are required to respond to multiple events and goals simultaneously. The cognitive architecture must be responsible for serving high priority goals, while still serving other necessary low-level goals.

3. **Robustness**

   Robustness can be defined in terms of the frequency that the architecture keeps operating, when there is a failure within its domain. The more often the architecture can recover from failure, the greater its robustness. A cognitive architecture should be able to adapt and cope with failure by relying on parts that are still functional [42]. It should be able to exploit the redundancy of the processing functions and its control is required to be decentralised to some extent [6]. This criterion is similar to the extended operation criterion proposed by Langley, Laird and Rogers [127]. Robots must be robust enough to keep from failing when they encounter unexpected situations and to keep from slowing down as they accumulate experience over long periods of time.

4. **Extensibility**

   Extensibility [42] can be defined in terms of the difficulty to change or add new functions and knowledge to improve performance. The easier it is for

the cognitive architecture to change or add parts, the greater its extensibility. Cognitive architecture should make learning possible [6]. This criterion is similar to the improvability criterion proposed by Langley, Laird and Rogers [127]. It is measured in terms of the agent's ability to perform tasks that it could not handle before the addition of knowledge. It involves the agent learning from the experience in the environment or with the internal processes.

A cognitive architecture that meets all these criteria will support service robots in undertaking tasks in open and complex environments. The more criteria a cognitive architecture meets, the greater it can govern service robots in undertaking tasks in open and complex environments, the faster it can accelerate the development, deployment and adoption of service robots in those environments.

Chapter 3

Existing Cognitive Architectures

In the previous chapter, I have described the background of robotics and cognitive architecture, as well as identified criteria to evaluate cognitive architectures. In this chapter, I identify the gaps in current cognitive architectures based on these criteria. I start by analysing major approaches in cognitive architectures based on the way knowledge is designed (see Section 3.1) and the way knowledge is used (see Section 3.2). I follow by analysing the current state of the art of cognitive architectures in Section 3.3. Finally, I summarise all the analyses and identify the gaps in Section 3.5.

3.1 Approaches based on Knowledge Design

Cognitive architectures are typically divided into three approaches based on the way knowledge is designed [145, 71]:

1. **Explicit-based Approach**

   An explicit-based approach is also known as a deliberative [246], top-down [145], symbolic [145], logical [145], neat [145] or knowledge-based approach. In an explicit-based approach, agents are designed to make decisions by deliberately reasoning about their decisions. They can provide reasons for their decisions if required. Each decision or action is a result of careful consideration, reasoning and/or planning. Agents are provided with necessary knowledge about the tasks so that they can (and are required to) use the knowledge to derive their decisions. The knowledge has to be encoded either through a

single (i.e. uniform) representation or multiple (i.e. heterogeneous) representations. A symbolic logic is typically used to encode the knowledge, but it is not a requirement of the explicit-based approach.

2. **Implicit-based Approach**

   An implicit-based approach is also known as a reactive [246], bottom-up [145], subsymbolic (i.e. connectionist) [145], analogical [145], scruffy or behaviour-based [43, 18, 145] approach. In an implicit-based approach, agents are designed to directly follow specified actions and decisions without deliberately reasoning about their decisions. They simply sense and react to the environment. They are not provided with complete knowledge about the tasks, instead they are provided with the actions to perform that have been planned by developers beforehand (i.e. implicit approach). They cannot (and are not required to) use their knowledge to derive their decisions since they do not have the complete knowledge.

3. **Hybrid Approach**

   A hybrid approach is a mixture of both explicit-based and implicit-based approaches [246, 145]. In a hybrid approach, agents are designed to make *some* levels of decisions by deliberately reasoning about their decisions and *other* levels of decisions by simply following specified actions. They are provided with both knowledge about the tasks and actions to perform so that they can use both depending on the situations or the design, e.g. explicit-based for high-level decisions and implicit-based for low-level decisions.

Each approach has its own strengths and weaknesses. An explicit-based approach tends to have higher computational cost than an implicit-based approach because every decision has to be reasoned deliberately. In addition, it tends to be more complicated than an implicit-based approach because complete knowledge has to be provided. In contrast, an implicit-based approach tends to be less flexible than an explicit-based approach because complete knowledge is not provided (hence behaviours cannot be modified as much as in an explicit-based approach). A hybrid approach has strengths as well as weaknesses of the two approaches, thus the focus is on how to balance the two approaches.

3.2 Approaches based on Knowledge Utilisation

In contrast to the typical classification described above, Russell and Norvig [192, pp. 46–54] classified approaches to govern agents (i.e. cognitive architectures) based on the way knowledge is utilised to perform actions (i.e. based on mechanisms of action selection). However, their classification does not include some state of the art of approaches described by Pirjanian in his taxonomy [176]. In this section, I reorganise all the approaches into a new classification to include the approaches described by Russell and Norvig and the approaches described by Pirjanian (see Figure 3.1). In this new classification, cognitive architectures can be divided into two major approaches based on the way knowledge is utilised:

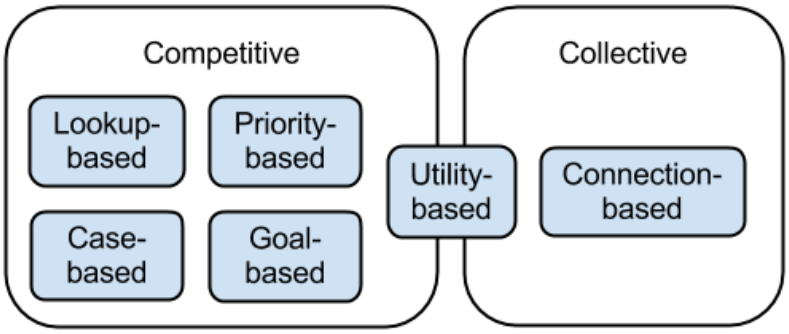


Figure 3.1: Decision Making Categories

1. **Competitive Approach**

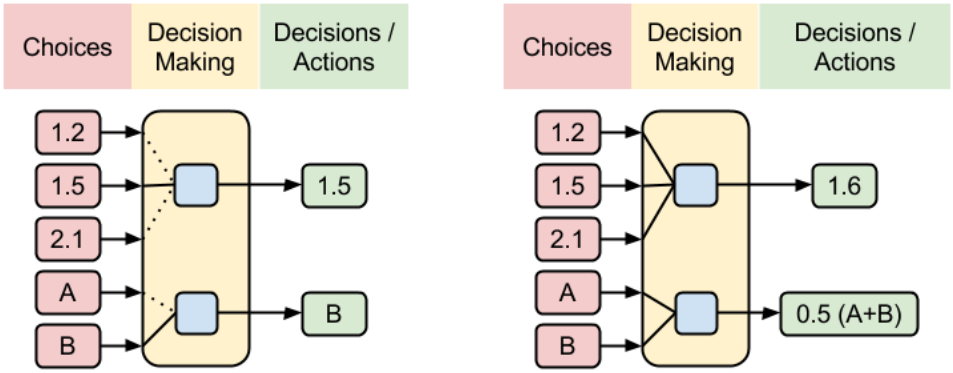
   A competitive approach is called an arbitration approach in Pirjanian's taxonomy [176, p. 9]. It describes the use of a single criterion to make decisions (i.e. single criteria decision making). In a competitive approach, actions are not decided by compromising all the available choices or alternatives. Instead, a choice is selected among the available choices and actions are decided based on this selected choice (see Figure 3.2a). For example, given three paths with different travel time, distances and obstacles, an agent may decide to walk on the path with the shortest time, despite the path may have a longer distance and more obstacles.

   I further divide the competitive approach into five types, namely lookup-based, case-based finite state machine, priority-based or hierarchical-based, goal-based and utility-based competitive approaches.

2. **Collective Approach**

   A collective approach is called a fusion approach in Pirjanian's taxonomy [176, p. 9]. It describes the use of multi-criteria to make decisions (i.e. multi-criteria decision making). In a collective approach, actions are decided by compromising all the available choices or alternatives (see Figure 3.2b). For example, given three paths with different travel time, distances and obstacles, an agent may decide to walk on the path with the average travel time, distance and obstacles (i.e. compromised between the travel time, the distance and obstacles).

   I further divide the collective approach into two types, namely connection-based and utility-based collective approaches.



(a) Competitive (b) Collective

Figure 3.2: Competitive vs. Collective Decision Making

Note that not all cognitive architectures fall neatly into a category. A cognitive architectures can have multiple decision making mechanisms that fall into a few categories.

In the following subsections, I describe each of the decision making approaches. I describe what agents are provided with, describe how actions and criteria preferences are determined, list example(s) of decision making mechanisms that fall into the approach and discuss strengths and weaknesses of the approach.

Throughout the following discussions of strengths and weaknesses of each approach, a number of dimensions will recur as common characteristics that best differentiate the approaches. For the sake of clarity, they are listed here:

1. **Computational complexity**: the complexity of computing decisions

2. **Knowledge complexity**: the complexity of constructing the required knowledge

3. **Knowledge size**: the size of knowledge required to achieve a task

4. **Real-time modification**: the capability to modify provided knowledge in real-time

5. **Flexibility**: the ease of modifying knowledge in order to meet new task requirements

3.2.1 Lookup-based Approach

In a lookup-based approach [192, pp. 46–47], agents are provided with a list of pre-computed decision making instructions describing every action to perform (i.e. output) given a situation or what the agent senses (i.e. input). In this case, actions are determined simply by reading and looking up instructions in the list, rather than performing real-time computation. In order to pre-compute the list, developers have to know all possible inputs that will be encountered by an agent in an environment (i.e. a complete model of the world within the operational range of the agent). Criteria and preferences to make decisions are implicitly determined by the list (i.e. specifying a particular action to perform over other actions in the list implicitly means that a criterion is preferred over others). The list can be implemented as a lookup table.

Inputs and outputs have to be described in the list explicitly. They can be continuous or discrete depending on the environment. An agent can be provided with a list of discrete inputs and outputs even though the agent works in a continuous environment. However, these continuous inputs and outputs have to be digitised or converted to discrete inputs and outputs in real-time so that they can be used by the list. This conversion requires a real-time computation (often a small amount), which is not the aim of the lookup-based approach.

Table 3.1 shows an example of incomplete instructions for a robot soccer scenario. Inputs and outputs in the table are continuous, because the robot soccer is a continuous domain. In the actual complete table, an action will be provided for every increment of x-coordinate in robot position, y-coordinate in robot position, yaw angle (i.e. θ) in robot position, x-coordinate in ball position and y-coordinate

in ball position.

No	Sense					Action		
	Robot Position			Ball Position		Leg Speed		Head
	X	Y	θ	X	Y	Forward	Turn	Angle
1	3.99	3.74	15.18°	-	-	-	-	−45°
2	6.32	-4.18	139.99°	6.04	0.03	0.16	-0.82	−87.02°
3	0.68	0.34	91.42°	8.74	0.88	-0.98	0.05	90°
4	0.69	0.34	91.42°	8.74	0.88	-0.98	0.05	90°

Table 3.1: Lookup Table Example

There are four instructions described in the table. Number 1 shows that the robots can sometimes perform only one action. Number 2 describes the instruction to chase and track the ball when the distance to the ball is near. Number 3 describes the instruction to defend an attack while tracking the opponent's goal when the ball is far. Number 4 shows that an instruction has to be provided explicitly for a different situation although it is very similar to the instruction in Number 3.

Strengths. Decision making can be performed very quickly because the complexity of its computation is reduced to memory retrieval once the list is defined and tabulated.

Weaknesses. The complexity to construct the list is exponential. *Exact* knowledge about the situation has to be provided so that it can be searched in the list. It would be difficult to specify exact knowledge for all possible situations, especially in continuous or partially observable environments. The performance of agents in continuous environments is determined by the resolution of the discretisation of the environment.

3.2.2 Case-Based Finite State Machine Approach

In a case-based Finite State Machine (FSM) approach (called a state-based approach in Pirjanian's taxonomy [176, pp. 11–16]), tasks are divided into a finite number of cases (represented as states) where a behaviour is developed to handle each case. Agents are provided with the states, state transitions and state structure (i.e. connection of states). A state is activated when a defined condition is matched, such

as ball position is greater than 0.0 or an opponent robot is seen. There is only one state active at a time and actions are determined based on the active state. Criteria preferences are implicitly determined by the structure of the states (i.e. being in one state over others means that a criterion is preferred over others). The complexity to construct the states can be linear or exponential depending on how independent the states are.

A state can cover a range of situations. For example, a state *ball_at_right* can be created to cover a range of ball positions with x-coordinate from 0.0, 0.1, 0.2, ..., 1.0, 1.1, ..., to n, where n is the end of the field (see Algorithm 3.1). Instead of providing a separate instruction for each different situation as in a lookup-based approach, some situations can be grouped and covered by a state, thus reducing the size of the lookup table.

```
1 if x-coordinate of ball position > 0.0 then
2 |   state ← state_ball_at_right
3 else
4 |   state ← state_ball_at_left
5 end
```

Algorithm 3.1: Case-based Finite State Machine Algorithm Example

There are two types of a state transition, namely deterministic and stochastic (or probabilistic) state transitions. In the deterministic state transition, action a performed in state s always results in the same new state s'. In the stochastic state transition, action a performed in state s will only have a particular probability of resulting in the state s' and thus may result in other states. Agents are provided with these probabilities of actions and outcomes in addition to states, state transitions and state structure.

Strengths. Decision making can be performed very quickly because it is determined based on the active state that can be retrieved immediately. Exact knowledge for each situation is not required. The size of the lookup table can be reduced to a number of states.

Weaknesses. The states and their structure are required to be predefined and cannot be updated in real-time. The performance of agents in continuous environments is determined by the resolution of the discretisation of the environment into states.

3.2.3 Priority-based or Hierarchical-based Approach

In a priority-based or hierarchical-based approach [176, pp. 10–11], agents are provided with a list describing priorities (or a hierarchy) of actions or criteria. Actions are determined based on these priorities. Unlike lookup-based and case-based approaches, criteria preferences are explicitly determined by the list. The complexity to construct the priority list can be linear or exponential depending on whether the priorities are independent or not independent, respectively. An example of priority-based mechanism is CP-nets [35].

Table 3.2 shows an example of priorities for a robot soccer scenario. Priorities or hierarchies can be different according to the conditions in the environment. They can be totally ordered (see numbers 5 and 6 in the table) or partially ordered if the total order is unknown (see numbers 3 and 4 in the table).

Number	Condition	Priority
1	Ball near	chase > defend, ball > opponent's goal
2	Ball far	defend > chase, opponent's goal > ball
3	Ball near, chase	time > distance, time > obstacle
4	Ball near, defend	distance > time, distance > obstacle
5	Ball far, chase	obstacle > time > distance
6	Ball far, defend	obstacle > distance > time

Table 3.2: Priority Table Example

The table shows that the robot prefers to chase a ball and track the ball than to defend an attack and track the opponent's goal respectively when the ball is near, but opposite otherwise. In chasing the near ball, the robot prefers the fastest route so that it can get to the ball as soon as possible. However, it prefers the shortest route when defending an attack even if the ball is near, so that it does not need to make a big turn which may leave an open area for the opponent to attack (see Figure 3.3). In the far ball condition, time and distance are not the main priorities. The robot prefers to chase the ball and defend an attack via the safest path (i.e. least obstacles).

A priority-based or hierarchical-based approach can be used with a case-based finite state machine approach, e.g. hierarchical finite state machine, in order to make decisions when more than one action can achieve the same goal.

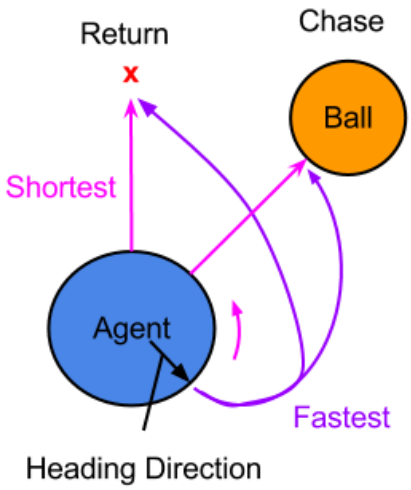


Figure 3.3: Priority Scenario

Strengths. The priority list can be modified in real-time independent of other knowledge. Decision making can be performed very quickly because searching a priority is linear. Exact knowledge for each situation is not required. The size of the lookup table can be reduced to a priority list.

Weaknesses. Priorities or hierarchies in this priority-based or hierarchy-based approach can sometimes be unclear, difficult to determine or inconsistent, e.g. $time > distance > obstacle > time$. Although a list of priorities can be modified, it is not flexible enough to meet new task requirements by modifying only priorities or hierarchies because knowledge related to the actions (e.g. outcomes or effects) are unknown. This priority-based or hierarchy-based approach does not have a measurement of how important a priority or hierarchy is (i.e. there is no notion of distance). For example, it cannot capture the difference between time that is twice preferred than a distance and time that is three times preferred than a distance.

3.2.4 Goal-based Approach

In a goal-based approach [192, p. 52], agents are not provided with the right actions to perform. Instead, they are explicitly provided with goals and action descriptions, so that they can plan and choose an action themselves that has the effect of satisfying the given goals. Both actions and criteria preferences are determined by the knowledge about goals and actions. An example of a goal-based decision making

mechanism is the STRIPS planner [70].

Table 3.3 shows an example of action descriptions for a robot soccer scenario, where $loc(o)$ is the location of object o. The search ball, chase ball and kick ball actions will be chosen if the robot's goal is to attack, because no other action can achieve the goal. However, the robot cannot choose an action (i.e. action conflict) if the goal is to defend. There is more than one action that can achieve the goal to defend, i.e. either chase and steal the ball or defend and block the ball. Thus, the robot needs criteria preferences to choose a preferred action over other conflicting actions.

Action	Conditions	Effects
Search ball	$\neg loc(ball)$	$loc(ball)$
Search opponent's goal	$\neg loc(opponent\_goal)$ $\wedge \neg loc(own\_goal)$	$loc(opponent\_goal)$
Calculate own / opponent's goal	$loc(opponent\_goal)$ $\vee loc(own\_goal)$	$loc(opponent\_goal)$ $loc(own\_goal)$
Calculate block position	$loc(ball) \wedge loc(own\_goal)$	$loc(block)$
Chase ball	$loc(robot) \neq loc(ball)$ $\wedge loc(ball)$	$loc(robot) = loc(ball)$
Defend	$loc(robot) \neq loc(block)$ $\wedge loc(block)$	$loc(robot) = loc(block)$
Kick ball	$loc(robot) = loc(ball)$	$attacked$
Steal ball	$loc(robot) = loc(ball)$	$defended$
Block ball	$loc(robot) = loc(block)$	$defended$

Table 3.3: Action Description Table Example

The goal-based approach can be used with the priority-based approach to solve an action conflict (called conflict resolution) – that is, to help make the decision when there is more than one action that can achieve the same goal. Expert systems are an example of goal-based approach systems that often use a priority-based approach for conflict resolution.

Strengths. Reasoning for the decisions can be provided and traced. Goals and effects of actions can be modified in real-time independent of other knowledge.

The goal-based approach is flexible because the goals and effects of each action are known and can be modified to meet new task requirements.

Weaknesses. Decision making is slower than the lookup-based, case-based and priority-based approaches because a decision cannot simply be retrieved, but must be reasoned deliberately. Explicit knowledge of goals and effects of each action are required to be provided. A separate mechanism is required for conflict resolution. Similar to a priority-based or hierarchy-based approach, this goal-based approach does not have a measurement of how effective a goal has been achieved.

3.2.5 Connection-based Approach

In a connection-based approach, tasks are modelled as layers of connected nodes. Agents are provided with the nodes and the right actions that are encoded as weights. Actions are determined by linearly calculating inputs and weights. Similar to a lookup-based approach, agents are designed to simply lookup the predetermined right actions. An example of a connection-based decision making mechanism is a neural network [192, p. 727].

A neural network can have an input layer, an output layer and any number of hidden layers. A layer can contain any number of nodes. Both sensations of the situations and actions are represented as nodes in the input layer and output layer respectively. They are connected through hidden layers in between, thus adjusting the weight of nodes in the hidden layers will affect the decision making.

Nodes act as gateways between layers to activate other nodes in the next layer based on the values of nodes in the previous layer. They are activated based on their activation function that depends on their weights. The activation function of node i in layer j is denoted by a_i^j or $a(i, j)$.

Weights are real numbers that have no semantic meaning. Similar to a lookup-based approach, situations and actions to perform are predetermined. However, they are not provided in a list or a lookup table, instead their relationships are converted into weights. Mathematically calculating the sensations with the weights will provide the specified actions. Weights are difficult to be specified manually, so they are typically learned from data or observations through supervised learning.

Figure 3.4 shows an example of neural network nodes for a robot soccer scenario. The nodes in the input layer are the values of robot position (i.e. x-y coordinate and yaw angle) and ball position (i.e. x-y coordinate). The nodes in the output layer are *chase/defend* node and *ball/opponent goal* node.

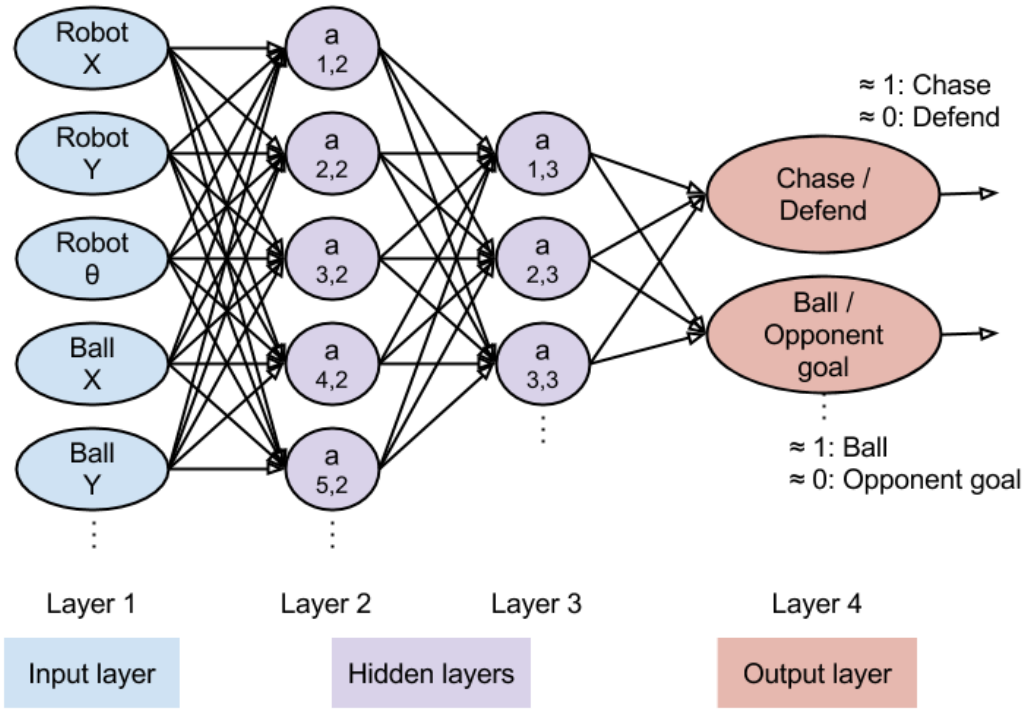


Figure 3.4: Neural Network Example

After calculating all of the activation functions, the action to chase a ball will be chosen if the value of the *chase/defend* node is close to 1 and the action to defend the attack will be chosen otherwise. In addition, the action to track the ball will be chosen if the value of *ball/opponent goal* node is close to 1 and the action to track the opponent's goal will be chosen otherwise.

Strengths. Decision making can be performed very quickly because the complexity of its computation is reduced to the linear calculation of weights once the weights are learned.

Weaknesses. The weights cannot be modified in real-time independent of other knowledge. Weights are difficult to specify manually. They can also be difficult to learn in some tasks because not all tasks can be successfully learned through supervised learning, i.e. other techniques may be required. The complexity to learn the weights can be exponential.

3.2.6 Utility-based Approach

A utility-based approach [192, p. 53] can be used for both competitive and collective decision making. Agents are provided with the utility of each action or criteria. Actions are determined based on the aggregate functions of the utilities. Examples of utility-based mechanisms are expected utility, multi-attribute utility theory (MAUT), analytical network processes (ANP) [227], soft-constraints [189, p. 9] and voting mechanisms.

There are different kinds of aggregate functions depending on whether the utility-based approach is used for competitive or collective decision making. Utility-based competitive decision making has an aggregate function that will select a single utility over others, such as the maximum (i.e. highest) or minimum (i.e. lowest) functions. In Pirjanian's taxonomy, this kind of approach is described as the winner-take-all approach [176, pp. 16–18]. In contrast, utility-based collective decision making has an aggregate function that will take the influences of multiple utilities, such as the sum or average functions. In Pirjanian's taxonomy, this kind of approach is further divided into voting, fuzzy and superposition approaches [176, p. 10].

Utilities allow actions and criteria to be modelled quantitatively. They capture the desirability of actions, outcomes of actions or criteria. The preferred actions when there is more than one action that can achieve the same goal can be determined by measuring their utilities. Not only can agents determine the preferred actions or criteria, but also they know how much the actions are preferred.

Utilities and utility functions do not necessarily need to be fixed in this utility-based approach. However, they can be fixed if desired at the design time or real-time. They are normally fixed when the policy used in the utility-based approach is optimal. A utility-based approach with fixed utilities can operate like a priority-based or hierarchy-based approach where the utilities define the priorities or hierarchies of actions or criteria.

In stochastic environments, a probability theory can be used in a utility-based approach to model uncertainty. Agents are provided with the probabilities of each action or criteria in addition to the utilities. Expected utility is an example of a probabilistic model of utility-based decision mechanisms. Actions are determined based on the weighted probability sum of all utilities.

A utility-based approach can be combined with a case-based finite state machine

approach. Examples of mechanisms that combine a utility-based approach with a case-based finite state machine approach are decision tree, Markov Decision Process (MDP), Partially-Observable Markov Decision Process (POMDP) and reinforcement Q-learning. These mechanisms can have fixed utilities and do not need to recompute the utilities in real-time when their policies are optimal.

Strengths. A utility-based approach can be used for both competitive and collective decision making. Utilities can be modified in real-time independent from other knowledge. Reasoning for decisions can be provided in terms of utility or number. Conflict resolution can be determined by measuring the utilities of the decisions. Unlike a priority-based or hierarchy-based approach and a goal-based approach, utilities can be used as a measurement of how effective a decision is.

Weaknesses. Decision making can be slower than lookup-based, case-based and priority-based approaches because a decision cannot simply be retrieved, but must be calculated (unless utilities are fixed). Explicit knowledge of the utility (and the probability for the stochastic environment) for each action is required to be provided.

3.3 Current State of The Art

In the previous sections, I have analysed major approaches in cognitive architectures based on knowledge design and knowledge utilisation. These approaches have been employed by the current state of the art in cognitive architecture. In this section, I analyse the current state of the art in cognitive architecture.

There is an enormous number of existing and developing cognitive architectures: there is almost a new cognitive architecture developed for each new kind of task. A *complete* systematic analysis of all cognitive architectures is clearly not feasible within the constraints of this dissertation due to the number of cognitive architectures available, limited time and limited resources. Despite the fact that various cognitive architectures have been extensively compared and reviewed in previous work [127, 232, 248, 148, 18, 246], there are still many cognitive architectures that were not covered in those reviews.

In this dissertation, I analyse nine most well-known and significant architectures

(and common to the reviews in previous work), namely Soar, ACT-R, Subsumption, Situated Automata, PRS, Agent Network, ICARUS, 3T and Polyscheme. Researchers may have different opinions on which one they will refer to as a cognitive architecture. Currently, there is still no widely accepted definition of a cognitive architecture (see Section 2.4). Among the nine cognitive architectures listed previously, Subsumption, 3T and Polyscheme are probably the most common architectures that researchers would not agree on with each other (if any). Laird refers to 3T as a cognitive architecture [120, p. 10] and Polyscheme as a cognitive architecture that is closer to a framework [120, p. 7], but he considers Subsumption as a framework rather than a cognitive architecture [120, p. 6]. In contrast, Langley, Laird and Rogers [127] refer to Polyscheme as a cognitive architecture, and Vernon, Metta and Sandini [232] refer to Subsumption as a cognitive architecture. In this review, I refer and include all Subsumption, 3T and Polyscheme as a cognitive architecture.

The nine cognitive architectures (or their derivations) are still in use today, although many of them were *first* published over years ago. Unlike a number of recently developed cognitive architectures, these nine cognitive architectures have been proven to be effective for general problems, different domains and different physical robots.

The nine cognitive architectures also covers the trends of cognitive architectures described by Bryson [43] as seen in Table 3.4.

No.	In Bryson's Analysis	Representative In My Analysis
1	Behaviour-based architectures:	
	Society of mind	Polyscheme
	Subsumption	Subsumption
	Behaviour-based diversification	Agent Network
2	Multi-layered architectures	3T
3	PRS (beliefs, desires and intentions) approach	PRS
4	Soar	Soar
5	ACT-R	ACT-R

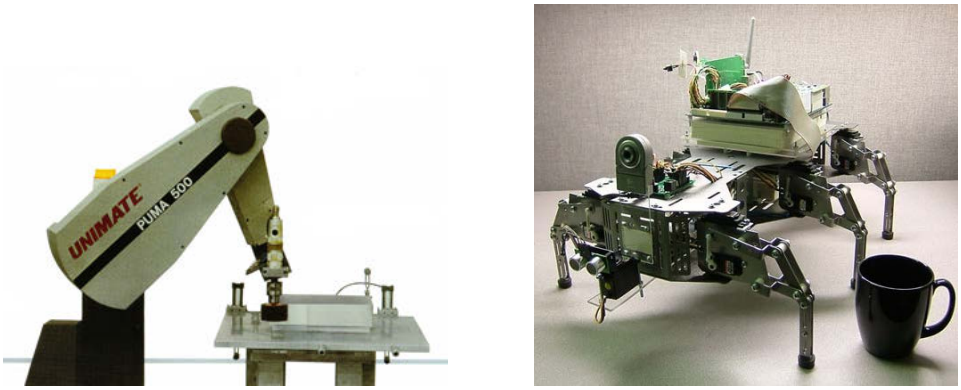
Table 3.4: Comparison to Bryson's Trend of Cognitive Architectures

In the following subsections, I evaluate each of the nine cognitive architectures against the four criteria described in Section 2.5, namely versatility, reactivity, ro-

bustness and extensibility. Details are given more to those architectures that provide practical examples (e.g. coding) and that have been implemented in real robotic applications. Other reviewers may disagree with the details of my evaluation, however, the general gaps in the architectures is unlikely to be controversial.

3.3.1 Soar

Soar [120, 121], initially developed in 1983, is one of the oldest and most widely recognised cognitive architectures. It is mainly used for cognitive modelling and governing intelligent agents. It was originally implemented in Lisp programming language and known as SOAR (all upper case) – the acronym for State, Operator and Result. Now, Soar has been implemented in C++ programming language and is no longer an acronym. It has been deployed in robots, such as the Unimation PUMA (the implemented system is called Robo-Soar) [122] and the Parallax HexCrawler [94], see Figure 3.5.



(a) Unimation PUMA [182] (b) Parallax HexCrawler [94]

Figure 3.5: Robots Governed by Soar

Soar's history can be traced back to and a direct successor of the General Problem Solver (GPS) created in 1959 [154]. GPS was designed to model human information processing by matching data with human performance (i.e. cognitive modelling). It has influenced STanford Research Institute Problem Solver (STRIPS) [70] which has successfully governed a physical robot called *Shakey* [158]. Both GPS and STRIPS use means-ends analysis to make decisions. However, GPS focuses on the cognitive modelling whereas STRIPS focuses on the planning of intelligent agents. Soar is like a combination of both GPS and STRIPS because it is not only used for cognitive modelling but also governing intelligent agents.

The Soar architecture is designed based on Newell's three theories, as listed as follows:

1. **Unified Theory of Cognition**

   The *unified theory of cognition* [153] proposes that all cognition and intelligent behaviours can be captured by a single set of mechanisms. In Soar, all tasks are treated as a problem that can be solved by searching and manipulating symbolic operators.

2. **Physical Symbol System**

   The *physical symbol system* theory [155] proposes that "a physical symbol system has the necessary and sufficient means for general intelligent action". A physical symbol system is a system that contains expressions and processes which can be created, modified, reproduced and destroyed to produce other expressions over time. An *expression* (also called symbol structure) contains physically related entities (called symbols) that obey the laws of physics. In Soar, knowledge about the tasks is encoded and manipulated in symbolic form.

3. **Problem Space Theory**

   The *problem space theory* [152] proposes that decision making and problem solving can be achieved by representing problems as states and operators in a space and finding operators through the space that can change the current state to the goal state. In Soar, situations (including goals) and actions are encoded as states and operators respectively. A task is solved by finding actions that could change the current situation to the desired situation (i.e. goal).

Soar consists of modules such as perception, action, working memory and long-term memory (see Figure 3.6). Inputs are retrieved from the environment by the perception module and held in the perceptual short-term memory (STM). Symbolic structures are extracted from perceptual short-term memory and added to Soar's working memory. The content of working memory is matched against procedural knowledge which is represented as production rules in procedural long-term memory.

During the matching process, preferences are generated by production rules to help the decision procedure to select an operator. An impasse will occur on the occasion when knowledge is inadequate to determine the operator (i.e. no operator

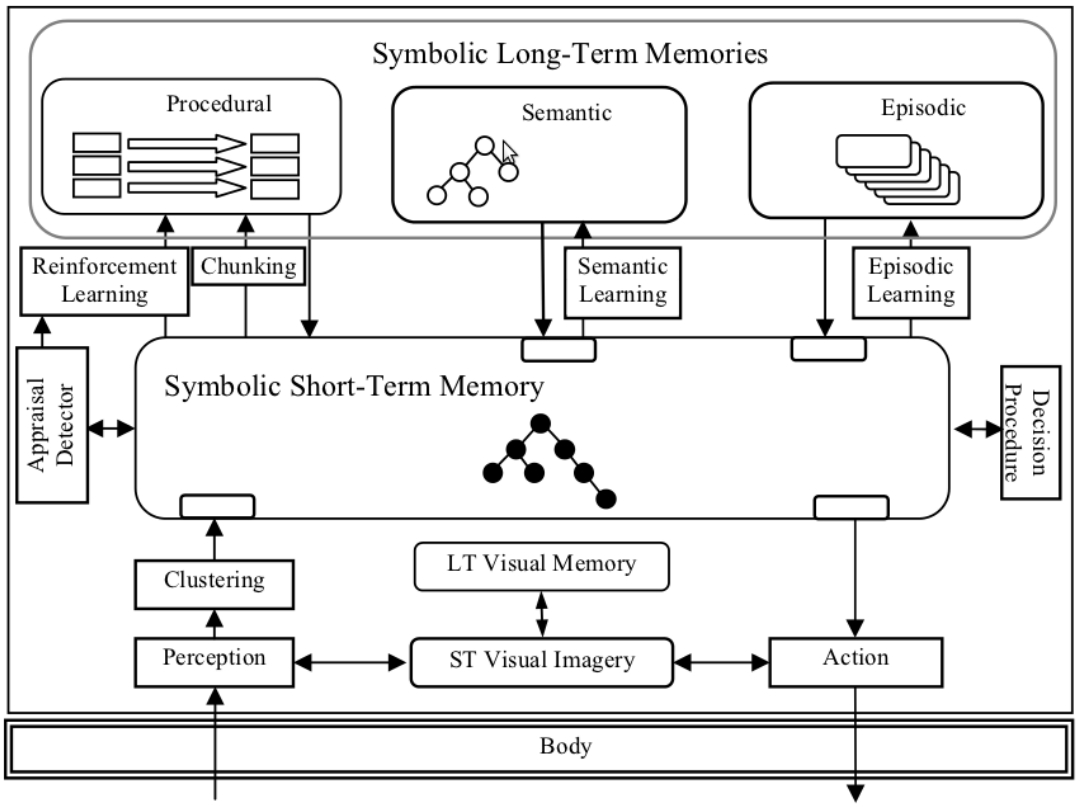


Figure 3.6: Soar Architecture [120]

can be selected), hence a new sub-goal will be created. Otherwise, an operator is selected and applied to change the content of the working memory (i.e. previous state is changed to new state).

Change in the content of working memory can initiate actions or trigger further knowledge from semantic or episodic memory. The new content is then rematched against production rules and the process is repeated until the new content is matched against the goal (i.e. the goal state has been achieved).

Versatility

Soar is not versatile. Recent versions of Soar has employed various techniques, such as symbolic representation, numeric preferences, mental imagery, appraisals and reinforcement learning. Each technique is carefully designed, so it can be integrated with other techniques. Soar is not capable of integrating undesignated techniques that are developed independently. In addition, Soar does not have an explicit mechanism to manage resources. It selects an operator with the assumption that the operator has enough resources to

be performed. As a result, robotic developers must accommodate resources manually when designing the procedural knowledge in order to avoid resource conflict.

Reactivity

Soar is not reactive. It does not guarantee that the action will be selected within a period of time despite that it typically has a fast execution cycle of accessing procedural knowledge and selecting an action. Its action selection is determined by knowledge search in the problem space. Reactive behaviour is lost if the knowledge search is unbounded. In addition, Soar cannot achieve multiple goals simultaneously because it chooses only a single action at a time.

Robustness

Soar is semi-robust. It can still generate partial plans from the available knowledge when some procedural knowledge is missing. However, Soar cannot exploit redundancy because actions are controlled by a central procedural system. Its control is centralised. In addition, there is no guarantee that Soar will not slow robots down since its problem space will grow larger as knowledge is accumulated.

Extensibility

Soar is extensible. Knowledge is separated from other components of the architecture. Adding procedural knowledge to improve tasks does not involve modifying any part of the architecture. In addition, Soar has chunking learning to create sub-goals when no operator can be selected. It also has reinforcement learning to tune procedural knowledge.

3.3.2 Adaptive Control of Thought – Rational (ACT-R)

The Adaptive Control of Thought – Rational (ACT-R) [12] is one of the oldest and most widely recognised cognitive architectures. It is mainly used for cognitive modelling. It is also known as Adaptive *Character* of Thought - Rational [3, G3]. It is the successor of several architectures that date back to 1973: Human Associative Memory (HAM) [14], ACT [13] and Adaptive Control of Thought (ACT*) [11]. It is implemented in the Lisp programming language, but alternative implementations

in Java and Python have been made available. It has been deployed in robots, such as the iRobot B21r [223] and the NASA Robonaut [208] (see Figure 3.7).

(a) iRobot B21r [223] (b) Nasa Robonaut [149]

Figure 3.7: Robots Governed by ACT-R

Like Soar, ACT-R is designed based on the production system approach and the unified theory of cognition assumption. It encodes knowledge using a set of if-then production rules and solves problems by using a single and fixed set of mechanisms.

Agents are provided with explicit knowledge that is divided into two types, namely, procedural and declarative knowledge. *Procedural knowledge* consists of rules and stepwise instructions to perform actions or to achieve goals, such as how to chase or kick a ball. They are encoded using a production structure. In contrast, *declarative knowledge* consists of facts, such as a ball is round or goalposts are stationary. They are encoded using a custom structure in ACT-R called *chunks*.

ACT-R [12] has a total of eight modules: visual, aural, manual, vocal, imaginal, declarative, goal and procedural (see Figure 3.8). The visual and aural modules perceive image and audio inputs from the external environment respectively. The manual and vocal modules produce motor and voice outputs to the external environment respectively. The imaginal module holds a current *mental* representation of the problem. The declarative module stores and retrieves the declarative knowledge to and from the memory. The goal module consists of the current goal and the agent's intention. Finally, the procedural module is a special module that retrieves the procedural knowledge to process information of other modules. All modules except the procedural module have a buffer that will only hold information in a single chunk (i.e. declarative knowledge). The procedural module does not have a buffer

and it is the only module that can examine and modify buffers in other modules.

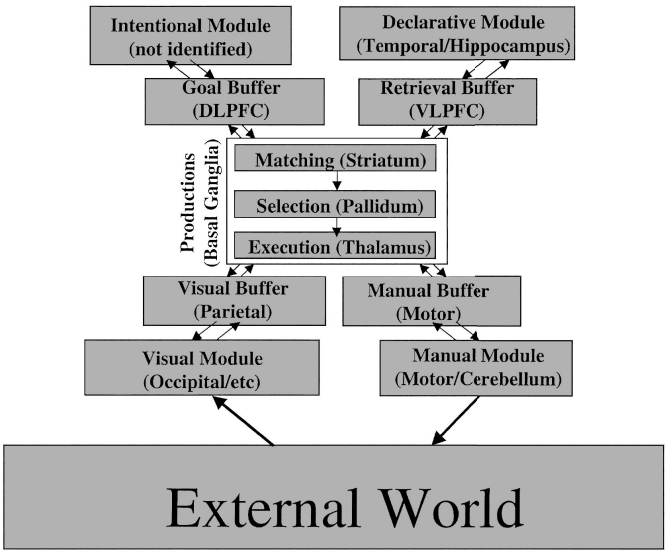


Figure 3.8: ACT-R Architecture

ACT-R's decision making is similar to Soar's, except that measures of utility (i.e. reward and cost) are used for decision making. Visual and aural modules perceive inputs from the environment and hold the inputs in their buffers. The decision making process cycle commences when there is content in the modules' buffers. The content is matched against procedural knowledge (i.e. production rules) in the procedural module. Multiple productions may be found and matched but only one production rule with the highest utility will be chosen and executed. This production rule will further update the content of the buffers and the cycle is repeated. Although each cycle is aimed to take about 50ms to complete, the cycle can take longer if there are retrievals from declarative memory involved. In this case, ACT-R does not guarantee that an action selection or decision making is performed every 50 ms.

The utility of a production rule is calculated by the Equation 3.1. It is the difference between the expected reward and the cost. It is positive if the expected reward is higher than the cost and negative otherwise. The expected reward is calculated by multiplying the success ratio by the reward. A higher success ratio will return a higher expected reward and vice versa.

$$U_i = \left(\frac{S_i}{S_i + F_i} G\right) - C_i \tag{3.1}$$

Where:

U_i is the utility value of production i
S_i is the number of success in achieving the goal using production i
F_i is the number of failure in achieving the goal using production i
G is the reward if the goal is achieved
C_i is the cost of using production i to achieve the goal

Versatility

ACT-R is not versatile. It has employed various techniques, such as symbolic representation, utility numeric representation, mental representation and reinforcement learning. Each technique is carefully designed in order to be integrated with other techniques. ACT-R is not capable of integrating undesignated techniques that are developed independently. In addition, ACT-R does not have an explicit mechanism to manage resources. It selects a production with the assumption that the production has enough resources to be performed. As a result, robotic developers must accommodate resources manually when designing the procedural knowledge in order to avoid resource conflict.

Reactivity

ACT-R is not reactive. It is modelled to match the execution cycle in humans, which is about 50 ms. It does not guarantee that the action will be selected within a period of time. In addition, it cannot perform simultaneous action selection. It chooses only a single action at a time.

Robustness

ACT-R is semi-robust. It can still generate partial plans from available knowledge when some procedural knowledge is missing. However, ACT-R cannot exploit redundancy because actions are controlled by a central procedural system. Its control is centralised. In addition, there is no guarantee that ACT-R will not slow robots down as declarative and procedural knowledge are accumulated.

Extensibility

ACT-R is extensible. Adding procedural knowledge to improve tasks does not involve modifying any part of the architecture. In addition, tasks can be improved easily by adjusting the cost and probability of the production rule

through reinforcement learning. ACT-R also involves production learning to combine a successive pair of productions into a single production.

3.3.3 Subsumption

Subsumption architecture [41, 42] was initially developed in 1986 to govern intelligent mobile robots. Its design was motivated by the lack of responsiveness in intelligent agents at that time. Governing agents in complex environments using approaches like Soar, STRIPS or ACT-R is computationally expensive, because every action is required to be deliberated. Subsumption avoids deliberative decision making using simple tables connecting sensory inputs to reactive actions.

Unlike Soar and ACT-R, agents are designed to follow the specified behaviours without undertaking deliberate reasoning about the consequences of their actions. The effort to provide knowledge about tasks is avoided [41]. Agents are provided with pre-designed actions to perform in response to sensor information instead of using explicit knowledge about goals and actions. Thus, they cannot (and are not required to) derive their actions deliberately because they do not have the required explicit knowledge.

Behaviours are divided into multiple layers starting from the simplest behaviour at the lowest layer to the most complex behaviour at the highest layer. Each layer has an individual goal to pursue. A higher layer can suppress and inhibit a lower layer, but not vice versa. For example, while the third and second layers can suppress and inhibit the second and first layers respectively, the first layer cannot suppress and inhibit the layers above (see Figure 3.9).

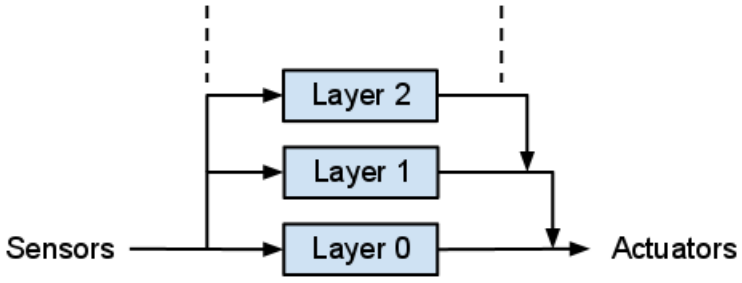


Figure 3.9: Subsumption Layer [42]

Each layer contains a set of modules. Each module is an augmented finite state machine with inputs, outputs, and a reset signal. Outputs of a module can be passed

to agents' actuators, passed to the inputs of the next module or used to suppress and inhibit the inputs and outputs of the next module respectively (see Figure 3.10).

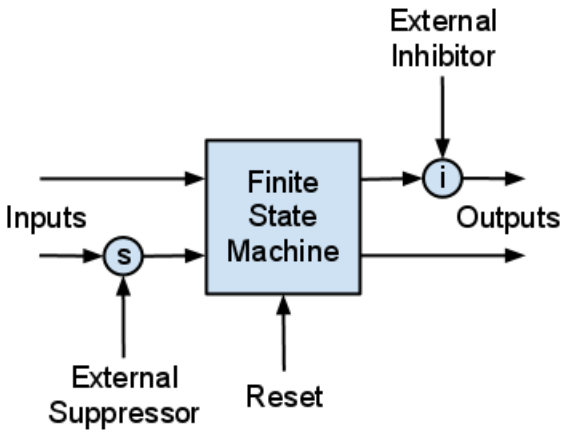


Figure 3.10: Subsumption Module [42]

Actions are performed simultaneously in multiple layers. They are prearranged hierarchically at design time. Hierarchies are used to determine a dominant action over subsumed actions when there is an action conflict.

Subsumption is designed for robustness. Layers are debugged and tested thoroughly from the lowest to the highest layer. Behaviours are not designed to be modified in real-time. Subsumption does not have a learning mechanism to change its behaviours.

Versatility

Subsumption is not versatile. Although independent techniques are easily created as modules layer by layer, connection between modules and layers are pre-arranged or wired manually during the design stage. In addition, Subsumption does not have an explicit mechanism to manage resources. Robotic developers must accommodate resources manually when arranging or wiring the modules and layers in order to avoid resource conflict.

Reactivity

Subsumption is reactive. It does not undertake deliberate reasoning to select actions. It has multiple layers that achieve individual goals concurrently. Each module can perform an action, thus multiple actions can be selected and performed at a time (i.e. simultaneous action selection).

Robustness

Subsumption is robust. Modules are built layer by layer after thorough debugging and testing. Modules in the lower layers will keep performing the next action despite a failure in the higher layers. In addition, Subsumption can exploit redundancy because actions can be controlled and executed by each module instead of by only a single module. Its control is decentralised and modular. Furthermore, Subsumption will not slow robots down because it does not accumulate experience and knowledge.

Extensibility

Subsumption is not extensible. Modules can be easily created and debugged, but adding or changing modules requires manual re-arrangement or rewiring of connections which takes much time and effort. In addition, Subsumption does not have learning mechanisms to improve performance and make the extension easy.

3.3.4 Situated Automata

Situated automata [105, 187] was initially developed in the mid-1980s for the purpose of governing intelligent agents. Its design motivation is similar to the Subsumption architecture, in that it seeks to compensate for the lack of responsiveness in agents. However, unlike Subsumption, knowledge is seen to be necessary instead of being avoided.

Knowledge is compiled into reactive behaviours in a form of digital circuits (i.e., logical gates) in order to stay responsive without being overwhelmed by the use of knowledge. Inputs from the environment are passed through the circuits and outputs are generated by the circuits without much computation (see Figure 3.11). Agents are provided with digital circuits instead of knowledge so that they can quickly react to the environment by looking up the outputs of the circuits without reasoning their knowledge and deliberating their decisions.

Several tools have been created to help make the development of agents easier, including Rex, Gapps and Ruler. Rex is a low-level language to specify digital circuits. Ruler and Gapps are declarative programming languages that are implemented on top of Rex to specify perception and action (or goal) of an agent respectively.

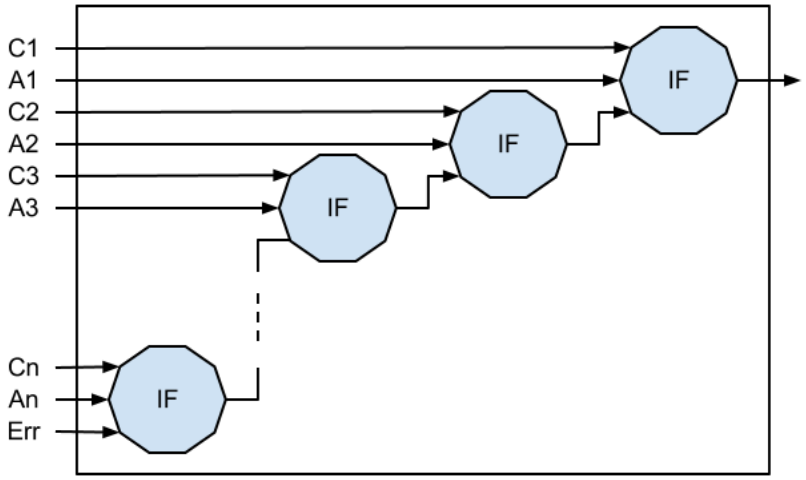


Figure 3.11: Circuit Generated by Gapps

Versatility

Situated automata is not versatile. The Rex, Gapps and Ruler custom languages can help the development of agents, but the situated automata cannot integrate techniques that cannot be described by the languages. In addition, robotic developers have to manually accommodate resources in the design of the circuits to avoid resource conflict.

Reactivity

Situated automata is reactive. Knowledge is compiled into digital circuits, which can generate fast outputs. In addition, Gapps supports prioritised goal lists and it can express multiple goals by using conjunctive expressions. Thus, it allows Situated automata to perform simultaneous outputs.

Robustness

Situated Automata is robust. Gates in the circuit will keep receiving inputs and producing outputs (i.e. actions) when there is a failure in other gates. In addition, Situated Automata can exploit redundancy because actions can be controlled by each gate instead of by only a single gate. Its control is decentralised. Furthermore, Situated Automata will not slow robots down because it does not accumulate experience and knowledge.

Extensibility

Situated automata is not extensible. Adding or changing functions requires the entire circuit to be recompiled. In addition, Situated automata does not

have learning mechanisms to improve performance and make the extension easy.

3.3.5 Procedural Reasoning System (PRS)

Procedural Reasoning System (PRS) [85, 84] was initially developed in 1986 for the purpose of governing intelligent agents. Its development has been continued and extended by different institutes, which results in following architectures: distributed multi-agent reasoning system (dMARS) [62], JACK [44] and CoJACK [160].

PRS is designed based on Bratman's theory of belief, desire and intention (BDI) [37]. Although PRS is not the only architecture based on BDI approach, it remains the most widely known example of BDI-based architecture. PRS consists of five major components (see Figure 3.12):

1. Database

   Database contains representations of current beliefs or facts about the world. That is, it contains a set of facts that is believed to be true at the current instant of time. The beliefs are described using first-order predicate calculus. Some of the beliefs may be provided initially by the developers.

2. Goals

   Goals express a set of current desires to be realised. Each goal is denoted by an action type or a set of state sequences described using a temporal predicate. For example, the expression ($walkab$) denotes the set of state sequences that embody walking actions from point a to point b.

3. Procedures

   Procedures (also called knowledge areas) represent the knowledge about how to accomplish given goals or react to certain situations. They express a declarative fact about the effects of performing certain sequences of actions under certain conditions. Each procedure has a body and an invocation condition. The body of the procedure is represented as a graphic network and can be viewed as a plan or plan schema. It consists of possible sequences of subgoals to be achieved instead of possible sequences of primitive actions. The invocation condition describes under what situations the procedure is useful.

4. Stack

   Stack represents the working memory or short-term memory of the architecture. It expresses the current intention of the system based on available goals and beliefs. It holds the selected procedure that is going to be executed by the interpreter.

5. Interpreter

   An interpreter is the inference mechanism or reasoner that controls the entire flow of the architecture. Its tasks involve performing consistency maintenance routines, finding procedures that are relevant to goals, choosing one procedure among relevant procedures, placing a procedure onto the process stack and executing a procedure.

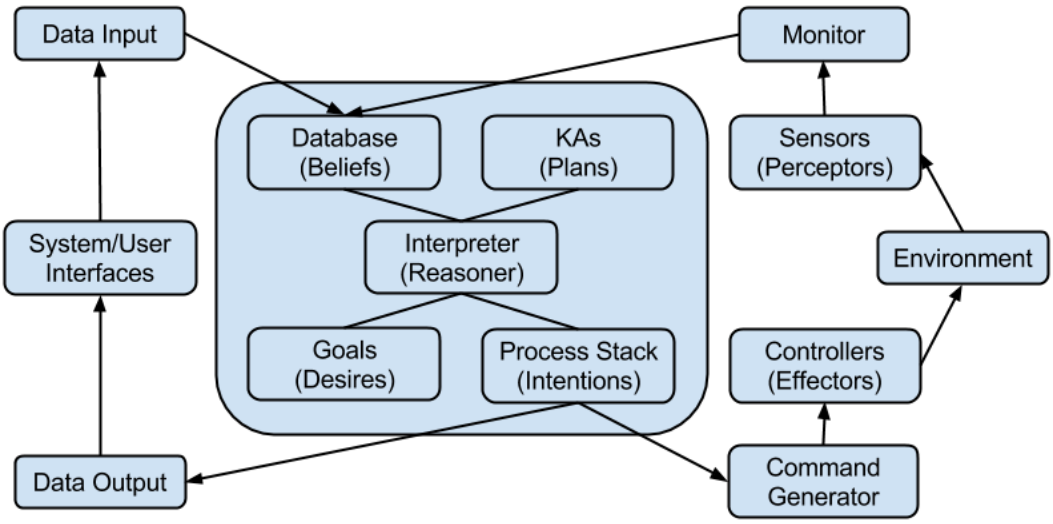


Figure 3.12: Procedural Reasoning System

PRS's decision making cycle starts every time a new goal is pushed onto the goal stack or a new belief is added to the database, which may be caused by changes in the environment. When a new goal is pushed, the interpreter finds procedures that are relevant to the new goal. When a new belief is added, the interpreter performs consistency maintenance routines that may activate relevant procedures. In both situations, the interpreter then chooses one procedure among the relevant procedures and places the selected procedure on the process stack. Finally, it executes the procedure that is placed in the process stack. Executing this procedure will derive new sub-goals and beliefs, thus the cycle is repeated.

Versatility

PRS is not versatile. Tasks must be able to be represented by beliefs, desire and intention. PRS cannot integrate undesignated techniques. In addition, procedures selected by PRS are assumed to have enough resources to be performed. Robotic developers have to manually ensure that resources are available for the procedures in order to avoid resource conflict.

Reactivity

PRS is semi-reactive. It finds and executes relevant procedures only when sufficient information is available. It expands plans dynamically and incrementally. Its planning can be interrupted whenever the situation demands. However, it cannot achieve multiple goals because it chooses one procedure among the relevant procedures at a time.

Robustness

PRS is not robust. The interpreter will stop selecting and executing the next action when there is a failure in any part of its database, goal, procedure or stack. The interpreter needs all the components to make decisions. In addition, PRS cannot exploit redundancy because actions are controlled by a central interpreter, although the use of multiple PRS instantiations to exploit redundancy has been discussed [85]. PRS's control is centralised. Furthermore, there is no guarantee that PRS will not slow robots down as beliefs are accumulated.

Extensibility

PRS is semi-extensible. Procedures are separated from the beliefs database (i.e. beliefs or facts) and goals. They can be added or changed without affecting the database or the goals. However, PRS does not have a learning mechanism to improve performance and make extension easy.

3.3.6 Agent Network Architecture (ANA)

The Agent Network Architecture (ANA), also known as the spreading activation network [137, 136] was initially developed in 1990 for the purpose of governing intelligent agents. It is implemented in the Lisp programming language. Similar to the situated automata, its design motivation is to use knowledge while compensating

for the lack of responsiveness in agents. However, knowledge about conditions and effects of modules are made explicit and used in real-time decision making instead of being compiled as in situated automata.

ANA consists of connected *competence modules* that correspond to agents' behaviours (see Figure 3.13). There are two types of competence modules, namely, action and belief modules. Action modules are designed to perform physical actions when they are activated whereas belief modules are designed to change agents' beliefs when they are activated.

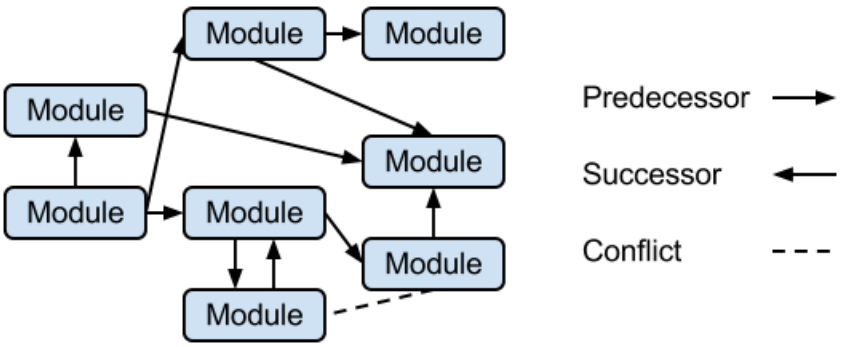


Figure 3.13: Agent Network Architecture

Every competence module has five properties that contain knowledge about the module: condition-list, add-list, delete-list, activation-level and implementation. The condition-list property specifies the conditions for which the module can be qualified to be activated. The add-list and delete-list properties specify the knowledge to be added and removed when the module is activated respectively (i.e. effects of the module). The activation-level property specifies a real number that represents how relevant the module is in the current situation. The implementation property specifies processes and behaviours that the module will carry out when it is activated.

Modules are connected based on their add-list and condition-list properties instead of being arranged manually as in Subsumption. For example, the *recognise cup* module is connected as the predecessor of the *pick up cup* module if it adds the same knowledge (i.e. add-list: cup-observed) required by the condition of the *pick up cup* module (i.e. condition-list: cup-observed).

The activation levels of modules are accumulated and calculated based on several conditions:

- **Increment based on sensor data**

  The activation levels of modules are increased if the modules' execution conditions match some sensors' data, as formalised as follows:

  $$\exists x \in \text{sensor\_data} : x \in \text{condition-list}$$

- **Increment based on goals**

  The activation levels of modules are increased if the modules add knowledge that matches some of the goals, as formalised as follows:

  $$\exists x \in \text{goals} : x \in \text{add-list}$$

- **Decrement based on protected goals**

  The activation levels of modules are decreased if the modules remove knowledge that matches some of the protected goals, as formalised as follows:

  $$\exists x \in \text{goals} : x \in \text{delete-list}$$

- **Increment based on predecessors**

  The activation levels of modules are increased (by a fraction of their predecessors' activation levels) if the modules' predecessors pass the conditions to be executed (i.e. condition-list properties are satisfied). In other words, successor modules are stimulated by the readiness of predecessor modules.

- **Increment based on successors**

  The activation levels of modules are increased (by a fraction of their successors' activation levels) if the modules' successors *do not* pass the conditions to be executed (i.e. condition-list properties are not satisfied). In other words, predecessor modules are stimulated by the failure of successor modules, so that the chances of predecessor modules being selected are increased.

- **Decrement based on conflicting modules**

  The activation levels of modules are decreased (by a fraction of conflicting modules' activation levels) if the modules are in conflict with other modules.

- **Decrement based on decay**

  The activation levels of all modules are decreased by the decay function.

Modules are qualified to be activated if their *condition-list* properties are satisfied and their activation levels surpass certain thresholds. ANA will activate a module that has the highest activation level among qualified modules (if there is any qualified module). Once a module is activated, its specified processes or behaviours will be executed and its activation level reset to zero.

Versatility

ANA is not versatile. Although it treats modules as black boxes where a greater diversity of techniques can be integrated as modules, it requires modules to be connected based on knowledge propositions using the *condition-list* and *add-list* properties. Thus, it limits the techniques that can be integrated to those that can be associated with knowledge propositions. In addition, robotic developers have to manually accommodate resources in the design of the modules in order to avoid resource conflict.

Reactivity

ANA is reactive. Modules can perform physical actions or change the beliefs of the agents as soon as they are activated. ANA can achieve multiple goals simultaneously by having each module perform a different action.

Robustness

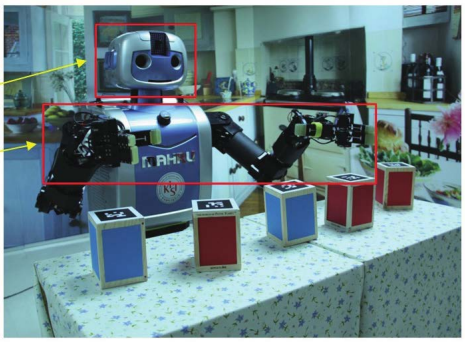
ANA is robust. Modules will keep executing actions when there is a failure in other modules because they are independent. In addition, ANA can exploit redundancy by duplicating modules to perceive the environment and control actions. Its control is decentralised. Furthermore, it will not slow the robots down because a slow module can be treated as a failed module that does not affect other modules (i.e. other modules will keep executing actions).

Extensibility

ANA is extensible. Modules can be added or changed independently and easily since they are connected automatically by their add-list and condition-list. In addition, ANA has two learning mechanisms to improve agents' performance and make the extension easy, namely to obtain new knowledge and to speed up the module selection by compiling (or chunking) existing knowledge [137].

3.3.7 ICARUS

ICARUS [126, 128] was initially developed in 1991 for governing intelligent physical agents. It is implemented in the Lisp programming language and has been deployed in physical agents, such as the KIST Mahru-Z [111] and the Adept MobileRobots Pioneer P3-DX [229] (see Figure 3.14). Its design motivation is to reproduce qualitative characteristics of human behaviour instead of matching quantitative human data. However, the way to measure human qualitative characteristics is not suggested in ICARUS.



(a) KIST Mahru-Z [111]

(b) Adept MobileRobots Pioneer P3-DX [4]

Figure 3.14: Robots Governed by ICARUS

ICARUS is based on five principles. First, cognition is grounded in perception and action. Second, concepts are differentiated with skills. Third, hierarchical concepts and skills are acquired cumulatively. Fourth, long-term memory is structured hierarchically. Fifth, long-term and short-term memory structures have a strong relationship.

Like ACT-R, knowledge is divided into two types, namely *concept* and *skill*. Concepts and skills can further be divided based on their relationships to the external environment into *primitive concept*, *non-primitive concept*, *primitive skill* and *non-primitive skill*. Primitive concepts are used to define state or meaning in terms of perception of the external environment, so that the concept is grounded to the external environment. In contrast, non-primitive concepts are used to define state or meaning in terms of other concepts. For example, a primitive concept of *holding-object* can be associated with a perception of an object on an agent's gripper. A primitive skill is an action that can be executed in the external environment whereas a non-primitive skill is an action that refers to other skills.

Each concept and skill have a header and a body. The header contains name and arguments. The body contains fields that are varied between concepts, primitive skills and non-primitive skills.

The body of a concept contains three fields, namely percepts, relations and tests. The *percepts* field describes the types and attribute values of objects perceived in the environment. The *relations* field describes the relationship to other concepts. The *tests* field describes the conditions that must be held for the concept to be valid.

The body of a primitive skill contains four fields, namely percepts, start, requires and actions. The *percepts* field is the same as those in concept. The *start* field describes the conditions for the skill to be selected. The *requires* field describes the conditions that must be held throughout the execution for the skill to be executed. The *actions* field describes the actions that would be performed by the skill.

The body of a non-primitive skill contains three fields, namely percepts, start and subgoals. The *percepts* field is the same as those in primitive skill or concept. The *start* field is the same as those in primitive skill. The *subgoals* field is similar to the *actions* field in primitive skill. It describes the order of subgoals that would be achieved by the skill.

ICARUS has six kinds of memory, namely perceptual, conceptual, belief, goal or intention, skill and motor memories (see Figure 3.15). On every decision cycle, objects in the external environment are perceived and stored in the perceptual buffer or memory. The information about the objects is matched against the bodies of concepts in the conceptual memory. Any supported beliefs are inferred and added to the belief memory. The process is repeated until all beliefs that can be inferred are added.

ICARUS will select a goal with the highest priority from the goal/intention memory as the current goal. The goal/intention memory contains an ordered list of objectives to be achieved by agents. A hierarchy of a skill path is constructed and planned from the current belief to the goal, based on the knowledge about skills in the skill memory. An action in the final skill of the skill path is executed to change the environment. It will trigger and cause new perceptions about the environment and the cycle is repeated.

Versatility

ICARUS is not versatile. Tasks must be able to be represented by knowledge

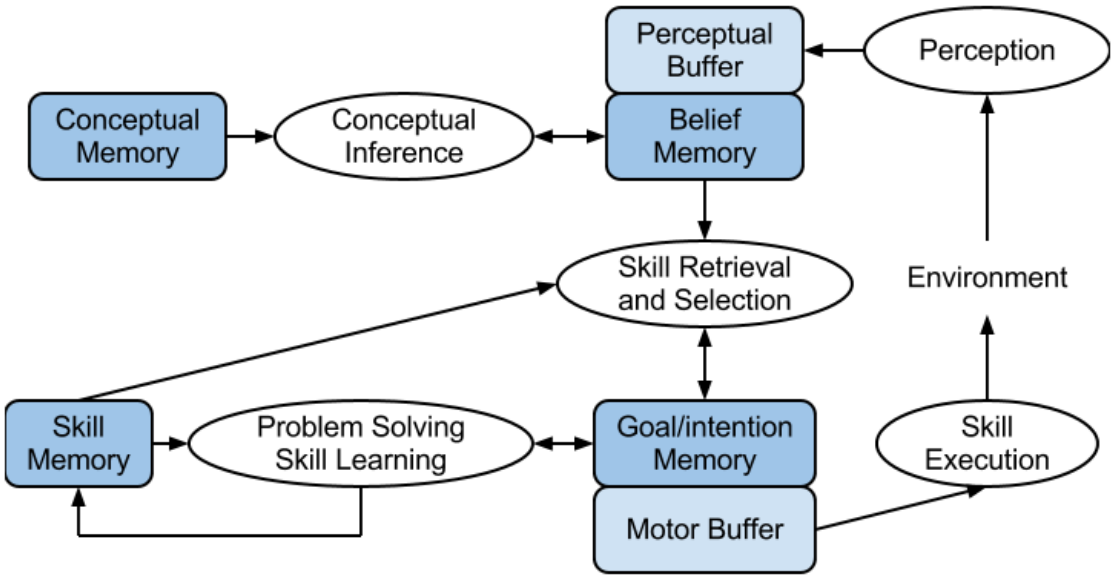


Figure 3.15: ICARUS Architecture

in concepts and skills. ICARUS does not provide a way to integrate other undesignated techniques. In addition, robotic developers have to manually accommodate resources in the design of the concept and skill to avoid resource conflict.

Reactivity

ICARUS is semi-reactive. It can adjust the degree to which it monitors the environment while carrying out actions. It also handles interruption with priorities. However, it can only access the highest goal in the prioritised goal stacks (i.e. achieve a goal at a time).

Robustness

ICARUS is not robust. Modules (called *memories*) are interdependent to each other. A failure in a *memory* causes other *memories* to stop working. ICARUS does not have redundancy of the processing functions. Its control is not decentralised. In addition, there is no guarantee that the robot will not slow down as they accumulate experience over long periods of time.

Extensibility

ICARUS is extensible. Concepts and skills can be added or changed without manually rearranging their connections since they are planned on every

decision cycle. In addition, ICARUS incorporates a learning module that creates a new skill whenever problem solving and execution achieve a goal, which improves agents' performance and makes the extension easy.

3.3.8 Three Tier / Three Layers Architecture (3T)

Three tier (3T) or three layers architecture [81, 34] was initially developed in 1996 for the purpose of governing intelligent agents. It is the successor of *A Three-Layer Architecture for Navigating Through Intricate Situations* (ATLANTIS) [80], which was developed in 1991. Its design motivation is to integrate deliberation and reactivity in agents.

Tasks are abstracted into three layers or tiers, namely, deliberation, sequencing and reactive skills (see Figure 3.16). In the deliberation layer, plans are created, selected and monitored by an Adversarial Planner (AP) in order to achieve goals. A plan does not call actions to control agents' actuators directly, instead it calls packages of actions called Reactive Action Packages (RAPs).

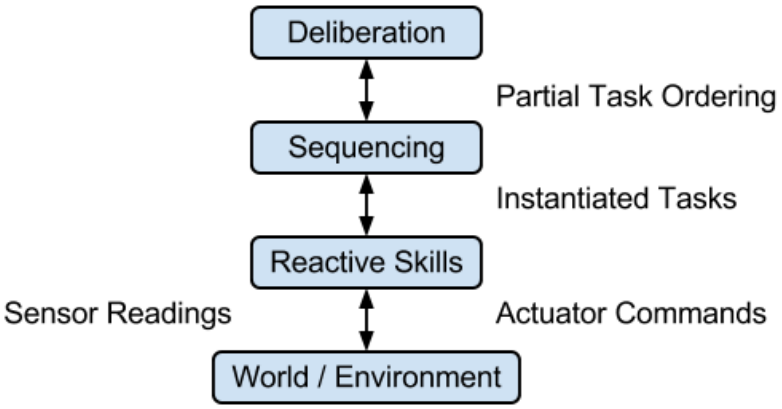


Figure 3.16: Three Tier / Three Layers Architecture [34]

In the sequencing layer, selected RAPs are interpreted and monitored by the RAP interpreter (i.e. sequencer). A RAP is a sub-plan that consists of context-dependent sets of reactive skills to achieve a task under a variety of conditions. A different set of reactive skills in a RAP can be activated and deactivated by the RAP interpreter depending on the situation or context. For example, a RAP *pass-ball* can have a set of *move to target* and *soft kick ball* skills and a set of *move to target* and *hard kick ball* skills that can be activated when a teammate is nearby and far respectively.

In the reactive layer, activated skills are coordinated and monitored by the skill manager. Reactive skills are low-level behaviours that continuously control motors and interpret sensors in real-time in order to achieve or maintain a particular state in the world, such as move position toward a target, soft kick ball and hard kick ball. Each skill has four functions and a specification about input and output, as follows:

1. **Input and output specification**

   A skill provides required inputs and expected outputs so that its outputs can be linked to the inputs of other skills in order to create a network of skills.

2. **Initialise function**

   A skill can be initialised when the system is started, e.g. setup communication ports.

3. **Enable function**

   A skill can be enabled or disabled by the sequencer. Any necessary start up routines may be performed every time a skill is enabled.

4. **Disable function**

   A skill will be disabled when it is not needed. Any necessary clean up routines are performed by the disable function.

5. **Computational transform function**

   A skill continuously recomputes its outputs based on its current inputs by using the computational transform function when it is enabled.

Versatility

3T is not versatile. Tasks must be able to be described by AP language and RAPs. 3T cannot integrate undesignated techniques. In addition, robotic developers have to manually design the content of the planner and the RAPs to avoid resource conflict.

Reactivity

3T is semi-reactive. Its reactive skills continuously control motors and interpret sensors in real-time. However, as also suggested by the authors, the

concurrency support of the planner and the sequencer to achieve multiple goals only remains in discussion and has not been implemented [34].

Robustness

3T is semi-robust. A failure in a reactive skill does not cause the whole robot to fail since reactive skills can work asynchronously. However, 3T cannot exploit redundancy of the processing functions. Its control is not decentralised. In addition, there is no guarantee that the robot will not slow down as they accumulate experience over long periods of time.

Extensibility

3T is semi-extensible. Adding or changing knowledge in a layer does not affect other layers. However, 3T does not have a learning mechanism to improve performance and make the extension easy. Learning remains future work in 3T [34].

3.3.9 Polyscheme

Polyscheme [48] was initially developed in 2002 for the purpose of governing intelligent agents. It is written in the Java programming language and has been implemented in robots, such as the NASA Robonaut [208] and the Nomadic Nomad 200 [224] (see Figure 3.17). Its design motivation is to use multiple representations (i.e. heterogeneous) instead of a single representation (i.e. uniform).

Polyscheme is designed based on the *Society of Mind* theory [144]. This theory proposes that intelligent behaviours emerge from the interactions among a diversity of processes instead of a single perfect principle. It can be seen as the opposite of the unified theory of cognition adopted by Soar and ACT-R.

Polyscheme encapsulates techniques in modules called *specialists*. Each specialist has its own representation to encode knowledge, but all specialists use the same propositional language to communicate and share knowledge among each other. Knowledge is shared by calling four functions that are implemented in each specialist, as listed below:

1. **The *StanceOn(P)* function**

(a) Nasa Robonaut [149]

(b) Nomadic Nomad 200 [159]

Figure 3.17: Robots Governed by Polyscheme

This function returns the truth value the specialist believes about the proposition P.

2. **The *ReportOpinion(P,S,tv)* function**

   This function gets the truth value of proposition P from specialist S and stores the truth value in tv.

3. **The *RequestedFoci()* function**

   This function returns a set of propositions that the specialist would like to focus on.

4. **The *Groundings(P)* function**

   This function returns a set of closed propositions that ground the open proposition P

At every decision making cycle, Polyscheme's *focus manager* chooses a proposition among requested propositions based on the level of urgency and factors specified by developers. Once a proposition is selected, sequences of three functions are called in each specialist. First, the *StanceOn* function is called to determine each specialist's stance on the proposition's truth value. Second, the *ReportOpinion* function is called to get the opinion about the proposition's truth value from each specialist. Finally, the *RequestedFoci()* function is called to get a new proposition that

each specialist would like to focus on. The focus manager will then choose a new proposition and this cycle is repeated. In this cycle, relevant specialists (e.g. motor specialist) will perform actions based on their own knowledge. However, the article [48] does not clearly describe when and how exactly a specialist performs the action.

Versatility

Polyscheme is not versatile. Although each specialist can have its own representation of knowledge, all specialists have to use the same propositional language to communicate and share their knowledge with each other. However, not all knowledge required by open and complex tasks can be represented by a propositional language. In addition, actions are performed by specialists with the assumption that they have enough resources. Robotic developers have to manually accommodate resources in the design of the specialists to avoid resource conflict.

Reactivity

Polyscheme is reactive. In Polyscheme, actions are executed by each specialist. Polyscheme responds to the situations fast because it does not wait for all specialists to finish processing their information before it selects and performs actions. Instead, some specialists can perform actions while other specialists are still processing their information. In addition, Polyscheme can achieve multiple goals simultaneously because each specialist can execute an action to achieve a goal.

Robustness

Polyscheme is robust. Each specialist uses a different technique (such as representation, reasoning and mechanism) to solve the same problem. If some specialists fail or cannot solve a problem, then there are still other specialists that may be able to solve the problem by using their techniques. In this case, specialists can keep solving problems and performing actions despite a failure in other specialists. In addition, for the same reason, specialists can be made redundant by solving the same problem using different techniques. Polyscheme's control is decentralised. Furthermore, it will not slow robots down because a slow specialist can be treated as a failed specialist that does not affect other specialists performing actions.

Extensibility

Polyscheme is semi-extensible. Specialists can be added or changed independently. However, Polyscheme does not have a learning mechanism to improve performance and make extension easy.

3.4 Related Architectures

While I described current state of the art in cognitive architecture in the previous section, there are still other architectures that are worth to discuss due to their similarities to the work in this dissertation. However, these architectures are not general purpose cognitive architectures and therefore do not fit in the previous section. They are robot architectures designed for specific robotic tasks and/or they are multi-agent architectures designed to govern multiple robots. They are outside the scope of this dissertation. In this section, I describe them briefly. In Section 5.5, I discuss how their mechanisms are similar and different to the mechanism proposed in this dissertation.

- **Autonomous Robot Architecture (AuRA)**

  Autonomous Robot Architecture (AuRA) [18, 20, 19] was initially developed in 1986 for robot navigation purposes. It has a schema controller that manages, controls and monitors all *motor schemas* at run time. Each motor schema can operate asynchronously and generates a response vector to control motors. AuRA sums and normalises all vectors received from all motor schemas before sending the result to the motors for execution.

- **ALLIANCE**

  ALLIANCE [173] was initially developed in 1994 to address fault tolerant issues. It is a multi-agent architecture. It consists of *motivational behaviours*. At all times, each behaviour defines its activation level based on inputs, including sensory feedback, inter-robot communication, inhibitory feedback from other active behaviours and internal motivation. Its activation level is a non-negative number. When this activation level exceeds a given threshold, the corresponding behaviour set becomes active. There are two variables used to determine the activation level, namely robot impatience and robot acquiescence. The robot impatience enables a robot to handle situations when other robots (outside itself) fail in performing a given task. The robot acquiescence

motivation enables a robot to handle situations in which it, itself, fails to properly perform its task.

- **Distributed Architecture for Mobile Navigation (DAMN)**

  Distributed Architecture for Mobile Navigation (DAMN) [186] was initially developed in 1997 for robot navigation purposes. Initially, it used three voting mechanisms to select actions, namely constraint arbitration, actuation arbitration and effect arbitration. However, these three voting mechanisms were later replaced by the utility fusion voting mechanism, which was proposed to solve issues encountered by the three mechanisms. DAMN consists of independent, distributed and asynchronous modules that handle the robot's behaviours to achieve tasks. Every module concurrently suggests an opinion (i.e. vote) for *every* candidate or action depending on the situations in the environment. DAMN takes these opinions to decide on an action.

- **Team Member Agent Architecture**

  Team member agent architecture [214, 213] was initially developed in 1998 for creating teams of agents in periodic team synchronisation (PTS) domains. It is designed based on a principle defined as the locker-room agreement. In this architecture, each agent keeps track of three types of state, namely the world state, the locker-room agreement and the internal state. The agent also has two types of behaviours, namely internal and external behaviours. They are both sets of condition/action pairs where conditions are logical expressions over inputs, and actions are the behaviours themselves. The internal and external behaviours are organised as directed acyclic graphs (DAGs). Various learning techniques have been applied to this DAGs structure in order to make decisions, such as reinforcement learning, decision tree learning and memory-based learning.

- **Kismet's Architecture**

  Kismet's architecture [39] was initially developed in 2000 for governing Kismet robot. This architecture consists of a behaviour system designed based on homeostatic regulatory mechanisms from the ethology literature. The behaviour system is organised into loosely layered heterogeneous hierarchies of behaviour groups. Each group contains behaviours that compete for activation (through activation levels) with one another. Each behaviour determines

its own activation level based on its relevance to the situation by taking into account perceptual factors as well as internal factors. At the highest level, behaviours are organised into competing functional groups (i.e. the primary branches of the hierarchy) where each group is responsible for maintaining one of the three homeostatic functions, namely to be social, to be stimulated by the environment and to occasionally rest. Only one functional group can be active at a time.

- **Emotionally Grounded (EGO) Architecture**

  Emotionally Grounded (EGO) architecture [77] was initially developed in 2003 for governing Sony robots, namely Sony QRIO and Sony AIBO. It is designed based on homeostatic regulatory mechanisms from the ethology literature. It consists of behaviours (called modules) that compete for activation using their activation levels. Each module determines its own activation level and requires resources. Modules are organised into a hierarchical tree structure, which can be thought like a layered hierarchical groups in Kismet's behaviour system. Modules that have the same parent in the tree will share the same target. Unlike Kismet's behaviour system, multiple modules can be activated if they have different targets (i.e. multiple modules in different groups can be active at a time). In addition, modules are selected concurrently in the competition from the highest activation level to the lowest activation level until there is no remaining resource available. Once a module is selected, it is given permission to execute the behaviour implemented in the module as a state machine (i.e. executing a state machine behaviour). Both module organisation and competition mechanisms are designed to manage resources.

- **TraderBots**

  Several auction-based architectures have been proposed in the literature [61, 86]. One example is TraderBots [61] developed in 2004 for coordinating robots. It is a multi-agent architecture designed based on the economic approach of free market system. In TraderBots, a larger task is divided into sub-tasks. Each task is associated with a revenue and a cost. A team of robots places bids for tasks. The goal of the team is to execute some plans such that its profit is maximised or its cost is minimised. TraderBots involves mathematical and principle approach towards choosing these bid values.

3.5 Summary and Gaps

In previous sections, I have analysed major approaches in cognitive architecture based on knowledge design and knowledge utilisation, as well as analysed the current state of the art in cognitive architecture. In this section, I will summarise these analyses and identify the gaps in cognitive architectures.

Table 3.5 summarises the approaches in cognitive architecture based on knowledge design. More expressive and flexible approaches tend to be more complicated and have higher computational costs (i.e. lower efficiency), and vice versa. A hybrid approach focuses on balancing expressiveness, flexibility, complexity and computation cost.

Criteria	Explicit-based	Implicit-based	Hybrid
Simplicity	Low	High	Low-high
Efficiency	Low	High	Low-high
Expressiveness	High	Low	Low-high
Flexibility	High	Low	Low-high

Table 3.5: Comparison of Approaches in Cognitive Architecture based on Knowledge Design

Table 3.6 summarises the approaches in cognitive architecture based on knowledge utilisation. Approaches that have low operation complexity, low knowledge complexity and small knowledge size tend to be inflexible, and vice versa.

These two summaries suggest that a cognitive architecture should be able to accommodate a variety of approaches in order to solve various problems or tasks efficiently. They show clearly that there is no the best approach in cognitive architecture. Each approach has strengths and weaknesses that suit different problems or tasks.

In addition to the two summaries, Table 3.7 summarises the current state of the art of cognitive architectures. The + and - signs indicate that the criteria is fully supported or semi-supported respectively. The table shows that these cognitive architectures do not meet all the evaluation criteria described in Section 2.5, especially versatility. As a result, current cognitive architectures lack the capability to fully support service robots.

Criteria	Lookup	Case	Priority	Goal	Connection	Utility
Operation complexity	Low	Low	Low	High	Low	Medium
Knowledge complexity	High	Low-high	Low-high	High	High	Medium
Knowledge size	Large	Small	Small	Large	Small	Small
Real-time modification	No	No	Yes	Yes	No	Yes
Flexibility	Low	Low	Medium	High	Low	Medium-high

Table 3.6: Comparison of Approaches in Cognitive Architecture based on Knowledge Utilisation

Analysing the existing cognitive architectures shows that there are four important issues that cause the inability to meet all the evaluation criteria (i.e. inability to fully support service robots in open and complex environments). The first three issues arise from the difficulty of integrating techniques. The last issue arises from the deficiencies in the robot operation. These four issues are as described as follows:

1. **Incapable of Managing Resources**

   Current cognitive architectures do not have an explicit mechanism to manage resources. As a result, robotic developers must take responsibility for managing the resources manually

2. **Incapable of Integrating Independently Developed Techniques**

   Many current cognitive architectures are carefully designed to integrate selected techniques, which may not suit all problems. They have limited techniques that can be integrated. They are often designed with some constraints that cause the use of any kind of independently developed technique to be difficult.

3. **Inflexible to Adapt to Changes**

   Many current cognitive architectures are inflexible and unable to adapt to changes of techniques and structure. They are difficult and require consider-

Architecture	Versatility	Reactivity	Robustness	Extensibility
Soar			-	+
ACT-R				+
Subsumption		+	+	
Situated Automata		+		
PRS		-		-
ANA		+	+	+
ICARUS		-		+
3T		-	-	-
Polyscheme		+	+	-
ASMO	+	+	+	+

Table 3.7: Comparison of Current State of The Art of Cognitive Architectures

able time and effort to add new techniques, change the techniques and change the structure.

4. **Inadequate Support to Robot Operation**

   Many current cognitive architectures are inadequate for supporting fast responses, simultaneous multiple goals, fault tolerance, redundancy, long-term operation, maintainability and learning.

The purpose of this dissertation is to address these significant gaps so as to accelerate the development, deployment and adoption of service robots undertaking tasks in open and complex environments. This dissertation develops the novel bio-inspired ASMO cognitive architecture that has been designed and developed to address all four identified shortcomings of current cognitive architectures.

Chapter 4

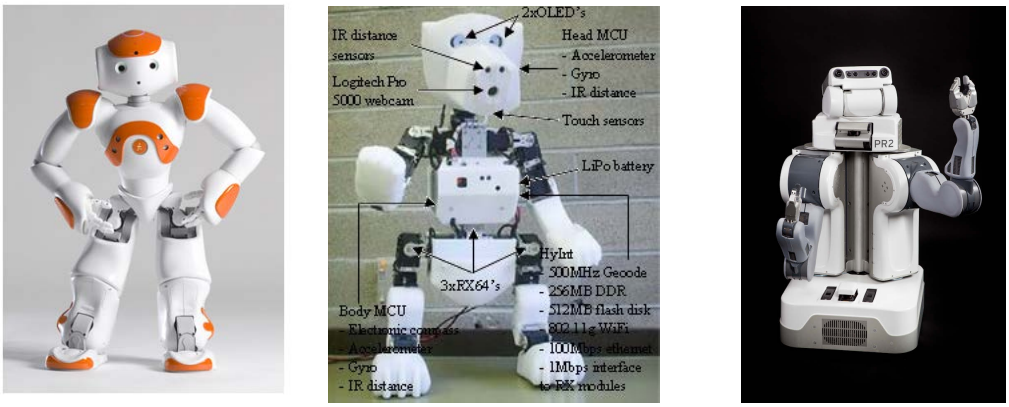
ASMO Cognitive Architecture

In the previous chapter, I analysed major approaches in cognitive architecture and the current state of the art of cognitive architectures. I also identified four shortcomings of current cognitive architectures. In this chapter, I introduce the novel ASMO cognitive architecture that addresses all of the four shortcomings. I begin by describing the overall design of ASMO cognitive architecture in Section 4.1. I follow by discussing how ASMO can be integrated with other systems in Section 4.2. Finally, I describe cognitive capabilities supported by ASMO in Section 4.3.

I will further describe the details of ASMO cognitive architecture: attention mechanism, emotion mechanism and learning mechanisms in the following Chapters 5, 6 and 7 respectively.

4.1 Overall Design

Attentive and Self-Modifying (ASMO) cognitive architecture [167, 168, 164, 166, 163, 165] (see `www.ronynovianto.com`) is developed in this dissertation to address the four gaps identified in current cognitive architectures. Its design motivation is to accelerate the development, deployment and adoption of service robots in society. It is developed for governing intelligent agents and robots. It is written in Erlang and Python programming language. Its version using the Robot Operating System (ROS) framework is also available. It has been deployed in a number of robots, including Aldebaran Nao [163], Tribotix Hykim [167, 168, 164, 166] and Willow Garage Personal Robot 2 (PR2) (see Figure 4.1).



(a) Aldebaran Nao [7] (b) Tribotix Hykim [228] (c) Willow Garage PR2 [242]

Figure 4.1: Robots Governed by ASMO

In ASMO, intelligent behaviours are emerged from interactions between constituent processes, rather than from careful design or fine-tuning. An emergent system consisting of simpler or smaller processes can raise intelligence although the processes themselves may lack of intelligence [144]. It produces more complex behaviours and properties that are different to simply the sum of its parts [15]. However, if desired, processes and their interactions can be carefully designed and fine-tuned to achieve desired behaviours and effects (see Section 5.4.1). The more carefully designed the processes, the less emergent the behaviours.

ASMO consists of processes, memory, an attention mechanism, an emotion mechanism and learning mechanisms (see Figure 4.2).

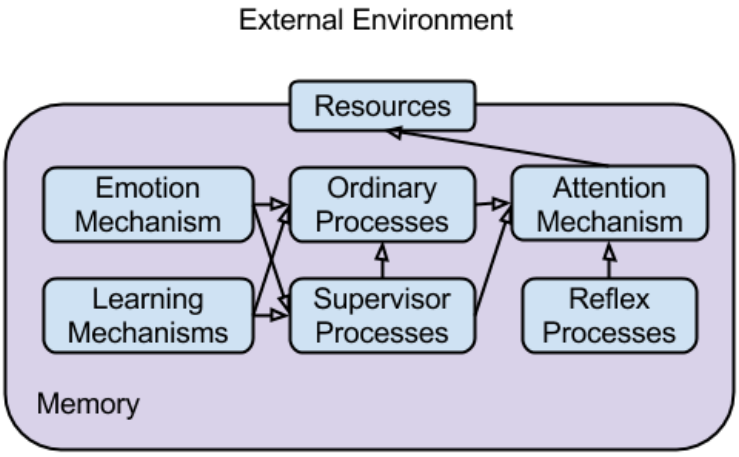


Figure 4.2: ASMO Cognitive Architecture

A process (also called a module) is an autonomous, self-contained, concurrent,

modular and distributed black box. It uses any variety of techniques (such as algorithms, representations and mechanisms) to perceive situations, process information and propose actions in order to achieve tasks. An action can be a high-level function or a low-level command to actuators and hardware. Examples of high-level functions are recognising objects (i.e. perception) and finding the shortest path (i.e. planning). Examples of low-level commands are walking to a specific location and storing data to memory.

An action require resources (e.g. leg actuators, arm actuators, etc) to execute. A module needs to have an access to resources required by its proposed actions in order to perform the actions. Modules are in conflict when they require the same resources at a time. When this happens, ASMO must select a module among the conflicting modules in order to determine which actions to perform. Thus, in order to select modules and manage resources, modules are designed to compete for 'attention' to use resources (i.e. demand attention). Modules that win this attention competition will get the attention to access to their required resources in order to perform their proposed actions (see ASMO's attention mechanism in the following subsection).

In the following, I describe the overall designs of memory and ASMO's attention, emotion and learning mechanisms. Details of the three mechanisms will be described in the next three chapters respectively.

4.1.1 Memory

A module can store (and retrieve) data, information and knowledge to (and from) its local storage or to a global memory (called ASMO's memory). The data stored in the local storage is private and only available to the module itself. The data stored in ASMO's memory is public and available to or can be accessed by other modules. Thus, ASMO's memory mediates the sharing and communication between modules.

A module can store data in any representation (i.e. ASMO's memory does not restrict how data is stored). For example, a module may store data in a numerical representation used by a neural network while another module may store data in a symbolic logic representation. ASMO's memory will contain both representations of situations in an environment (e.g. there is a ball on the left) and representations of mental states inside a robot (e.g. a belief that chasing a ball would give higher performance than defending an attack).

A module is mainly designed to produce and use its own data (i.e. aimed to be as independent as possible). However, if desired, it can still be designed to use and combine data produced by other modules that is stored in ASMO's memory. The advantage of this design of having common memory in ASMO distributed architecture is that the module can have low coupling and yet it is still allowed to communicate and share with other modules. The disadvantage of this design is that the module is required to know the representation of the data stored by other modules (i.e. not the aim of ASMO).

The design of emergence and independent modules creates some difficulties to coordinate the modules. However, sharing the global ASMO's memory helps to solve these difficulties. For example, consider an independent module (called search_ball) requires a robot's head to look in a particular direction in order to search for a ball and another independent module (called chase_ball) requires the robot's legs to chase for the ball. For the search_ball module to propose to look in the right direction, it needs the information from the body orientation, which comes from where the legs will be placed next by the chase_ball module. In order words, a module needs to anticipate what other modules will do if they win an attention competition. Without the global ASMO's memory, the search_ball module cannot anticipate the location of the legs. With the global ASMO's memory, the chase_ball module can store the location where it proposes the legs to step to and the search_ball module can use this information to propose the right direction for the head.

In addition, sharing the global ASMO's memory also provides developers with the flexibility to develop modules. For example, consider two modules (say to search a ball and to search for landmarks) require an access to the same camera to obtain images. Developers do not have to create these two modules to compete for the same camera resource. Instead, developers can create another module to require the camera to capture images and then share these images to ASMO's memory so that the two modules (i.e. search the ball and search landmarks modules) can use the images.

Modules can retrieve data in ASMO's memory (i.e. access ASMO's memory) whenever they like or desire. This retrieval resembles the *Hollywood principle* of software design: "don't call us, we'll call you" [233]. In the Hollywood principle, data is not passed to other components or modules directly after the data is produced. Instead, the data remains available to be fetched for use on demand or when needed.

In the current implementation, data is stored with an address and keywords in a tuple form [83]. It can later be retrieved by specifying its address or searching its keywords (or contents). ASMO's memory will operate as a random access memory if data is retrieved by an address or operate as a content-addressable memory (i.e. associative memory) [116] if data is retrieved by keywords or contents. A search engine is required to search data by keywords or contents. If an exact match is not found, then a list of data is provided according to matching scores of the data.

The address of the data is formatted using a uniform resource identifier (URI)-like string, such as 'asmo/perception/vision/ball/location'. A URI address is compatible with the Representational State Transfer (REST) [69] protocol that allows the data to be exchanged through the HTTP connection without predefining the function interfaces.require

4.1.2 Attention Mechanism

ASMO's attention mechanism [163] selects modules and manages resources required by modules explicitly and automatically through attention competitions (also called attention elections). Modules are designed to compete for attention to use resources. They continuously update their demand of attention based on tasks and situations in an environment and propose actions to achieve the tasks. ASMO will then select modules and perform their proposed actions on every attention competition.

ASMO's attention mechanism continuously runs or calls for an attention competition at any time or any condition designed by developers (e.g. every interval or on-demand). In every attention competition, it ranks competing modules based on their attention demand, and selects modules as winners of the competition based on their rankings and the availability of their required resources as described as follows. It checks every module from the highest ranking to the lowest ranking. If resources required by the module are available, then it selects the module as one of the winners and marks the resources as unavailable. Otherwise, it skips the module and continues checking the next ranking. It repeats this process until all resources are occupied or all modules have been checked.

In this process, a module with a higher ranking is given the chance to occupy resources before a module with a lower ranking. However, the higher ranking module does not always win over the lower ranking module. Instead, the lower ranking

module can still win over the higher ranking module if resources required by the higher ranking module are not available. In addition, modules that have no conflict in requiring resources will eventually be selected despite their rankings, because they have no competitors.

ASMO's attention mechanism selects multiple winners in an attention competition. These winners are selected with respect to resources. They occupy different resources. Multiple winners allow multiple actions that require different resources to perform simultaneously. Hence, ASMO can make multiple decisions at one time in order to achieve multiple goals simultaneously.

4.1.3 Emotion Mechanism

ASMO's emotion mechanism [166] extends the design of a module to incorporate a subjective bias based on emotions. Modules update their own demand of attention not only based on tasks and situations in an environment, but also based on a subjective bias given by ASMO's emotion mechanism. The purpose of ASMO's emotion mechanism is to provide modules with subjective biases in order to influence the selection of modules.

ASMO's emotion mechanism continuously runs its operation at any time or any condition designed by developers (e.g. every interval or on-demand). It uses causal Bayesian networks to calculate the probabilities of emotions will be elicited given a situation. It uses these probabilities to determine the amount of subjective biases that should be given to modules. Modules incorporate these amount of subjective biases when updating their demand of attention. These subjective biases will affect the modules' rankings and their chances of winning an attention competition. As a result, they may cause different winners to be selected, and thus different actions to be performed.

4.1.4 Learning Mechanisms

ASMO's learning mechanisms [167, 168, 164] extend the design of a module to incorporate learning. Each module is given a boost in order to affect its demand of attention. The higher the boost, the lower the attention needed by the modules to win an attention competition. The purpose of ASMO's learning mechanisms is to

provide modules with the right boosts in order to modify the selection of modules, so as to improve an agent's or a robot's performance.

ASMO's learning mechanisms are divided into several mechanisms, namely habituation, sensitisation, operant conditioning, classical conditioning and observational learning mechanisms. Similar to ASMO's emotion mechanism, ASMO's learning mechanisms affect the attention demanded by modules. Unlike ASMO's emotion mechanism, however, modules do not incorporate boosts when updating their demand of attention. Thus, they do not accumulate the effects of the boosts.

ASMO's habituation and sensitisation mechanisms [164] modify the boosts possessed by modules based on the significance of a situation perceived by the modules. ASMO's habituation mechanism decreases the boost possessed by a winning module when a situation is perceived to be insignificant in an environment (i.e. it is not necessary for the module to have a higher attention demand). In contrast, ASMO's sensitisation mechanism increases the boost possessed by a losing module (i.e. module that does not win an attention competition) when a situation is perceived to be significant in an environment (i.e. it is necessary for the module to have a higher attention demand).

ASMO's operant conditioning mechanism [167] increases or decreases the boosts possessed by modules based on explicit feedback received from an environment. It is similar to ASMO's habituation and sensitisation mechanisms, except that its evaluation of a situation is provided by feedback in the environment explicitly, rather than a significance perceived internally by modules. ASMO's operant conditioning mechanism punishes modules that are supposed to win an attention competition but receive negative feedback, and reinforce modules that are not supposed to win an attention competition but receive positive feedback.

ASMO's observational learning mechanism modifies the boosts possessed by modules based on a model or patterns learned from labelled data. Experts provide correct boosts for situations encountered during training (i.e. labelled data). ASMO's observational learning mechanism will learn a model or patterns of these correct boosts in order to predict future situations and to modify the boosts possessed by modules.

Unlike ASMO's habituation, sensitisation, operant conditioning and observational learning mechanisms, ASMO's classical conditioning mechanism [168] does not modify the boosts possessed by modules directly. Instead, it learns the asso-

ciations between stimuli that have no responses (i.e. neutral stimuli) and stimuli that will be responded by modules (i.e. unconditioned stimuli). It then triggers the modules to propose actions (i.e. to respond) on the presence of the neutral stimuli despite that the unconditioned stimuli are not actually present. In this case, the neutral stimuli would be called conditioned stimuli, and its presence substitutes the presence of the unconditioned stimuli.

4.2 Share Management of Resources

ASMO cognitive architecture can share a responsibility with other systems to manage resources. Its self-contained and modular processes would allow it to work or integrate easily with existing architectures, frameworks or systems without much need to redevelop the whole system. It will incorporate benefits from other effort.

ASMO cognitive architecture can potentially be integrated with other ASMO cognitive architectures to create a multi-agent architecture. Each ASMO will run an attention competition to manage its own resources, while interacting with other ASMOs as their superior, peer or subordinate. This interaction creates a complex ASMO hierarchy. The higher the position in the hierarchy, the more resources ASMO can control. The work of multi-agent architecture is beyond the scope of this dissertation and remains future work (see Section 9.2). In the following, I describe briefly the three possible types of interactions between ASMO and other systems:

1. **Superior Type**

   In a superior type, ASMO has a full control over resources. A superior type of interaction is achieved by making ASMO as a main system while embedding other systems as ASMO's modules (i.e. ASMO's subsystems). Other systems help ASMO perform tasks by perceiving situations, processing information and proposing actions, but final decisions and actions are selected and performed by ASMO.

2. **Peer Type**

   In a peer type, ASMO has a partial control over resources. A peer type of interaction is achieved by sharing ASMO's memory with other systems. ASMO will only make decisions and perform actions for resources that it manages.

3. **Subordinate Type**

   In a subordinate type, ASMO does not have a control over resources. A subordinate type of interaction is achieved by embedding ASMO as a subsystem within other systems. It helps the other systems to perform tasks by perceiving situations, processing informations and proposing actions, but final decisions and actions are selected and performed by the other systems.

4.3 Cognitive Capabilities

ASMO supports cognitive capabilities required by agents as proposed by Gärdenfors and Williams [92] (see Figure 4.3). According to Gärdenfors and Williams, an agent (such as a robot) is a self-contained autonomous system whose inner world interacts with an outer world (the environment). It requires cognitive capabilities that support sensations, actuations, perceptions, conceptions, simulations and planning in order to achieve effective collaboration with other agents. These cognitive capabilities are as described as follows:

1. Sensations are sensory impressions or data that report what is happening now in the outside world and within the body.

2. Actuations are outputs such as information or actions.

3. Perceptions are interpreted or processed sensory impressions.

4. Conceptions are concise representations of classes of entities.

5. Simulations are knowledge generated from the reconstruction or manipulation of perceptions, conceptions and plans.

6. Plans are recipes for action.

These cognitive capabilities are represented using two types of representations proposed by Gärdenfors [90], namely cued and detached representations. Perceptions are represented using the cued representation whereas conceptions and simulations can be represented using both types. A cued representation always represents something that is *present* in the current situation, such as the ball is seen on the left. In contrast, a detached representation represents something that may or may

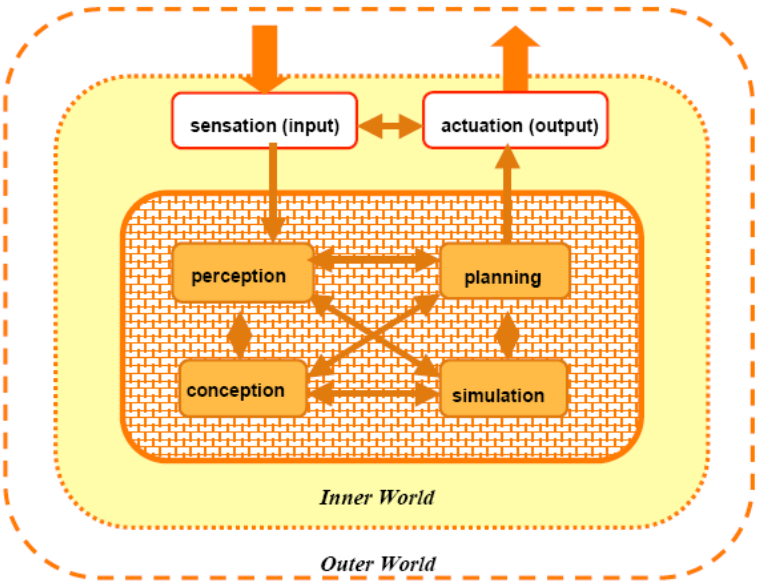


Figure 4.3: Cognitive Capabilities proposed by Gärdenfors and Williams

not be present in the current situation, such as the ball is imagined to be on the left although the ball cannot be currently seen. A detached representation can also represent something that may not exist at all, such as ball with legs, unicorn or a dragon.

4.3.1 Outer World, Inner World, Sensation and Actuation

In ASMO, *outer world* is the (external) environment or the space where the robot lives whereas *inner world* is the space inside the body of the robot (i.e. inside the robot). Inner world separates the robot from the environment. Data and information inside the inner world are private and can only be shared with other robots through communication (e.g., by speech or network connection). In contrast, data and information in the outer world are public and can be directly perceived by others.

A *body* is a set of resources that can be controlled by ASMO (see Figure 4.4). A robot requires a physical body to represent its presence and identity in the environment. In biological agents, resources that can be controlled by the brain are confined within a single physical body. In computational agents including robots, the resources may be located within a single body but can also be distributed across different physical locations connected by a network.

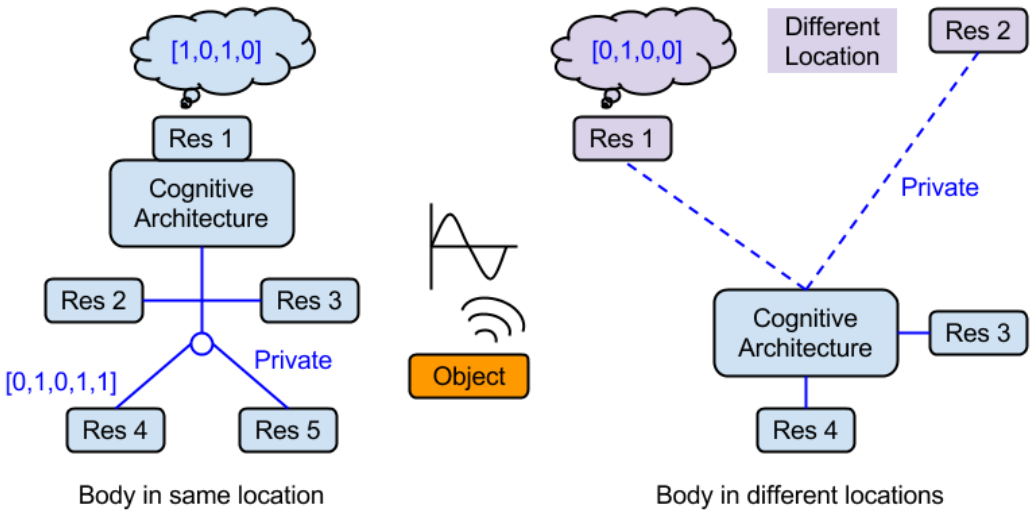


Figure 4.4: ASMO Body

A body has sensors and actuators that act as the interface between the inner world and the outer world. Sensors convert stimuli in the environment into input data that can be processed by the robot. In other words, *sensations* are the raw or unprocessed input data received from the environment through a robot's sensors. They are modality specific, e.g. visual sensation, auditory sensation, etc. In contrast, actuators convert output data or command back into stimuli that can change the environment. In other words, *actuations* are the low-level commands performed to affect the environment through a robot's actuators. Stimuli (e.g. light, sounds, etc) are physical matters that have physical characteristics (e.g. wavelengths, frequencies, chemistries) whereas sensations and actuations are virtual matters (e.g. image data).

4.3.2 Perception, Conception, Simulation and Planning

In ASMO, perceptions, conceptions, simulations and plans are created by modules:

1. **Perceptions**

   Modules can interpret sensations and fuse them with conceptions, simulations and plans to form a perception. In other words, a *perception* is the interpretation of sensations mixed with conceptions, simulations and plans. A perception can be created from either a single modality of sensation (e.g. ball perception created from image data only) or multiple cross-modalities of sensations (e.g.

ball perception created from both image and audio data). Robots may make different interpretations of the same sensations even if the same sensors are used (i.e. robots may 'feel' and interpret the same sensation differently). For example, a module may perceive a red blob in the image data as a ball while another module may perceive the same blob as a button on the robot (i.e. both modules make different interpretations of the same sensation received from the same sensor).

2. **Conceptions**

   Modules can classify similar perceptions, simulations or plans into a group called *conception*. They can use a conceptual space [91] to represent each concept. A conceptual space is a multi-dimensional space where each dimension represents a similarity feature and a collection of instances or objects is placed as points in the space based on their similarity features.

3. **Simulations**

   Modules can reconstruct or manipulate perceptions, conceptions and plans to create a simulation. ASMO can simulate the interaction between modules by running attention competitions without performing the winning actions.

4. **Plans**

   Modules can use their own planners to create plans. Actions can be planned using different planners independently. ASMO does not commit to a specific planner.

In ASMO, modules are free to use their own algorithms, representations and mechanisms (i.e. no restriction to use a particular kind). In this case, they can use both cued and detached representations. They store all perceptions, concepts, simulations and plan in the inner world with confidence scores. Modules with different algorithms, representations and mechanisms may produce different results and confidence scores given the same situation. The confidence score helps modules to decide which perceptions, concepts, simulations and plans to use.

Chapter 5

ASMO's Attention Mechanism

In the previous chapter, I have described the overall design of ASMO cognitive architecture. In this chapter, I describe the details of ASMO's attention mechanism and how it selects modules (i.e. makes decisions). I start by explaining the needs for attention in decision making in Section 5.1. I follow by describing theories of human attention in Section 5.2. I describe the interpretation of these theories in Section 5.3. These theories form the basis of the design of ASMO's attention mechanism. Finally, I describe the implementation of ASMO's attention mechanism in Section 5.4.

5.1 Need for Attention in Decision Making

Decision making is the process of choosing an action among a set of alternatives [95, p. 657] [243, p. 220]. Hastie distinguishes decision making from a judgement, as follows: "decision making refers to the entire process of choosing a course of action. Judgement refers to the components of the larger decision-making process that are concerned with assessing, estimating, and inferring what events will occur and what the decision-maker's evaluative reactions to those outcomes will be" [95, p. 657]. In other words, decision making is a multi-faceted process that involves reasoning, judgement and problem solving to select an action (i.e. a choice is made). In this dissertation, I only consider action selection that involves decision making. Thus, the term *decision making* and *action selection* will be used interchangeably.

Decision making is part of planning. Decision making selects an action on a

single or episodic situation. In contrast, planning simulates and projects a series of decisions on sequential situations in order to achieve the goal. As such, planning is essentially a sequential decision making process.

Decision making is formally studied in the field of *decision theory*. While most current decision theories are developed to account for the choice of one action at one point in time [95, p. 665], ASMO is designed to account for the choice of multiple actions at one time (i.e. multiple decision making).

A decision is *rational* if it is expected to optimise its performance given and with respect to the available knowledge. As described by Russell and Norvig [192, p. 37], "for each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has". Rational agents act to achieve the best outcome or best expected outcome when there is uncertainty [192, p. 4]. Performance is a measurement of an outcome in achieving a goal. An agent is said to be *irrational*, if it chooses an action that is expected not to maximise its performance or does not expect to achieve its goal given its available knowledge.

Attention is required in decision making or action selection to manage resources due to limited resources. Neumann [151] suggested that attention is important in action selection to prevent disorganisation of behaviours because all potential actions might be simultaneously trying to seize control of the resources. In addition, Anderson [9] proposed attention as a mechanism for the allocation of processing and physical resources. This allocated portion changes depending on the demands of the task and the resources available.

ASMO does not only need to support decision making or action selection as it is required for being a cognitive architecture (see Section 2.4), but it also needs an attention mechanism because of the following reasons:

- **Requirement to address the gaps**

  Managing resources is one of the requirements of ASMO in order to address the gaps identified in the research literature.

- **Sharing limited resources**

  Managing resources is necessary because robots have limited resources (such as legs, arms, CPU and memory). A robot cannot perform different actions

that require the same resources simultaneously. For example, it cannot go to different places at the same time because it has only a pair of legs.

- **Avoid conflicting actions**

  Making decisions based on resources ensures that ASMO will only perform actions if the resources required by the actions are available.

5.2 Theories of Attention

Attention is classically described by William James [103, pp. 403–404] as follows: "[attention] is the taking possession by the mind, in clear and vivid form, of one out of what seems like several simultaneously possible objects or trains of thought ... It implies withdrawal from some things in order to deal effectively with others". In other words, attention is the process of selecting from several simultaneous options: choosing some while rejecting the others.

ASMO's attention mechanism is designed based on the inspiration of human attention. Theories of human attention are divided into three major areas, namely selective attention, divided attention and automaticity [68].

5.2.1 Selective or Focused Attention

Selective or *focused* attention refers to the ability to select or to focus on task relevant information while ignoring simultaneous irrelevant information. This information may include stimuli, perception and thoughts. One classical example of selective attention is observed in the cocktail party phenomenon [53]. In a cocktail party, people can focus on a single conversation despite simultaneous background noises and other people's conversations. Experiments in *dichotic listening* have been conducted to study the cocktail party phenomenon and selective attention [53, 147, 87, 225]. In a dichotic listening experiment, participants are asked to listen to audio messages in two different channels on a headphone at the same time: one channel on the left ear and the other channel on the right ear. They are then asked to attend or to *shadow* one of the channels.

Experiments in dichotic listening have shown few different findings. Cherry [53] discovered that participants were able to identify physical characteristics in the

unattended channel, such as whether the channel was a human voice or a noise, whether the speech was made by a male or female speaker, and whether the sex of the speaker changed during the test. However, participants did not notice the detailed aspects of the channel, such as the language it was spoken in, the semantic content and the individual words or phrases.

Moray [147] repeated Cherry's experiments and showed that participants still did not notice the individual words or phrases in the unattended channel eventhough the words were repeated many times. However, when he changed the audio messages in the channels to include the participants' names, he discovered that participants could switch their attention immediately from the attended channel to the unattended channel when they heard their name mentioned in the unattended channel. This finding suggested that the unattended channel was not completely ignored as initially discovered by Chery (i.e. since recognising a name required the analysis for semantic content in the unattended channel).

Gray and Wedderburn [87] as well as Treisman [225] have reconfirmed Moray's findings when they conducted similar experiments with more meaningful audio messages other than participants' names. They showed that participants could switch attention based on the meaning rather than the physical characteristics of the messages in different channels.

These dichotic listening studies have lead to three major theories that attempted to explain findings in selective or focused auditory attention:

1. **Filter theory**

   Broadbent [40] first developed a selective attention theory called the *filter theory*. He proposes that stimuli are filtered based on their physical characteristics to allow one of them to pass through, while the others to be held in the buffer for later processing. According to his theory, participants in the dichotic listening experiment focus on the physical characteristics of the messages in the attended channel. The messages in the unattended channel are not analysed for meaning, because they are on hold and not passed through. This theory is an early selection theory, because the relevant channel is selected before the meaning of messages is analysed. This theory cannot explain Moray's findings on why people can still hear their names in the unattended channel.

2. **Deutsch and Deutsch's Late-selection theory**

J. A. Deutsch and D. Deutsch [60] proposed an alternative theory to Broadbent's filter theory. They argue that all stimuli are allowed to pass through for meaning analysis and the most important stimulus is selected to determine the response. The filter happens near the response system not at the perceptual system. This theory is a late selection theory because a relevant channel is selected after the meaning of messages is analysed.

3. **Attenuation theory**

   Treisman [225] proposed a modified version of Broadbent's filter theory called the *attenuation theory*. She argues that all stimuli are filtered based on their physical characteristics, but all of them are allowed to pass through at different levels instead of some being held in the buffer. The stimulus that matches the physical characteristics will be passed at a normal level while others will be passed at an attenuated level. Stimuli are not eliminated by attention, instead they are rather enhanced or attenuated.

   This theory can explain Moray's finding on why participants can still hear their names in the unattended channel. Participants' names are not blocked entirely since stimuli are not held in the buffer. Instead, their names are still processed and allowed to pass through at an attenuated level. They switch their attention after the semantic content of the stimuli has been analysed.

5.2.2 Divided Attention

Divided attention refers to the ability of attention to accommodate and perform multiple tasks simultaneously (i.e. the ability of attention to be shared or divided between each task). Studies in divided attention explore the limit of simultaneous tasks and the possibility of whether multiple tasks can be performed without affecting the performance of each task. The ability to perform more than one task at the same time is called *multitasking*. There are three main theories that explain divided attention:

1. **Bottleneck theory**

   Welford [239] proposed the *bottleneck theory*. He argues that there is a bottleneck in the attention that can only accommodate one process at a time, so multitasking is achieved by continuously switching and alternating between

each task rapidly. Tasks are performed serially but they appear as if they are performed in parallel. Performance degradation does not occur when performing individual tasks separately, instead it occurs in multitasking because time is needed to switch between the tasks (i.e. context switching). This theory supports the selective attention theory that a selection is needed to choose a process among other multiple concurrent processes.

2. **Central capacity theory**

Kahneman [106] proposed the *central capacity theory*. He argues that instead of a bottleneck, there is a limited central capacity of attention that can be allocated across concurrent processes. The central capacity is like a *pool* of resources. When a process demands more capacity, there will be less capacity available for other processes, hence the performance of the other processes will be diminished. Multitasking can be achieved as long as there is enough capacity. According to this theory, simultaneous tasks will always interfere with each other because there is only one source of capacity shared among the tasks (i.e. the central capacity).

3. **Multiple resource theory**

Wickens [240] proposed the *multiple resource theory*. He argues that instead of a single central capacity, there are multiple capacity of attention, like multiple pools of resources, that can be allocated across concurrent processes. Multitasking can be achieved if tasks do not share the same type of resources. Wickens divides resources into three dimensions, namely stage, modality and code dimensions. The stage dimension consists of perception, cognition and responding. The modality dimension consists of visual and auditory perceptions. The code dimension consists of spatial and verbal activities. Resources are independent with respect to their dimensions. For example, a task or process that uses the visual modality in the perception stage will not interfere with the task or process that uses the auditory modality in the same perception stage. The multiple resource theory has also been supported by other findings, which propose that there will be interference when tasks share the same modalities [226] or responses [143].

5.2.3 Automaticity

Automaticity is the ability to perform tasks without much attention. One classical example of automaticity is the effect discovered by Stroop called the Stroop effect [215]. Stroop discovered that identifying the colour that does not match the reading of a text is more prone to errors and takes longer time than the colour that matches the reading of a text (see Table 5.1). For example, we get distracted and take longer time to identify the colour of *Blue* text than the colour of *Blue* text. In our minds, we automatically read and interpret the text (i.e. blue) despite our intention to identify its colour (i.e. red or blue colours).

Match:	Red	Green	Blue
Mismatch:	Red	Green	Blue

Table 5.1: Stroop Effect

Posner [179] defines three characteristics of automaticity, namely it occurs without conscious awareness, without intention and without requiring much effort and attention. For example, people can be walking automatically while reading a book, without being aware of the height they move their legs, without having intention to place each step at a particular location and without requiring much effort and attention to guide the walk. They simply walk automatically.

Theories of automaticity are divided into two main perspectives:

1. **Biological Perspective: Basic Reflexes**

   McKinley, O'Loughlin and Bidle [142, p. 564] defined a basic reflex to be an innate and pre-programmed response. It is a built-in and unlearned response that occurs automatically without conscious effort [202, p. 179]. It occurs based on a direct sensation of an environment performed outside the brain. For example, a person pulls out his hand away from a burning hot object before his brain can process the burning signals.

2. **Cognitive Perspective: Automatic Processing Actions**

   Neumann defined an action as a "sequence of movements that is controlled by the same internal control structure and that is *not* a reflex" [151, p. 375]. An action produced by an automatic process is developed through extensive and repeated learning, practice or training [198]. A task that initially requires

much effort and attention can be performed without much effort and attention after an extensive practice. An automatic processing action occurs based on the processing performed in the brain. It is very difficult to unlearn or change once it has been learned. It requires deliberate effort, attention and practice to modify. It is a context dependant, e.g. when a ball is seen, a striker may chase the ball whereas a goalkeeper may defend the goal.

Note that a basic reflex is different to an acquired (or conditioned) reflex. Sherwood [202, p. 179] proposed that there are two types of reflexes, namely a simple (or basic) reflex and an acquired (or conditioned) reflex. The simple (or basic) reflex is the innate action as defined by McKinley, O'Loughlin and Bidle described in the biological perspective above, whereas the acquired (or conditioned) reflex is the learned action as defined by Neumann described in the cognitive perspective above. In this dissertation, the term *reflex* is used to refer to the basic reflex instead of the acquired reflex.

There are three main theories that further explain automatic processes:

1. **Two-Process Theory**

   Schneider and Shiffrin [199, 203] proposed a *two-process* theory. They argued that there are two different kinds of processes, namely controlled and automatic processes. A controlled process demands attention. It is limited by an attention capacity. Only one process can be controlled at a time without interference. In contrast, an automatic process demands no or limited attention. It can run without a control and a capacity limitation. It is located in the long-term memory and activated by appropriate stimuli.

   In addition, Schneider and Shiffrin argued that an automatic process is developed only if mapping or connections between stimuli and responses are consistent throughout repeated learning, training and practices. When positive feedback is received, the connections are strengthened and their priorities are increased [198]. Consistent connections will increase their priorities and eventually responses can be performed with little effort or attention (i.e. become automatic with respect to the stimuli).

2. **Multi-Level Theory**

   Norman and Shallice [161] proposed a *multi-level* theory. They argued that instead of two kinds, there are three different kinds of processes, namely fully

automatic, partially automatic and deliberate control processes. Fully automatic processes are controlled by schemas (i.e. organised plans) and occur with little conscious awareness. Partially automatic processes are controlled by *contention scheduling* and occur with a higher degree of conscious awareness than fully automatic processes. Deliberate control processes are controlled by the *supervisory attentional system* and occur with fully conscious awareness.

Contention scheduling and supervisory attention are two separate systems. Contention scheduling resolves conflicts among schemas and selects a schema based on the information in the environment and current priorities. The supervisory attentional system oversees and controls the contention scheduling by influencing schema activation probabilities.

Neumann [150] supported the multi-level theory. He argued that processes are neither controlled nor automatic (i.e. not two level), instead their difference is on the level of control required (i.e. multi-level). He also argued that an automatic processing is not uncontrolled, instead it is controlled below the conscious awareness level.

3. **Instance Theory**

   Logan [135] argued that automaticity is memory retrieval of past solutions. He argued that "[people] can respond with the solution retrieved from memory or the one computed by algorithm. At some point, they may gain enough experience to respond with a solution from memory on every trial and abandon the algorithm entirely" [135, p. 493]. His theory suggests that processes become automatic if they can be performed by retrieving solutions from memory without computing the solutions.

   In addition, Logan argues that a controlled performance is limited by a lack of knowledge instead of resources. Automaticity does not occur because resources cannot be divided efficiently to perform multiple tasks simultaneously. Instead, it does not occur because knowledge is limited.

5.3 Computational Design and Model

Currently there is no widely accepted theory of selective attention, divided attention and automaticity. Instead of designing ASMO's attention mechanism based on a

specific theory of attention, I look at the previously discussed theories of attention from a computational perspective and accommodate all of the theories to design ASMO's attention mechanism. From a computational perspective, the theories can be realised into the same mechanism even though some of the theories may have different or opposing views from a psychological perspective.

In my perspective, attention is a noumenon–a phenomenon of the mind–that describes the selection of processes among other simultaneous processes. The processes are said to be *attentive* or given the *attention* if they are selected. Process selection is necessary because the system has limited resources, thus attention is the mechanism to mediate control of resources. Processes demand attention as long as they have not been served or satisfied.

In the following, I describe how each theory in the previous section inspires the design of ASMO's attention mechanism.

- **Central Capacity Theory**

  Inspired by Central Capacity Theory, ASMO's attention mechanism allows modules to compete for computing resources in order to perform their actions, such as central processing unit (CPU) or random access memory (RAM). These computing resources serve as a central capacity, because when a module demands some computing resources, there will be less resources available for other modules. Hence, these computing resources will affect the executions and performances of the modules.

- **Multiple Resource Theory**

  Inspired by Multiple Resource Theory, ASMO's attention mechanism allows modules to compete for any kind of multiple resources that they require (including computing resources) in order to perform their actions. Modules that do not require the same resources (non-conflicting modules) can be selected simultaneously. In contrast, modules that require the same resources (conflicting modules) cannot be selected simultaneously and will have a bottleneck (see also the design in the Bottleneck Theory below).

  In agent and robotic systems, resources are not necessarily divided into three dimensions as proposed by Multiple Resource Theory, namely stages, modality and codes. Instead, they can be any hardware or software component required by modules, such as arm actuators, leg actuators, camera sensors, CPU, RAM

and virtual resources (a competition over virtual resources is described further in Section 5.4.1).

- **Bottleneck Theory**

  The design in Multiple Resource Theory above already suggests that a bottleneck will occur in conflicting modules. ASMO's attention mechanism can only select one module among the conflicting modules at a time. It may need to continuously select different modules rapidly in order to achieve all tasks. In contrast, non-conflicting modules will have no bottleneck, because they do not interfere resources required by other modules.

- **Filter Theory**

  Inspired by Filter Theory, ASMO's attention mechanism selects modules only if the resources they competed for (i.e. their required resources) are available. Modules that are selected (called winners) will occupy and gain access to their required resources and their actions will be performed. Other modules that are not selected will have to re-compete and therefore appear to be held in the buffer.

- **Attenuation Theory**

  Inspired by Attenuation Theory, ASMO's attention mechanism requires modules to demand attention as a mean to compete for resources. In other words, modules compete for resources by competing for attention to use the resources. ASMO's attention mechanism will run this attention competition, will rank modules based on their attention demand and will select them from the highest to the lowest ranking. This selection means that a module with a higher ranking will be selected to occupy its required resources earlier than a module with a lower ranking. It may cause resources required by the lower ranking module to be unavailable, and thus the lower ranking cannot be selected. In general, this selection causes a module with a higher ranking to have higher chance to be selected than a module with a lower ranking. As a result, a module with a higher ranking will be selected more often (i.e. higher rate) than a module with a lower ranking. The higher the ranking, the less attenuated the module.

- **Deutsch and Deutsch's Late Selection Theory**

Inspired by Deutsch and Deutsch's Late Selection Theory, ASMO's attention mechanism allows modules to run concurrently to perceive situations, process information, propose actions and compete for resources. However, when conflicting modules occur, ASMO's attention mechanism will only select the most important module (i.e. the module with the highest attention demand) among the conflicting modules in order to perform the module's proposed actions.

In case of a dichotic listening experiment, two modules can analyse and propose to dictate an audio message from the left and right channel respectively. However, they will be in conflict, because dictating a message requires a control over a mouth and a person cannot dictate two messages with one mouth simultaneously. While they can analyse or propose actions concurrently, ASMO's attention mechanism will only select the most important module among these conflicting modules in order to dictate one of the messages (either from the left or right channel). This mechanism shows that the analysis and processing occur at an earlier stage, but the filtering of messages appears to be at near the response system.

- **Instance Theory**

  Inspired by Instance Theory, ASMO's attention mechanism allows modules that demand attention in a competition to be made automatic by retrieving solutions from memory and then boosting their attention demand in order to win the competition. This technique suggests that a module will have two properties, namely attention demand and boost. In this case, modules are now ranked based on both their attention demand and boost, instead of only their attention demand.

- **Two-process Theory**

  Inspired by Two-Process Theory, ASMO's attention mechanism allows modules to demand high attention explicitly (i.e. control their demand), or to be automatically boosted with high boosts. Their boosts will be acquired through consistent and repeated learning, training and practices until eventually they have high boosts and can be selected with little attention demand.

- **Multi-level Theory**

  Inspired by Multi-Level Theory, ASMO's attention mechanism requires the boosts possessed by modules to range continuously rather than in binary.

These boosts determine the automaticity of the modules. Continuous boosts means that modules are neither controlled nor automatic. Instead, the higher the boost, the more automatic the module, because modules with high boosts require less attention demand to win and be selected. Thus, modules can appear to be fully automatic, partially automatic or deliberately controlled.

Modules are said to be more controlled if they win an attention competition because they explicitly demand attention higher than their competitors (i.e. they are aware of the need of attention). In contrast, modules can be said to be more automatic if they win an attention competition because they have boosts higher than their competitors although they explicitly demand low attention (i.e. they are not aware of the need of attention).

In addition, inspired by Multi-Level Theory, ASMO's attention mechanism distinguishes modules into ordinary and supervisor types. Ordinary and supervisor modules are the same, except that supervisor modules can control ordinary modules deliberately and influence the probabilities of ordinary modules being selected (i.e. through the ordinary modules' attention demand and boosts).

- **Theories of Basic Reflexes**

  Inspired by theories of basic reflexes, and in addition to the design in Multi-Level Theory above, modules are now divided into ordinary, supervisor and reflex types. Ordinary and supervisor modules are non-reflex modules that demand attention and have boosts. They need to acquire boosts through learning in order to be automatic. Their rankings in an attention competition are determined by their attention demand and boosts. In contrast, a reflex module is pre-programmed to handle innate reflexes without demanding attention or having a boost. It is simply given a higher priority in the competition, which results in a higher ranking, than a non-reflex module.

5.4 Implementation

Previous section describes the design of modules and ASMO's attention mechanism based on theories of attention. Modules run concurrently to perceive situations, process information, propose actions and compete for resources in order to perform

their proposed actions. They are divided into three types, namely ordinary, supervisor and reflex. Non-reflex modules (i.e. ordinary and supervisor modules) compete for resources by demanding attention. They possess boosts to help their attention demand. Their boosts range continuously and are modified through learning. They are ranked based on their attention demand and boosts. In contrast, reflex modules compete for resources by using their priorities. They are simply given higher priorities (and thus higher ranking) than non-reflex modules. ASMO's attention mechanism selects modules to be the winners in an attention competition based on their rankings and the availability of their required resources (see illustration in Figure 5.1). The winners will gain access to their required resources, and their actions will be performed.

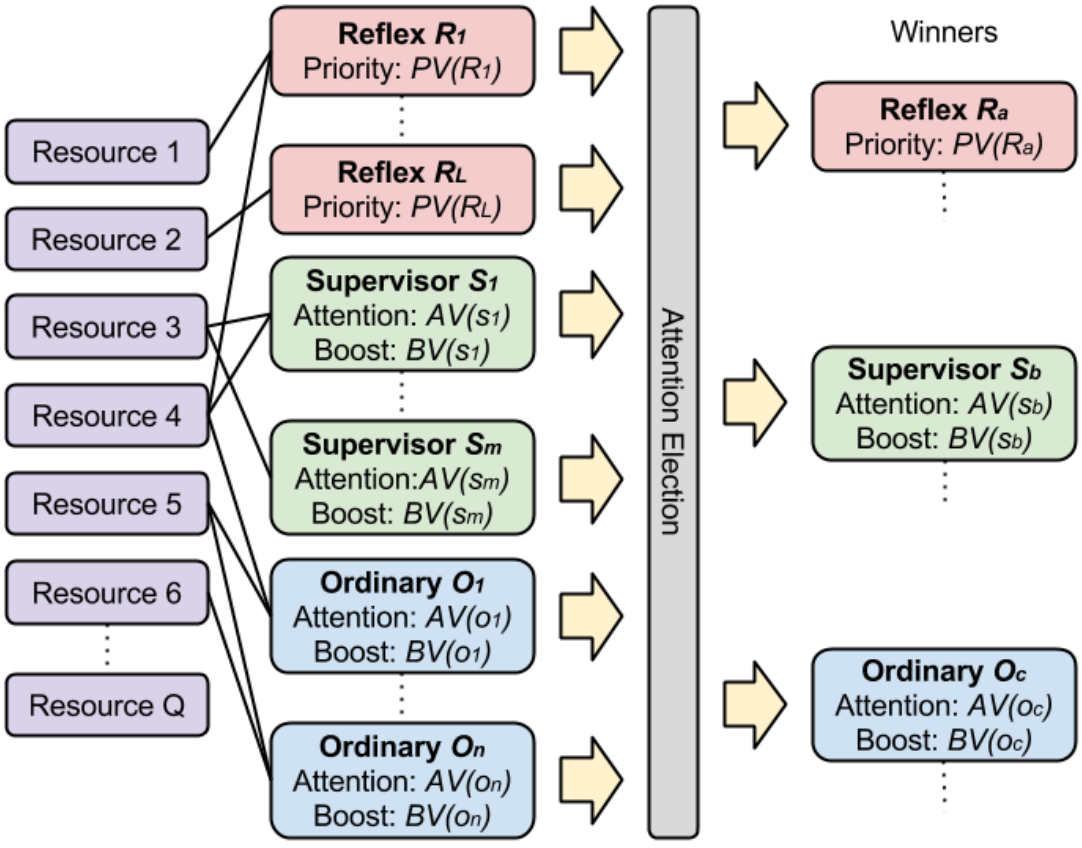


Figure 5.1: Attention Competition or Election

In the following, I describe the implementation of modules, attention demand, boosts and reflex priorities, and attention competitions.

5.4.1 Modules

A module is implemented as an autonomous, self-contained, concurrent, modular and distributed black box. It can store data in any representation to its local storage or to the global ASMO's memory. It can use any variety of techniques (such as algorithms, representations and mechanisms) to perceive situations, process information and propose actions in order to achieve tasks. Its actions require resources to perform. Modules have to compete for attention to gain access to their required resources in order to perform their proposed actions.

ASMO's attention mechanism allows modules to compete for any kind of resources that they require, such as hardware and software components. It also allows a competition over virtual resources, which is useful when actions proposed by the modules are high-level functions (e.g. learning) rather than low-level commands (e.g. move arms). A virtual resource can be required by modules that have different techniques to compete for solving the same problem. For example, one module may use a habituation learning to improve a robot's object tracking, whereas another module may use an operant conditioning learning. Without a competition of a virtual resource, both modules (i.e. two learning techniques) can be chosen simultaneously because they do not require any resource and conflicts in learning have to be handled separately. With a competition of a virtual resource, the two modules can be designed to compete for the same virtual resource, so only one module (i.e. one learning technique) will automatically be selected at a time.

In addition, a virtual resource can be used by modules that compete for the same goal but require different resources. This mechanism allows redundancy of modules that will make ASMO robust without having the modules to perform duplicate outcomes. For example, one module may require a speaker resource to deliver a message via voice, whereas another module may require a screen resource to display the same message. These two modules are redundant. When one module fails, the other module will still deliver the message. Without a competition of a virtual resource, however, ASMO's attention mechanism will select both modules since they require different resources, and ASMO will deliver the message via voice and via display (i.e. duplicate outcomes). While duplicate outcomes are fine in some applications, they are not desired in other applications. With a competition of a virtual resource, both modules can be designed to compete for the same virtual resource, so only one module will automatically be selected at a time. The other

module is ready to substitute the first module if failure occurs.

Similar to data in ASMO's memory (see Section 4.1.1), a resource is identified by an address formatted using a uniform resource identifier (URI)-like string, such as 'asmo/ram' or 'asmo/actuators/legs/left'. Reading this address provides a numerical real value that captures the available capacity of the resource. This capacity is bounded by convention between 0.0 and 100.0 (i.e. equivalent to a scaled value between 0.0 and 1.0). In discrete resources, the capacity of a resource is either 100.0 if the resource is available or 0.0 if the resource is unavailable. A module requires a resource by specifying the resource's address with a quantity (if a quantity is not given, then the module requires the whole resource). For example, a module that requires 'asmo/actuators/legs/left' means that it requires the whole left leg actuator whereas a module that requires ('asmo/ram', 25.8) means that it requires 25.8% of the whole random access memory.

Modules have different operations, which effect how their rankings are determined. The three types of modules are implemented as follows:

1. **Ordinary Module**

   An ordinary module has an *attention value* and a *boost value* to represent its attention demand and boost respectively (further described below in Section 5.4.2 and 5.4.3). By convention, it can only inspect and update its own attention value and boost value. In the current implementation, the sum of these two values determine the module's *total attention level*, which determines the module's ranking in an attention competition (further described below in Section 5.4.4). The higher the total attention level, the higher the ranking, the higher the chance of winning the attention competition.

2. **Supervisor Module**

   A supervisor module is like an ordinary module. It has an attention value and a boost value. Its ranking is determined by its total attention level, which is determined by the sum of its attention value and boost value. However, unlike an ordinary module, a supervisor module is allowed to inspect and modify the attention values and/or boost values of other non-reflex modules (i.e. ordinary and supervisor modules). There are two ways to modify these values:

   (a) **Competing Modification**

In competing modification, a supervisor module modifies the attention values and/or boost values of other non-reflex modules when it wins an attention competition, just like its proposed actions can be performed only when it wins an attention competition. Several competing supervisor modules that attempt to modify the same non-reflex module will compete in an attention competition and only the supervisor module with the highest ranking among them can be the winner to modify the non-reflex module. A competing modification is used to select one influencer at a time, e.g. to select a suitable learning technique at a time depending on the situations.

(b) **Direct Modification**

In direct modification, a supervisor module modifies the attention values and/or boost values of other non-reflex modules directly when necessary. Several supervisor modules that attempt to modify the same non-reflex module will not need to compete, and instead all of them will be included. The final attention value and/or boost value of the non-reflex module will be determined by the net result or effect given by all of the supervisor modules. A direct modification is used to combine influences from several supervisor modules at a time, e.g. to combine the effects from several learning techniques at a time to solve a problem. Note that this direct operation is only applicable to a supervisor module to modify attention values and/or boost values of other non-reflex modules. A supervisor module will still need to compete for attention in order to perform its proposed actions.

This design creates a separation (i.e. hierarchy) of controls between non-reflex modules. It allows a supervisor module to plan the interactions of non-reflex modules (by determining their attention values and/or boost values) in order to achieve goals and tasks. In ASMO, a supervisor module can use an existing AI planner like how the planner is normally used. However, the actions of the planner will not be commands to actuators directly. Instead, these actions will be to determine the attention values and/or boost values of non-reflex modules, so as the winners of attention competitions are determined by the planner. For example, a supervisor module can use an existing Markov Decision Process (MDP) planning system [192, p. 652] to plan a soccer attack behaviour by having actions from the MDP system to determine the attention values or

boost values of search ball module, chase ball module, kick ball module and other relevant modules.

In addition, this design (including a supervisor module modifying other supervisor modules) allows the description of complex behaviours. For example, in playing a soccer, an agent may be influenced by the fear emotion to defend an attack. However, the agent may learn from observations that chasing the ball is better than defending the attack. Meanwhile, after several attacks, the agent may habituate to the opponent's attack, thus becoming less fear to the attack. This example can be described in ASMO by handling the defend and chase behaviours as ordinary modules and by handling the emotion, observational learning and habituation learning as supervisor modules. Handling the emotion and observational learning as supervisor modules mean that they can influence the attention value and the boost value of defend and chase ordinary modules. In addition, handling the habituation learning as a supervisor means that it can modify the boost value of the emotion supervisor module to learn to become less fear (i.e. a supervisor module modifies another supervisor module).

In this dissertation, I focus on examples and applications of supervisor modules modifying ordinary modules. In addition, supervisor modules are not allowed to modify each other back and forth, and this issue remains future work (see Section 9.2).

3. **Reflex Module**

Unlike non-reflex modules (i.e. ordinary and supervisor modules), a reflex module does not have an attention value and a boost value. Thus, it does not have a total attention level to determine its ranking. Instead, it has a priority and it is simply given higher ranking than non-reflex modules. When competing with other reflex modules, its ranking is determined based on its priority.

Giving a reflex module higher ranking than non-reflex modules means that non-reflex modules cannot win an attention competition against a reflex module no matter how high they demand attention. However, humans seem to be able to perform non-reflex actions intentionally to overcome reflexes that are supposed to be triggered (i.e. non-reflex actions win over reflexes). For example, people can hold something that is hot to avoid spilling it. This capability

to overcome reflexes is not studied in ASMO and remains future work (see Section 9.2).

In ASMO, intelligent behaviours are emerged from interactions between constituent processes (i.e. modules), rather than from careful design or fine-tuning. However, if desired, modules and their interactions can be carefully designed and fine-tuned to achieve desired behaviours and effects. This careful design is achieved by determining and adjusting the priorities of reflex modules, the attention values of non-reflex modules and/or the boost values of non-reflex modules (e.g. manually adjusting the priorities and the values until desired behaviours are achieved or using a planner in a supervisor module to plan the priorities and the values). The more carefully designed the modules, the less emergent the behaviours.

Both the emergence-based ASMO and the carefully design-based ASMO have advantages and disadvantages. The emergence-based ASMO allows modules to be easily developed independently, but it can be difficult to coordinate modules in order to achieve desired combined behaviours. In contrast, the carefully designed-based ASMO allows modules (therefore behaviours) to be coordinated and combined as planned, but it can be difficult to develop modules independently. The manual adjustment in the careful design is a difficult and tedious process, but the use of a planner can help reducing the difficulties in coordinating or combining behaviours. ASMO is flexible and highly versatile. The combination of the emergence and careful design allows ASMO to support various types of developments of intelligent systems with less difficulties across a given set of tasks and environments.

5.4.2 Attention Value

An attention value captures the degree of attention a non-reflex module demands based on the tasks. It is an abstract numerical real value that can be set and read by the module locally (i.e. property of the module). It is bounded by convention between 0.0 and 100.0 (i.e. equivalent to a scaled value between 0.0 and 1.0), although theoretically it can be set to any value. The higher the attention value, the higher the attention demand sought by the module.

Attention values vary over time in accordance not only with the demand of the task, but also with the intention, motivation, goals and experiences of a robot. Both ordinary and supervisor modules have to modify or update their attention

values to win an attention competition, so that they can be selected to perform their proposed actions. An attention value is updated based on Equation 5.1, which involves following six parameters:

$$\underbrace{AV_t}_{\text{next value}} \leftarrow \underbrace{\alpha_t\, AV_{t-1}}_{\text{accumulation}} + \underbrace{\beta_t\, O_t}_{\text{objective}} + \underbrace{\gamma_t\, S_t}_{\text{subjective}} \tag{5.1}$$

Where:
$0 \leq \alpha_t \leq 1$
$0 \leq \beta_t \leq 1$
$0 \leq \gamma_t \leq 1$
AV_t is the attention value of the module at time t
α_t is the accumulation rate of the module at time t
β_t is the objective rate of the module at time t
γ_t is the subjective rate of the module at time t
O_t is the objective weight of the module at time t
S_t is the subjective weight of the module at time t

1. **Previously Accumulated Attention Value**

   The previously accumulated attention value, AV_{t-1}, is the attention value accumulated by the module in the previous update. Adding this value to the equation means that the current decision making is affected by past decision making (i.e. the decision making is sequential instead of episodic). Supervisors can be used to manage and manipulate the attention values in several updates in order to produce coherent sequences of decision making (i.e. plans).

2. **Objective Weight**

   The *objective weight*, O_t, denotes the change of attention sought by the module based on the objective evaluation of the tasks. For example, the objective weight of defending a goal module can be associated with the distance of the ball. The closer the ball, the higher the attention the module seeks to perform the defensive action.

3. **Subjective Weight**

   The *subjective weight*, S_t, denotes the change of attention sought by the module based on the subjective evaluation of the tasks. For example, the subjective weight of defending a goal module can be associated with the personality of

the goalkeeper. An offensive goalkeeper may have higher subjective weight to leave the goal area than a defensive goalkeeper. Subjective weight is optional. A module does not necessarily need to consider a subjective weight when updating its attention value. A subjective weight is described further in Section 6.4.3.

4. **Accumulation Rate**

   The *Accumulation rate*, α_t, denotes the proportion of the previously accumulated attention value. It can range between 0.0 to 1.0. An accumulation rate of 1.0 will make the module fully consider the accumulated attention value from the previous update (i.e. setting the decision making to sequential) whereas an accumulation rate of 0.0 will make the module ignore the accumulated attention value from the previous update (i.e. setting the decision making to episodic).

5. **Objective Rate**

   The *objective rate*, β_t, denotes the proportion of the objective weight (i.e. how much attention is affected by the objective evaluation). It can range between 0.0 to 1.0. An objective rate of 1.0 will make the module fully consider objective evaluation whereas an objective rate of 0.0 will make the module not consider objective evaluation at all (i.e. objective evaluation will be disabled). In other words, the module will only consider subjective evaluation when the objective rate is 0.0.

6. **Subjective Rate**

   The *subjective rate*, γ_t, denotes the proportion of the subjective weight (i.e. how much attention is affected by the subjective evaluation). It can range between 0.0 to 1.0. A subjective rate of 1.0 will make the module fully consider subjective evaluation whereas a subjective rate of 0.0 will make the module not consider subjective evaluation at all (i.e. subjective evaluation will be disabled). In other words, the module will only consider objective evaluation when the subjective rate is 0.0.

The accumulation, objective and/or subjective rates control the change in the attention value. They can help avoiding oscillations in the attention value. For example, the accumulation rate can be set to a value close to 1.0 and the objective

and subjective rates can be set to a value close to 0.0 in order to change the attention value slowly, and thus avoiding oscillations. These rates are similar to the learning rate and discounted factor in Q-learning reinforcement learning [236]. Changing these rates can produce various effects. Some of the effects are listed in the following Table 5.2.

α_t	β_t	γ_t	Effect	Description
0.0	0.0	0.0	$AV_t = 0$	Clear to zero
0.0	0.1	0.0	$AV_t = O_t$	Direct set to a new value
0.0	0.0	1.0	$AV_t = S_t$	Direct set to a new value
1.0	0.0	0.0	$AV_t = AV_{t-1}$	Maintain at previous attention value
1.0	1.0	0.0	$AV_t = AV_{t-1} + O_t$	Change from previous attention value
1.0	0.0	1.0	$AV_t = AV_{t-1} + S_t$	Change from previous attention value

Table 5.2: Effects for Different Rates in Attention Value

5.4.3 Boost Value and Reflex Priority

A boost value captures the degree of attention bias a non-reflex module has acquired based on tasks. In other words, it captures the degree of automaticity possessed by the module toward the tasks. Similar to an attention value, it is an abstract numerical real value that can be set and read by the module. The higher the boost value, the more automatic is the module.

Unlike an attention value, however, a boost value is modified as a result of extensive learning, practice and training instead of at every attention update. Modifying a boost value can be used to correct the total attention level of a non-reflex module that demand attention inappropriately (i.e. misbehave), e.g. when modules only care about getting their own proposed actions accepted.

Non-reflex modules can be made to behave more automatic or 'involuntarily' by increasing their boost values. Non-reflex modules that have high boost values require only low attention values to win the competition. They are likely to be selected before resources are running out (i.e. suffer no limitation of resources). Thus, they appear to run without attention demand, control and capacity limitation.

In ASMO, non-reflex modules do not become more automatic by reducing their attention demands (i.e. reducing attention values). Instead, they become more

automatic through higher boosts (i.e. higher boost values), so that they can easily win the attention competition and be selected without requiring much attention values.

Non-reflex modules that become more automatic through boosting have different operations to reflex modules in achieving automaticity. They still have to compete for attention through their attention and boost values. They may not necessarily win the attention competition. There is a chance for them to lose the attention competition over other modules without boosting or with zero boost values (i.e. automatic processes lose over controlled processes).

In contrast, reflex modules are special modules that compete for attention without having attention values and boost values. They are given a higher priority than ordinary and supervisor modules in the competition. They always win the attention competition against ordinary and supervisor modules (because of the higher priority). In this case, they do not have to compete with ordinary and supervisor modules. The higher priority is given to ensure that highly critical actions can be performed when necessary without the need to wait for sufficiently high attention.

5.4.4 Attention Competition

An attention competition (also called *attention election*) is used to mediate the selection of modules, the emergence of complex behaviours and the management of limited resources. Modules can be selected and their proposed actions can be performed only if their required resources are available. Since a robot has limited resources, modules have to compete for attention to gain access to resources.

Modules that are selected as winners will gain access to the resources they requested so as to perform their actions. Their access to the resources will be cleared in the next attention competition. As a result, their actions will be interrupted if the actions have not finished yet. This interruptive behaviour is great to prevent any module from locking the resources and to keep responsiveness of the system. It motivates developers to develop and use non-blocking actions.

ASMO's attention mechanism can be configured to run an attention competition at any suitable situations (e.g. every interval or on-demand basis). In every competition, modules propose actions to perform and ASMO's attention mechanism will select modules as the winners based on their types, their attention levels and

the availability of their required resources (see Algorithm 5.1), as described in the following five steps:

1. **Rank Modules**

   ASMO's attention mechanism ranks or sorts modules from the highest to the lowest ranking: starting with reflex modules from the highest priority to the lowest priority, and followed by non-reflex modules from the highest total attention level to the lowest total attention level. A non-reflex module's total attention level is determined based on Equation 5.2.

$$\begin{aligned} TAL(p) &\leftarrow BV_t(p) + AV_t(p) \\ \underbrace{AV_t(p)}_{\text{next value}} &\leftarrow \underbrace{\alpha_t(p)\ AV_{t-1}(p)}_{\text{accumulation}} + \underbrace{\beta_t(p)\ O_t(p)}_{\text{objective}} + \underbrace{\gamma_t(p)\ S_t(p)}_{\text{subjective}} \end{aligned} \tag{5.2}$$

   Where:
   $0.0 \leq \alpha_t \leq 1.0$
   $0.0 \leq \beta_t \leq 1.0$
   $0.0 \leq \gamma_t \leq 1.0$
   $TAL(p)$ is the total attention level of module p
   $BV_t(p)$ is the boost value of module p at time t
   $AV_t(p)$ is the attention value of module p at time t
   $\alpha_t(p)$ is the accumulation rate of module p at time t
   $\beta_t(p)$ is the objective rate of module p at time t
   $\gamma_t(p)$ is the subjective rate of module p at time t
   $O_t(p)$ is the objective weight of module p at time t
   $S_t(p)$ is the subjective weight of module p at time t

2. **Check Required Resources**

   ASMO's attention mechanism checks each module from the highest to the lowest ranking whether the resources required by the module are available or not.

3. **Update Available Resources, Select Module, Perform Actions**

   If the resources required by the module are unavailable, then ASMO's attention mechanism checks the next module. Otherwise, if the required resources are available, then ASMO's attention mechanism will perform the following actions:

(a) remove the required resources from the list of available resources (i.e. the required resources become unavailable).

(b) select the module as one of the winners.

(c) perform actions proposed by the module.

4. **Repeat Until Finish**

   ASMO's attention mechanism repeats Step 1 to Step 3 until either all modules have been checked or all available resources are used (i.e. no more remaining resources)

```
1  available_resources ← get_possible_resources()
2  winners ← [ ]
3  # Reflex modules from highest to lowest priority then non-reflex
   modules from highest to lowest attention levels
4  ranked_modules ← sort(reflex_modules, supervisor_modules,
   ordinary_modules)
5  # The highest ranking is 1
6  for i ← 1 to count(ranked_modules) do
7      m ← ranked_modules[i]
8      if (required_resources(m) in available_resources) then
9          # arithmetic operators are applicable to matrix and array
10         available_resources ← available_resources - required_resources(m)
11         winners ← winners + m
12         a ← get_proposed_actions(m)
13         perform(a)
14     end
15     if available_resources is empty then  break the loop ;
16 end
```

Algorithm 5.1: Module Selection Algorithm in ASMO's Attention Mechanism

Modules can propose actions at any time. However, they will not be selected and their proposed actions will not be performed until an attention competition. This attention competition allows modules to have enough time to propose actions and aggregate attention. In addition, it balances between the reactivity of the system and the computing power used by the system. The more frequent the attention

competition, the more reactive the system, the more computing power used by the system.

In an attention competition, there can be multiple winners where each winner occupies different resources. In this case, each resource is used by no more than one winner. Multiple winners allow multiple actions that require different resources to be performed simultaneously. Hence, ASMO can make multiple decisions at one time (i.e. simultaneous action selection). It can achieve multiple goals simultaneously.

Modules are ranked and then selected from the highest to the lowest ranking, in order to ensure that modules with a higher ranking have the privilege to use their required resources before the modules with a lower ranking. In this case, reflex modules are given the priority to lock and use their required resources before non-reflex (i.e. ordinary and supervisor) modules. In addition, a non-reflex module that has a higher attention level is given the privilege to occupy and use its required resources before a non-reflex module that has a lower attention level. Modules that have no conflict in using their required resources will eventually be selected despite low ranking, because they have no competitors (i.e. they will win the competition for the resources they require).

In the current implementation of ASMO, non-reflex modules are ranked simply based on the amount of their total attention levels (i.e. the sum of attention values and boost values). A non-reflex module is placed in a ranking higher than other non-reflex modules as long as its total attention level is higher than theirs, even if its total attention level is just slightly higher. For example, a non-reflex module with a total attention level of 1.01 has a higher ranking than a non-reflex module with a total attention level of 1.00. This approach of determining ranking can be improved in future work.

Sometimes a ranking cannot be determined because non-reflex modules have the same total attention levels. When this happen, ASMO's attention mechanism has to use a *conflict resolution strategy* to select which module is to be given a higher ranking. Some of the possible conflict resolution strategies are as described as follows:

- **Random**

  In *random* strategy, ASMO's attention mechanism randomly places one of the equivalent modules in a higher ranking.

- **Minimal Resources**

  In *minimal resources* strategy, ASMO's attention mechanism places the module that uses the least resources (i.e. the lowest number of resources) in a higher ranking. This strategy opens more opportunity for other non-reflex modules to win on the unused resources. Hence, it is a good strategy for producing more winners. ASMO's attention mechanism has to employ another strategy when there is more than one module that uses the least resources.

- **Maximal Resources**

  In *maximal resources* strategy, ASMO's attention mechanism places the module that uses the most resources (i.e. the highest number of resources) in a higher ranking. This strategy creates less opportunity for other non-reflex modules to win on the unused resources. Hence, it is a good strategy for producing less winners. ASMO's attention mechanism has to employ another strategy when there is more than one module that uses the most resources.

- **Previous Winner**

  In *previous winner* strategy, ASMO's attention mechanism places the module that wins the previous competition in a higher ranking. The module is relevant if it wins the previous competition. ASMO's attention mechanism has to employ another strategy when there is more than one module that wins the previous competition or none of the equivalent modules wins the previous competition.

- **Most Frequent Winner**

  In *most frequent winner* strategy, ASMO's attention mechanism places the module that wins the most competitions in a higher ranking. The module is important if it wins many competitions. ASMO's attention mechanism has to employ another strategy when there is more than one module that wins the most competitions or none of the equivalent modules has ever won previously.

- **New Winner**

  In the *new winner* strategy, ASMO's attention mechanism places the module that has never won a competition in a higher ranking. This strategy explores more variety of actions. ASMO's attention mechanism has to employ another strategy when there is more than one module that has never won a competition.

All strategies described above, except the random strategy, require ASMO's attention mechanism to employ another strategy when there is a further tie among the equivalent modules. ASMO's attention mechanism can use a mix of strategies to resolve the ties. For example, it can first select the equivalent modules based on the *most frequent winner* strategy, followed by the minimal resources strategy if none of the equivalent modules has ever won previously, and finally followed by the *random* strategy if there is more than one module that uses the least resources.

5.5 Other Work

In this section, I discuss computational models that are similar to ASMO's attention mechanism (i.e. used for decision making or action selection), regardless of whether they are relevant to attention or not. This is because there is a lack of computational models of attention designed for decision making or action selection. Research on attention in the literature have been focusing on the area of visual attention, rather than other areas such as decision making or action selection. The majority of computational models of attention have also been developed for the purpose of visual attention [75, 101].

Majority of these previous models have been realised into robot architectures and/or multi-agent architectures. I have described these architectures in Section 3.4. They are outside the scope of this dissertation. In the following, I briefly describe how their mechanisms are similar and different to ASMO's attention mechanism.

Hambuchen [93] has proposed a computational model of attention in her dissertation called Sensory Ego-Sphere (SES) attention network. This model is not used directly for action selection. Instead, it is used with a behaviour system implemented in ISAC robot to select actions. It selects the most salient event in the environment based on the event's incidence, task relevancy and regularity. Similar to ASMO's attention mechanism, SES selects the most salient event among competing events as the winner, and then behaviours are selected based on the winner.

Breazeal [39] has developed a behaviour system for action selection in her dissertation. This behaviour system has been implemented in Kismet robot. It is designed based on homeostatic regulatory mechanisms from the ethology literature. It is organised into loosely layered heterogeneous hierarchies of behaviour groups. Each group contains behaviours that compete for activation (through activation levels)

with one another. Each behaviour determines its own activation level based on its relevance to the situation by taking into account perceptual factors as well as internal factors. At the highest level, behaviours are organised into competing functional groups (i.e. the primary branches of the hierarchy) where each group is responsible for maintaining one of the three homeostatic functions, namely to be social, to be stimulated by the environment and to occasionally rest. Only one functional group can be active at a time. This behaviour system (i.e. Kismet's behaviour system), behaviours and activation levels are similar to ASMO's attention mechanism, modules and total attention levels respectively.

Similarly, Fujita et al. [77] have proposed an action selection mechanism based on homeostatic regulatory mechanisms from the ethology literature. This mechanism is embedded in a robot architecture called the emotionally grounded (EGO) architecture. In this architecture, behaviours (called modules) compete for activation using their activation levels. Each module determines its own activation level and requires resources. Modules are organised into a hierarchical tree structure, which can be thought like a layered hierarchical groups in Kismet's behaviour system. Modules that have the same parent in the tree will share the same target. Unlike Kismet's behaviour system, multiple modules can be activated if they have different targets (i.e. multiple modules in different groups can be active at a time). In addition, modules are selected concurrently in the competition from the highest activation level to the lowest activation level until there is no remaining resource available. Once a module is selected, it is given permission to execute the behaviour implemented in the module as a state machine (i.e. executing a state machine behaviour). Both module organisation and competition mechanisms are designed to manage resources. EGO's modules, activation levels, competition are very similar to ASMO's modules, total attention levels and competition respectively (which I developed independently).

Rosenblatt [186] has proposed an action selection mechanism based on voting in his dissertation. This mechanism is embedded in a robot architecture called Distributed Architecture for Mobile Navigation (DAMN). Initially, he proposed three voting mechanisms to select actions, namely constraint arbitration, actuation arbitration and effect arbitration. However, he later replaced these three mechanisms with the utility fusion voting mechanism. He proposed that this utility fusion voting mechanism can solve issues encountered by the first three mechanisms. DAMN consists of independent, distributed and asynchronous modules that handle the robot's

behaviours to achieve tasks. Every module concurrently suggests an opinion (i.e. vote) for *every* candidate or action depending on the situations in the environment. DAMN takes these opinions to decide on an action. DAMN's voting mechanism and modules are similar to ASMO's attention mechanism and modules respectively. ASMO allows a module to provide an opinion about other modules' actions by biasing their total attention levels. The module may increase or decrease their total attention levels when it favours or disfavours their proposed actions respectively.

Arkin [20] has proposed an action selection mechanism based on vector responses in his dissertation. This mechanism is embedded in a robot architecture designed specifically for navigation called Autonomous Robot Architecture (AuRA). This architecture has a schema controller that manages, controls and monitors all *motor schemas* at run time. Each motor schema can operate asynchronously and generates a response vector to control motors. AuRA sums and normalises all vectors received from all motor schemas before sending the result to the motors for execution. AuRA's vector-based action selection mechanism and motor schemas are not really similar to ASMO's attention mechanism and modules. However, I discuss them because their differences to ASMO's attention mechanism and modules are interesting (see the discussion below).

Parker [173] has proposed an action selection mechanism based on motivation in her dissertation. This mechanism is similar to the action selection mechanism in EGO architecture. It is embedded in a multi-agent architecture called ALLIANCE to address fault tolerant issues. ALLIANCE consists of *motivational behaviours*. At all times, each behaviour defines its activation level based on inputs, including sensory feedback, inter-robot communication, inhibitory feedback from other active behaviours and internal motivation. Its activation level is a non-negative number. When this activation level exceeds a given threshold, the corresponding behaviour set becomes active. There are two variables used to determine the activation level, namely robot impatience and robot acquiescence. The robot impatience enables a robot to handle situations when other robots (outside itself) fail in performing a given task. The robot acquiescence motivation enables a robot to handle situations in which it, itself, fails to properly perform its task. ALLIANCE's activation levels and behaviours are similar to ASMO's total attention levels and modules respectively.

Several auction-based action selection mechanisms have been proposed in the

literature [61, 86]. One example is the mechanism embedded in TraderBots multi-agent architecture developed by Dias in her dissertation [61]. It is designed based on the economic approach of free market system. In TraderBots, a larger task is divided into sub-tasks. Each task is associated with a revenue and a cost. A team of robots places bids for tasks. The goal of the team is to execute some plans such that its profit is maximised or its cost is minimised. This mechanism involves principle approach towards choosing these bid values. It is not particularly similar to ASMO's attention mechanism, but it is interesting for comparison (see the discussion below).

Stone did not propose an action selection mechanism, rather he has proposed an organisation of behaviours that effects the behaviour selection [213, 214]. This organisation is embedded in a multi-agent architecture called Team Member Agent Architecture (TMAA). He has evaluated this architecture in a RoboCup Soccer Small Size League competition. This architecture is designed for creating teams of agents in periodic team synchronisation (PTS) domains. In this architecture, agents collaborate based on pre-determined multi-agent protocols called locker-room agreements. Each agent has *behaviours* that are represented as directed acyclic graphs (DAGs). A behaviour is a set of condition/action pairs where conditions are logical expressions over inputs, and actions are the behaviours themselves. This architecture allows parts of the agent's behaviours (i.e. sub-graphs) to be switched based on the agent's role and formation. Various learning techniques have been applied to this DAGs structure in order to help make decisions, such as decision tree learning and memory-based learning. Similarly, ASMO has been evaluated in a RoboCup Soccer SPL competition and various learning techniques have been applied to ASMO to improve decision making (see Chapter 8).

Previous works have been done to address varieties of issues. It is worth to note that all of those described above, except EGO, are results and development from doctoral dissertations. Despite much work and several dissertations, they still lack of some capabilities described below. ASMO and its attention mechanism are designed to address this lack. They are different to previous works in the following:

- **Common: Resource Management**

  ASMO's attention mechanism manages resources required by modules automatically through attention competitions without the need for developers to manually organise modules. In contrast, SES, Kismet's behaviour system, EGO's motivation mechanism, DAMN's voting mechanism, AuRA's vector

mechanism, ALLIANCE's motivation mechanism, TraderBots' auction mechanism and TMAA's system require developers to manage resources manually. EGO has a similar competition technique to ASMO, however it still requires developers to organise modules manually.

- **Common: Action Selection**

  ASMO allows mechanisms (such as its learning mechanisms) to correct and modify the action selection through boost values. The competition of non-reflex modules is not only determined by attention values that are determined by the modules themselves, but also influenced by boost values that can be determined by other modules. In contrast, the competition of behaviours in Kismet's system, the competition of modules in EGO and the voting of modules in DAMN are determined by activation levels or opinions that are determined only by the modules themselves.

- **Common: Critical Actions**

  ASMO's attention mechanism allows reflex modules to serve critical actions. In contrast, SES, Kismet's behaviour system, EGO's motivation mechanism, DAMN's voting mechanism, AuRA's vector mechanism, ALLIANCE's motivation mechanism, TraderBots' auction mechanism and TMAA's system do not have a mechanism to handle critical actions.

- **SES: Non-Goal Related Attention**

  In ASMO's attention mechanism, an unexpected event (handled by a module) can still gain attention even though it is not related to the goal. This is because an unexpected event can lead to a new goal that is turn out to be more important than the current goal. For example, an alarm may unexpectedly ring and the robot needs to evacuate when the robot is about to grab an object. In this case, the alarm event leads to a new goal of survival that is unrelated to but more important than the robot's current goal of grabbing an object. In contrast, SES increases the salience of an unexpected event only if the event is related to the robot's goal [93, p. 46].

- **Kismet: Unrestricted Responsibility of Modules**

  ASMO's attention mechanism does not require modules to have certain responsibilities, whereas behaviours in Kismet's system are divided into functional groups that have responsibilities limited to three homeostatic functions.

- **Kismet: Concurrent Purposes or Functional Groups**

  ASMO's attention mechanism allows multiple modules to perform concurrently despite that they have different purposes or are belong to different functional groups. In contrast, Kismet's behaviour system only allows behaviours to perform concurrently if they are within a group, but not if they are in different functional groups, since only one functional group can be active at a time.

- **EGO: Unrestricted Behaviour Representation**

  ASMO's attention mechanism allows modules to use any kind of representation, whereas EGO's modules are limited to use a state machine.

- **DAMN: Free Opinion or Voting**

  ASMO's attention mechanism allows modules to freely provide opinion when necessary. Modules may not provide a bias if they have no opinion, or they may choose to not provide a bias even though they have an opinion. In contrast, DAMN's voting mechanism requires every module to provide an opinion for every action, which can be difficult in developing the modules because an opinion may be unavailable for certain actions or choices.

- **AuRA: Guarantee Within Possible Choices**

  ASMO's attention mechanism allows the aggregation of modules' total attention levels, not the aggregation of actions. It guarantees that its selected actions are always within the possible choices. For example, one module may propose to turn left (i.e. -90deg) while another module may compete equally strong to propose to turn right (i.e. 90deg). The action selected by ASMO's attention mechanism will be either to turn left or to turn right, even if the two modules have equal total attention levels. In contrast, AuRA's vector mechanism aggregates all vector responses for actions. It may produce actions that are not feasible. For example, a motor schema may generate a vector in the left direction of -90deg while another motor schema may generate an equivalent vector response in the opposite right direction of 90deg. The sum of these vector responses will result in a value near 0deg (i.e. to go straight), which is not within the possible choices of turning left or right. In many applications, it is not fine for a robot to take an action that is not within the possible choices. For example, given two possible paths on the left and right,

it may not be fine for a robot to take the middle path to go straight because there may be no path in the middle at all.

- **ALLIANCE: No Threshold**

  ASMO's attention mechanism selects non-reflex modules based on their required resources and total attention levels. There is no involvement of thresholds in selecting the modules. ASMO's attention mechanism will not select non-reflex modules even though they have high total attention levels, if their required resources are not available or another module has higher total attention level than theirs. In contrast, ALLIANCE's motivation mechanism selects behaviours as long as their activation levels exceed a given threshold. In this case, developers are required to define appropriate thresholds, which can be difficult to define.

- **TraderBots: Revenue–Cost Model**

  ASMO's attention mechanism selects non-reflex modules based on their total attention levels, whereas robots in TraderBots' auction mechanism select tasks based on the tasks' revenues and costs.

- **TMAA: Evaluation and Learning**

  ASMO and its attention mechanism have been evaluated in a RoboCup Soccer Standard Platform League competition. This competition does not allow modifications of robots' hardware, so ASMO and its attention mechanism could be purely evaluated. In addition, various learning techniques are applied to ASMO and its attention mechanism to modify the total attention levels of non-reflex modules in order to improve action selection or decision making. These learning techniques are habituation, sensitisation, operant conditioning, classical conditioning and observational learning (see Chapter 7). In contrast, TMAA has been evaluated in a RoboCup Soccer Small Size League competition. This competition allows modifications of robots' hardware to improve performance. Thus, a robot software architecture (such as TMAA) demonstrated to work in this competition is difficult to evaluate whether that is purely due to the software architecture itself or both the robot's hardware and the software architecture. In addition, various learning techniques are applied to TMAA's directed acyclic graphs structure to help action selection or decision making. These learning techniques are decision tree learning and memory-based learning, which are different to those applied to ASMO.

Chapter 6

ASMO's Emotion Mechanism

In the previous chapter, I described how ASMO's attention mechanism selects modules (i.e. performs decision making). In this chapter, I describe the details of ASMO's emotion mechanism and how it subjectively biases the selection of modules. I start by explaining the needs for emotion and subjective bias in Section 6.1. I follow by exploring theories of emotion in Section 6.2. I describe the interpretation of these theories in Section 6.3. These theories form the basis of the design of ASMO's emotion mechanism. I then describe the implementation of ASMO's emotion mechanism in Section 6.4. Finally, I review other work on emotion mechanisms and compared them with ASMO's emotion mechanism in Section 6.5.

6.1 Needs for Emotion and Subjective Bias

Studies have found that emotions bias and influence attention [249, 73, 63, 250] and decision making [96, 134, 8, 204]. Anderson [8] argues that decision making incorporates rational evaluations and inferences along with anticipated emotions.

An emotion provides a *subjective bias*: the tendency towards something as a result of personal evaluations or judgements. These emotions and subjective biases have different influences on an individual's behaviours. For example, some people would undertake actions based on the pursuit of goals despite their emotions (e.g. do bungee jumping to receive money despite the fear of height), while others would choose differently in the same situation based on their emotions despite their goals (e.g. refuse bungee jumping because of fear of heights, despite the money offered).

In ASMO, intelligent behaviours emerge from the interaction of modules by selecting modules via the attention mechanism. In some circumstances, the selection of modules has to be biased subjectively to guide the emergence, as described as follows:

- **Social Interaction**

  Robots can be required to behave socially when operating in society such as an everyday environment. This social behaviour requires modules to be selected not only based on the goals, objective functions or performance functions, but also based on social factors or subjective bias, such as emotions and personality (i.e. social factors are included as part of the goals).

- **Soft-constraints and Refinement of Solutions**

  Robots can have several choices of acceptable solutions to achieve goals. While they must meet some constraints required by the goals (i.e. hard-constraints), they may have other constraints that are desirable but not necessarily required by the goals (i.e. soft-constraints). A subjective bias provides a mechanism for the robots to refine their choices based on soft-constraints.

  For example, in a soccer game, a robot has a choice to chase the ball either offensively or defensively. Both offensive and defensive chase actions are acceptable, because either of them will lead the robot to the ball. While both chase actions require the robot to know the location of the ball (i.e. hard-constraints), a subjective bias allows the robot to refine and choose one of the chase actions.

- **Human Modelling**

  Robots can be required to model or mimic human behaviours. Humans have individual preferences that subjectively bias their selection of actions. Modelling human behaviours requires the selection of modules to accommodate similar bias found in humans.

- **Selection Simplification**

  Selecting modules based on optimal and accurate calculations can be complicated and can incur high computational cost. Subjective bias may provide approximate and less accurate selection of modules that is less complicated and has lower computational cost.

- **User's Influence**

  Subjective bias in the selection of modules provides a mechanism for a user to influence the system created by the developers. Developers do not need to change the algorithms and techniques used in the modules for accommodating different users. Instead, they can develop modules to achieve the tasks and use subjective bias to influence the selection of the modules according to the user's preference.

ASMO supports biasing the selection of modules subjectively through its emotion mechanism. The use of this subjective bias is optional. In circumstances outside those described above for example, the selection of modules may not need to be biased (i.e. subjective bias may not be necessary).

6.2 Theories of Emotion

ASMO's emotion mechanism is inspired by human emotions. Theories of human emotions are divided into three major areas, namely representation, causality and evaluation.

Emotions have been defined differently from various perspectives [209]. However, there is something common about emotion in general. Emotions can only be experienced within an individual, so they must be perceived and observed by others externally. They can be perceived explicitly through the individual's statements (i.e. verbal communication) or implicitly through the individual's behaviours and physiological responses (i.e. non-verbal communication). The genuineness of an emotion cannot be confirmed since we cannot inspect the inside of the person's mind to recognise the person's emotion. A person who looks down and shows tears may be perceived as being sad although the person may be faking it.

6.2.1 Representation

Theories of emotional representation explore the composition of emotions and how emotions are connected to each other. There are two major theories that describe the representation of emotions:

1. **Basic or Discrete Emotion Theory**

Basic emotion theories (also known as discrete emotion theories) [171, 66] propose that there are basic emotions (also called primary emotions) that are inherited (i.e. innate) and universally the same across different cultures. According to this theory, a more complex emotion (also called secondary emotion) is represented by or composed of a combination of basic emotions. This representation of emotion is analogous to the representation of a colour using a combination of three primary colours of red, green and blue (i.e. tristimulus values).

Researchers have proposed various basic emotions ranging from two to eleven basic emotions [185, p. 309]. Currently there is no widely accepted theory on which emotions are considered basic. For example, Ekman [66] proposes six basic emotions, namely anger, disgust, fear, happiness (i.e. joy), sadness and surprise. Shaver et al. [200] propose six different basic emotions, namely anger, fear, joy, love, sadness and surprise. The difference to Ekman's is that Shaver et al. propose love instead of disgust as one of the basic emotions. In contrast, Plutchik [177] proposes eight basic emotions, namely anger, anticipation, disgust, fear, joy, sadness, surprise and trust.

2. **Dimensional Theory of Emotion**

Dimensional theories of emotion propose that emotions are represented by the degrees of one or more dimensions. This representation of an emotion is similar to the representation of a colour based on the degrees of hue, saturation and intensity dimensions. According to this theory, two emotions can be represented by the same basic emotions, but they are distinguished by the degrees of the dimensions. For example, both annoyance and frustration can be represented by the same basic emotion of anger, but annoyance has a higher degree of arousal than frustration.

Researchers have proposed various dimensions for representing emotions [191, 177]. Currently there is no widely accepted theory on which dimensions are correct. For example, Russell [191] proposes two dimensions, namely valence and arousal, which indicate the pleasantness and the excitement of a situation respectively. Watson and Tellegen [237] propose two different dimensions, namely positive affect and negative affect. In contrast, Plutchik [177] proposes four dimensions, namely joy versus sorrow, anger versus fear, acceptance versus disgust and surprise versus expectancy.

There are two kinds of dimensions used in the dimensional theory of emotion, namely bipolar dimension [191] and bivariate dimension [162, 237]. A bipolar dimension captures the degrees of two opposite qualities in a single variable. Increasing one of the qualities implies that the other quality is decreased by the same amount. A bipolar dimension suggests that two opposite qualities cannot occur at the same time. For example, a valence dimension can be used to capture the degrees of pleasantness and unpleasantness. A valence that indicates a situation is 70% pleasant implies that the situation is 30% unpleasant. If it indicates that the situation is 80% pleasant instead of 70%, then it implies that the situation is 20% unpleasant.

In contrast, a bivariate dimension captures the degree of only one quality in a single variable. The degrees of two opposite qualities have to be captured by using two separate dimensions. Increasing one of the qualities does *not* necessarily imply that the other quality is decreased by the same amount. A bivariate dimension suggests that two opposite qualities can occur at the same time. It is supported by the findings that show people can feel both happy and sad at the same time [129]. For example, a positive valence dimension and a negative valence dimension can be used to capture the degrees of pleasantness and unpleasantness respectively. The positive valence can indicate that the situation is 70% pleasant while at the same time the negative valence can indicate that the situation is 60% unpleasant. Changing the positive valence does not necessarily change the negative valence. Both positive valence and negative valence dimensions can be independent.

6.2.2 Causality

Theories of emotional causality explore the cause of emotions in terms of a feeling, a physiological state, a behaviour and an appraisal (see Figure 6.1). Most would think in common sense that physiological states and behaviours are elicited because of our feelings. We cry because we are sad, we laugh because we are happy and we fight because we are angry. However, four major theories of emotional causality explain the cause of emotions differently:

1. **James–Lange Theory**

   James–Lange theory refers to the two theories developed independently by William James [102] and Karl Lange (also known as Carl Lange) [125, 124] respectively. This theory proposes that physiological states and behaviours

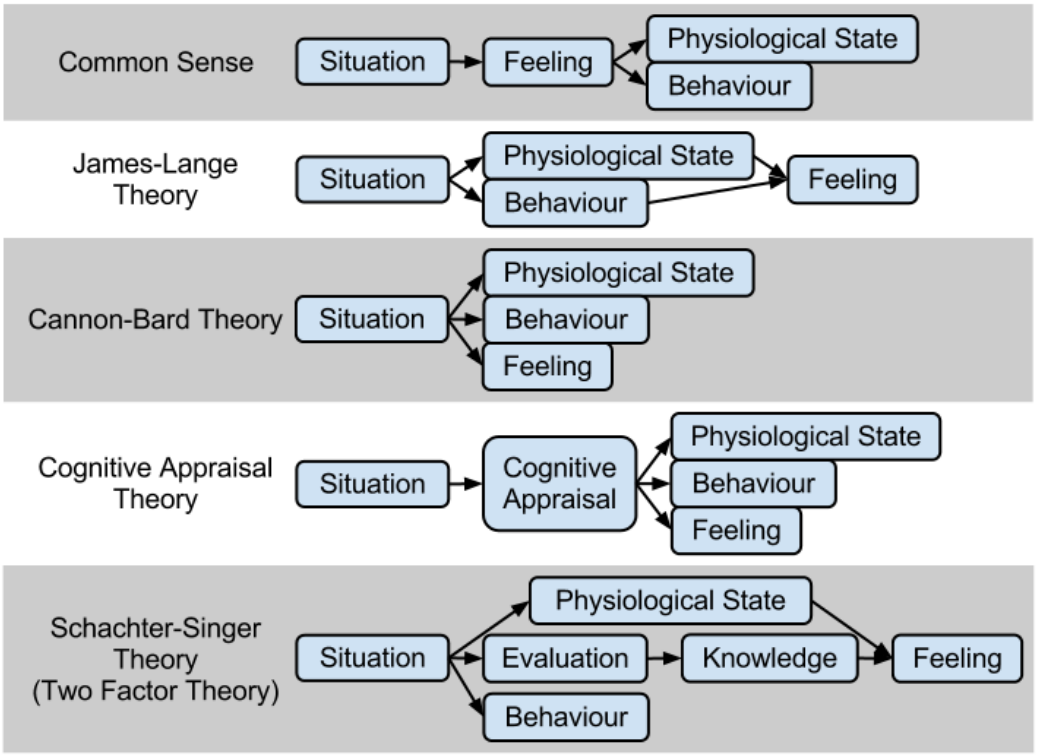


Figure 6.1: Theories of Emotional Causality

are not elicited because of a feeling (as suggested by common sense), instead a feeling is elicited because of physiological states and behaviours (i.e. the order is reversed). This theory can explain the reason why humans can feel surprise based on the startle reflex caused by a sudden loud noise [123]. According to this theory, we are sad because we cry, we are happy because we laugh and we are angry because we fight. A feeling is just the label given to the behaviours and the physiological states that are evoked automatically based on the situation in the environment.

2. **Cannon–Bard Theory**

Cannon–Bard theory [27] proposes that physiological states, behaviours and feelings are not elicited one after the other (as proposed by common sense and James–Lange theory). Instead, they are all elicited independently at the same time. According to this theory, we are sad and cry concurrently, happy and laugh concurrently and angry and fight concurrently. This theory is formulated based on the following two discoveries [47]:

(a) Physiological states are too slow to elicit a feeling.

(b) An artificial induction of physiological states does not evoke a feeling.

This theory suggests that emotions cannot be distinguished solely based on physiological states. Individuals can have *different* physiological states and responses for the same feeling, e.g. we may or may not cry when we are sad. On the other hand, individuals can also have the *same* physiological state and response for different feelings, e.g. our crying does not necessarily mean that we are sad since we may also cry for a happy situation.

3. **Cognitive Appraisal Theory**

Cognitive Appraisal theory proposes that physiological states, behaviours and feelings are elicited based on an individual's subjective interpretation or evaluation of a situation in terms of the relevance and impact on personal well-being [197, 131, 21]. A feeling cannot be elicited without a reason or justification. According to this theory, we are sad because we lose money, we are happy because we receive money and we are angry because we do not received the money that has been promised to us.

This theory can explain the reason why the same situation can be evaluated differently by individuals. A person whose confession of love is rejected may be sad or cry because the person interprets the situation as he/she will not get a potential partner (i.e. an impact on his/her well-being). In contrast, a different person in the same situation may *not* be sad or cry because the person interprets the same situation as they were not a suitable couple, hence it is better to get rejected earlier than later in the relationship.

4. **Schachter–Singer Theory / Two-factor Theory of Emotion**

Schachter–Singer theory (also known as the *two-factor theory of emotion*) [196] proposes that both physiological states and cognitive appraisals can be elicited independently, however the two must occur together in order to elicit a feeling. A physiological state alone is not sufficient to elicit a feeling (as similarly proposed by the Cannon–Bard theory). In addition, if people have an explanation for their physiological states, then they will not evaluate the situation and they will label their feelings based on that explanation. Otherwise, they will explore possible explanations based on the situation in order to label their physiological states and describe their feelings.

According to this theory, our heart rate can increase or decrease through an

artificial induction (i.e. our physiological state is elicited), but we should not feel sad, happy or angry without any reason. In contrast, if we understand the reasons for our crying, laughing and fighting, we should not feel sad, happy or angry unless accompanied by changes in our heart rate. In addition, we do not need to find a reason for our crying, laughing and fighting if we have an explanation, but we will search for a possible explanation to identify them if we do not have the explanation.

6.2.3 Evaluation: Innate or Learned

Theories of emotional evaluation explore the elicitation of emotion based on how a situation is evaluated. They are divided into two perspectives [185, p. 304], namely biological perspective and cognitive perspective. Major representatives of the theories for the biological perspective and cognitive perspective are basic emotion theories and Cognitive Appraisal theory respectively. In Section 6.2.1, I have described basic emotion theories in terms of representation of emotions. In Section 6.2.2, I have described Cognitive Appraisal theory in terms of causes of emotions. In the following, I describe basic emotion theories and Cognitive Appraisal theory in terms of the elicitation of emotions:

1. **Biological Perspective: Basic Emotion Theory**

   Basic emotion theories propose that emotion is inherited and innate [171, 66]. Feelings are elicited from an intuition or instinct without an explicit reasoning or rational deliberation. According to this theory, people can have an emotion about a situation regardless of its relevance or impact on them. They may not have a reason or may believe that they do not need a reason to justify their emotions. This theory is supported by the studies that show that infants respond emotionally to taste (i.e. prefer sweet taste) [64] and to voices (i.e. smile to high-pitched human voices) [244] despite their lack of developed cognitive functions (i.e. have not fully developed yet). Major theories in the biological perspective of emotion are described further in Reeve's work [185, p. 309].

   Actions as a result of an intuition are similar to reflexes because both are innate. However, these actions elicit emotions whereas reflexes do not. Reflexes have been described in Section 5.2.3.

2. **Cognitive Perspective: Cognitive Appraisal Theory**

   Cognitive Appraisal theory supports the view that emotion is learned. Feelings are elicited from an individual's evaluation of a situation with explicit reasoning or rational deliberation, rather than from the situation itself [130]. According to this theory, people cannot have an emotion without a reason. For example, a decrease in a currency value is not likely to cause a person to be sad without a reason, unless the person has a foreign currency investment and thinks that his/her well-being is affected by the money lost in the investment. In addition, the outcome of a situation can be the same, but the appraisal of the situation causes different emotions to be elicited [238]. For example, a feeling of sadness is produced when one believes that losing money was self-caused, and a feeling of anger is produced when one believes that someone else has caused that loss (e.g. a funds investment manager).

   Several theories of cognitive appraisal have proposed criteria that can be used to evaluate situations in order to determine the kind of emotion one should experience, for example, the OCC theory [171] and Scherer's theory [197]. Major theories in the cognitive perspective of emotion are described further in Reeve's work [185, p. 311].

6.3 Computational Design and Model

Currently, there is no widely accepted theory of emotional representation, emotional causality and emotional evaluation. Instead of designing ASMO's emotion mechanism based on a specific theory, I look at the previously discussed theories of emotion from a computational perspective and accommodate all of the theories to design ASMO's emotion mechanism. From a computational perspective, the theories can be realised into the same mechanism even though some of the theories may have different or opposing views from a psychological perspective.

From my perspective, emotion is a noumenon that describes the following activities that occur inside a person: feeling, behaviour, physiological state and appraisal activities. A system or an agent that has these activities and can convince others about its emotion should be considered to have an 'emotion', just like we are convinced by other people that they have an emotion.

In the following, I describe how each theory in the previous section inspires the

design of ASMO's emotion mechanism (see Figure 6.2).

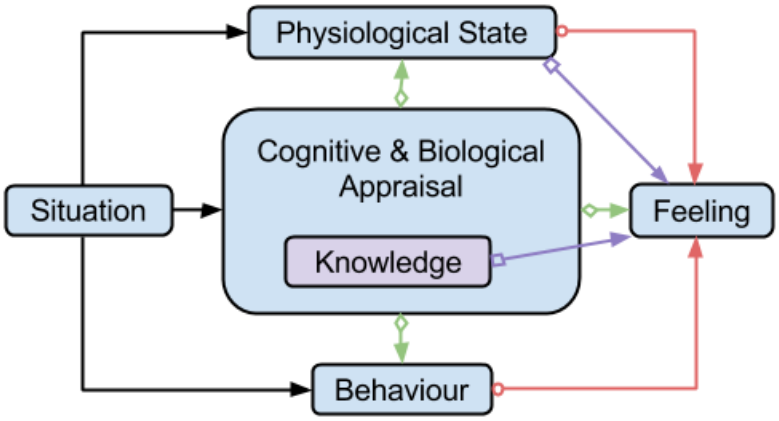


Figure 6.2: Design of ASMO's Emotion Mechanism

1. **Basic Emotion Theory**

   Inspired by the basic emotion theories, ASMO's emotion mechanism represents emotions in terms of 'basic' feelings.

2. **Dimensional Theory of Emotion**

   Inspired by the dimensional theory of emotion, 'basic' feelings in ASMO's emotion mechanisms are further represented using one or more bivariate dimensions.

   A basic emotion theory and a dimensional theory of emotion may seem to be opposite to each other, which may suggest that a mechanism cannot be designed or modelled based on the inspiration from both theories. However, I argue that both theories are not opposite. Instead, they just represent emotions from different perspectives, just like a colour can be represented by a combination of three primary colours (i.e. red, green and blue) and by the degrees of hue, saturation and intensity dimensions.

   From a computational perspective, a dimensional representation has a finer control than a basic emotional representation. It can represent a basic emotion, but not vice versa. Its value can be set to high (e.g. 1.0) or low (e.g. 0.0) to indicate whether it represents the basic emotion or not respectively. For example, the contempt emotion that is composed of the anger and disgust basic emotions [178] can be represented by using two dimensions, called *anger* and *disgust* respectively, with a value of 1.0 each.

In addition, from a computational perspective, it is more compelling to use bivariate dimensions over bipolar dimensions, because emotions represented using bipolar dimensions can also be represented using bivariate dimensions, but not vice versa.

3. **Biological Perspective**

   Inspired by the biological perspective of emotion, ASMO's emotion mechanism elicits feelings based on an intuition or instinct mechanism, which is modelled by evaluating a situation against built-in knowledge (called biologically-based evaluation or biological appraisal). Developers can simply provide this built-in knowledge without the necessary to have evidence, rational thought, inference or observation.

4. **Cognitive Perspective**

   Inspired by the cognitive perspective of emotion (and in addition to the biological perspective of emotion), ASMO's emotion mechanism elicits feelings by evaluating a situation against learned knowledge (called cognitively-based evaluation or cognitive appraisal). ASMO's emotion mechanism uses this learned knowledge to explicitly reason about the feelings.

5. **Cognitive Appraisal Theory**

   Inspired by Cognitive Appraisal theory, the knowledge used in a cognitive appraisal is learned based on the relevance and impact to personal well-being. ASMO's emotion mechanism elicits physiological states, behaviours and feelings based on a cognitive appraisal against this kind of knowledge (see the diamond-tail green arrows in Figure 6.2). This theory is the same as the cognitive perspective theory of emotion.

6. **Cannon–Bard Theory**

   Inspired by Cannon–Bard theory, ASMO's emotion mechanism elicits feelings (through appraisals) concurrently and independently with physiological states and behaviours (see the diamond-tail green arrows in Figure 6.2 which are similar to Cognitive Appraisal theory). Behaviours and physiological states can be evoked automatically as responses to a situation in the environment (see the black arrows in Figure 6.2).

   In ASMO, physiological states, behaviours and feelings can be elicited concurrently, because modules can run concurrently and simultaneously. In addition,

cognitive appraisals are assumed to be very fast. Thus, even though feelings are elicited through and after appraisals, they appears to be elicited concurrently with physiological states and behaviours.

7. **James–Lange Theory**

James–Lange theory is seen to be contradicted with Cannon–Bard theory. It suggests that a feeling is elicited after or based on physiological states and behaviours. Meanwhile, Cannon–Bard theory suggests that a feeling is elicited independently and concurrently with physiological states and behaviours as they are too slow to elicit a feeling.

From a computational perspective, I interpreted these two theories as follows: a feeling is elicited independently and concurrently with physiological states and behaviours, however the awareness (or the recognition) of the feeling often occurs after physiological states and behaviours have occurred. The feeling can be recognised better when information or knowledge about physiological states and behaviours are available (see the circle-tail red arrows in Figure 6.2).

Inspired by this interpretation, ASMO's emotion mechanism elicits feelings independently and concurrently, and uses knowledge that drives physiological states and behaviours to recognise the feelings. In ASMO, this knowledge is incorporated as factors in biological and cognitive appraisals.

8. **Schachter–Singer Theory / Two-factor Theory of Emotion**

Inspired by Schachter–Singer theory, ASMO's emotion mechanism does not appraise a situation to determine physiological states if an explanation or knowledge about the physiological states is available in ASMO's memory. Instead, it should use the knowledge directly to determine the physiological states (see the purple knowledge box in Figure 6.2).

In addition, ASMO's emotion mechanism elicits feelings based on both cognitive appraisals and knowledge about physiological states (see the square-tail purple arrows in Figure 6.2) (as also inspired by Cognitive Appraisal theory and James–Lange theory).

6.4 Implementation

Previous section describes the design of modules and ASMO's emotion mechanism based on theories of emotion. In ASMO, emotions are represented in terms of 'basic feelings', which are further represented using bivariate dimensions. Modules are designed to run concurrently and independently to elicit feelings, behaviours and physiological states (this concurrent and independent design has been suggested earlier in the design described in Section 5.3). ASMO's emotion mechanism is designed to elicit feelings based on biological appraisals and cognitive appraisals. It uses knowledge to help recognise the feelings and to determine physiological states. Biological and cognitive appraisals are modelled by evaluating a situation against built-in and learned knowledge respectively. This knowledge is learned based on the relevance and impact to personal well-being.

The purpose of ASMO's emotion mechanism [166] is to bias the selection of modules subjectively based on elicited emotions or feelings. This bias is achieved and expressed by modifying the attention values of non-reflex modules via their subjective weights. This modification will change the modules' rankings, and hence cause the selection of modules to be biased.

ASMO's emotion mechanism uses *causal Bayesian networks* [175] to predict the emotions that will be elicited given a situation and uses this prediction to determine subjective weights. A causal Bayesian network is a directed acyclic graph whose nodes denote probabilistic cause–effect relationships of random variables. It is a special case of Bayesian network where parent nodes are required to be the causes of their children nodes. For example, in a causal Bayesian network, if chasing the ball causes the robot to get the ball, then a node that denotes chasing the ball action can be the parent of a node that denotes the result of getting the ball, but not vice versa.

Nodes in the causal Bayesian networks used by ASMO's emotion mechanism are divided into four types according to the dimensional representation, basic feeling representations, biological appraisal and cognitive appraisal described earlier. These four types are dimension, label/feeling, biological factor and cognitive factor nodes respectively (see Figure 6.3). They are created with the following constraints:

1. **Connection based on Node Types**

   Nodes have to be connected according to their types. For example, a biological

factor node cannot be connected directly to a dimension node. Instead, it has to be connected to a feeling node before the feeling node can be connected to the dimension node.

2. **Acyclic Connection**

   Connections of nodes cannot be cyclic. The descendent of a node cannot be connected back to the node itself.

3. **Non-negative Probability**

   The probability associated with a node cannot be negative. Instead, it has to be a positive number ranging from 0.0 to 1.0 (equivalent to a range from 0.0% to 100.0%).

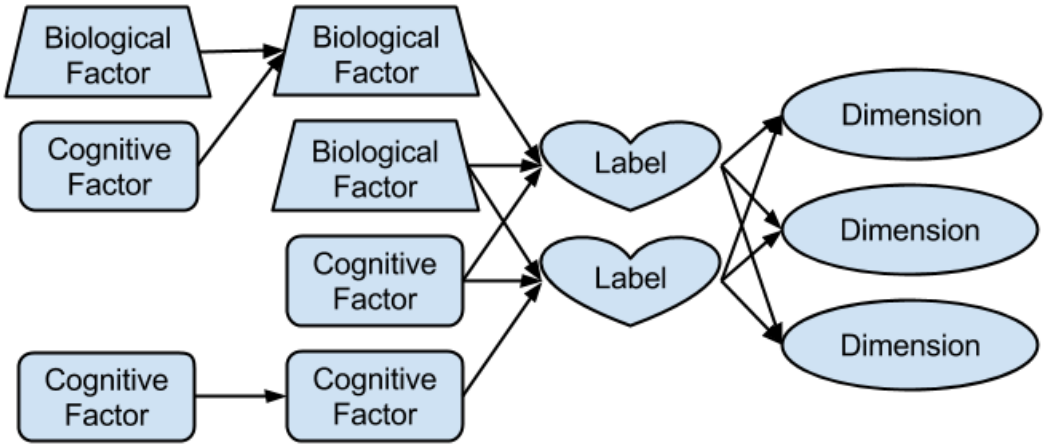


Figure 6.3: Types of Nodes Used by ASMO's Emotion Mechanism

In the following subsections, I will describe the four types of nodes in ASMO's emotion mechanism and how the nodes are used to determine subjective weights.

6.4.1 Dimension and Label Nodes

As described in Section 6.3, emotions are represented in terms of 'basic' feelings that are further represented by bivariate dimensions. These bivariate dimensions and basic feelings are implemented as dimension nodes and label nodes in a causal Bayesian network respectively.

Dimension nodes denote the bivariate dimensions that capture the degrees of *emotional qualities*, such as positive valence, negative valence and arousal dimensions

that capture the degrees of pleasantness, unpleasantness and excitement respectively. Any number of dimensions may be chosen by developers according to their preferred specific emotion theory. Dimension nodes are created at the design stage. They are the lowest child nodes (i.e. they are not connected to other nodes). Modules can calculate the conditional joint probability of a dimension node in order to determine how likely the corresponding dimension represents the emotion experienced in a situation.

Note that dimension nodes represent bivariate dimensions instead of bipolar dimensions. Bivariate dimensions allow the degrees of emotional qualities to be captured independently. It allows two opposite emotional qualities (such as pleasantness and unpleasantness) to occur or to be experienced at the same time. In bivariate dimensions, a situation is not necessarily unpleasant when pleasantness has not occurred or been experienced. For example, a situation that has a probability of 0.7 to cause pleasantness to occur, implies that it has the probability of 0.3 to cause pleasantness to *not* occur, but *does not* imply that it has the probability of 0.3 to cause unpleasantness to occur.

Label nodes denote the feelings experienced in a situation, such as happy and sad. They can be created at the design stage or in real-time through learning. They are connected as parents to the dimension nodes. Modules can calculate the conditional joint probability of a feeling node in order to determine how likely the corresponding feeling is elicited in a situation, or how likely the corresponding basic feeling represents the emotion experienced in a situation.

ASMO's emotion mechanism can have a causal Bayesian network with any number of dimension nodes and label nodes. It does not limit the kind of dimension and feeling being denoted. However, in many robotic tasks, developers typically represent emotions, by using the three following dimensions: positive valence, negative valence and arousal. In addition, they typically consider the two following feelings in their systems: happy and sad.

6.4.2 Biological Factor and Cognitive Factor Nodes

Biological appraisals and cognitive appraisals are implemented through biological factor and cognitive factor nodes in a causal Bayesian network respectively. *Biological factor nodes* denote built-in knowledge used for evaluating a situation, such as

a sweet taste elicits a happy feeling or a red colour from a robot soccer ball elicits an excitement feeling. They are created at the design stage. They are connected as parents to the label nodes and/or other biological factor nodes. Modules can calculate the conditional joint probability of a biological node in order to determine how likely the emotion experienced in a situation is caused by the corresponding biological factor.

By convention, biological factor nodes are fixed across a robot's life span (i.e. the number of biological factor nodes is fixed). They can be used to embed permanent or innate characteristics and personalities desired in the robots, such as a characteristic to avoid harmful actions or a personality to be risk-averse.

In contrast, *cognitive factor nodes* denote learned knowledge used for evaluating a situation based on the relevance and impact to personal well-being, such as the belief that being praised indicates a stronger social acceptance. They can be created at the design stage or in real-time through learning. They are connected as parents to the label nodes, biological factor nodes and/or other cognitive factor nodes. Modules can calculate the conditional joint probability of a cognitive node in order to determine how likely the emotion experienced in a situation is caused by the corresponding cognitive factor.

Cognitive factor nodes can be varied across a robot's life span (i.e. the number of cognitive factor nodes is not fixed). The number of cognitive factor nodes is changed based on the robot's experience. Cognitive factor nodes can be used to build personalities learned during the life span of the robot, such as a personality to be happy for being praised (i.e. the robot learns to smile when praised).

Both biological factor and cognitive factor nodes affect the computational cost of the emotion mechanism. The higher the number of nodes, the higher the computational cost. Performing exact inference on these nodes in order to evaluate the situation can be computationally expensive. ASMO's emotion mechanism uses an approximate inference method such as the Monte Carlo Algorithm [192] in order to maintain fast evaluation.

6.4.3 Subjective Weight

In the attention value update equation (i.e. Equation 5.1), a subjective weight denotes the change of attention sought by the module based on subjective evalua-

tions of tasks. This subjective weight is not only modified by the module based on tasks, but it is also modified by ASMO's emotion mechanism as a mean to bias the module. In order to modify this subjective weight, ASMO's emotion mechanism is implemented in a supervisor module (called *emotion_mechanism*). This is because a supervisor module is allowed to inspect and modify other non-reflex modules, whereas an ordinary module is not allowed.

A subjective weight (S_t), an objective weight (O_t), and a boost value (BV_t) are used to determine the total attention level of a non-reflex module (see Equation 5.4.4, which is also shown below). However, they have different purposes. An objective weight is the weight of an attention value calculated based on objective evaluations. A subjective weight is the weight of an attention value calculated based on subjective evaluations. A change in objective weight and subjective weight (i.e. their effects) will be accumulated in an attention value in proportion to the accumulation rate (α_t). In contrast, a boost value is similar to an attention value, except that its change or effect will not be accumulated. A boost value can be equivalent to an objective and a subjective weight when the accumulation rate is zero.

$$
\begin{aligned}
TAL(p) &= BV_t(p) + AV_t(p) \\
AV_t(p) &= \underbrace{\alpha_t(p)\ AV_{t-1}(p)}_{\text{accumulation}} + \underbrace{\beta_t(p)\ O_t(p)}_{\text{objective}} + \underbrace{\gamma_t(p)\ S_t(p)}_{\text{subjective}}
\end{aligned}
$$

ASMO's emotion mechanism can use multiple causal Bayesian networks to determine a subjective weight. Nodes can be created and distributed in different causal Bayesian networks. As described in the previous subsections, the feeling nodes and cognitive factor nodes can be created in real-time through learning. However, this type of learning remains as a future work. In this dissertation, nodes in causal Bayesian networks are created at the design stage.

ASMO's emotion mechanism uses casual Bayesian networks to determine a subjective weight based on an algorithm described in the following two steps:

1. **Predict Feeling**

   ASMO's emotion mechanism predicts the feelings that will be elicited given a situation. It predicts these feelings by calculating the conditional joint probability of each feeling node using Bayes' rule (see Equation 6.1). The higher the

conditional joint probability, the more likely that the feeling will be elicited.

$$P(h|e) = \frac{P(e|h) \times P(h)}{P(e)} \tag{6.1}$$

Where:
$P(h|e)$ is the posterior probability. It is the probability of hypothesis h given or after evidence e is observed
$P(e|h)$ is the likelihood. It indicates how likely evidence e is associated with hypothesis h
$P(h)$ is the prior probability. It is the probability of hypothesis h before any evidence is observed
$P(e)$ is the marginal likelihood. It indicates how likely evidence e is observed regardless of the hypothesis

2. **Calculate Expected Desirability of Feelings**

   ASMO's emotion mechanism calculates the expected desirability of feelings based on the predicted feelings. It assigns this expected desirability of feelings as the subjective weight of an ordinary and a supervisor module (see Equation 6.2).

$$S_t = \frac{1}{N_a N_f} \sum_{i=0}^{N_a} \sum_{j=0}^{N_f} P(f_j|a_i, e) D(f_j) \tag{6.2}$$

Where:
$N_f \leq N_{fn}$
S_t is the subjective weight of a module at time t
N_a is the number of actions proposed by the module
N_f is the number of feelings that developers want the system to experience
N_{fn} is the total number of feeling nodes
a_i is the i-th action proposed by the module
e is the set of nodes that represents the situation (i.e. $e \leftarrow e_1, e_2 ... e_n$)
$P(f|a, e)$ is the conditional joint probability of feeling f given action a and a set of nodes e that represents the situation
$D(f)$ is the desirability of experiencing feeling f

For example, a robot may be desired to have emotions in a soccer scenario. ASMO's emotion mechanism can represent and have a model of possible feelings

that may occur in the scenario, such as happy, sad, anger and surprise. However, developers may consider only happy and sad feelings to be relevant in playing soccer. Thus, they may want the robot to have only these two feelings (i.e. $N_f = 2$), rather than all the possible feelings. Suppose that a module called 'chase_ball' proposes an action to run to a ball and an action to track the ball at the same time (i.e. $N_a = 2$). A subjective weight of this module will then be determined based on the probability of each feeling (i.e. happy and sad) will be elicited given each action (i.e. walk and track).

By default, ASMO's emotion mechanism determines a subjective weight based on the expected desirability of feelings. However, developers can change this default setting in order to determine a subjective weight based on the expected desirability of dimensions. When they use the expected desirability of dimensions, they have to change Equation 6.2 into Equation 6.3.

$$S_t = \frac{1}{N_a N_d} \sum_{i=0}^{N_a} \sum_{j=0}^{N_d} P(d_j|a_i, e) D(d_j) \tag{6.3}$$

Where:
$N_d \leq N_{dn}$
S_t is the subjective weight of a module at time t
N_a is the number of actions proposed by the module
N_d is the number of dimensions that developers want the system to experience
N_{dn} is the total number of dimension nodes
a_i is the i-th action proposed by the module
e is the set of nodes that represents the situation (i.e. $e \leftarrow e_1, e_2 ... e_n$)
$P(d|a, e)$ is the conditional joint probability of dimension d given action a and a set of nodes e that represents the situation
$D(d)$ is the desirability of dimension d to occur

Using the expected desirability of dimensions, modules are biased based on the groups the feelings belong to, instead of the particular feelings. For example, modules can be biased as long as their actions result in positive valence, regardless whether the feelings experienced in the situation are happy, surprise and energetic.

6.5 Other Work

Other emotion mechanisms or models have been proposed in the literature. Conati [54, 55] has proposed a similar probabilistic approach using Dynamic Decision Networks (DDN) in order to recognise students' emotions based on the OCC cognitive appraisal theory [170]. Like ASMO's emotion mechanism, this model contains nodes to represent situations, such as a student's goal, an interaction pattern, an appraisal and an agent's action. However, unlike ASMO's emotion mechanism, this model does not incorporate the non-cognitive perspective (i.e. the biological perspective) of emotions and how emotions bias attention to influence decision making and behaviours.

Considering the non-cognitive perspective of emotions, Rosis et al [188] proposed a Belief-Desire-Intention (BDI) emotion model that distinguishes cognitive evaluations with intuitive appraisals. In this model, a cognitive evaluation is a rational judgement supported by reasons, whereas an intuitive appraisal is a non-rational judgement based on associative learning and memory instead of justifiable reasons. These cognitive evaluation and intuitive appraisals in this model are similar to cognitive appraisals and biological appraisals in ASMO's emotion mechanism respectively. However, this model does not incorporate intuitive appraisals based on innate knowledge (such as the preference for sweet taste that is probably encoded in our DNA) and how emotions bias attention to influence decision making and behaviours.

A number of emotion models have also been implemented in robots [113, 28, 110, 112, 195, 38]. Many of them focus on emotion expressions (e.g. facial expression, body language, etc), while some focus on influencing decision making and behaviours. One common example of those models designed to influence decision making and behaviours is the emotion model proposed by Sawada, Takagi and Fujita [195]. This model generates 6 emotions (proposed by Ekman's basic emotion theory [66]) and a neutral emotion in a robot based on the following internal needs: hunger, fullness, pain, comfort, fatigue and sleepiness. Like ASMO's emotion mechanism, this model is used to influence behaviours instead of for emotional expression or recognition. Unlike ASMO's emotion mechanism, this model only incorporates the biological perspective of emotions and does not incorporate the cognitive perspective of emotions.

Another common example is the emotion model proposed by Breazeal [38]. This

model generates emotions in a robot based on mechanisms called emotive releasers, affective appraisals, emotion elicitors and emotion activation. Each emotive releaser can be thought of as a simple 'cognitive' evaluation with respect to the robot's well-being and its goals. It has an activation level that is determined by evaluating attributes in the environment, such as the presence or absence of a stimulus (and for how long), the context of the stimulus (e.g. toy-related or person-related), or whether the stimulus is desired or not. Each releaser with activation above threshold will be tagged by the affective appraisals based on three dimensions in order to create a somatic marker, namely arousal, valence and stance. These markers are mapped to some distinct emotions (e.g. anger, fear, etc) by filtering them through the emotion elicitors. Finally, these distinct emotions compete for activation and they are selected by the emotion activation based on the winner-take-all scheme. The emotive releasers, affective appraisals and emotion elicitors in this model are similar to the cognitive factor nodes, dimension nodes and label nodes in the causal Bayesian network used by ASMO's emotion mechanism respectively. The difference is that this model maps the three dimension markers into the distinct emotions (i.e. labelled emotions), whereas ASMO's emotion mechanism maps reversely, which maps the labelled emotions (i.e. label nodes) into dimensional representation (i.e. dimension nodes). In addition, this model does not incorporate the biological perspective of emotions.

A summary of emotion models that integrate causes and effects has been provided in the literature [55]. In addition, a recent comparison of emotion models in autonomous agents has been presented [190]. However, I am not aware of any emotion model that integrates innate emotion (biological appraisal), learned emotion (cognitive appraisal), attention and decision making.

Chapter 7

ASMO's Learning Mechanisms

In the previous two chapters, I described how ASMO's attention mechanism selects modules and how ASMO's emotion mechanism subjectively biases the selection. In this chapter, I describe the details of ASMO's learning mechanisms and how the selection of modules is modified in order to improve performance. I start by defining learning in Section 7.1. I follow by exploring theories of learning for improving performance in Section 7.2. I describe the interpretation of these theories in Section 7.3. These theories form the basis of the design of ASMO's learning mechanisms. I then describe the implementation of ASMO's learning mechanisms in Section 7.4. Finally, I review other work on learning mechanisms and compared them with ASMO's learning mechanisms in Section 7.5.

7.1 Needs for Learning

Learning is defined in Oxford English Dictionary [172] as "the action of receiving instruction or acquiring knowledge" or "a process which leads to the modification of behaviour or the acquisition of new abilities or responses, and which is additional to natural development by growth or maturation". In other words, learning involves knowledge acquisition, behaviour modification or both of them. A learning that involves knowledge or skill acquisition does not necessarily involve behaviour modification and vice versa.

Tom Mitchell formally defines learning [146] as follows: "a computer program is said to learn from experience E with respect to some class of tasks T and perfor-

mance measure P, if its performance at tasks in T, as measured by P, improves with experience E".

Learning is necessary not only because it can improve the versatility, robustness and extensibility criteria required by a cognitive architecture including ASMO (see Section 2.5), but also for the following reasons:

- To handle situations that change unexpectedly or were anticipated incorrectly by developers.
- To automate the development of behaviours. Anticipating all possible situations for developing behaviours can be difficult or not possible due to the enormous number of possibilities of situations. In addition, developing behaviours by hand can also be difficult.
- To find a better decision or action that can increase performance. Some information or parameters that lead to a better performance may not be available during design, but can be acquired when interacting with the environment.

7.2 Theories of Learning

ASMO's learning mechanisms are inspired by human learning. There are two main theoretical perspectives of human learning, namely behaviourist and cognitivist [169]. The behaviourist perspective focuses on low-level learning involving conditions (stimuli) and behaviours (responses), whereas the cognitivist perspective focuses on high-level cognitive processes. In this dissertation, I have approached the low-level forms of learning first (i.e. the behaviourist perspective), which consists of habituation, sensitisation, classical conditioning, operant conditioning and observational learning. The high-level cognitive processes are developed based on lower-level learning but this remains as future work.

7.2.1 Habituation and Sensitisation

Habituation is the decrease in response to a repeated stimulus as a result of a lack of significance or reinforcement [180, p. 100] [65]. It involves the elimination of unnecessary behaviours that are not needed by the individual. The stimulus will

not be attended to or responded to (i.e. given less attention) if it does not hold any significance for the individual even though it may be repetitive. For example, when tracking an object, an individual who initially pays attention to (and is distracted by) repeated movements of other people, can become habituated to (thus ignore) these movements if these movements do not hold any significance to the object tracking task.

Habituation is not a form of sensory adaptation (e.g. fatigue). While both habituation and sensory adaptation involve the decrease in response to a repeated stimulus, habituation occurs in the brain as attention phenomenon and it can be consciously controlled, whereas sensory adaptation occurs directly in the sense organ and it cannot be consciously controlled [212, p. 138]. For example, in habituation, people can still pay attention and force themselves to hear a loud noise although they are habituated to the noise. In sensory adaptation, people cannot force themselves to smell an odour once their smell organ has adapted to the smell, regardless of how hard they try to pay attention to the odour. The characteristics of habituation that can distinguish habituation from sensory adaptation are further discussed in the work of Thompson and Spencer [221] and Rankin et al. [184].

In contrast to habituation, sensitisation is the increase in response to a repeated stimulus because the stimulus holds a significance for the individual [180, p. 101] [65]. The stimulus will be attended to or responded to (i.e. given more attention) as long as it holds some significance for the individual, even if it is not intense. For example, when tracking an object, an individual who initially ignores and does not pay attention to the movements of other people can become sensitised to (thus focus on) these movements if these movements hold any significance to the object tracking task (e.g. when the object is often occluded by the movements).

Habituation and sensitisation are automatic processes that are developed as a result of an adaptation to the environment. They are a form of *non-associative learning*, which is a learning that does not involve an association with other stimuli or responses. They occur without feedback from the environment. Individuals do not depend on explicit feedback from the environment to modify their responses. Instead, they internally judge the effectiveness of their responses in order to modify their behaviours.

7.2.2 Operant Conditioning

Operant conditioning (or instrumental conditioning) is a modification of behaviours by their consequences or feedback [180, p. 228] [207]. This feedback regulates the probability of exhibiting the behaviours. There are three possible outcomes of the feedback received from the environment:

1. **Reinforcement**

   Reinforcement is received to increase or strengthen the tendency of exhibiting the behaviour (i.e. to encourage the behaviour).

2. **Punishment**

   Punishment is received to decrease or weaken the tendency of exhibiting the behaviour (i.e. to discourage the behaviour).

3. **No Consequences or Feedback**

   No feedback is received either to increase or decrease the tendency of exhibiting the behaviour (i.e. neither to encourage nor discourage the behaviour).

Both reinforcement and punishment can be either positive or negative depending on whether a stimulation is added or removed respectively [50, p. 133–134]:

1. **Positive Reinforcement**

   Positive reinforcement is the addition of stimulation to encourage the behaviour. For example, a player is given bonus money for every kick that scores a goal.

2. **Negative Reinforcement**

   Negative reinforcement is the removal of stimulation to encourage the behaviour. For example, a player is not required to attend to the extensive kicking training (which the player does not like) if the player performs a good kick.

3. **Positive Punishment**

   Positive punishment is the addition of stimulation to discourage the behaviour. For example, a player is required to do push-ups for every wrong kick the player performs.

4. **Negative Punishment**

   Negative punishment is the removal of stimulation to discourage the behaviour. For example, a player is not given a rest time if the player performs a wrong kick.

Operant conditioning is a form of *associative learning*, which is a learning that involves an association between stimuli or responses. It occurs with feedback from the environment. Individuals do not need to manually judge the effectiveness of their responses to modify their behaviours. Instead, they receive explicit feedback for their responses after they perform their behaviours, which they can use to correct those behaviours.

In artificial intelligence and machine learning, the area of *reinforcement learning* has been inspired by operant conditioning. Similar to operant conditioning, reinforcement learning uses reinforcement to encourage the behaviour and punishment to discourage the behaviour. However, unlike operant conditioning, there is no distinction whether the reinforcement and punishment are positive or negative (i.e. whether the stimulation is added or removed respectively).

In reinforcement learning, the reinforcement is often represented by a positive value (also called positive reward) whereas the punishment is often represented by a negative value (also called negative reward). Note that the terms *positive* and *negative* here represent the sign of the values instead of the type of reinforcement or punishment. Thus, the terms *positive* and *negative* in reinforcement learning may refer to the reinforcement and punishment respectively, whereas the terms *positive* and *negative* in operant conditioning refer to the addition and the removal of stimulation respectively.

7.2.3 Classical Conditioning

Classical conditioning (or Pavlovian conditioning) [180, p.109–110] [174] is an association of a neutral stimulus that does not elicit a response (called the *conditioned stimulus* or CS) with another stimulus that elicits a response (called the *unconditioned stimulus* or US), such that the presence of the CS will elicit the same response that would be elicited by the US, despite the US not actually being present. For example, if John repeatedly asks Mary to cook him the same dish every time he visits Mary, then Mary may develop the association between his visit and the dish,

such that his presence will trigger Mary to *accidentally* start cooking the dish, even though John had asked Mary to go to a restaurant.

In classical conditioning, the CS initially does not elicit any particular response whereas the US elicits a response (called the *unconditioned response* or UR). However, after the association is made, the presence of CS will elicit the UR even though the US is not actually present. In other words, the CS will elicit the same response as if the US was present. This response elicited by the CS is called the *conditioned response* or CR, and is the same as the UR.

Similar to operant conditioning, classical conditioning is a form of associative learning. However, classical conditioning creates an association between *involuntary* behaviours and stimulus before the behaviours are performed, whereas operant conditioning creates an association between *voluntary* behaviours and their consequences after the behaviours are performed [49, p. 132].

7.2.4 Observational Learning

Observational learning (or vicarious learning) [180, p. 563] [50, p. 281] is the reproduction of results or behaviours of others (called the teachers or experts) through observation, demonstration, examples, teaching or supervision. Less formally, this learning may also be called learning with a teacher, learning from the experts, learning by observation or learning by demonstration. For example, a player can learn how to move its legs to kick a ball by observing other players' behaviours, even though he has not experienced the kick for himself.

Observational learning is different to imitation. Paul Chance [49, p. 281] argued that observational learning may occur even if imitation has failed. Some parts of the behaviours may have been learned even if they cannot be imitated perfectly. He also argued that observational learning may not occur even if imitation is successful. Behaviours are not learned if there is no permanent modification in the internal mechanism even if they can be imitated perfectly.

Observational learning is a form of social learning [26] as it involves more than one individual. It can produce different results and behaviours depending on the teachers or the demonstrators.

In artificial intelligence and machine learning, supervised learning [192, p. 695]

is typically used as a mechanism for observational learning. It is often referred to as learning from labelled data or inductive machine learning [119]. In supervised learning, the labelled data is assumed to be correct. The objective of the supervised learning is to model the labelled data as precisely and with as much generality as possible.

7.3 Computational Design and Model

Each theory of learning described in the previous section is translated and modelled into a learning mechanism in ASMO, as described as follows:

1. **Theory of Habituation and Sensitisation**

   Inspired by theories of habituation learning, ASMO's habituation mechanism is created to decrease the boost value of a non-reflex module based on the significance of a repetitive stimulus. Decreasing the boost value will eventually make the module lose the attention competition and thus the stimulus will not be responded to. Similarly, inspired by theories of sensitisation learning, ASMO's sensitisation mechanism is created to increase the boost value of a non-reflex module based on the significance of a repetitive stimulus. Increasing the boost value will eventually make the module win the attention competition and thus the stimulus will be responded to.

2. **Theory of Operant Conditioning**

   Inspired by a theory of operant conditioning learning, ASMO's operant conditioning mechanism is created to increase and decrease the boost value of a non-reflex module based on positive and negative feedback respectively. Increasing the boost value of a module will increase the module's chance of winning the attention competition, thus it will also increase the probability of the module's actions being executed. Similarly, decreasing the boost value of a module will decrease the module's chance of winning the attention competition, thus it will also decrease the probability of the module's actions being executed.

3. **Theory of Classical Conditioning**

   Inspired by a theory of classical conditioning, ASMO's classical conditioning mechanism is created to trigger non-reflex modules to propose actions based

on the presence of the *conditional* stimulus even though the *unconditional* stimulus is not actually present.

4. **Theory of Observational Learning**

   Inspired by a theory of observational learning, ASMO's observational learning mechanism is created to modify the boost value of a non-reflex module based on labelled examples provided by (or observed from) the experts. Modifying the boost values of modules will change the winners of the attention competition, thus it will also change the actions performed by ASMO according to the experts' examples.

7.4 Implementation

ASMO's learning mechanisms uses multiple algorithms. Wolpert [245] has shown that there is no best algorithm in learning. Each algorithm has strengths and weaknesses that solve some problems better than other problems. Thus, using multiple algorithms provides ASMO with the capability to solve a wider range of problems more effectively.

Each learning mechanism is implemented in a *supervisor* module. It can be enabled or disabled by simply enabling and disabling the corresponding module respectively. Recall from Section 5.4.1 that a supervisor module is a module that has a meta-level control of the system and that can modify the attention values and/or boost values of other non-reflex modules (i.e. ordinary and supervisor modules). Recall also that a supervisor module can modify these values in two ways, namely competitive modification and direct modification.

All ASMO's learning mechanisms (i.e. supervisor modules) affect the boost values of other non-reflex modules. If the supervisor modules are designed in a competitive modification, then only one supervisor module (i.e. one learning mechanism) can win an attention competition to modify the boost values. If the supervisor modules are designed in a direct modification, then they all can modify the boost values and their net results will determine the final boost values. The higher the change of boost value given by a supervisor module, the more dominance the learning mechanism.

These competitive and direct modifications serve as a mechanism to prevent

supervisor modules having contradictory effects, which may cause them to cancel each other (i.e. causing learning to not occur). This prevention mechanism will not work for supervisor modules that have equal strength in direct modification. However, having contradictory effects is fine just like what happens in humans (i.e. inspired by biological agents). For example, in playing a soccer, we can habituate to the opponent's attack and become less fear to the attack, but at the same time, we can also be reinforced to feel fear of the attack. In this case, our habituation learning and reinforcement learning (i.e. operant conditioning learning) will have contradictory effects, which may cancel each other.

7.4.1 Habituation and Sensitisation

ASMO's habituation and sensitisation mechanisms [164] decrease and increase the boost values of non-reflex modules. Habituation occurs when the boost value of a winning module is decreased because the repeated situation is not significant (i.e. it is *not* necessary for the module to have higher attention). In contrast, sensitisation occurs when the boost value of a module that does not win the attention is increased because the repeated situation is significant (i.e. it is necessary for the module to have higher attention).

There are three functions used by ASMO's habituation and sensitisation mechanisms:

1. **Significance Function**

   The *significance* function measures the significance of the repeated stimulus. It provides a value between -1.0 to 1.0. A value of less than zero is insignificant whereas a value of greater than zero is significant.

   ASMO's habituation and sensitisation mechanisms require at least the significance function to perform habituation and sensitisation learning. They will not modify the boost value of a module (i.e. skip the module) if this function is not provided.

2. **Delta (δ) Function**

   The *delta* function determines the amount by which the boost value will be decreased or increased (i.e. for habituation and sensitisation respectively).

The boost value of a module is modified in proportion to this function (see Equation 7.1).

This delta function can be provided optionally in the competing non-reflex modules. If it is not provided, then ASMO's habituation and sensitisation mechanisms will use the default delta function defined internally in the habituation and sensitisation supervisor module (see Equation 7.2). By default, the mechanisms will modify the boost value of a module in proportion to the significance function.

3. **Termination Function**

   The *termination* function specifies the condition for the learning to stop. It can be provided optionally in the competing non-reflex modules. If it is not provided, then ASMO's habituation and sensitisation mechanisms will use the default termination function defined internally in the habituation and sensitisation supervisor module (see Equation 7.3). By default, the mechanisms will terminate when the root mean square deviation of the previous 20 consecutive boost values is less than or equal to a defined threshold (see Equation 7.4).

$$B_t(p) = B_{t-1}(p) + \delta(p) \tag{7.1}$$

$$\delta(p) = Significance(p) \times C \quad -1.0 \geq Significance(p) \leq 1.0 \tag{7.2}$$

$$Termination(p) = \begin{cases} True & \text{if } RMSD(p) \leq T_{\text{RMSD}} \\ False & \text{otherwise} \end{cases} \tag{7.3}$$

$$RMSD(p) = \sqrt{\frac{1}{n} \sum_{i=t-n}^{t} \left(B_i(p) - B_{i-1}(p)\right)^2} \tag{7.4}$$

Where:
$B_t(p)$ is the boost value of process p at time t
$\delta(p)$ is the delta function of process p
$Significance(p)$ is the significance function of process p
C is the maximum value of change
$Termination(p)$ is the termination function of process p
$RMSD(p)$ is the root-mean-square deviation of process p
T_{RMSD} is the threshold to stop learning
n is the number of previous boosts

7.4.2 Operant Conditioning

ASMO's operant conditioning mechanism increases and decreases the boost values of non-reflex modules by performing the following four steps:

1. **Pick A Candidate**

   ASMO's operant conditioning mechanism uses its *policy* function to pick a non-reflex module as the candidate for the winner of the attention competition with respect to the module's required resources. This candidate can happen to be the module that has the highest total attention level (i.e. the module that was originally supposed to win), or another competing module that has lower total attention level (i.e. the module that was originally not supposed to win).

2. **Promote The Candidate**

   ASMO's operant conditioning mechanism makes the candidate temporarily win the attention competition by increasing the candidate's boost value to a value such that the candidate's total attention level is higher than the total attention level of other competing modules.

3. **Reinforce or Punish The Candidate**

   After the candidate wins the attention competition, ASMO's operant conditioning mechanism reverts its boost value back to its original value (i.e. to the value it had before its boost value was temporarily increased).

   ASMO's operant conditioning mechanism will use the candidate's *reward* function to check for feedback and measure the candidate's performance. If the reward function returns a non-zero value, then ASMO's operant conditioning mechanism will reinforce or punish the candidate as follows:

   - The mechanism will increase the candidate's boost value by the value given by the reward function (i.e. reinforce the candidate) if the candidate was originally not supposed to win, but the candidate receives a positive reward.
   - The mechanism will decrease the candidate's boost value by the value given by the reward function (i.e. punish the candidate) if the candidate was originally supposed to win, but the candidate receives a negative reward.

4. **Repeat Step 1 to 3 Until Termination**

   ASMO's operant conditioning mechanism repeats Step 1 to Step 3 and stops learning when a condition given by the *termination* function is reached.

There are three functions used by ASMO's operant conditioning mechanism:

1. **Policy Function**

   The *policy* function determines a candidate as the new winner of the competition with respect to required resources. It is defined internally in the operant conditioning supervisor module. By default, it determines a candidate randomly. However, it can be redefined by developers.

2. **Reward Function**

   The *reward* function checks for feedback and measures the candidate's performance. It provides a positive value for reinforcement, a negative value for punishment and zero for no feedback. It is also used to determine the amount by which the boost value will be decreased or increased. The boost value of a module is modified in proportion to this function (see Equation 7.5).

   ASMO's operant conditioning mechanism requires at least the reward function to perform operant conditioning learning. It will not modify the boost value of a module (i.e. will skip the module) if this function is not provided.

3. **Termination Function**

   The *termination* function specifies the condition for the learning to stop. It can be provided optionally in the competing non-reflex modules. If it is not provided, then ASMO's operant conditioning mechanism will use the default termination function defined internally in the operant conditioning supervisor module (see Equation 7.6). By default, the mechanism will terminate when the root mean square deviation of the previous 20 consecutive boost values is less than or equal to a defined threshold (see Equation 7.7).

$$B_t(p) = B_{t-1}(p) + Reward(p) \tag{7.5}$$

$$Termination(p) = \begin{cases} True & \text{if } RMSD(p) \leq T_{\text{RMSD}} \\ False & \text{otherwise} \end{cases} \tag{7.6}$$

$$RMSD(p) = \sqrt{\frac{1}{n}\sum_{i=t-n}^{t}\left(B_i(p) - B_{i-1}(p)\right)^2} \tag{7.7}$$

Where:
$B_t(p)$ is the boost value of process p at time t
$Reward(p)$ is the reward function of process p
$Termination(p)$ is the termination function of process p
$RMSD(p)$ is the root-mean-square deviation of process p
T_{RMSD} is the threshold to stop learning
n is the number of previous boosts

ASMO's operant conditioning mechanism is similar to ASMO's habituation and sensitisation mechanisms, except that developers do not need to specify how to evaluate the effectiveness of the behaviours. Instead, the effectiveness of the behaviours is reflected in the environment where explicit feedback can be directly perceived.

7.4.3 Classical Conditioning

ASMO's classical conditioning mechanism [168] triggers modules to propose actions based on only the presence of the conditional stimulus. It performs the following four steps:

1. **Capture sequences of stimuli**

   ASMO's classical conditioning mechanism captures sequences of stimuli. It represents each sequence of stimuli by using a Markov chain where each node represents a stimulus.

2. **Find the stimulus that will most likely occur given the current stimulus**

   When a stimulus occurs (i.e. the conditioned stimulus occurs), ASMO's classical condition mechanism calculates each probability that another stimulus

will occur afterwards (i.e. each possible unconditioned stimulus) given the occurrence of the current stimulus. It calculates this probability by using the maximum likelihood estimation algorithm [33, p. 615]. It will then find the stimulus that has the highest probability (i.e. the most likely unconditioned stimulus).

3. **Trigger modules to propose actions**

   ASMO's operant conditioning mechanism adds the most likely unconditioned stimulus to ASMO's memory as if this stimulus is currently occurring. The addition will cause non-reflex modules to believe that this stimulus is present, despite the fact that this stimulus is not physically present. As a result, it will trigger non-reflex modules to propose actions in order to respond to this stimulus. Hence, the conditioned stimulus has triggered actions that are associated with the unconditioned stimulus, despite the fact that the unconditioned stimulus is not physically present.

4. **Repeat step 2 and 3 for other occurring stimuli**

   ASMO's operant conditioning mechanism repeats step 2 and step 3 if there are other stimuli that are currently occurring.

The Markov chain model used by ASMO's operant conditioning mechanism may require many observations to provide an accurate estimation of reality. However, many observations are often not available and can be difficult to obtain. Thus, the mechanism uses a smoothing technique, such as the *Laplace smoothing* (also called *additive smoothing*) [192], to smooth the observations in order to provide a better estimation.

In the current implementation, ASMO's classical conditioning mechanism works with symbolic data instead of non-symbolic data. For example, in order to associate a ball at coordinate (0.5, 0.8, 0.0) and an action to move to coordinate (0.6, 0.7, 0.0), these data have to be discretised into *ball_location(region1)* and *move(forward)* before the mechanism can learn to associate them.

7.4.4 Observational Learning

ASMO's observational learning mechanism uses a neural network [192, p. 727] to modify the boost values of non-reflex modules (see Figure 7.1)). The inputs of

the neural network are situations in the environment (stored in ASMO's memory). The outputs of the neural network are the boost values of the non-reflex modules. The neural network learns the patterns of what should be the right boost values given the situations. ASMO's observational learning mechanism does not correct the actions proposed by the non-reflex modules, instead it corrects the boost values of the non-reflex modules.

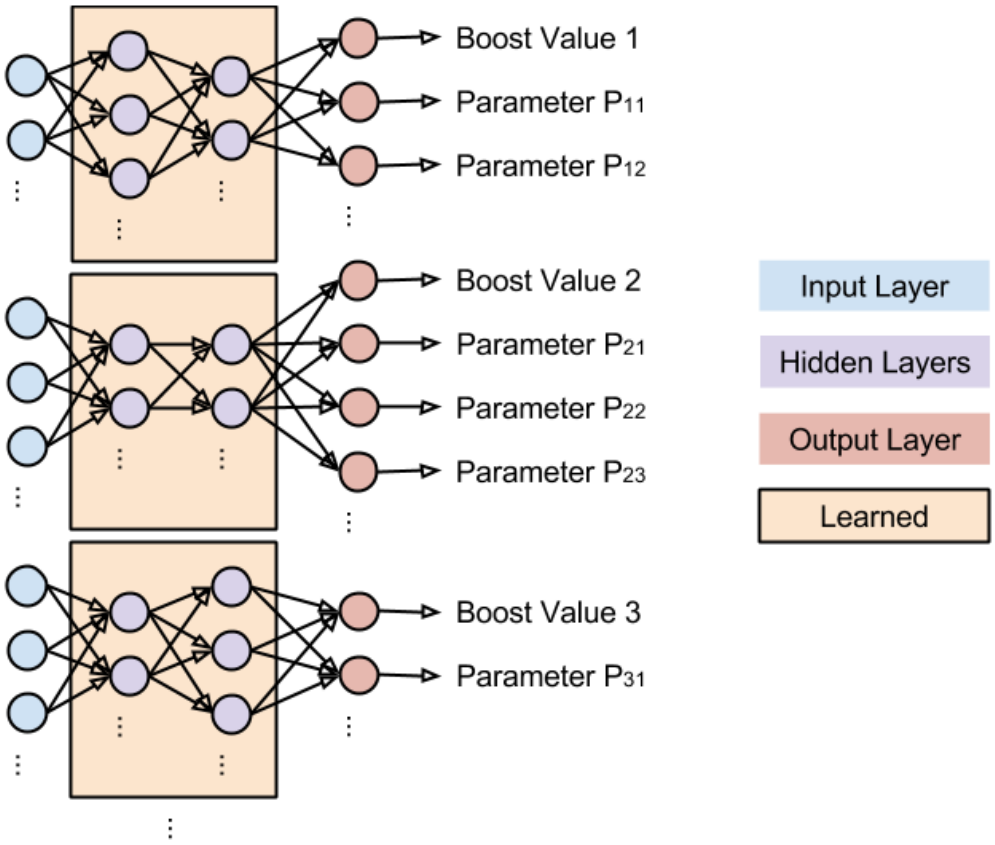


Figure 7.1: Neural Networks for Observational Learning

The neural network is trained from examples provided by experts. Each example is a pair of situations and correct (or desired) boost values. Experts do not need to provide the correct (or desired) boost values manually. Instead, they can provide which modules they think should be the winners of an attention competition (i.e. called experts' winners). ASMO's observational learning mechanism will compare these winners selected by experts with the winners originally selected by ASMO's attention mechanism to determine the correct boost values automatically, as described in the following two conditions (see Algorithm 7.1):

1. **Winners selected by experts and ASMO are the same**

If a winner selected by experts and ASMO is the same, then ASMO's observational learning mechanism does not need to increase the winner's ranking. Thus, it will use the existing boost values as the correct boost values.

2. **Winners selected by experts and ASMO are different**

If a winner selected by the experts is not selected by ASMO (i.e. because its ranking is less than the rankings of its competitors), then ASMO's observational learning mechanism has to increase its ranking to be higher than its competitors' rankings. In the current implementation of ASMO, this means that the winner's total attention level has to be higher than the competitors' highest total attention level.

Developers can choose between two equivalent modes in ASMO's observational learning mechanism to increase the winner's ranking:

- **Increase Mode**

  In the increase mode, ASMO's observational learning mechanism will not change the boost values of the winner's competitors. Instead, it will just increase the winner's boost value in order for the winner's ranking to be higher than the competitors' rankings. It determines the correct boost value for the winner based on the competitors' highest total attention level and the winner's attention value (Equation 7.8 shows the derivation of the boost value needed).

- **Decrease Mode**

  In the decrease mode, ASMO's observational learning mechanism will not change the winner's boost value. Instead, it will decrease the boost values of the competitors that have total attention levels higher than the winner's total attention level, so that the winner's ranking will be higher than the competitors' rankings. It determines the correct boost values for the competitors based on the winner's total attention level and the competitors' attention values (Equation 7.8 shows the derivation of the boost value needed).

$$\left.\begin{aligned} TAL(w) &> TAL(hc) \\ AV(w) + BV(w) &> TAL(hc) \\ BV(w) &> TAL(hc) - AV(w) \\ BV(w) &= TAL(hc) - AV(w) + delta \end{aligned}\right\} \text{increase mode}$$
$$\left.\begin{aligned} TAL(w) &> TAL(hc) \\ TAL(w) &> AV(hc) + BV(hc) \\ TAL(w) - AV(hc) &> BV(hc) \\ BV(hc) &= TAL(w) - AV(hc) - delta \end{aligned}\right\} \text{decrease mode} \tag{7.8}$$

Where:
w is the winner selected by the experts
hc is the highest competitor, which is the module that has the highest ranking among the competitors
$TAL(m)$ is the total attention level of module m
$AV(m)$ is the attention value of module m
$BV(m)$ is the boost value of module m
$delta$ is the difference desired between the winner and the highest competitor

7.5 Other Work

Other learning mechanisms or models have been proposed in the literature. In each of the following subsections, I describe other learning models of habituation and sensitisation, operant conditioning, classical conditioning and observational learning respectively.

7.5.1 Habituation and Sensitisation

In the literature, there have been more computational models of habituation than sensitisation. The majority of computational models of habituation are based on neural networks [116, 139, 51, 205, 140, 56], which range from feed-forward neural networks and recurrent neural networks to artificial spiking neural networks. Some of these models have been implemented in robots [51, 205, 140, 56]. While these

```
1  used_resources = [ ]
2  # expert_winners:  winners selected by experts
3  ranked_expert_winners = sort(expert_winners)
4  # The highest ranking is 1
5  for i ← 1 to count(ranked_expert_winners) do
6      w ← ranked_expert_winners[i]
7      # asmo_winners:  winners selected by ASMO
8      if w not in asmo_winners and required_resources(w) not in
       used_resources then
9          # find_modules(r) returns sorted modules that require
           resources r
10         competing_modules ← find_modules(required_resources(w))
11         highest_module ← competing_modules[1]
12         # total attention level = attention value + boost value
13         if is_increasing_mode then
14             # increase the boost value of the winner selected by
               the expert (i.e.  w)
15             w.boost_value ← highest_module.total_attention_level -
               w.attention_value + delta
16         else
17             # decrease the boost values of other competing modules)
18             for j ← 1 to count(competing_modules) do
19                 if competing_modules[j].total_attention_level ≥
                   w.total_attention_level then
20                     competing_modules[j].boost_value ← w.total_attention_level
                       - competing_modules[j].attention_value - delta
21                 end
22             end
23         end
24     end
25     used_resources ← used_resources + required_resources(w)
26 end
```

Algorithm 7.1: ASMO's Observational Learning Mechanism

networks have various details to simulate habituation effects, all these networks share a common fundamental approach. They have to be trained based on some prior input stimuli. Thus, they cannot learn online and they can only habituate to stimuli that have been trained.

Sirois and Mareschal [206] have reviewed a range of computational models of habituation that include neural-based and non neural-based models. They classify the models into six categories, namely function estimators, symbolic, simple recurrent networks, auto-encoders, novelty filter and auto-associators. The function estimators and symbolic categories are not based on neural network whereas the other four are based on neural network. Sirois and Mareschal assess these models against seven features, namely temporal unfolding, exponential decrease, familiarity-to-novelty shift, habituation to repeated testing, discriminability of habituation items, selective inhibition and cortical-subcortical interactions. Their analysis discovers that none of these models accommodates the familiarity-to-novelty shift feature. Thus, Sirois later proposes a hebbian-based neural network model of habituation that can accommodate this familiarity-to-novelty shift feature [205].

While as to our understanding, there is no existing computational model designed specifically for sensitisation, there are computational models that can accommodate both habituation and sensitisation. The most commonly found model of both habituation and sensitisation is the following model that estimates the exponential strength of a response over time and defines the change in response as a single differential equation (see Equation 7.9). It is often credited to Stanley [211], although in his paper, he used few different equations to describe his model. He refers to his model as a simplification of the dual-process theory of Groves and Thompson [89]. This model accounts for the exponential decay or growth in responses as found in biological habituation and sensitisation respectively. It also accounts for recovery of responses (i.e. dishabituation or desensitisation). However, it lacks a mechanism to accommodate the significance and specificity of the stimulus. The response is a function of stimulus intensity. A stimulus is represented by its strength. The response decays when a stimulus is on and recover when a stimulus is off regardless of the significance of the stimulus.

$$\tau \frac{dy(t)}{dt} = \alpha(y_0 - y(t)) - S(t) \tag{7.9}$$

Where:

$y(t)$ is the strength of the response at current time t
τ is the time constant that adjusts the exponential decay or growth
α is the rate of recovery
y_0 is the initial strength before habituation or sensitisation
$S(t)$ is the intensity of the stimulus

Damper, French and Scutt have proposed alternative computational model of habituation and sensitisation based on a spiking neural network [59]. They have implemented this model on a Khepara robot called ARBIB. This model is connected such that if neuron A synapses onto neuron B then the repetition of fire of neuron A causes the synaptic strength to decrease, which causes the response of neuron B to decrease too. In this model, habituation is achieved by decreasing the *weight* of a synapse every time it fires such that the the weight will return toward the *base weight* with a time constant determined by *recovery*. In addition, sensitisation is achieved by using a facilitatory interneuron I with synapse-on-synapse connection to an $A \rightarrow B$ synapse.

ASMO's habituation and sensitisation mechanisms are real-time non-neural network models. They have been implemented in a physical robot and embedded in a cognitive architecture. They are different from previous works in the following: (i) they account for both habituation and sensitisation, and (ii) they accommodate the significance of the situation.

7.5.2 Operant Conditioning

In computer science (artificial intelligence and machine learning), an operant conditioning has been studied in and designed as a *reinforcement learning* [219]. In this subsection, I describe computational models of operant conditioning that are both designed and not designed as reinforcement learning.

There are surveys on computational models of reinforcement learning in the literature [115, 133]. An example of well-known computational models of reinforcement learning is Q-learning [235]. This model updates a utility (called Q-value) of every pair of state and action based on the reward received for that state (see Equation 7.10). This pair means that this model will have different Q-values for different actions even though the actions were executed from the same state. This model can use the Q-values to update its policy and to correct actions. It can determine

a sequence of actions by finding the action that has the highest Q-value in every state.

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t) \times \Big[R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a_t) - Q_t(s_t, a_t) \Big] \quad (7.10)$$

Where:
$Q_t(s_t, a_t)$ is the Q-value of state s and action a at time t
$\alpha_t(s_t, a_t)$ is the learning rate of state s and action a at time t
R_t is the reward at time t
γ is the discount factor

Unlike reinforcement learning, some computational models of operant conditioning are developed based on an artificial neural network (ANN) and its variant [57, 82, 24, 88]. Other computational models of operant conditioning have been demonstrated to also model classical conditioning [25, 222]. Touretzky and Saksida [222] proposed a model that attempts to capture some aspects that are missing from simple reinforcement learning. Balkenius [25] proposed a model of visual attention that also models habituation, classical conditioning and operant conditioning.

ASMO's operant conditioning mechanism is very similar to Balkenius' computational model of attention incorporating operant conditioning [25]. However, while Balkenius' model focused on visual attention, ASMO's operant conditioning mechanism involves a general kind of attention that can be used for visual, motor coordination and processes. This mechanism is different from previous works in the following: (i) it integrates learning with action selection and attention, (ii) it is embedded in a cognitive architecture, (iii) it is a real-time model and not based on neural network, and (iv) it is implemented in a physical agent.

7.5.3 Classical Conditioning

Computational models of classical conditioning can be divided into models depending on whether they are trial-level or real-time, and whether they are designed based on neural network or not [78, p. 47]. In trial-level models, the association between the stimuli is computed after all relevant stimuli have been observed and terminated. In real-time models, the association between stimuli is computed at every time-frame and the computation can cope with those frames being arbitrarily

small. Furze [78, pp. 47–59] has reviewed a large number of both trial-level and real-time computational models of classical conditioning (for both neural network and non-neural network models) in his dissertation:

- The trial-level neural network models reviewed were the Pearce and Hall model and the Kehoe model.
- The trial-level non-neural network models reviewed were the Stimulus Substitution model, the Rescorla–Wagner model and the Mackintosh's Attention model.
- The real-time neural network models reviewed were the Grossberg model, the Grossberg–Schmajuk (G.S) model, the Klopf model (also called the drive-reinforcement model), the Schmajuk–DiCarlo (S.D) model and the Schmajuk–Lam–Gray (S.L.G) model.
- The real-time non-neural network models reviewed were the Temporal Difference (TD) model, the Sometimes Opponent-Process (SOP) model and the Sutton–Barto (S.B) model.

In this subsection, I focus more details on the real-time non-neural network models: SOP, TD and S.B models. This is because most robots are required to operate in real-time. In addition, non-neural network models allow robots to learn without the need to be trained based on some prior input stimuli. Thus, they allow robots to predict stimuli that have not been trained previously.

The SOP model [234] represents a stimulus in one of three states: A1 (high activation), A2 (low activation) or I (inactive). A stimulus in the A1 state will elicit a primary A1 response (observed as an unconditioned response) whereas a stimulus in the A2 state will elicit a secondary A2 response. Two stimuli that are both in the A1 state will become associated and cause the strength of their association to increase. A stimulus that is either in the A1 or A2 state will induce its associated stimuli to enter their A2 states, which will then elicit their A2 responses (observed as conditioned responses). This inducement occurs in proportion to the strength of the association between the two stimuli. This model supports different phenomena of classical conditioning. However, it requires a stimulus to be represented in one of the three states and it is not implemented in robots.

The Temporal Difference (TD) model [218] is an extension of the Sutton–Barto model [217] proposed by the same authors. These two models rely on reinforcement (or rewards) and eligibility to determine the association strength of a stimulus (7.11). They have the same operations and equations, except that the reinforcement is determined by R_{TD} for the TD model (7.12) or R_{SB} for the SB model (7.13). Unconditioned stimuli have a starting association strength of a positive value. Other stimuli have a starting association strength value of zero.

$$\begin{aligned} \Delta V_t(i) &= \beta R \times \alpha(i)\overline{X}(i) \\ \overline{X}_{t+1}(i) &= \delta \overline{X}_t(i) + (1-\delta)X_t(i) \end{aligned} \tag{7.11}$$

$$R_{TD} = \lambda_t + \gamma Y_t - Y_{t-1} \tag{7.12}$$

$$R_{SB} = \dot{Y}_t = Y_t - Y_{t-1} \tag{7.13}$$

Where:
$R \in R_{TD}, R_{SB}$, $0 < \beta < 1$, $0 < \alpha < 1$
$V(i)$ and $\Delta V(i)$ are the association strength and the change of the association strength of stimulus i respectively
β and α are the constant reinforcement and eligibility learning rates respectively
$X_t(i)$ and $\overline{X}_t(i)$ are the strength and the weighted average strength (called eligibility trace) of conditioned stimulus i at time t respectively
δ is the decay rate of the eligibility trace
λ_t is the strength of the unconditioned stimulus at time t
γ is the discount factor
Y_t is the prediction made at time t of the unconditioned stimulus being associated

ASMO's classical conditioning mechanism is a real-time non-neural network model. It has been implemented in a physical robot and embedded in a cognitive architecture. It is different from previous works in the following: (i) it does not require reinforcement values to learn, and (ii) it incorporates attention and memory manipulation in the learning.

7.5.4 Observational Learning

There are extensive work on computational models of observational learning in the literature (also called imitation learning or learning from demonstration) [17, 46, 31].

Argall et al. divided these models into three approaches based on policy derivation from demonstration data [17]:

1. **Mapping Function**

   In this approach, demonstration data is used to directly approximate the underlying function mapping from the robot's state observations to actions ($f() : Z \rightarrow A$). Models in this approach are further categorised into classification models or regression models depending on whether the outputs of the algorithm are discrete or continuous respectively. Examples of these models have been proposed and developed using several techniques, such as Gaussian Mixture Model [108], Bayesian inference [52], hierarchical neural networks [32], and a combination of Hidden Markov Model and Gaussian Mixture Regression [45].

2. **System Model**

   In this approach, demonstration data is used to determine a model of the world dynamics ($T(s'|s, a)$), and possibly a reward function ($R(s)$). A policy is then derived using this information. This approach is typically formulated within the structure of reinforcement learning (see Section 7.2.2 and 7.5.2). Models in this approach are further categorised into models with engineered reward functions or models with learned reward functions depending on whether the reward functions are manually defined by the user or learned from demonstration respectively. Examples of these models include the model that uses Expectation-Maximization based reinforcement learning [118], the model that incorporates future directed rewards [220] and the model that uses apprenticeship learning via inverse reinforcement learning [1].

3. **Plans**

   In this approach, demonstration data, and often additional user intention information, is used to learn rules that associate a set of pre-conditions and post-conditions with each action ($L(preC, postC|a)$), and possibly a sparsified state dynamics model ($T(s'|s, a)$). A sequence of actions is then planned using this information. Example of models in this approach have been proposed and developed using several techniques, such as skill trees construction [117] and behaviour network [156].

ASMO's observational learning mechanism has been implemented in a physical robot and embedded in a cognitive architecture. It falls into an approach similar to the system model approach, because it does not directly map the robot's perceptions to actions. Neither it does generate rules that associate conditions with each action. However, unlike the system model approach, it uses demonstration data without a reward function to determine a learning model (i.e. boost values of non-reflex modules), whereas the system model approach uses demonstration data with a reward function to determine a model of the world dynamics. Learning with a reward function (i.e. based on the reward feedback from the environment) is classified as an operant conditioning, which is different to an observational learning.

Chapter 8

Evaluation

In Chapter 5, 6 and 7, I described ASMO's attention, emotion and learning mechanisms respectively. In this chapter, I evaluate ASMO and its mechanisms based on the four criteria of cognitive architecture identified in Section 2.5, namely versatility, reactivity, robustness and extensibility. I start by describing the application of ASMO in the RoboCup Soccer SPL standard benchmark problem in Section 8.1. I follow by describing the application of ASMO in the *Smokey robot companion* problem in Section 8.2. Finally, I analyse ASMO and discuss how ASMO meets all of the four criteria, which results in addressing the gaps identified in the research literature in Section 8.3.

8.1 RoboCup Soccer SPL Standard Benchmark Problem

As described in Section 1.1, the RoboCup Soccer SPL standard benchmark problem is an excellent problem for evaluating the ASMO cognitive architecture, because of the following main reasons:

1. **New Grand Challenge**

   This problem has been viewed as a new grand challenge in artificial intelligence after computer chess [114].

2. **Common Benchmark Problem**

This problem provides a common benchmark where different cognitive architectures can be compared, examined and shared.

3. **Regular Refinement**

   This problem is evaluated through international competitions where their rules are refined annually to advance the state of the art of artificial intelligence.

4. **Most Difficult Properties**

   This problem has the most difficult properties characterised by Russel and Norvig [192, p. 44]: partially observable, non-deterministic, sequential, dynamic and continuous.

I have evaluated ASMO in this RoboCup Soccer SPL standard benchmark problem at the RoboCup SPL competition in 2010 (i.e. based on the rules in 2010) [163]. I was part of the team (called *WrightEagleUnleashed!*) that participated the RoboCup SPL competition 2010. Our team was (and is) a joint team between two institutes. It consisted of students and researchers in two physically separated locations. The students had various backgrounds. They might or might not know advanced programming.

In our team, we developed techniques independently as modules to achieve the robot soccer task. We used ASMO to integrate all of the independently developed techniques. We applied ASMO in each robot to govern the robot's behaviours to play soccer.

In the RoboCup SPL competition 2010, robots must achieve many goals simultaneously. However, for simplicity, I describe only five of these goals in this dissertation (see Figure 8.1), as listed as follows:

1. Chase the ball
2. Defend an attack
3. Track the ball
4. Track the opponent's goal
5. Get up autonomously when fall down

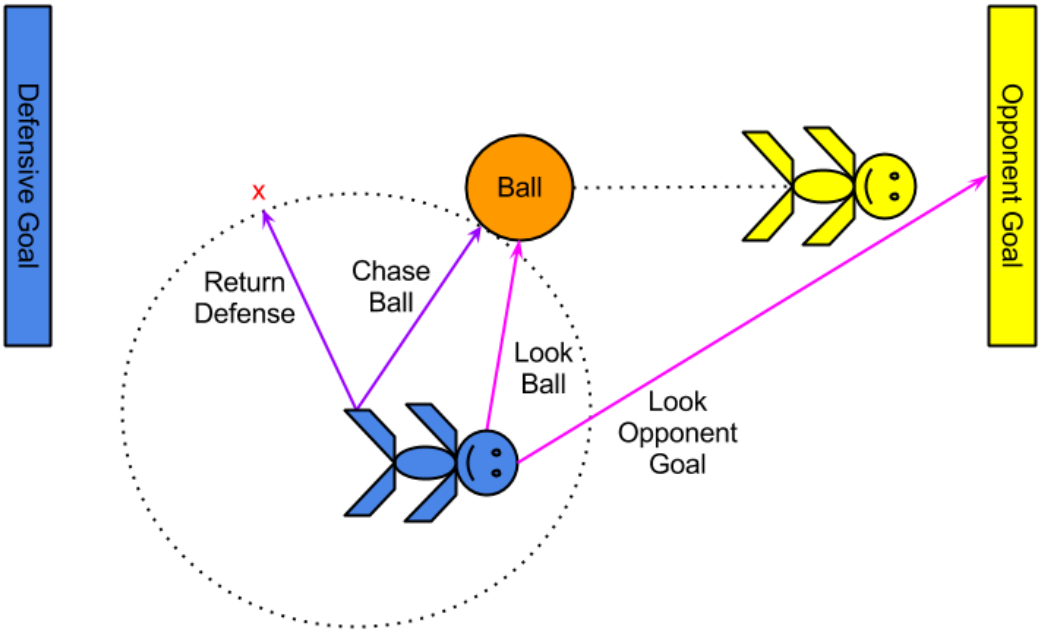


Figure 8.1: RoboCup Soccer Scenario

The robots cannot pursue all the five goals simultaneously, because they have limited resources (e.g. head, legs, CPU, RAM, etc). They cannot chase the ball and defend an attack at the same time, because they have only a pair of legs. In addition, they cannot track the ball and track the opponent's goal at the same time because they have only one head.

Instead, the robots have to decide between chasing the ball and defending an attack. Chasing the ball may increase the chances of controlling the ball sooner. However, it may also leave a defensive position unguarded or unattended, thus opening an opportunity for the opponent to attack. In contrast, defending an attack may decrease the chances of the opponent's attack being successful. However, it may also decrease the chances of controlling the ball because the opponent may get to the ball sooner.

In addition, the robots have to decide between tracking the ball and tracking the opponent's goal. Tracking the ball may allow the robots to keep track of the location of the ball, thus increase the confidence about the location of the ball. However, it may cause the robots to not see the opponent's goal, thus decreasing the confidence about the location of the opponent's goal. In contrast, tracking the opponent's goal may increase the confidence about the location of the opponent's goal. However, it may decrease the confidence about the location of the ball.

In order to achieve the five goals above, six modules are created in ASMO (see Figure 8.2):

1. **The update_world Ordinary Module**

   The *update_world* module captures images and processes the images to recognise objects in the environment, such as the ball, the field, field lines, the opponent's goal, the own team's goal, opponent robots and same team robots. It proposes an action to store the identified objects in ASMO's memory and update the model of the environment. This action is set to require a visual perception virtual resource identified as '/asmo/perception/visual'. The update_world module is the only module that requires this resource. Thus, it will always be selected in an attention competition.

   The update_world module mainly perceives objects in the environment and updates the model of the environment, as described as follows:

   - **Update Confidence**

     The environment is partially observable. The robot player may not see some objects in the environment. It has to remember the objects that it has previously seen. The update_world module increases the confidence about the locations of objects when the objects are recognised in the environment. It reduces the confidence when the objects are not seen for a long time or not seen at the expected locations. The robot is said to know or not know the location of the object if its confidence is high (i.e. above or equal to a threshold) or low (i.e. below a threshold) respectively.

   - **Determine Nearest Opponent to Ball**

     The update_world module will determine the location of the nearest opponent to the ball when the ball is known and at least one opponent is recognised (see Equation 8.1).

$$no = \arg\min_{op} |b - op| \tag{8.1}$$

     Where:
     no is the location of the nearest opponent
     b is the location of the ball
     op is the location of an opponent

- **Infer Opponent's Goal or Own Team's Goal**

  The update_world module will infer the location of the opponent's goal from the location of the own team's goal when only the own team's goal is known, and vice versa. The opponent's goal and the own team's goal are opposite to each other in x-dimension.

2. **The chase_ball Ordinary Module**

   The *chase_ball* module proposes an action only if the location of the ball is known. It proposes to chase the ball by the fastest route if the ball is far from where the robot can kick (i.e. the kicking distance). Otherwise, if the ball is within the kicking distance, then the module proposes to align the robot with the opponent's goal in order to kick the ball effectively. Both of these actions require control of the robot's left leg and right leg (i.e. both leg resources).

3. **The defend_attack Ordinary Module**

   The *defend_attack* module proposes an action to block the opponent's attacking angle and then approach the ball while blocking this angle (i.e. this action is like chasing the ball by the safest route in terms of defending the attack). It proposes this action only if the location of the own team's goal (or the opponent's goal) is known. Note that the location of the own team's goal can be inferred from the location of the opponent's goal, and vice versa. This defend action requires control of both of the robot's legs (i.e. both leg resources). The defend_attack module requires the same resources as required by the chase_ball module. Thus, ASMO will only select one among the two modules in an attention competition.

4. **The track_ball Ordinary Module**

   The *track_ball* module proposes an action to look at the ball if the location of the ball is known. Otherwise, it determines the location to search for the ball and proposes to look at this location if the location of the ball is unknown. This looking action requires control of the robot's head (i.e. head resource).

5. **The track_opponent_goal Ordinary Module**

   The *track_opponent_goal* proposes an action to look at the location to search for the opponent's goal only if the location of the opponent's goal or the own team's goal is unknown. This action requires control of the robot's head (i.e. head resource). The track_opponent_goal module requires the same resources

as the track_ball module. Thus, ASMO will only select one among the two modules in an attention competition.

6. **The get_up Reflex Module**

   The *get_up* module analyses the robot's accelerometer to detect whether the robot has fallen down. It proposes an action to get up if the robot has fallen down. This action requires a control to the whole body (i.e. whole body's resources).

   Recall that a reflex module is given a higher priority than an ordinary or a supervisor module. Designing the get_up module as a reflex module ensures that the module and its proposed action (i.e. to get up) can use the whole body's resources immediately when necessary, without the need to compete with other ordinary and supervisor modules. Its action will occupy the whole body's resources, so other modules cannot be selected. Performing this get up action immediately is crucial, because a fallen robot is unable to play, walk or look at objects and is potentially subject to removal from the game.

Note that modules do not always propose actions in every attention competition. Instead, they propose actions only when necessary, depending on the situation in the environment. For example, the chase_ball module proposes an action only when the location of the ball is known whereas the track_ball module proposes an action regardless of whether the location of the ball is known or not.

In the following subsections, I describe how ASMO's attention mechanism manages resources and select modules to achieve goals. I also describe the potential use of ASMO's emotion and learning mechanisms to bias and modify the selection of modules.

8.1.1 ASMO's Attention Mechanism

ASMO's attention mechanism runs attention competitions to select modules (i.e. perform decision making) and manage limited resources. It selects modules based on their rankings. Recall from Section 5.4.1 that a reflex module is given a higher ranking when competing against ordinary and supervisor modules. Otherwise, its ranking is determined based on its priority when competing among each other. An ordinary or a supervisor module has to compete for attention and its ranking is

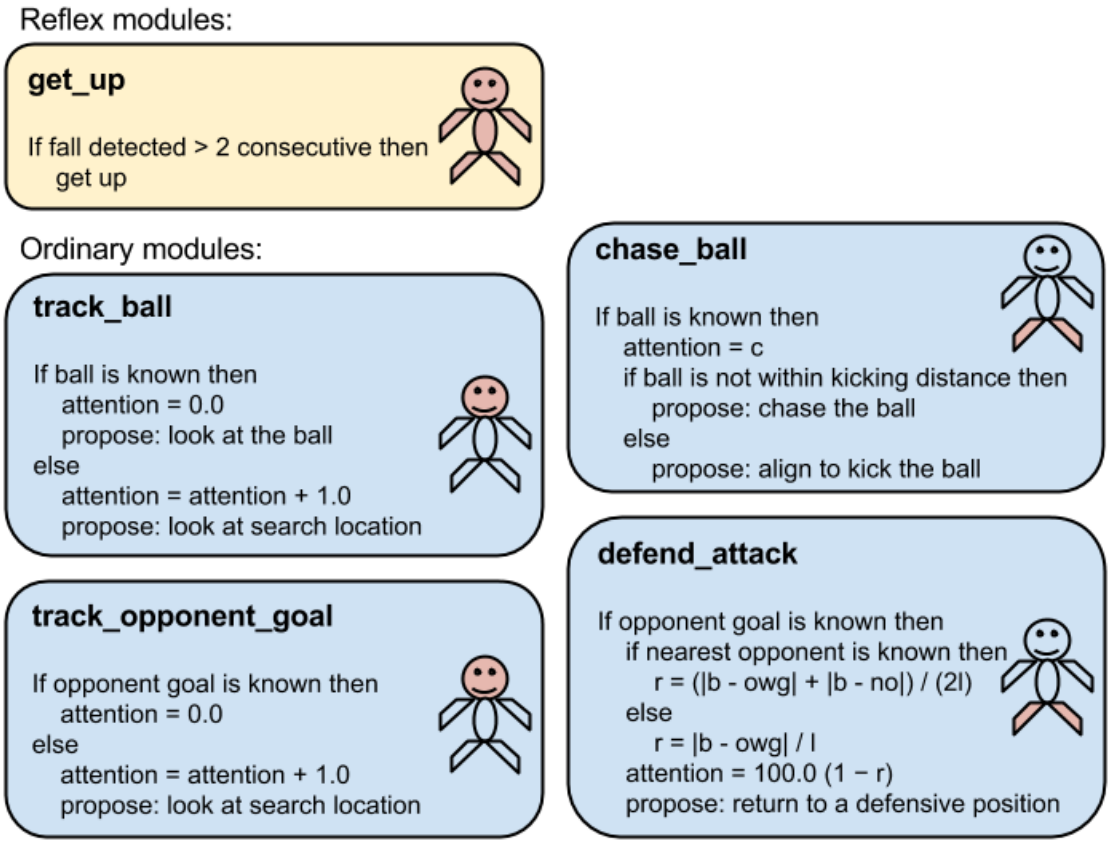


Figure 8.2: Pseudocode of Some Modules Used in RoboCup Soccer System

determined by its total attention level, which is determined by the sum of its boost value and attention value. It determines its own attention value based on Equation 5.1, as also shown below:

$$\underbrace{AV_t}_{\text{next value}} \leftarrow \underbrace{\alpha_t\, AV_{t-1}}_{\text{accumulation}} + \underbrace{\beta_t\, O_t}_{\text{objective}} + \underbrace{\gamma_t\, S_t}_{\text{subjective}}$$

In the RoboCup SPL competition 2010, ASMO's attention mechanism is set to run an attention competition at every image frame. While the get_up module does not have an attention value since it is a reflex module, the other five ordinary modules update their own attention values, as described as follows:

1. **update_world**

   The update_world module sets its attention value to a constant value of 100.0 (see Equation 8.2). This module is necessary to have a very high attention value, because it has to perceive objects in the environment and update the model of the environment in order to play soccer. However, its attention

value actually does not affect the outcome of the winners selected by ASMO, because it does not have a conflict of resources with other modules. It requires a resource that is not required by other modules. Thus, it will still be selected even if it has just an attention value of 0.0.

$$AV_t(uw) = O_t(uw) = 100.0 \tag{8.2}$$

Where:
$\alpha_t = 0.0$, $\beta_t = 1.0$, $\gamma_t = 0.0$
$AV_t(uw)$ is the attention value of the update_world module at time t

2. **chase_ball**

   The chase_ball sets its attention value to simply a constant value when the location of the ball is known (see Equation 8.3). This constant value means that the robot player demands a constant attention to chase the ball. When the location of the ball is not known, this module's attention value will not affect the attention competition, because this module does not propose an action.

   The constant value, c, is determined by developers depending on how strong the opponent team is and whether the team wants to play more offensive or defensive against the opponent team. For example, in a normal match, this value is typically set to 75.0.

$$AV_t(cb) = O_t(cb) = c \tag{8.3}$$

Where:
$\alpha_t = 0.0$, $\beta_t = 1.0$, $\gamma_t = 0.0$
$AV_t(cb)$ is the attention value of the chase_ball module at time t
c is the constant demand of attention

3. **defend_attack**

   The defend_attack module modifies its attention value based on the distance of the ball to the own team's goal and the distance of the ball to the nearest opponent (see Equation 8.4). The closer the ball to the own team's goal or to the nearest opponent, the higher the attention value. This equation will provide a maximal attention value when the ball is in the own team's goal (i.e. when an opponent has scored a goal). As a result, it will cause

the defend_attack module to propose a defend action although the attack has already over. However, this behaviour was not a problem in the competition, because referees would remove the ball and would restart the game when a goal was scored.

Note that the distance of the ball to the own team's goal, $|b - owg|$, can be inferred from the distance of the ball to the opponent's goal, since the location of the opponent's goal is the opposite of the own team's goal. When the location of the own team's goal is not known, this module does not propose an action and it does not update its attention value.

$$r = \begin{cases} \dfrac{|b - owg| + |b - no|}{2l} & owg \text{ and } no \text{ are known} \\ \dfrac{|b - owg|}{l} & owg \text{ is known} \end{cases} \tag{8.4}$$
$$AV_t(da) = O_t(da) = 100.0(1 - r)$$

Where:
$\alpha_t = 0.0$, $\beta_t = 1.0$, $\gamma_t = 0.0$
$|b - owg| \leq l$, $|b - no| \leq l$
$AV_t(da)$ is the attention value of the defend_attack module at time t
r is the ratio of the distance of the ball to the own team's goal and the distance of the ball to the nearest opponent
b is the location of the ball
owg is the location of the own team's goal
no is the location of the nearest opponent
l is the length of the field

4. **track_ball**

The track_ball module increases its attention value linearly to search for the ball when the location of the ball is unknown (see Equation 8.5). The longer the robot does not know where the ball is, the higher the attention value demanded by the robot in order to search for the ball. When the ball is found or when the location of the ball is known, this module resets its attention value

to zero (i.e. demands the least attention to look at the ball).

$$
\begin{aligned}
O_t(tb) &= \begin{cases} 1.0 & \text{Ball is not known} \\ -AV_{t-1}(tb) & \text{Otherwise} \end{cases} \\
AV_t(tb) &= AV_{t-1}(tb) + O_t(tb) \\
&= \begin{cases} AV_{t-1}(tb) + 1.0 & \text{Ball is not known} \\ 0.0 & \text{Otherwise} \end{cases}
\end{aligned} \tag{8.5}
$$

Where:
$\alpha_t = 1.0$, $\beta_t = 1.0$, $\gamma_t = 0.0$
$AV_t(tb)$ is the attention value of the track_ball module

5. **track_opponent_goal**

   The track_opponent_goal module increases its attention value linearly to search for the opponent's goal when the location of the opponent's goal or the own team's goal is unknown (see Equation 8.6). The longer the robot does not know where the opponent's goal is, the higher the attention value demanded by the robot in order to search for the opponent's goal. When the opponent's goal or own team's goal is found, or when the location of the opponent's goal or own team's goal is known, this module resets its attention value to zero. The location of opponent's goal can be inferred geometrically from the location of own team's goal and vice versa.

$$
\begin{aligned}
O_t(tog) &= \begin{cases} 1.0 & \text{Opponent's goal is not known} \\ -AV_{t-1}(tog) & \text{Otherwise} \end{cases} \\
AV_t(tog) &= AV_{t-1}(tog) + O_t(tog) \\
&= \begin{cases} AV_{t-1}(tog) + 1.0 & \text{Opponent's goal is not known} \\ 0.0 & \text{Otherwise} \end{cases}
\end{aligned} \tag{8.6}
$$

   Where:
   $\alpha_t = 1.0$, $\beta_t = 1.0$, $\gamma_t = 0.0$
   $AV_t(tog)$ is the attention value of the track_opponent_goal module

Both the track_ball module and the track_opponent_goal may have the same attention value since they update their attention values in the same way. This means

that ASMO can choose one of them during an attention competition. However, the track_ball module is more preferred than the track_opponent_goal module, because knowing the location of the ball gives the robot more choices than knowing the location of the opponent's goal. Knowing the location of the ball gives the robot a choice between chasing the ball and defending an attack. In contrast, knowing the location of the opponent's goal (hence the own team's goal) only gives the robot a choice to defend an attack.

In order to ensure that the track_ball module wins the attention competition against the track_opponent_goal module, the boost value of the track_ball module is set to 1.0 while the boost values of other modules (i.e. update_world, chase_ball, defend_attack and track_opponent_goal) remain unchanged at an initial value of 0.0. Note that a boost value is normally learned during the operation of ASMO. However, in this case, it is simply set at the design stage to simulate the learning.

Figure 8.3 shows the result of the robot playing soccer. It shows that the chase_ball module's total attention level was constant over time when the robot knew the location of the ball. The defend_attack module's total attention level was increased overall as the ball was getting closer to the own team's goal and the nearest opponent was getting nearer to the ball. It was eventually higher than the chase_ball module's total total attention level and the robot decided to defend the attack rather than to chase the ball. The track_ball and track_opponent_module's total attention levels were increased to a maximum value when the robot could not find the ball and its vision to the opponent's goal was blocked by referees and other robots. The track_opponent_goal module's total attention level was bumping (i.e. increasing and decreasing) repeatedly as the robot demanded attention to look at the opponent's goal when the confidence about the location of the opponent's goal was low and to look back at the ball when the confidence was high repeatedly.

8.1.2 ASMO's Emotion and Learning Mechanisms

Our team did not use ASMO's emotion and learning mechanisms in the RoboCup Soccer SPL competition, because we did not have much time to test these mechanisms. Thus, the use of these mechanisms in the RoboCup Soccer SPL problem remains future work (see Section 9.2). In the following, I speculate how these mechanisms will be used for this problem:

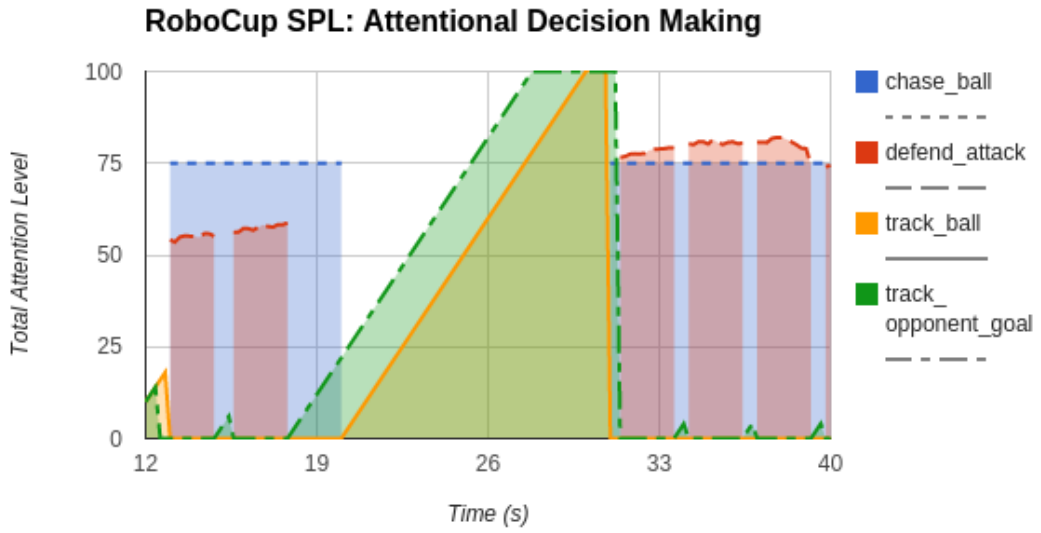


Figure 8.3: Attentional Decision Making in The RoboCup Soccer SPL Competition 2010

- **Aggressive Attitude**

  ASMO's emotion mechanism can potentially be used to build a more aggressive robot player by modifying the chase_ball module's subjective rate and weight (i.e. γ_t and S_t) to a non-zero positive value. A higher subjective weight results in a higher attention value demanded by the chase_ball module. It increases the chance of the module winning the attention competition, which increases the chance of performing the chase action. As a result, the player is more likely to chase the ball and thus becomes more aggressive.

- **Defensive Attitude**

  ASMO's emotion mechanism can potentially be used to build a more defensive robot player by modifying the defend_attack module's subjective rate and weight (i.e. γ_t and S_t) to a non-zero positive value. A higher subjective weight results in a higher attention value demanded by the defend_attack module. It increases the chance of the module winning the attention competition, which increases the chance of performing the defend action. As a result, the player is more likely to defend an attack and thus becomes more defensive.

- **Association Between Ball and Opponents**

  ASMO's classical conditioning learning mechanism can potentially be used to learn the association between the area of the ball and the direction of oppo-

nents. This association is used to chase the ball in the direction of where most opponents are going despite the ball is not seen. ASMO's classical conditioning mechanism will simulate the perception of the ball as if the ball is seen. It will indirectly cause the chase_ball module to propose the chase action even though the ball is not actually seen.

- **Effective Path to Chase the Ball**

  ASMO's operant conditioning learning mechanism can potentially be used to find most effective paths to chase the ball. Modules can use various techniques to propose different paths to chase the ball, such as the shortest path, the fastest path and the path that has the least opponents. ASMO's operant conditioning mechanism will increase or decrease the boost values of modules based on whether the chase action through a path is successful or not respectively. The more frequent the chase action through a path is successful, the higher the chance that the path is effective.

Section 8.3 further discusses the evaluation of ASMO in the RoboCup problem. In the next section, I will first explain the evaluation of ASMO in the Smokey Robot Companion problem.

8.2 Smokey Robot Companion Problem

In addition to the RoboCup soccer SPL standard benchmark problem, ASMO is evaluated in the Smokey robot companion project to demonstrate its application in another domain using a different type of robot. This project, created by the University of Technology Sydney, aims to bring a bear-like robot (called Smokey) to 'life' and explores the meaning of life by interacting socially with people. It has potential applications in nursing, healthcare and entertainment industries by providing companionship to people with disabilities, people with autism, the elderly and children.

Unlike the soccer problem, the robot companion problem is very subjective. There is no single correct way of accompanying a person and, even if there is, it would vary from person-to-person according to their unique personalities.

ASMO is evaluated in an experiment as part of the project. In the experiment, Smokey has to accompany or entertain a person (i.e. the target user) while simulta-

neously regulating the person's rest (see Figure 8.4). Smokey can play either a red ball game or drums to accompany the user. It can also go to sleep to encourage the user to rest (since it will not interact with the user when it is sleeping). When playing, Smokey will also pay attention to any motion in the environment from people other than the user.

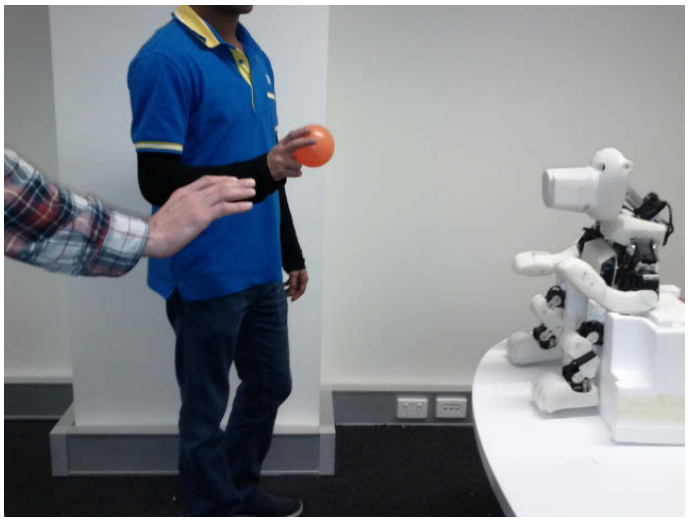


Figure 8.4: Smokey Robot Companion Experiment

In addition, Smokey can receive a request from the user through a graphical user interface to either play the red ball game, play the drums or go to sleep. Smokey will consider this request, but does not necessarily have to perform this request. As a companion robot, Smokey is designed to have an emotion towards the colour red and to like being praised. The more frequent Smokey receives the same request from a user, the more the user wants to see Smokey performing the request, the higher Smokey will believe that it will be praised for performing that request.

Furthermore, Smokey should adapt to its environment and should be able to improve its decisions over time in four situations :

1. **Improving Object Tracking**

   Paying attention to the motions of people other than the target user can cause Smokey to lose track of the ball or the drums. Smokey should be able to focus on significant motions and ignore insignificant motions in order to improve its object tracking.

2. **Adapting to Feedback**

When interacting with a user, Smokey can receive feedback from the user after it performs an action. The feedback indicates whether the user likes Smokey's behaviour or not. Smokey should be able to adapt to the user's feedback. It should improve its decisions to suit the user's personality based on the user's feedback.

3. **Predicting Requests**

   Smokey should be able to predict the request that the user tends to ask and to perform this request before the user asks. This prediction can make Smokey more personalised to the user, which results in better companionship.

4. **Taking Advantage of Training Data**

   Smokey should be able to make and improve its decisions based on training or labelled data, if provided by developers. Training data can be used to improve performance, especially when the model about the world is unknown.

In the experiment, improving object tracking, adapting to feedback, predicting requests and taking advantage of training data were achieved through ASMO's habituation/sensitisation, operant conditioning, classical conditioning and observational learning mechanisms respectively.

ASMO is used to govern Smokey's behaviours in the experiment. In ASMO, modules are created to use various techniques in order to solve Smokey's tasks. Note that ASMO does not put restrictions on how modules should be created and organised as well as how techniques should be used in the modules.

In this experiment, each module was simply created to handle each of Smokey's behaviours, as described as follows:

1. **The attend_motion Ordinary Module**

   The *attend_motion* module proposes an action when Smokey is not sleeping to look at the fastest motion of people other than the user in the environment. It does not propose any action if there is no motion detected in the environment or if Smokey is sleeping. It requires control of Smokey's head to look at the motion (i.e. head resource).

2. **The play_ball Ordinary Module**

The *play_ball* module proposes an action when Smokey is not sleeping to track the ball or search for the ball depending on whether the location of the ball is known or not respectively. It does not propose any action if Smokey is sleeping. It requires control of Smokey's head to track or to search for the ball (i.e. head resource).

3. **The play_drums Ordinary Module**

   The *play_drums* module proposes an action when Smokey is not sleeping to search for the drums when the location of the drums is unknown, to track the drums when the location of the drums is known but not within reach, and to beat the drums when the location of the drums is known and within reach. It does not propose any action if Smokey is sleeping. It requires a control of only Smokey's head to track and to search for the drums (i.e. head resource). Meanwhile, it requires a control of both Smokey's head and arms to beat the drums relative to the position of the drums (i.e. head, left arm and right arm resources).

4. **The go_sleep Ordinary Module**

   The *go_sleep* module proposes an action to go to sleep when Smokey is not sleeping and to wake up after Smokey has enough sleep. It requires control of Smokey's whole body (i.e. whole body's resources).

5. **The attend_request Supervisor Module**

   The attend_request module proposes an action to increase the boost value of the play_ball, play_drums or go_sleep module when Smokey is not sleeping and Smokey receives a request from the user to play the red ball game, play the drums or go to sleep respectively. It increases these boost values proportionally to the probability of the request (see Equation 8.7). The probability is 1.0 when a request is received through a graphical user interface. This module does not require any resource. Thus, it will always be selected when it participates in the attention competition.

$$\begin{aligned} BV(pb) &= P(b) \times 20.0 \\ BV(pd) &= P(d) \times 20.0 \\ BV(gs) &= P(s) \times 20.0 \end{aligned} \tag{8.7}$$

Where:
$BV(pb)$ is the boost value of the play_ball module
$BV(pd)$ is the boost value of the play_drums module
$BV(gs)$ is the boost value of the go_sleep module
$P(b)$ is the probability that the request to play ball is received
$P(d)$ is the probability that the request to play drums is received
$P(s)$ is the probability that the request to go to sleep is received

The attend_request supervisor module influences the boost value of an ordinary module, instead of the attention value of an ordinary module. Recall from Section 5.4.1 that a supervisor module can influence the attention value and boost value of an ordinary module to favour or disfavour the actions proposed by the ordinary module. Influencing the boost value is intended to create a long-term bias in the ordinary module, while influencing the attention value is intended to create a short-term bias in the ordinary module.

In addition to Smokey's actuators, the five modules above actually also require computing resources, such as memory and hard disk. However, in this experiment, these resources are assumed to be negligible. Thus, modules do not compete for these resources.

In the following subsections, I demonstrate the application of ASMO's attention, emotion and learning mechanisms in this Smokey robot companion problem.

8.2.1 ASMO's Attention Mechanism

ASMO's attention mechanism runs attention competitions to select modules (i.e. perform decision making) and manage resources in Smokey. It selects modules based on their rankings. Recall from Section 5.4.1 that a reflex module is given a higher ranking when competing against ordinary and supervisor modules. Otherwise, its ranking is determined based on its priority when competing among each other. An ordinary or a supervisor module has to compete for attention and its ranking is determined by its total attention level, which is determined by the sum of its boost value and attention value. It determines its own attention value based on Equation 5.1 as also shown below:

$$\underbrace{AV_t}_{\text{next value}} \leftarrow \underbrace{\alpha_t\, AV_{t-1}}_{\text{accumulation}} + \underbrace{\beta_t\, O_t}_{\text{objective}} + \underbrace{\gamma_t\, S_t}_{\text{subjective}}$$

In the experiment, the five modules determine their attention values as described as follows:

1. **The attend_motion Module**

   The attend_motion module determines its attention value based on only its objective weight (see Equation 8.8). It uses a simple connection-based neural network to determine its objective weight (thus its attention value), which is based on the speed and intensity of the fastest motion in the environment (see Figure 8.5). The neural network is trained such that the faster the motion that has a medium to high average of intensity, the more attention demanded by the module in order to look at the motion, the higher the attention value.

$$AV_t(am) = O_t(am) = NN(speed, average\_intensity) \tag{8.8}$$

   Where:
   am is the attend_motion module
   $\alpha_t(am) = 0.0$, $\beta_t(am) = 1.0$, $\gamma_t(am) = 0.0$
   $AV_t(am)$ is the attention value of the attend_motion module
   $NN(speed, average\_intensity)$ is the output of the neural network given the speed and average intensity

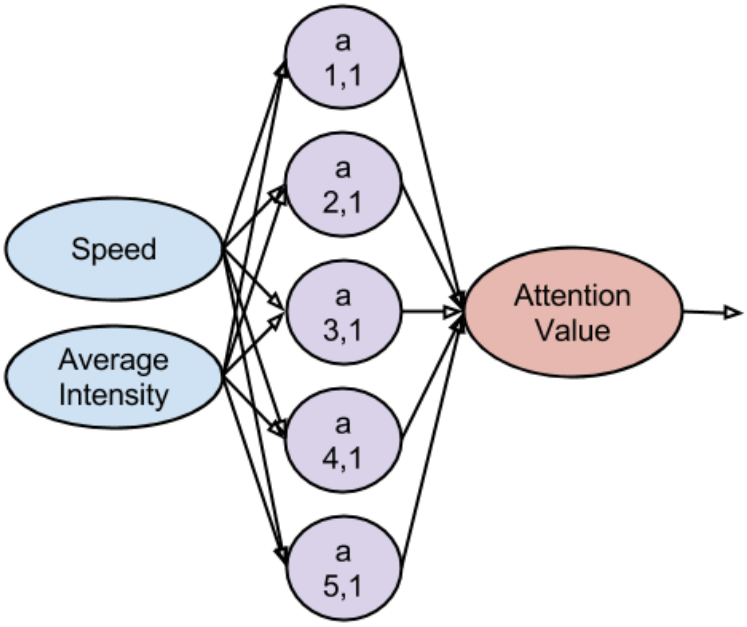


Figure 8.5: Neural Network Used by The attend_motion Module

2. **The play_ball Module**

The play_ball module determines its attention value based on the user's preference of playing ball (see Equation 8.9). Its subjective weight, $S_t(pb)$, is initially set to zero and will be determined by ASMO's emotion mechanism in the next subsection. The user's preference is determined by using a priority-based or hierarchical-based approach (see Table 8.1).

$$AV_t(pb) = O_t(pb) = \begin{cases} 60.0 & \text{if play ball is more preferred} \\ 50.0 & \text{Otherwise} \end{cases} \tag{8.9}$$

Where:
pb is the play_ball module
$\alpha_t(pb) = 0.0$, $\beta_t(pb) = 1.0$, $\gamma_t(pb) = 1.0, S_t(pb) = 0.0$
$AV_t(pb)$ is the attention value of the play_ball module

Name	Condition	Priority
Anshar	Morning	Play Ball = Play Drums
	Afternoon	Play Ball < Play Drums
	Evening	Play Ball > Play Drums
Ben	Before Meal	Play Ball > Play Drums
	After Meal	Play Ball < Play Drums
Evelyn	Before Exercise	Play Ball < Play Drums
	After Exercise	Play Ball > Play Drums
Michelle	All times	Play Ball < Play Drums
Xun	After Meal	Play Ball > Play Drums

Table 8.1: Users' Preferences of Playing Ball and Drums

3. **The play_drums Module**

The play_drums module determines its attention value based on the user's preference of playing drums (see Equation 8.10). Its subjective weight, $S_t(pd)$, is initially set to zero and will be determined by ASMO's emotion mechanism in the next subsection. The user's preference is determined by using a priority-

based or hierarchical-based approach (see Table 8.1).

$$AV_t(pd) = O_t(pd) = \begin{cases} 60.0 & \text{if play drums is more preferred} \\ 50.0 & \text{Otherwise} \end{cases} \tag{8.10}$$

Where:
pd is the play_drums module
$\alpha_t(pd) = 0.0$, $\beta_t(pd) = 1.0$, $\gamma_t(pd) = 1.0$, $S_t(pd) = 0.0$
$AV_t(pd)$ is the attention value of the play_drums module

4. **The go_sleep Module**

The go_sleep module determines its attention value based on its previously accumulated attention value and its objective weight (see Equation 8.11). Its accumulated attention value, $AV_{t-1}(gs)$, is initially set to 0.0. Its objective weight, $O_t(gs)$, is determined by three operators using a goal-based action description, as described as follows:

$$AV_t(gs) = AV_{t-1}(gs) + O_t(gs) \tag{8.11}$$

Where:
gs is the go_sleep module
$\alpha_t(gs) = 1.0$, $\beta_t(gs) = 1.0$, $\gamma_t(gs) = 0.0$
$AV_t(gs)$ is the attention value of the go_sleep module

(a) **Increase Operator**

The increase operator sets the objective weight to a constant value in order to increase the attention value linearly when the following pre-conditions are true:

$$\neg ReachLimit(AV(gs)) \wedge \neg Winner(gs)$$

The $ReachLimit(AV(gs))$ proposition is true if the attention value of the go_sleep module is more than or equal to the maximum value of the module's ranking (i.e. $AV(gs) \geq 100.0$).

The $Winner(gs)$ proposition is true if the go_sleep module wins the attention competition.

(b) **Maintain Operator**

The maintain operator sets the objective weight to 0.0 in order to maintain the attention value when the following pre-conditions are true:

$$Winner(gs) \wedge \neg enough\_sleep$$

The *enough_sleep* proposition is true if the duration of the sleep is greater than or equal to a defined period (i.e. $Time(sleep) \geq T_{\text{sleep}}$).

(c) **Clear Operator**

The clear operator sets the objective weight to the previous attention value (i.e. $O_t(gs) = -AV_{t-1}(gs)$) in order to clear the attention value to 0.0 when the following pre-condition is true:

$$enough\_sleep$$

The go_sleep module will repetitively select operators to modify its objective weight in order for Smokey to have a regular sleep (i.e. to have the *enough_sleep* proposition true and false regularly). An operator can only be selected when its pre-conditions are satisfied.

5. **The attend_request Module**

The attend_request module sets its attention value to a constant value of 10.0 (see Equation 8.12). This value does not hold any significant meaning. It does not have to be 10.0. It can be any value between 0.0 to 100.0. The reason is because the attend_request module does not need to compete for attention to gain access to resources since this module does not require any resource. Thus, this module will always be selected regardless of its attention value.

$$AV_t(ar) = O_t(ar) = 10.0 \tag{8.12}$$

Where:
ar is the attend_request module
$\alpha_t(ar) = 0.0$, $\beta_t(ar) = 1.0$, $\gamma_t(ar) = 0.0$
$AV_t(ar)$ is the attention value of the attend_request module

Figure 8.6 shows the result of the experiment without an emotional subjective bias and learning. It shows every module's total attention level (i.e. the sum of the attention value and the boost value) when Smokey was interacting with a user that

prefers Smokey to play ball than to play drums. It shows that the total attention level of the attend_motion module was fluctuating based on the speed and average intensity of the motion. When it was higher than other modules' total attention levels, Smokey chose to attend to the motion rather than to search for the ball or the drums. The total attention levels of the play_ball, play_drums and attend_request modules were constant. The total attention level of the play_ball module was higher than the attention value of the play_drums module, because the user prefers Smokey to play ball than to play drums. The total attention level of the go_sleep module increased linearly until it exceeded other modules' total attention levels and then it reset back to zero after Smokey had enough sleep.

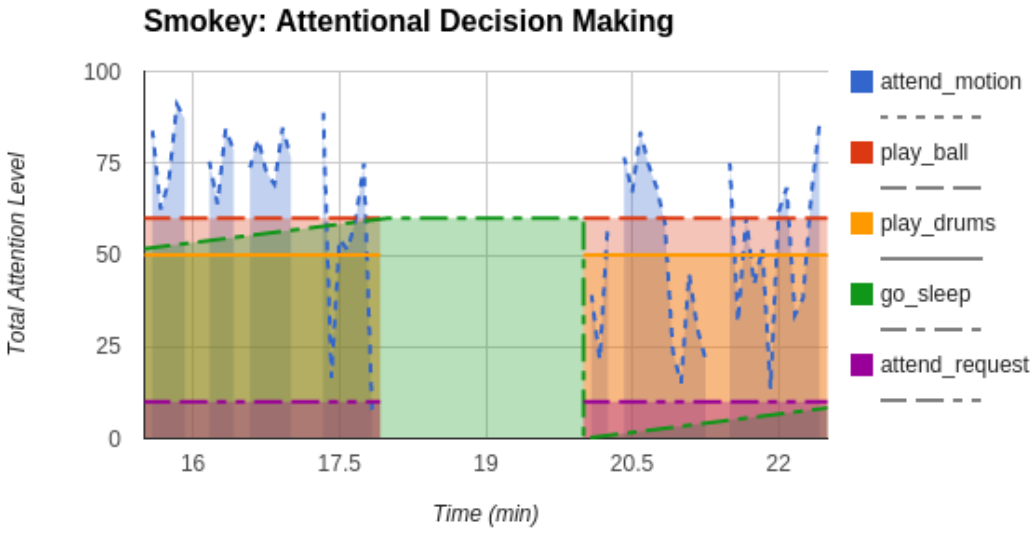


Figure 8.6: Attentional Decision Making in Smokey

8.2.2 ASMO's Emotion Mechanism

ASMO's emotion mechanism biases the selection of modules (i.e. decision making) in Smokey. In the experiment, Smokey is designed to have emotions toward the colour red and toward praise. These emotions are modelled in ASMO as nodes in a causal Bayesian network (see Figure 8.7):

1. **Action Node**

   The *action* node represents Smokey's actions that may affect its emotions.

2. **Red Sensation and Praise Nodes**

The *red sensation* and *praise* nodes represent the biological and cognitive factors that cause the emotions respectively.

3. **Robot's Happiness and User's Happiness Nodes**

   The *robot's happiness* and *user's happiness* nodes represent the happiness of Smokey and the happiness of the user respectively.

4. **Positive Valence, Negative Valence and Arousal Nodes**

   The *positive valence*, *negative valence* and *arousal* nodes represent the dimensions of the emotions.

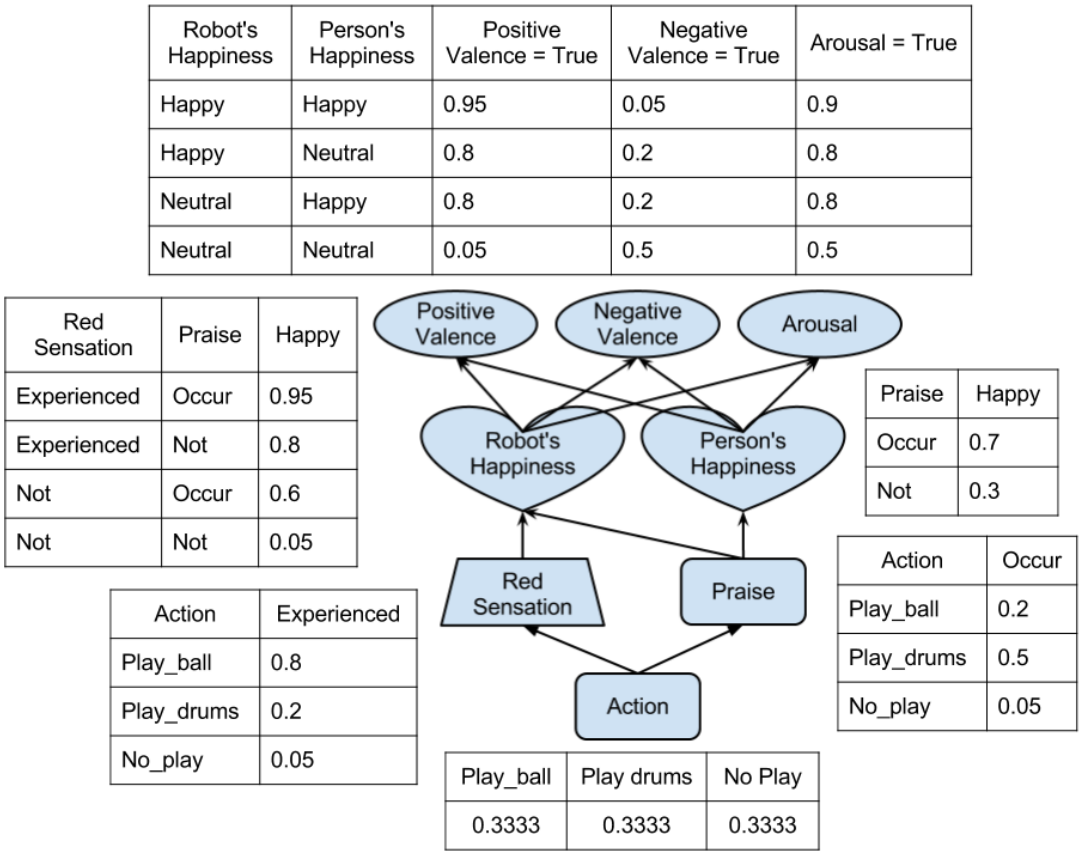


Figure 8.7: Causal Bayesian Network Used in Smokey

The emotion towards the colour red is an innate emotion and indicated by a biological node, because there is no an explicit reason for Smokey to like the colour red, apart from the design decisions of its developers (i.e. inherentance). In contrast, the emotion towards praise is a learned emotion and indicated by a cognitive node, because Smokey can learn that being praised is relevant to, and impacts, the

performance of accompanying the user. In this experiment, however, it is assumed that the desire to be praised is already learned.

ASMO's emotion mechanism is implemented in a supervisor module called *emotion_mechanism*. This module uses the causal Bayesian network described above to predict Smokey's and the user's happy feelings when Smokey plays the ball or the drums (i.e. based on Equation 6.1). It will then determine the subjective weights of the play_ball and play_drums modules (i.e. bias the modules) according to the expected desirability of these feelings (i.e. based on Equation 6.2).

Table 8.2 shows the predictions of Smokey's happy and the user's happy feelings and also the subjective weights of the play_ball and play_drums modules for different probabilities of receiving praise. The higher the probability of receiving praise, the higher the subjective weight of the play_drums module. Both desirabilities of Smokey's happy and the user's happy are set to 20.0. Symbols used in the table are as described as follows:
$P(pr|pda)$ is the probability of receiving praise given Smokey plays the drums
$P(rh|pba)$ is the probability of Smokey is happy given Smokey plays the ball
$P(uh|pba)$ is the probability of the user is happy given Smokey plays the ball
$P(rh|pda)$ is the probability of Smokey is happy given Smokey plays the drums
$P(uh|pda)$ is the probability of the user is happy given Smokey plays the drums
$S_t(pb)$ is the subjective weight of the play_ball module at time t
$S_t(pd)$ is the subjective weight of the play_drums module at time t

$P(pr\|pda)$	Play Ball			Play Drums		
	$P(rh\|pba)$	$P(uh\|pba)$	$S(pb)$	$P(rh\|pda)$	$P(uh\|pda)$	$S(pd)$
0.5	0.696	0.38	10.76	0.435	0.5	9.35
0.55	0.696	0.38	10.76	0.4585	0.52	9.785
0.6	0.696	0.38	10.76	0.482	0.54	10.22
0.65	0.696	0.38	10.76	0.5055	0.56	10.655
0.7	0.696	0.38	10.76	0.529	0.58	11.09
0.75	0.696	0.38	10.76	0.5525	0.6	11.525
0.8	0.696	0.38	10.76	0.576	0.62	11.96

Table 8.2: The Subjective Weights of the play_ball and play_drums Modules

Figure 8.8 shows the results of the two experiments without and with ASMO's emotion mechanism respectively. In the experiment, the user's preferences of Smokey

playing the ball and the drums were equal and Smokey received requests to play drums. These requests to play drums increased the probability of receiving praise given that Smokey plays the drums (i.e. $P(pr|pda)$).

Without ASMO's emotion mechanism, both the play_ball and play_drums modules had an equal total attention level, since the user's preference is equal (see Figure 8.8a). Thus, Smokey relied on a conflict resolution strategy (see Section 5.4.4) to choose between the two modules.

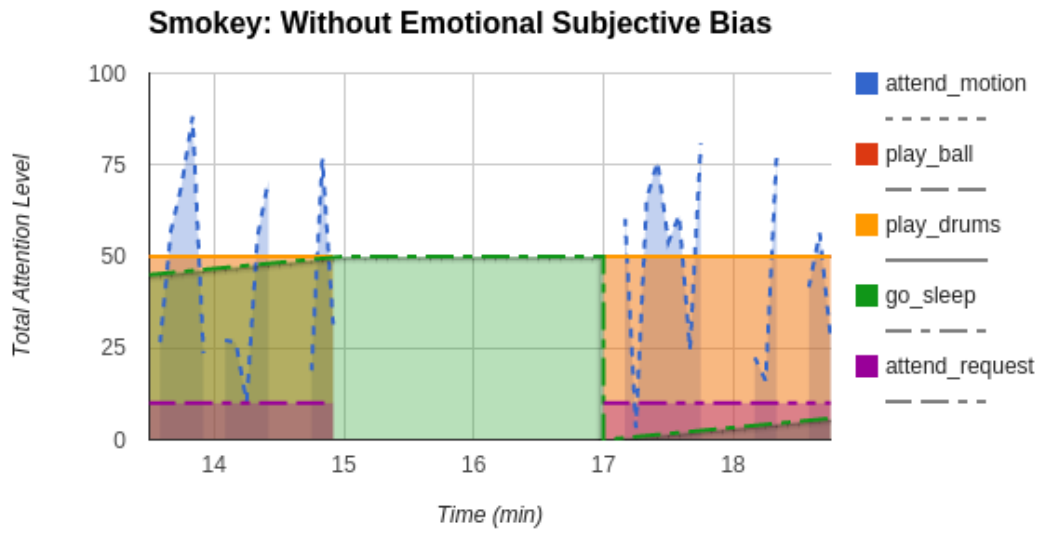
With ASMO's emotion mechanism, the play_ball and play_drums modules' total attention levels were increased by their subjective weights (see Figure 8.8b). The play_drum module's subjective weight was increased as the probability of receiving praise was increased. Over time, the total attention level of the play_drums module was slightly higher than the total attention level of the play_ball module. Thus, Smokey chose to play drums instead of to play ball when interacting with the user. In addition, Smokey's sleep time was shifted because the go_sleep module took a longer time to win the attention competition against the play_ball and play_drums modules.

8.2.3 ASMO's Habituation and Sensitisation Mechanisms

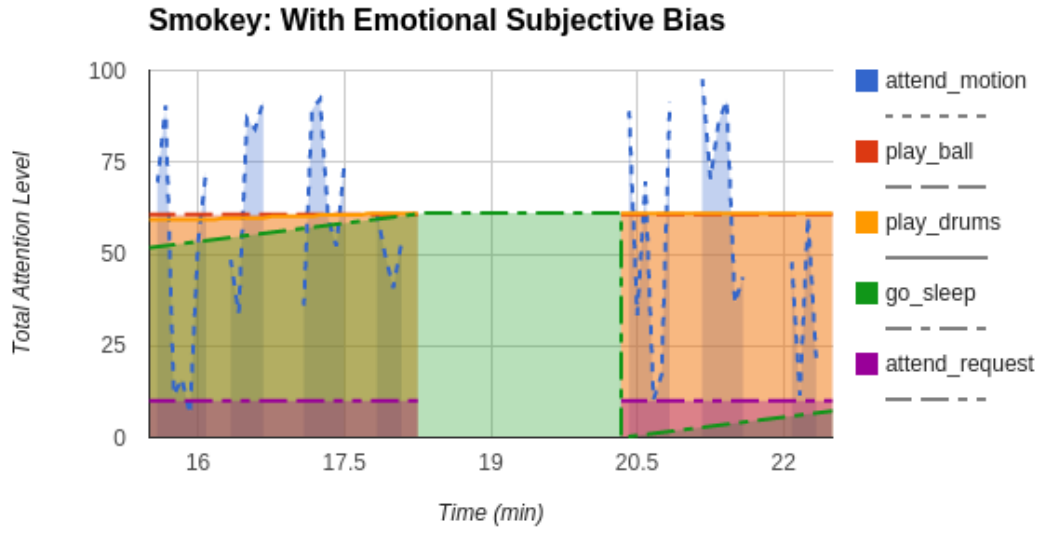
ASMO's habituation and sensitisation mechanisms learn to modify the boost values of non-reflex modules based on the significance of the modules. Recall from Section 7.4.1 that these mechanisms are implemented in a supervisor module and they modify the boost value of any non-reflex module that has defined a *significance* function.

In the experiment, ASMO's habituation and sensitisation mechanisms were enabled to learn to focus on significant motion and to ignore insignificant motion in order to improve object tracking (described on Page 178). Since attending to motion was handled by the attend_motion module, ASMO had to learn to modify the boost value of the attend_motion module in order to focus and ignore motion appropriately. Thus, the attend_motion module was added with a significance function required for habituation and sensitisation learning.

The significance function of the attend_motion module measures the significance of paying attention to motion based on the distance between the centre of the ball and the bounding sphere of the motion (see Equation 8.13). It returns a positive



(a) Without Emotional Subjective Bias



(b) With Emotional Subjective Bias

Figure 8.8: Experiments of Emotional Subjective Bias in Smokey

value for a distance less than or equal to the significant boundary threshold b, a negative value for a distance greater than b but within the space threshold, and a minimum negative value of -1.0 for a distance beyond the space threshold. Motion that occurs closer to the ball (i.e. the object of interest) has higher significance than motion that occurs far from the ball since Smokey's goal is to track the ball when it is not sleeping.

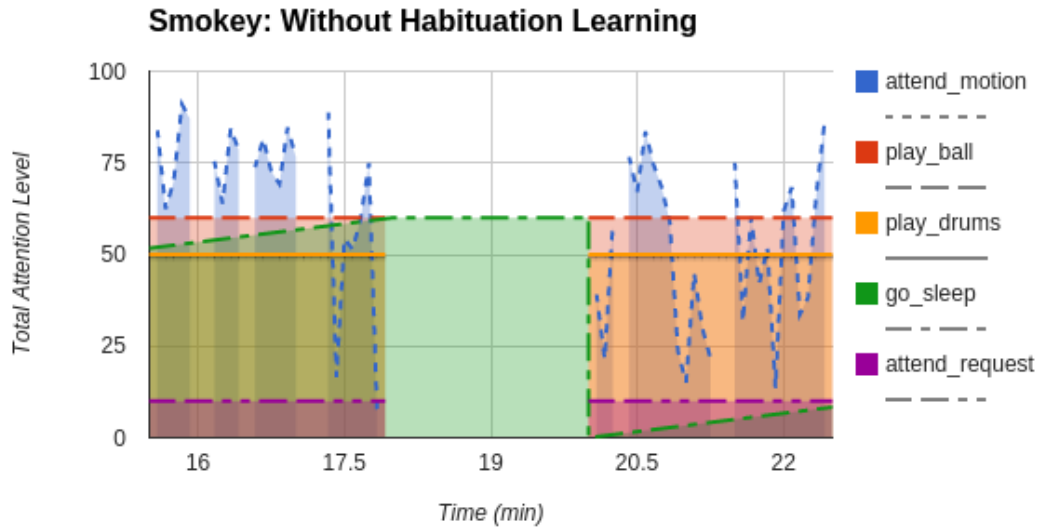
$$significance(\text{attend\_motion}) = \begin{cases} \frac{b-d}{b} & \text{if } d \leq b \\ \frac{-d}{l} & \text{if } d > b, d < l \\ -1.0 & \text{if } d > l \end{cases} \tag{8.13}$$

Where:
$b < l$
$-1.0 \leq significance(\text{attend\_motion}) \geq 1.0$
d is the distance between the centre of the ball and the bounding sphere of the motion
b is the threshold of the significant boundary from the ball
l is the space threshold

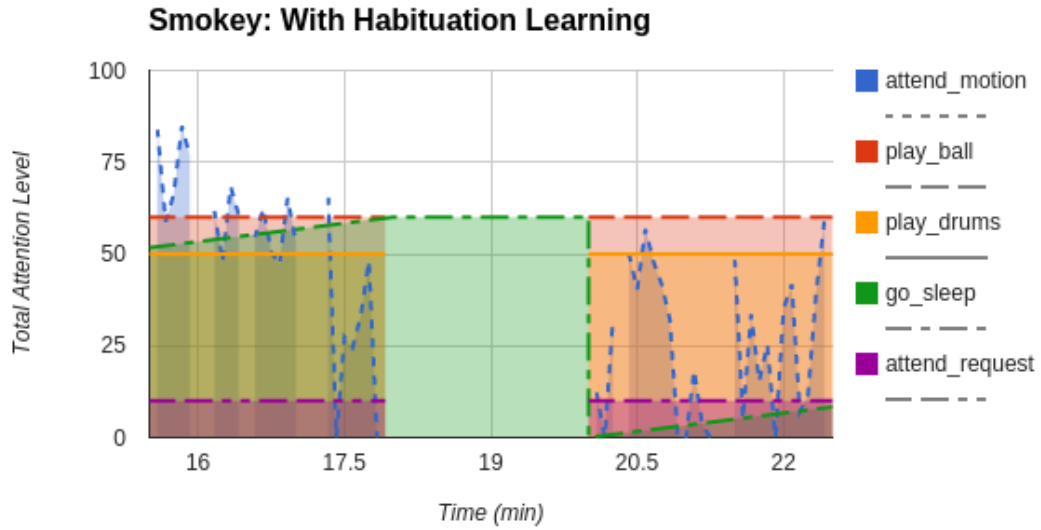
ASMO's habituation and sensitisation mechanisms modify the boost value of the attend_motion module directly. ASMO's habituation mechanism evaluates the significance of the attend_motion module when the module wins the attention competition and decrease the module's boost value if the significance is negative (i.e. not significant). In contrast, ASMO's sensitisation mechanism evaluates the significance of the attend_motion module when the module does not win the attention competition and increase the module's boost value if the significance is positive (i.e. significant).

Figure 8.9 shows the results of the experiments with and without ASMO's habituation and sensitisation learning mechanisms for two situations. The first situation was when people other than the target user were moving fast but far from the ball, whereas the second situation was when they were moving slowly but near to the ball. The attention value of the attend_motion (thus the response to the motion) changed based on the speed of the motion.

Without ASMO's habituation and sensitisation learning mechanisms (i.e. without accommodating the boost value of the attend_motion module), Smokey tended to look at the fast motion although that motion was far from the ball (see Figure

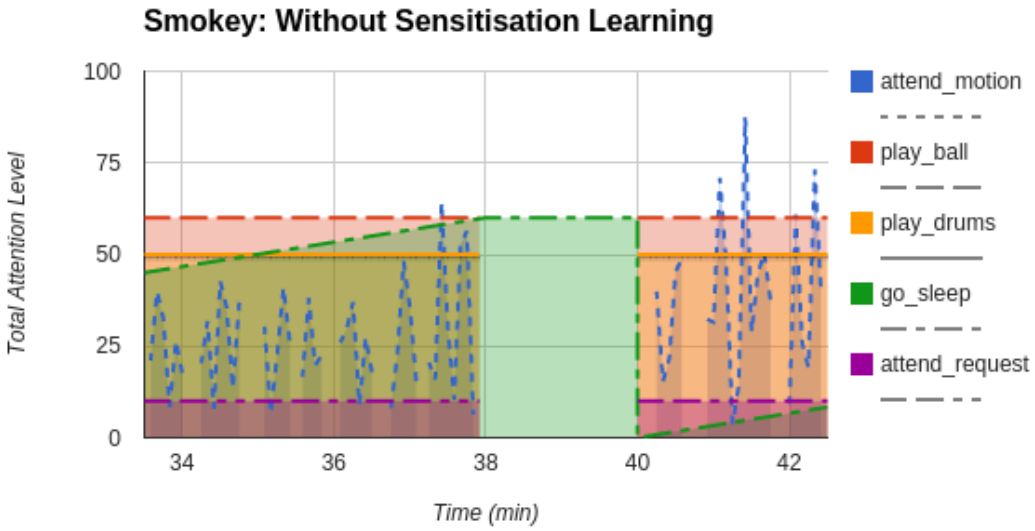


(a) Faster & Far Motion, Without Habituation Learning

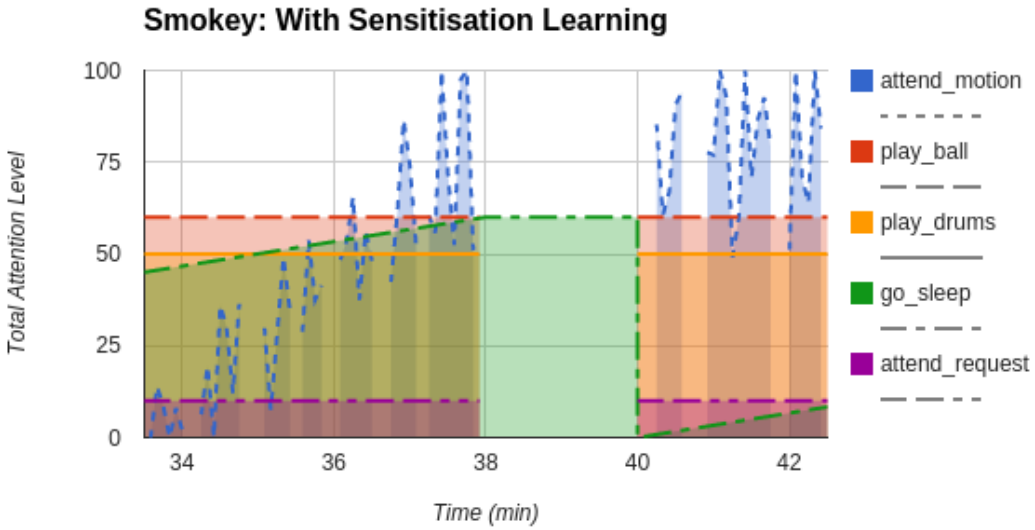


(b) Faster & Far Motion, With Habituation Learning

Figure 8.9: Habituation and Sensitisation Learning Experiment in Smokey



(c) Slower & Near Motion, Without Sensitisation Learning



(d) Slower & Near Motion, With Sensitisation Learning

Figure 8.9: Experiments of Habituation and Sensitisation Learning in Smokey

8.9a). In addition, Smokey also tended to ignore the slow motion although that motion was near to the ball (see Figure 8.9c). Any fast motion could easily distract Smokey from tracking the ball and any slow motion could easily be ignored by Smokey.

With ASMO's habituation and sensitisation learning mechanisms (i.e. accommodating the boost value of the attend_motion module), Smokey tended to ignore the motion far from the ball although that motion was fast (see Figure 8.9b). In addition, Smokey also tended to focus on the motion near to the ball although that motion was slow (see Figure 8.9d). Any fast motion did not easily distract Smokey from tracking the ball, and any slow motion was not easily ignored by Smokey. Thus, Smokey was habituated and sensitised to motion that was far from, and near to, the ball respectively.

8.2.4 ASMO's Operant Conditioning Mechanism

ASMO's operant conditioning mechanism learns to modify the boost values of non-reflex modules based on feedback received explicitly from the environment. Recall from Section 7.4.2 that it is implemented in a supervisor module and it modifies the boost value of any non-reflex module that has defined at least a *reward* function (to capture the feedback in the environment).

In the experiment, ASMO's operant conditioning mechanism was enabled to improve Smokey's decisions based on the user's feedback. It used customised policy and reward functions and the default termination function. The customised policy and reward functions are as described as follows:

- **Policy Function**

  The policy function picks a module as the trial candidate for the winner of the attention competition based on Algorithm 8.1, as described as follows:

  1. It picks the go_sleep module if the go_sleep module's total attention level is higher than the total attention levels of other modules.
  2. It picks either attend_motion, play_ball or play_drums module randomly if the go_sleep module's total attention level is less than or equal to the total attention levels of other modules and a motion is detected.

3. It picks either play_ball or play_drums module randomly if the go_sleep module's total attention level is less than or equal to the total attention levels of other modules and a motion is not detected.

```
1  Function policy():
2      am ← module(attend_motion)
3      pb ← module(play_ball)
4      pd ← module(play_drums)
5      gs ← module(go_sleep)
6      if (total_attention_level(gs) > total_attention_level(am)) or
       (total_attention_level(gs) > total_attention_level(pb)) or
       (total_attention_level(gs) > total_attention_level(pd)) then
7          candidate ← gs
8      else if motion is detected then
9          candidate ← random(am, pb, pd)
10     else
11         candidate ← random(pb, pd)
12     end
13 end
```

Algorithm 8.1: Policy Function Used in Smokey

- **Reward Function**

  All of the four ordinary modules (i.e. attend_motion, play_ball, play_drums and go_sleep modules) used the same reward function. The user was given two buttons to express his/her feedback, namely the *like* and *dislike* buttons. The reward function returned 2.0, -2.0 or 0.0 when the user pressed the *like* button, pressed the *dislike* button or did not press any button respectively.

As described in Section 7.4, ASMO's operant conditioning mechanism performs the two following actions to modify a candidate's boost value:

1. **Reinforce The Candidate**

   ASMO's operant conditioning mechanism will increase a candidate's boost value (i.e. reinforce) if the candidate is not supposed to win, but receives a positive feedback.

2. **Punish The Candidate**

   ASMO's operant conditioning mechanism will decrease the candidate's boost value (i.e. punish) if the candidate is supposed to win, but receives a negative feedback.

Figure 8.10 shows the results of the two experiments without and with ASMO's operant conditioning learning mechanism respectively. In the experiments, the user punished Smokey for playing ball and reinforced Smokey for playing drums.

Without ASMO's operant conditioning mechanism, the total attention level of the play_ball module was higher than the total attention level of the play_drums module (see Figure 8.10a). Thus, Smokey chose to play ball instead of to play drums when interacting with the user.

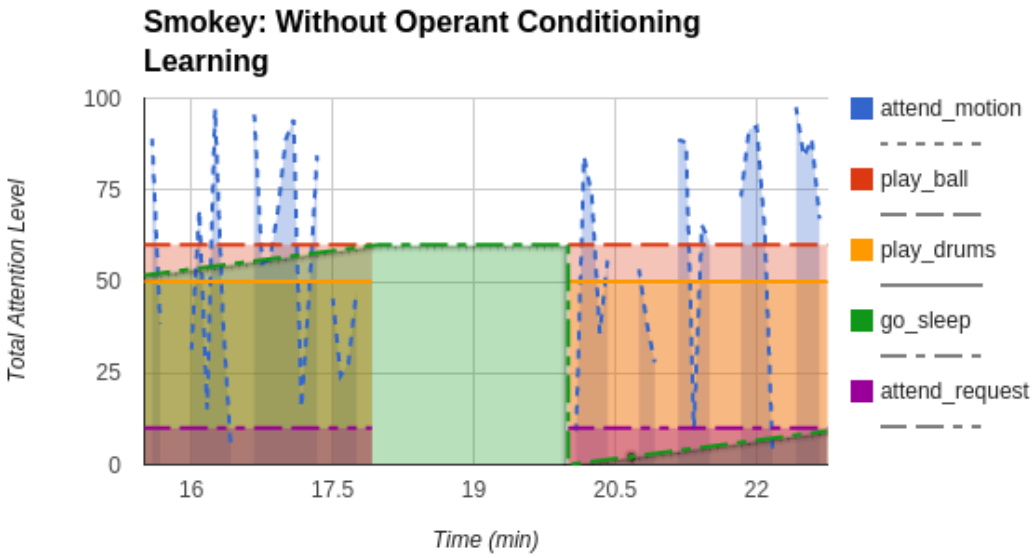
With ASMO's operant conditioning mechanism, both play_ball and play_drums modules' total attention levels were decreased and increased by their boost values as the user punished and reinforced Smokey respectively (see Figure 8.10b). As a result, the total attention level of the play_drums module was higher than the total attention level of the play_ball module. Thus, Smokey chose to play drums instead of to play ball when interacting with the user.

8.2.5 ASMO's Classical Conditioning Mechanism

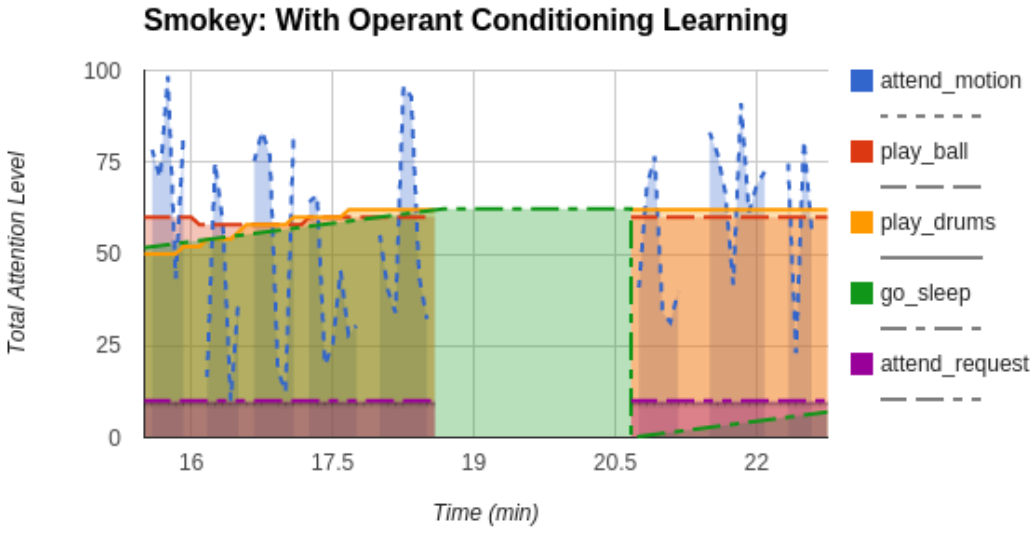
ASMO's classical conditioning mechanism learns to create associations between stimuli. Recall from Section 7.4.3 that it is implemented in a supervisor module, but it does not modify the boost value of a non-reflex module directly. Instead, it simulates the appearance of stimuli although they are not physically present, so that modules propose actions based on the simulated stimuli.

In the experiment, ASMO's classical conditioning mechanism was enabled to predict and perform users' requests without being asked whenever Smokey met the users. It learned the associations between the appearance of a user and his/her requests. It predicted a user's most likely request and inserted this predicted request into ASMO's memory every time the user appeared. It simulated the user's request although the user did not ask. As a result, it triggered the play_ball, play_drums and go_sleep modules to propose actions as if the user made an actual request.

Table 8.3 shows the requests received during the interaction with five users where



(a) Without Operant Conditioning Learning



(b) With Operant Conditioning Learning

Figure 8.10: Experiments of Operant Conditioning Learning in Smokey

-, *b*, *d* and *s* denote no request, the play ball request, the play drums request and the go to sleep request respectively. Table 8.4 shows the probability of each request that might be asked by each user given the user is seen. These probabilities were calculated based on the users' requests in Table 8.3 using the expectation maximization algorithm and Laplace smoothing with k of 1.0.

User	Requests
Anshar	b,s,d,d
Ben	d,d,d,d,d
Evelyn	-,-,-
Michelle	b
Xun	s,d,b,-,s

Table 8.3: Users' Requests

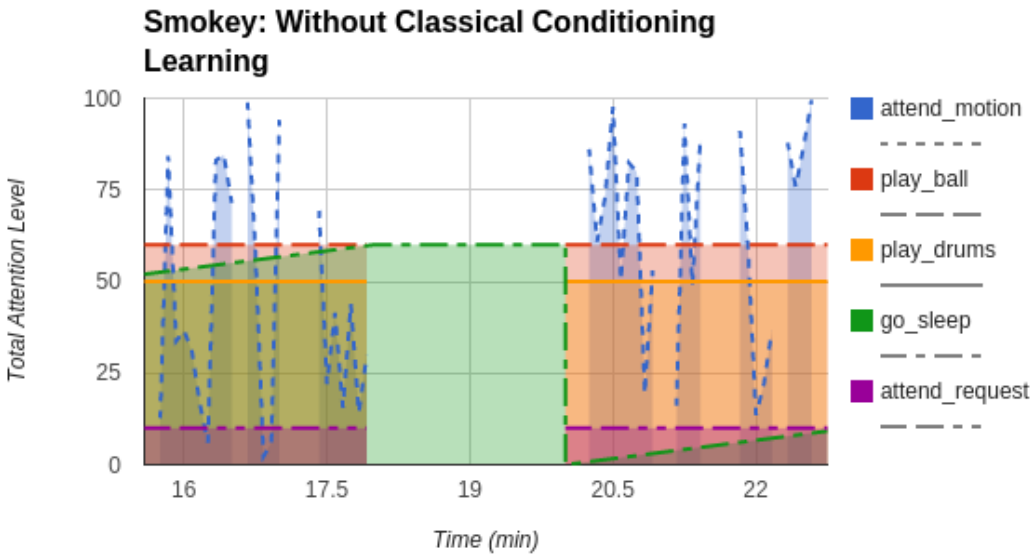
User	Probability of Request Given User is Seen			
	Play Ball	Play Drums	Go to Sleep	No Request
Anshar	0.25	0.375	0.25	0.125
Ben	0.1111	0.6666	0.1111	0.1111
Evelyn	0.1429	0.1429	0.1429	0.5714
Michelle	0.4	0.2	0.2	0.2
Xun	0.2222	0.2222	0.3333	0.2222

Table 8.4: Probability of Requests Asked by Users

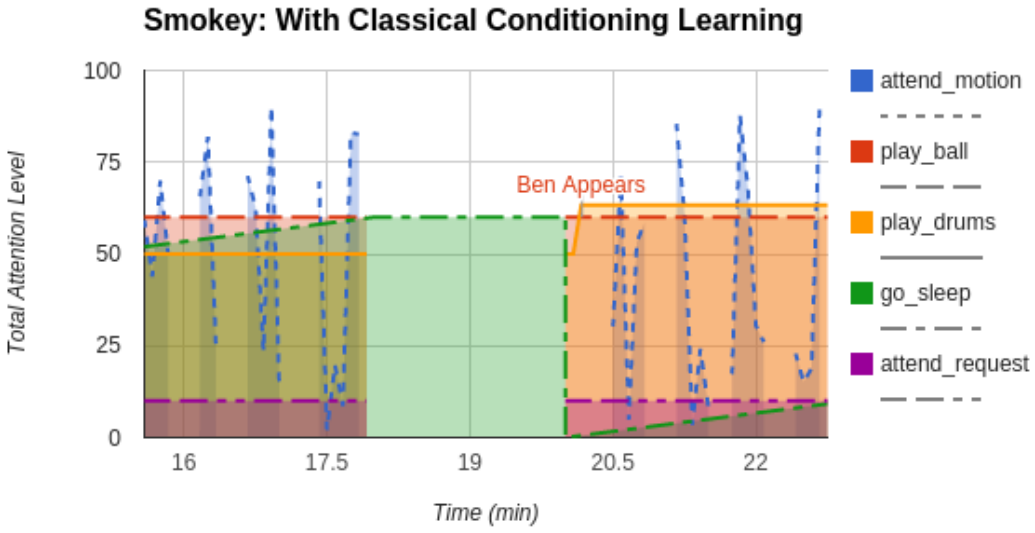
Figure 8.11 shows the results of the two experiments without and with ASMO's classical conditioning learning mechanism respectively. In the experiments, the first user preferred for Smokey to play ball rather than to play drums and the second user replaced the first user to interact with Smokey.

Without ASMO's classical conditioning mechanism, the total attention levels of the play_ball and play_drums modules did not change when Smokey saw the second user (see Figure 8.11a). Thus, Smokey still chose to play ball instead of to play drums when interacting with the second user.

With ASMO's classical conditioning mechanism, the play_ball and play_drums modules' total attention levels were increased by their boost values when Smokey saw the second user (see Figure 8.11b). The total attention level of the play_drums



(a) Without Classical Conditioning Learning



(b) With Classical Conditioning Learning

Figure 8.11: Experiments of Classical Conditioning Learning in Smokey

module was increased slightly higher than the total attention level of the play_ball module. Thus, Smokey chose to play drums instead of to play ball when interacting with the second user.

8.2.6 ASMO's Observational Learning Mechanism

ASMO's observational learning mechanism learns to modify the boost values of non-reflex modules based on labelled data. Recall from Section 7.4.4 that it is implemented in a supervisor module and it takes inputs represented by variables defined in ASMO's memory.

In the experiment, ASMO's observational learning mechanism was enabled to improve Smokey's behaviours based on how the user taught Smokey to behave. It used a neural network, which was learned based on situations in the environment to modify the boost values of attend_motion, play_ball, play_drums and go_sleep modules (see Figure 8.12).

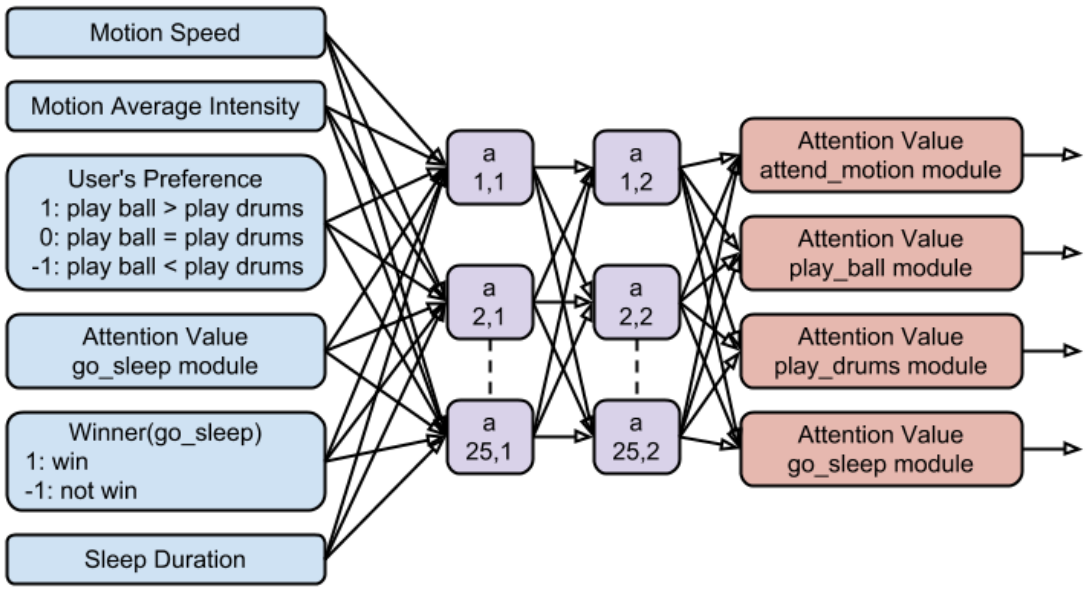


Figure 8.12: The Neural Network of The Observational Learning in Smokey Robot Companion

Figure 8.13 shows the results of the two experiments without and with ASMO's observational learning mechanism respectively. It shows that the modules' total attention levels without ASMO's observational learning mechanism were similar to the total attention levels determined with ASMO's observational learning mecha-

nism. The total attention level of the attend_motion module was different because it changed based on the motion in the environment (i.e. different motions during the two experiments). ASMO's observational learning mechanism had achieved similar behaviour without requiring the developers to hard-code the parameters.

8.3 Analysis and Discussion

The evaluations of ASMO in the RoboCup Soccer SPL standard benchmark and the Smokey robot companion problems have demonstrated that ASMO meets the aims of this dissertation. They validated my hypothesis about attention and emergence of techniques. They also demonstrated ASMO's capabilities to address the four issues identified in the research literature (see Section 3.5) and demonstrated ASMO's natural programming benefit. I have discussed the comparative analysis of existing architectures in Section 3.5. In the following, I describe the analysis of ASMO cognitive architecture:

1. **Complex and intelligent behaviours based on emergence**

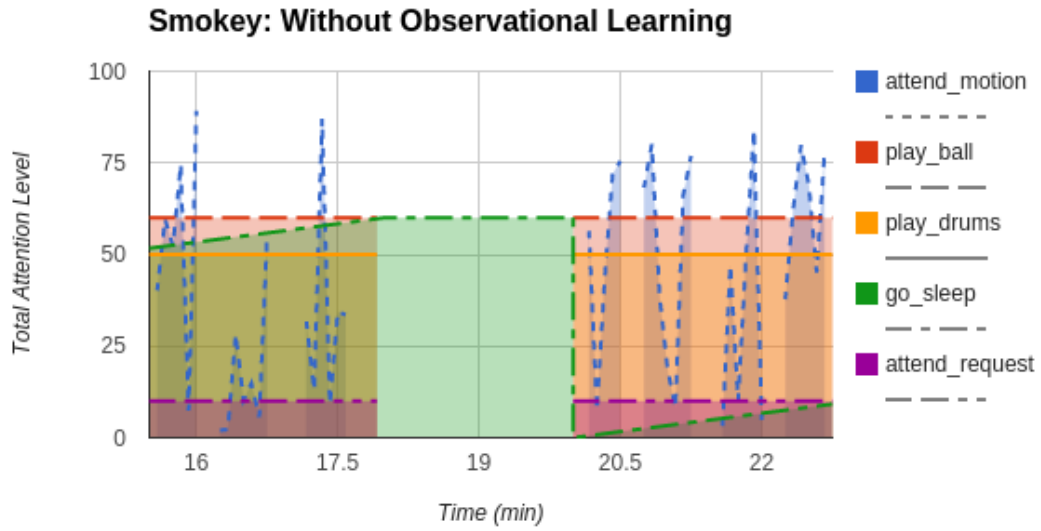
   Robots' complex and intelligent behaviours for playing soccer and accompanying users emerged from the interaction of simple modules (e.g. the modules simply demand constant or linear attention).

2. **Validation of theories of attention and emergence**

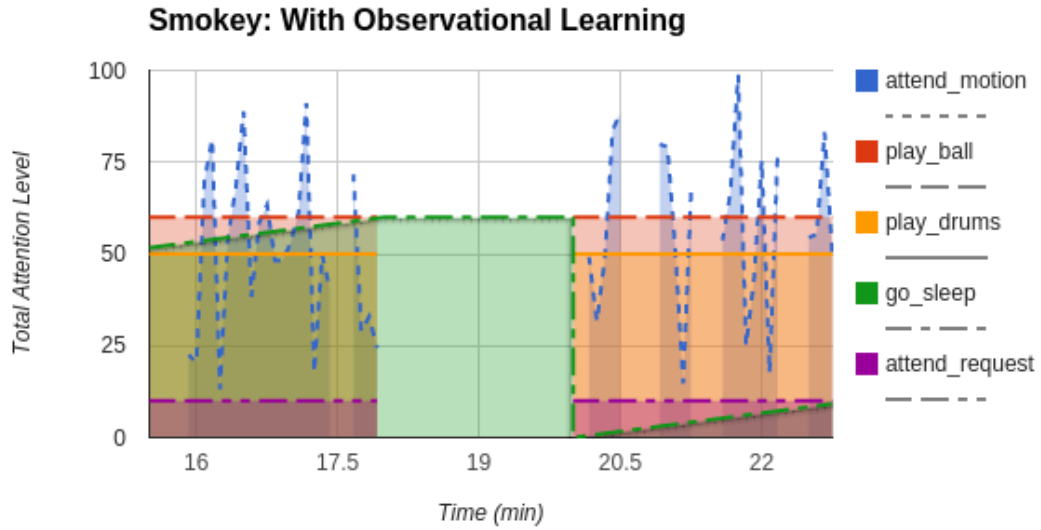
   Robots are capable of achieving complex soccer and companion tasks based on attention and the emergence of techniques. The evaluations validate the theories of attention and emergence.

3. **Resource management**

   ASMO's attention mechanism runs attention competitions to select modules and to control or manage limited resources explicitly and automatically. It selects modules based on their rankings and required resources. Developers do not need to manage the resources manually. Instead, they just need to specify the resources required by each module. ASMO will manage the resources automatically by ensuring that modules and their proposed actions can only be selected if their required resources are available.



(a) Without Observational Learning



(b) With Observational Learning

Figure 8.13: Experiments of Observational Learning in Smokey

In the RoboCup Soccer SPL standard benchmark problem, ASMO automatically ensured that the chase_ball and defend_attack modules were not selected at the same time. ASMO also ensured that track_ball and track_opponent_goal modules were not selected at the same time.

In the Smokey robot companion problem, ASMO automatically ensured that at most one of the attend_motion, play_ball, play_drums and go_sleep modules was selected at any given time.

4. **Integration of independently developed techniques**

   ASMO's attention mechanism and its black-box design allow a diversity of any kind of independently developed technique to be integrated as a module without the need to know its internal algorithm, representation or mechanism.

   In the RoboCup SPL competition 2010, students and team members were required to individually develop techniques. ASMO allowed their techniques to be integrated easily as modules.

   In the Smokey robot companion problem, the attend_motion module used a neural network representation. The play_ball and play_drums modules used a priority-based or hierarchical-based approach. The go_sleep module used a logic representation. The attend_request module used a simple event-driven approach. Despite having modules that used different representations and approaches, ASMO could still integrate and orchestrate this diversity of modules through a common currency of attention.

5. **Adaptation to design changes**

   ASMO's attention mechanism and its ranking system based on an attention value and a boost value enable various techniques (i.e. modules) to be (re)integrated or (re)structured on the fly with considerably less time and effort. The effects of techniques can be increased or eliminated on the fly by changing the attention values or boost values.

   In the RoboCup SPL competition 2010, changes in game strategies were achieved simply by changing the mix of the attention and/or boost values of modules as the means to improve system performance. In addition, in *less than a day*, students and team members were able to modify the system configured for soccer matches into a system for technical challenges (i.e. with different purposes and requirements).

In the development of Smokey robot companion system, techniques (i.e. modules) were tested simply by changing their attention and/or boost values. In addition, the go_sleep module was created by modifying or adapting the track_ball module in the robot soccer system into a goal-based approach. This modification or adaptation was achieved in considerably less time and effort than creating the go_sleep module from scratch.

6. **Support to robot operation**

   ASMO supports the following necessary capabilities required by robots:

   (a) **Fast Response**

   Since ASMO runs modules independently and concurrently, modules can propose actions as soon as they are ready without having to wait for other modules. In addition, attention competitions guarantee that modules will be selected at the specified time or conditions. Both the soccer robot and Smokey have demonstrated fast responses when playing soccer and accompanying users.

   (b) **Simultaneous Multiple Goals**

   Each module in ASMO can pursue different goals. ASMO's attention mechanism selects multiple modules simultaneously in every attention competition to perform their proposed actions in order to simultaneously achieve multiple goals.

   In the RoboCup Soccer SPL standard benchmark problem, each module pursued a different goal. ASMO selected multiple modules simultaneously to govern the robot's head and legs. For example, ASMO allowed the soccer robot to simultaneously chase the ball while searching for the opponent's goal.

   The Smokey robot companion problem was not a good example to demonstrate the capability to achieve multiple goals, because all modules happened to require the same resource of Smokey's head. However, if more modules that require different resources are developed, then ASMO can select them concurrently to achieve simultaneous multiple goals.

   (c) **Fault tolerant**

   ASMO does not have a centralised perception system, reasoner, planner or executor. Instead, ASMO's modules are distributed and each module can sense the environment, process information, reason the situation,

make plans and propose actions. ASMO's independent and distributed modules allow them to continue to sense, process, reason, plan and propose despite the failure of other modules (i.e. robots will not fail but keep operating and performing tasks).

In the RoboCup Soccer SPL standard benchmark problem, the soccer robots could still chase the ball based on the chase_ball module when the defend_attack module has failed, and vice versa.

In the Smokey robot companion problem, Smokey could still play the ball based on the play_ball module when the play_drums module has failed, and vice versa.

(d) **Redundant**

Since ASMO's modules are independent and distributed, ASMO can have a redundancy system simply by duplicating modules without the need to restructure its design. In addition to a duplication, ASMO can have different modules achieving the same tasks. Two or more modules can have different types of sensing, reasoning, planning and action proposal. In order to avoid duplicate outcomes, the modules can be designed to require the same virtual resource (see Section 5.4.1). They do not cause chaotic behaviours because ASMO's attention mechanism will choose one of them (if win) to perform.

In the robot soccer system and Smokey's system, all modules were duplicated and they kept working to provide redundant functionalities.

(e) **Long-term operation**

Slowing robots down due to the accumulation of knowledge and experience is not a major issue in ASMO, because ASMO can keep the system being responsive. While waiting for some optimal modules to process enormous knowledge, other suboptimal modules can still performing their tasks in order to respond to the environment.

ASMO further supports long-term operation by not letting failed modules to crash the whole system. This capability is possible because failed modules cannot participate in the attention competition due to non-responsiveness, which result in a loss of attention and, therefore, a loss of control over the system resources. Poorly designed modules tend to lose attention and not be enacted.

In the RoboCup SPL competition 2010, a module had returned the *division by zero* error. However, it did not crash the whole robot soccer system. Instead, it was simply ignored by ASMO as it did not propose actions nor request any control over the robot's resources.

In the Smokey robot companion experiment, a module contained an infinite loop was tested. The module became unresponsive, but Smokey's behaviours were still normal as expected.

(f) **Maintainable**

In ASMO, each module is a black-box, independent, distributed, self-contained, concurrent and modular. This design allows high cohesion and low coupling modules that will make maintenance and extensibility easy. It allows modules to be maintained in isolation. Modules can be added, removed, modified or replaced without requiring the system to be off and without affecting many other modules. Furthermore, the isolation of behaviours into distributed modules will assist with the distribution of work and maintenance across a team.

In the development of the soccer system for RoboCup SPL competition 2010, members modified and tested modules in both isolation and real-time (while the robot was on). They did not depend on the work of other team members to update modules. As a result, they saved considerable time and effort. They also reported that creating high cohesion and low coupling modules in ASMO was not difficult because modules work independently and are self-contained.

In Smokey's system, the maintainability was not really explored because there were not many modules to maintain. However, ASMO's modules and designs will allow Smokey to be easy to maintain and extend.

(g) **Learning**

ASMO has various learning mechanisms to improve performance and make extension easier, namely habituation, sensitisation, operant conditioning, classical conditioning and observational learning mechanisms.

In the RoboCup SPL competition 2010, ASMO's learning mechanisms were not activated due to the lack of time to test. However, if they were activated, they could help to improve the performance of the soccer robots.

In the Smokey robot companion experiment, each ASMO's learning mechanism improved Smokey's performance in each different situation described in Page 178.

7. **Natural programming approach to robots to accelerate the development, deployment and adoption of robots**

   As demonstrated by the RoboCup soccer competition in 2010, our team used ASMO cognitive architecture to integrate various representations and mechanisms for developing a robust robot soccer system. The team had limited time — less than three months — to build a fully functional robot soccer system from scratch based on new rules (i.e. rules in 2010). New rules meant that we had to work with a new robot platform (i.e. Aldebaran Nao). As a consequence the team did not have sufficient time to build and fine-tune a new robot soccer system for a new robot platform (unlike what we did in the 2008 competition). Utilising the novel more agile ASMO cognitive architecture allowed the team to integrate various representations and mechanisms as well as to have a fully functioning robot soccer team ready for competition in six weeks of development.

Chapter 9

Conclusion

In this dissertation, I described the novel ASMO cognitive architecture that addresses the gaps identified in the research literature to accelerate the development, deployment and adoption of service robots in society.

9.1 Significance and Contributions

This dissertation studies and develops ASMO cognitive architecture to address the gaps identified in the research literature (see Section 3.5). It is significant and provides important contributions:

1. **Accelerate the development, deployment and adoption of service robots**

   Service robots have not been as widely deployed and adopted as industrial robots. In this dissertation, I have demonstrated that ASMO manages resources automatically, integrates independently developed techniques, easily adapts to design changes and supports robotic tasks. These capabilities can help to accelerate the development, deployment and adoption of service robots.

2. **Develop a more intuitive approach to programming robots**

   Conventional robots are programmed in an unnatural manner and one that is completely different to the way humans are instructed to perform tasks. ASMO provides a more intuitive approach to programming robots based on managing

their attention. This approach is more natural and easier to understand by programmers because they themselves are attention-driven agents.

3. **Test theories of attention and emergence**

   Theories of attention and emergence are traditionally difficult to test because they involve the human mind that cannot be easily and fully inspected. In this dissertation, I have implemented theories of attention and emergence and tested them using robots. I have demonstrated that robots are capable of achieving complex tasks based on attention and the emergence of techniques. This successful implementation lends evidence to support the theories.

4. **Advance the study in interdisciplinary fields**

   Many robotics studies focus on a single discipline. Studies in interdisciplinary fields are more complicated than studies in a single field. This dissertation advances the study in interdisciplinary fields pertinent to robots by taking theories from cognitive science and psychology, and implementing the theories into a system in engineering and computer science.

5. **Put cognitive science into practice**

   Cognitive science is a new emerging field. Its problem is not as well-defined as in other fields, such as engineering. It involves vague terms and definitions and also philosophical issues that are not practical. In this dissertation, I have studied and designed ASMO cognitive architecture based on the cognitive science theories and implemented ASMO into a concrete working system in robots.

6. **Provide a better understanding of cognitive architectures**

   Previous work classifies cognitive architectures into three general categories, namely top-down, bottom-up and a hybrid approach. This dissertation provides a more specific classification based on knowledge utilisation, which are lookup-based, cased-based finite state machine, priority-based or hierarchical-based, goal-based, connection-based and utility-based approach.

7. **Motivate the use of a standard benchmark problem for evaluating and sharing significant work**

   Existing cognitive architectures are difficult to compare, because they are evaluated in a variety of different problems. This dissertation motivates the use of

a standard benchmark problem to compare or evaluate cognitive architectures, share the work and measure progress. ASMO was evaluated in the RoboCup Soccer SPL standard benchmark problem, where progress is evaluated each year. In addition, Smokey robot companion problem was introduced as a basis for benchmarking in the future.

9.2 Limitation and Future Work

The work in this dissertation provides a novel way and a number of capabilities to govern agents (including robots). However, it has a limitation, which opens new questions and opportunities in the field. These research questions stem not only within the wide scope of the work, but also from its effects and potential applications:

- **Attention Value and Boost Value**

  In this dissertation, an attention value and a boost value are abstract numerical values that quantify attention. They have no absolute measurement to the quantity of human attention. In the future work, a measurement can be developed to define these values. Future work may explore what it means to have a change of 1.0 in an attention value or boost value.

  In addition, the attention value is updated in this dissertation simply based on Equation 5.1. In future it may be possible to explore other attention update equations that, for example, more closely model empirical data relating to human or animal attention.

- **Total Attention Level**

  In this dissertation, a non-reflex module's ranking is determined by its total attention level, which is simply determined by the sum of its boost value and attention value. In future work, this total attention level can be determined by more complex methods, such as weighted sum or non-linear functions.

- **Cyclic Modification of Supervisor Modules' Attention and Boost Values**

  In this dissertation, a *cyclic modification* of supervisor modules is not studied and allowed. In other words, supervisor module A is not allowed to modify

the attention and boost value of supervisor module B if B also modifies the attention and boost value of A. Any form of transitivity is also not allowed. Future work may include study of cyclic modifications and a mechanism that allows for cyclic modifications.

- **Jitter or Oscillation Prevention**

  In this dissertation, ASMO produces an attentive system that is intended to be responsive and sensitive to changes, including small changes. This responsiveness may cause the system to jitter or oscillate. For example, two modules can win an attention competition back and forth, causing the robot to not achieve its goals. Future work may incorporate hysteresis or a mechanism that can prevent or adjust the sensitivity of winning an attention competition.

- **Overcoming Reflexes**

  In this dissertation, a reflex module is simply given higher ranking than non-reflex modules. As a result, non-reflex modules cannot win an attention competition against a reflex module no matter how high they demand attention. However, humans seem to be able to perform non-reflex actions intentionally to overcome reflexes that are supposed to be triggered (i.e. non-reflex actions win over reflexes). For example, people can hold something that is hot to avoid spilling it. Future work may include further study of theories of intention, reflexes and non-reflex actions, as well as a mechanism or conditions to overcome reflexes.

- **Multi-agent Architecture**

  In this dissertation, ASMO was evaluated in a multi-agent environment (i.e. the RoboCup Soccer SPL Standard Benchmark Problem), but it was used as a single agent architecture in every robot to manage the robot' resources. Future work may include further evaluation and extension of ASMO as a multi-agent architecture (e.g. by integrating ASMO with other ASMOs) to manage multiple robots' resources.

- **Emotion and Learning in the RoboCup Soccer SPL Standard Benchmark Problem**

  In this dissertation, I only speculate how emotion and learning mechanisms will be used in the RoboCup Soccer SPL Standard Benchmark Problem, because our team did not have much time to test these mechanisms in the RoboCup

Soccer SPL competition. Future work may include the evaluation of these mechanisms in this problem.

- **Comparative Analysis in the Standard Benchmark Problem**

  In this dissertation, only ASMO was evaluated in the RoboCup Soccer SPL Standard Benchmark Problem. Existing architectures reviewed in this dissertation were not evaluated in the same problem, because they have been designed with a specific goal in mind so evaluating them in the same problem would not provide a fair and good comparison. As a result, statistical results could not be obtained for comparing ASMO with existing architectures. Thus, the comparative analysis of architectures was discussed argumentatively or descriptively, rather than statistically. Future work may include mapping the problems where existing architectures were evaluated on to the RoboCup Soccer SPL Standard Benchmark Problem in order to provide a comparative analysis statistically. Alternatively, I hope that the RoboCup Soccer SPL Standard Benchmark Problem will be adopted as the standard benchmark problem for cognitive architecture, so developers will (re)-evaluate their architectures in this problem and therefore results can be compared.

- **Theorems and Proofs**

  In this dissertation, ASMO is not defined and described formally using mathematical theorems and proofs. In addition, there is no mathematical proof of the analysis between ASMO and existing architectures. It can be difficult to replicate and compare the work without such formal theorems and proofs. Future work may include mathematical theorems and proofs to formally define and describe ASMO, as well as to compare ASMO with existing architectures.

9.3 Final Thoughts

I began my doctoral studies with an innocent ambition to create human-level intelligent robotic systems. Of course, this long-term ambition is far beyond the scope of a doctoral dissertation. Thus, I had to start from an achievable goal, cognitive architecture, which may lead to my long-term ambition.

Due to practicality and high level of abstractions in cognitive science and psychology, many robotics researchers focus on the engineering problems of robotics.

However, I feel that studying multi-disciplinary areas of engineering, cognitive science and psychology, especially in cognitive architecture, is an important step to advance the robotics technology. I hope that this dissertation contributes some measure of progress towards a better world.

Appendix A

Non-Cognitive Architectures

This appendix describes non-cognitive architectures that are similar to ASMO, namely AuRA, DAMN and EGO. They are not considered as cognitive architectures and therefore not described in Section 3.3, because they are designed for specific robotic tasks rather than general intelligent behaviours.

A.1 Autonomous Robot Architecture (AuRA)

Autonomous Robot Architecture (AuRA) [18, 23, 20, 19] was initially developed in 1986 for the purpose of governing intelligent mobile robots. It focuses on governing robots for navigation. Its design motivation was to integrate deliberative and reactive components in order to compensate for the lack of responsiveness in agents. It has been deployed in robots, such as DARPA Demo II High Mobility Multi-purpose Wheeled Vehicle (HMMWV) [23] and Nomadic Nomad 150 [23] (see Figure A.1).

AuRA has two components, namely hierarchical deliberative and reactive components. The deliberative component consists of the mission planner, the spatial reasoner and the plan sequencer that are arranged from the highest to the lowest hierarchy. The reactive component consists of the schema controller.

The mission planner provides the high-level goal and constraints of the operation. It acts as the interface between the users and the agents. The spatial reasoner (originally known as the navigator) uses cartographic knowledge stored in long-term memory to generate a sequence of navigational paths in order to achieve the goal. The plan sequencer (originally known as the pilot) translates the path generated by

(a) DARPA HMMWV [23]

(b) Nomadic Nomad 150 [23]

Figure A.1: Robots Governed by AuRA

the spatial reasoner into a set of motor actions (called *motor schema*) for execution.

The schema controller is designed based on schema theory [16] to manage, control and monitor all *motor schemas* at run time. Each motor schema can operate asynchronously and it is associated with a perceptual schema to perceive the environment. It generates a response vector to control motors in a manner analogous to the potential field method proposed by Khatib [109]. The vectors are summed and normalised before sending the result to the motors for execution.

The deliberative component is not activated once the reactive component begins to execute, unless a failure is detected in the reactive component. A typical failure is caused by a velocity of zero or time-out. When a failure is detected, the deliberative component is reactivated one hierarchy at a time from the lowest to the highest hierarchy until the problem is resolved. First, the plan sequencer is reactivated to choose a different path stored in the short-term memory. Second, if the problem is not yet resolved, the spatial reasoner is reactivated to regenerate a new sequence of path that avoids the failed path. Lastly, if the problem is still not resolved, the mission planner is reactivated to inform the operator and ask for reformulation or abandonment of the entire mission and goal.

Versatility

AuRA is not versatile. It is a specific architecture used for navigation. It does not provide a way to integrate various techniques. All tasks must be be able to be represented by a motor schema. In addition, robotic developers have to manually accommodate resources in the design of the motor schemas in order to avoid resource conflict.

Reactivity

AuRA is semi-reactive. Once a path is found, the schema controller performs the tasks by managing, controlling and monitoring motor schemas without the deliberative component being activated. However, the mission planner cannot achieve multiple goals simultaneously. The spatial reasoner cannot construct multiple sequences of paths simultaneously.

Robustness

AuRA is robust. Motor schemas will keep sending vectors to the motors (i.e. perform actions) when there is a failure in other motor schemas since they are independent. In addition, AuRA can exploit redundancy because each motor schema can perceive the environment and control actions asynchronously. AuRA's control is decentralised. Furthermore, AuRA will not slow robots down because a slow schema can be treated as a failed schema that does not affect other schemas (i.e. other schemas will keep sending vectors to the motors).

Extensibility

AuRA is semi-extensible. Motor schema can be added or changed with less effects to other motor schemas because they are normalised before the result is sent to the motors. However, AuRA does not have learning mechanisms to improve performance and make the extension easy.

A.2 Distributed Architecture for Mobile Navigation (DAMN)

Distributed Architecture for Mobile Navigation (DAMN) [186] was developed in 1997 for mobile robot navigation. Its design motivation is to integrate various distributed, independent, asynchronous systems of different capabilities on mobile robots.

DAMN consists of independent, distributed and asynchronous modules that handle the robot's behaviours to achieve tasks. Several modules concurrently suggest opinions to influence decision making. DAMN does not choose an action based on the average of the opinions, instead it chooses an action that either satisfies all the opinions, has the most positive opinions or has the most expected positive opinions.

Currently, there are four action selection mechanisms in DAMN used to integrate opinions:

1. Constraint Arbitration

   In constraint arbitration, each module suggests an opinion in terms of an expected maximum value (i.e. constraint). DAMN will choose the minimum value among these maximum values to control agents' actuators (i.e. min-max function to satisfy all the maximum values) (see Figure A.2).

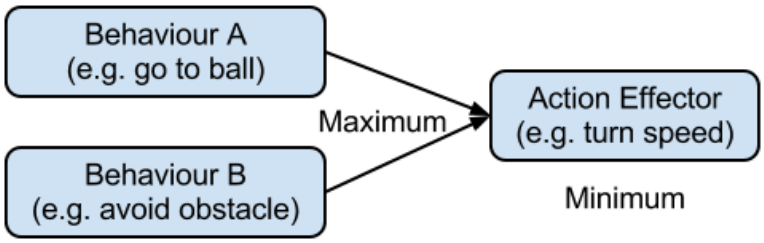


Figure A.2: DAMN Constraint Arbitration [186]

2. Actuation Arbitration

   In actuation arbitration, each module suggests an opinion by voting an actuator command. DAMN will choose and perform the actuator command with the highest positive vote (see Figure A.3).

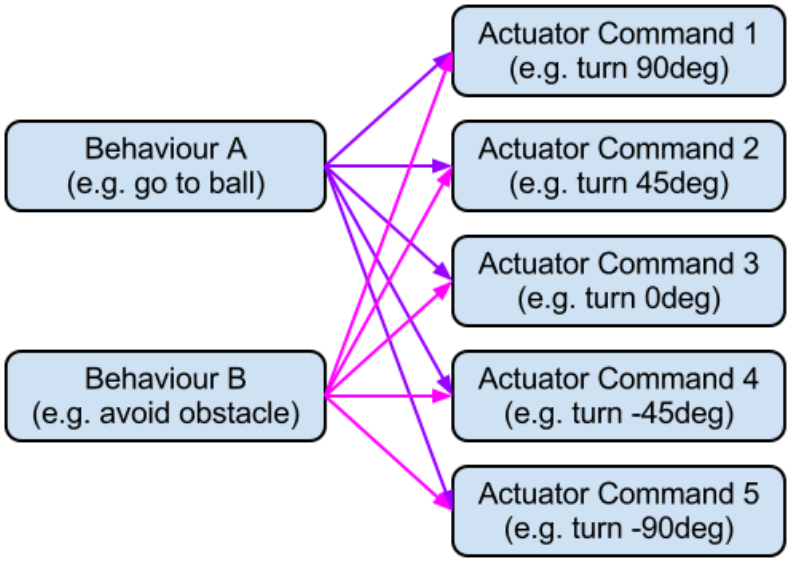


Figure A.3: DAMN Actuation Arbitration [186]

3. Effect Arbitration

   In effect arbitration, each module suggests an opinion by voting for an effect of an action (i.e. a state). DAMN will choose and perform the action with the highest positive vote on the state (see Figure A.4). Modules will vote for the effect of actions instead of the direct control of the actuators as in actuation arbitration.

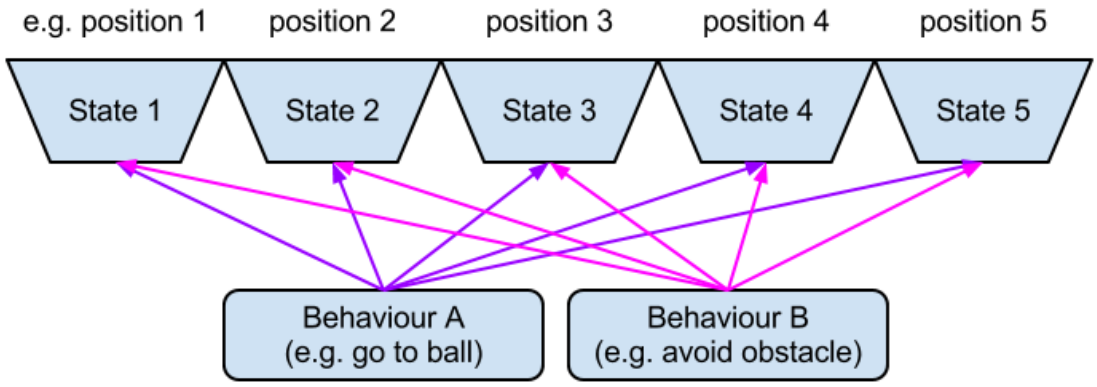


Figure A.4: DAMN Effect Arbitration [186]

4. Utility Fusion

   Utility fusion was proposed in DAMN architecture as the solution to the limitations that were observed in constraint, actuation and effect arbitration mechanisms. It merely refers to the utility theory mechanism. In utility fusion, opinions are provided in terms of the utility or the desirability of outcomes (i.e. states). DAMN will calculate the utility for each action and will choose and perform the action that has the highest utility (i.e. maximum expected utility) (see Equation A.1).

$$Decision \leftarrow \arg\max_{a} \underbrace{\sum_{s'} P(s'|a,k)U(s')}_{\text{Expected Utility}} \tag{A.1}$$

   Where:
   $EU(a|k)$ is the expected utility of action a given knowledge or evidence k
   $P(s'|a,k)$ is the probability of outcome s' given action a and knowledge k
   $U(s')$ is the utility value of outcome s'

Versatility

DAMN is not versatile. It is a specific architecture designed for mobile robot navigation. Although a voting mechanism can increase versatility if used properly, DAMN's voting mechanisms are not versatile because they require modules to provide an opinion for *every* candidate. In addition, robotic developers have to manually design the modules to avoid resource conflict.

Reactivity

DAMN is reactive. Modules can suggest opinions to control actuators directly through the arbitration and fusion mechanisms. In addition, modules can suggest opinions concurrently in order to achieve multiple goals.

Robustness

DAMN is robust. A failure in a module does not cause the whole robot to fail since modules are distributed, independent and asynchronous. The constraint arbitration mechanism allows modules to have redundancy of the processing functions without altering the minimum value, despite the application of this mechanism being limited. DAMN's control is decentralised and distributed. In addition, robots will not slow down as they accumulate experience because a slow module can be treated as a failed module. Other modules can still continue suggesting opinions in order to perform tasks.

Extensibility

DAMN is semi-extensible. Modules can be added or modified without affecting other modules since modules are distributed, independent and asynchronous. However, DAMN does not have a learning mechanism to improve performance and make extension easy.

A.3 Emotionally Grounded (EGO) Architecture

Emotionally Grounded (EGO) Architecture [195, 77, 76] was initially developed in 2001 [76] and later improved in 2003 [77] for governing entertainment robots, which are Sony SDR-4X (or simply called QRIO) [77] and Sony AIBO ERS-110 [210] (see Figure A.5). Its design motivation is to integrate various technologies and advanced technical features, such as real-time motion control, face recognition and speech recognition.

(a) Sony SDR-4X (QRIO) [77]

(b) Sony AIBO [210]

Figure A.5: Robots Governed by EGO

EGO is designed based on homeostatic regulatory mechanisms from the ethology literature. In these mechanisms, it activates behaviour modules (or just called modules) based on external stimuli and internal variables in order to regulate and maintain internal variables to be within proper ranges. EGO consists of five parts (see Figure A.6):

1. **Perception**

   The perception part contains three channels, namely visual, audio and tactile. The main functions of the visual channel are to detect a floor to walk while avoiding obstacles and to detect and identify human faces. The audio channel has a speech processing function that includes sound direction estimation, sound event identification and Large Vocabulary Continuous Speech Recognition (LVCSR). The tactile channel processes and identifies the signals from touch sensors into the several types of touching, such as 'hit' and 'pat'.

2. **Memory**

   The memory part contains two types of memory, namely a short-term memory (STM) and a long-term memory (LTM). The STM stores the result of speech, face and object recognition. It uses this information and kinematics to compute the direction of a sound based on multiple microphone localisation, the direction and the approximate distance to a face based on a stereo vision (assuming that the face size is roughly known) and the relative positions to the

detected objects (e.g. face and ball). It memorises these positions so that it can locate objects that are outside the robot's limited field of view.

The LTM is further divided into two types, namely associative memory and frame-type (symbol) memory. The associative memory is implemented in neural networks to associate a user with the information in STM (i.e. to remember an identified face, an identified voice and an acquired name). Using the associative memory, a user can be identified either from his/her face or voice alone. The frame-type symbol memory is used to remember details of users, such as birthday and favourite items. Using the frame-type symbol memory, the robot can behave differently towards each individual user.

3. **Internal State Model**

   Internal state model contains various internal state variables that change over time depending on incoming external stimuli.

4. **Behaviour Control**

   The behaviour control part contains three types of modules, namely reflexive, deliberative and situated. A reflexive module depends on the robot's mechanical configuration to provide a rapid response to external stimuli. A deliberative module performs computationally heavy tasks, such as path planning. A situated module performs two functions, namely action and monitor. The action function uses a state machine to decide action commands based on the robot's state and inputs (i.e. it executes a behaviour). The monitor function evaluates external stimuli and internal variables, and also calculates the module's *activation level*, which is a value indicating how much a given behaviour is relevant in the current situation. This function is periodically called by the behaviour control part. Some parameters of this function can be adjusted to make several different personalities. For example, giving more weight on the internal drives can result in behaviour selection that tends to satisfy internal motivation, which seems like a 'selfish' personality.

5. **Motion Control**

   The motion control part provides a real-time integrated adaptive motion control and a real-time gait pattern generation. These technologies enable QRIO to walk on unbalanced terrain, to adaptively move against an external force, to fall down with shock absorbing motions and to recover motions.

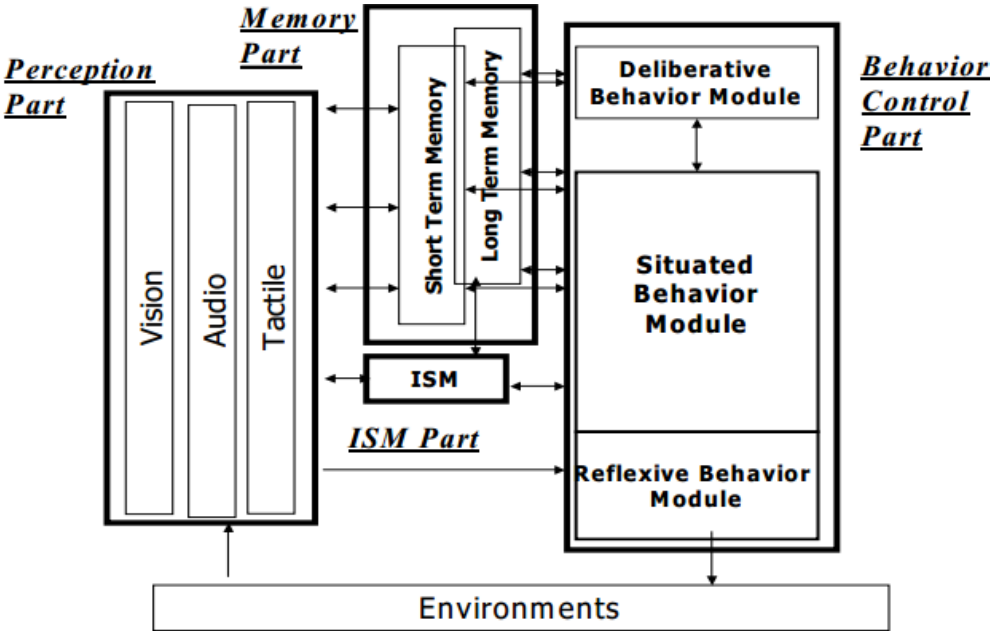


Figure A.6: EGO Architecture

EGO employs a hierarchical tree structure to manage resources and to coordinate, organise and connect modules. In this tree structure, developers manually decompose a task into multiple sub-tasks, which are then handled by individual modules (see an example in Figure A.7). Modules that have the same parent in the tree will share the same target. For example, all of the search_ball, approach_ball and kick_ball modules have the same parent of the soccer module to play a ball, so they share the same target of ball. Multiple modules can be executed concurrently if their targets are different from each other. In this case, resources are managed in the tree structure through the modules' targets.

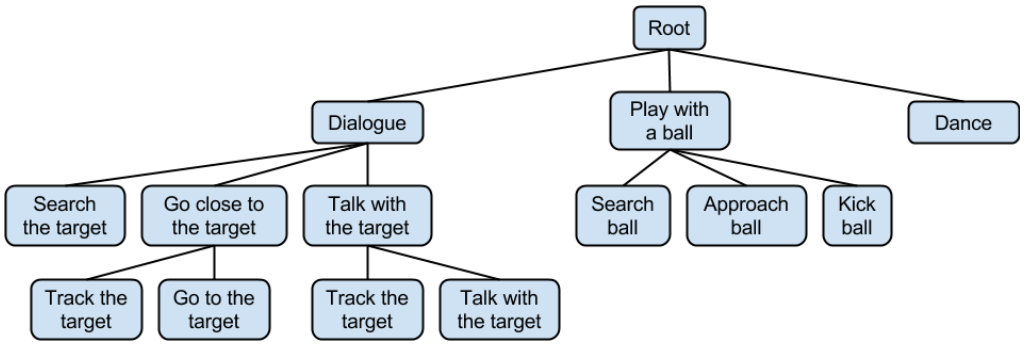


Figure A.7: An Example of A Module Tree Structure in EGO Architecture

EGO selects modules using a competition of their activation levels. The behaviour control part periodically calls each module's monitor function to evaluate

the module's activation level. During this process, the module also provides information regarding its required resources (such as head, arm, speaker, etc). EGO will then select modules based on their activation levels and required resources as described as follows. It first selects the module with the highest activation level. If there are still remaining resources available, another competition is conducted between the modules to select another module. This process is repeated until there is no remaining resource available. Once a module is selected, it is given permission to execute its action function, which executes the behaviour implemented in the module as a state machine (i.e. executing a state machine behaviour). As a result of this competition, multiple modules can be executed with different resources in parallel. In the case where there is no resource conflict among the modules, all modules are given permission to execute concurrently.

Both of the tree structure and the competition mechanisms are introduced to manage resources, but their capabilities have not been evaluated [77, p. 966]. In addition, they have an exception that is not described in the papers: they allow module to execute concurrently only if the modules do not share the same target and require different resources. This means that modules that share the same target cannot be executed concurrently despite that they require different resources.

Versatility

EGO is not versatile. It is a specific architecture designed for entertainment robotic tasks. Although a resource management through competition can increase versatility if it is used properly, EGO's resource management is not versatile because it requires modules to be manually decomposed into submodules in a tree structure, which can be difficult. In addition, behaviours in EGO's modules are limited to use a state machine.

Reactivity

EGO is reactive. Modules can be selected concurrently as long as they require different resources and share different targets.

Robustness

EGO is semi-robust. Its control is decentralised and distributed. However, a failure in a higher hierarchy module in a tree will cause lower hierarchy modules in the same branch to fail. EGO's hierarchical tree structure does not allow modules to have redundancy processing functions if they share the

same target. In addition, robots may slow down as they accumulate experience because a slow module will affect lower hierarchy modules in the tree.

Extensibility

EGO is semi-extensible. Modules can be added or modified without affecting other modules since modules are distributed, independent and asynchronous. However, EGO does not have a learning mechanism to improve performance and make extension easy.

Bibliography

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, New York, NY, USA, 2004. ACM.

[2] Abs survey of disability, ageing and carers 2009 (cat no. 4430.0). Australian Bureau of Statistics, 2009.

[3] Act-r: frequently asked questions list. `http://acs.ist.psu.edu/projects/act-r-faq/act-r-faq.html`, Jan 2012.

[4] Adept mobilerobots pioneer p3-dx. `http://www.mobilerobots.com/researchrobots/pioneerp3dx.aspx`.

[5] C.E. Agüero, J.M. Cañas, F. Martín, and E. Perdices. Behavior-based iterative component architecture for soccer applications with the nao humanoid. In *Proceedings of the 5th Workshop on Humanoid Soccer Robots*, pages 29–34, Nashville, USA, Dec 2010.

[6] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An architecture for autonomy. *The International Journal of Robotics Research*, 17(4):315–337, 1998.

[7] Aldebaran nao. `http://www.aldebaran-robotics.com`.

[8] Christopher J. Anderson. The psychology of doing nothing: Forms of decision avoidance result from reason and emotion. *Psychological Bulletin*, 129:139–167, 2003.

[9] John Anderson. *Cognitive psychology and its implications*. Worth Publishers, 7 edition, 2009.

[10] John R. Anderson and Christian Lebiere. The newell test for a theory of cognition. *Behavioral and Brain Sciences*, 26(05):587–601, 2003.

[11] John Robert Anderson. *The architecture of cognition*. Cognitive science series. Lawrence Erlbaum Associates, Inc, Hillsdale, NJ, England, 1983.

[12] John Robert Anderson. *How can the human mind occur in the physical universe?* Oxford Series on Cognitive Models and Architectures. Oxford University Press, USA, 2007.

[13] J.R. Anderson. *Language, memory, and thought*. The Experimental Psychology Series. Taylor & Francis, 1976.

[14] J.R. Anderson and G.H. Bower. *Human associative memory*. Winston & Sons, 1973.

[15] P. W. Anderson. More is different. *Science*, 177(4047):393–396, 1972.

[16] Michael Arbib. Schema theory. In S Shapiro, editor, *The Encyclopedia of Artificial Intelligence*, pages 1427–1443. Wiley Interscience, New York, NY, 2 edition, 1992.

[17] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[18] R.C. Arkin. *Behavior-based robotics*. MIT Press, 1998.

[19] Ronald C. Arkin. Path planning for a vision-based autonomous robot. In Nelson Marquina and William J. Wolfe, editors, *Proceedings of the SPIE Conference on Mobile Robots*, volume 727, pages 240–250, Cambridge, MA, Oct 1987.

[20] Ronald Craig Arkin. *Towards Cosmopolitan Robots: Intelligent Navigation in Extended Man-Made Environments*. Ph.d. dissertation, University of Massachusetts, 1987.

[21] Magda B. Arnold. *Emotion and personality*. Columbia University Press, New York, NY, USA, 1960.

[22] Jayen Ashar, David Claridge, Brad Hall, Bernhard Hengst, Hung Nguyen, Maurice Pagnucco, Adrian Ratter, Stuart Robinson, Claude Sammut, Benjamin Vance, Brock White, and Yanjin Zhu. Robocup standard platform league - runswift 2010. In Gordon Wyeth and Ben Upcroft, editors, *Proceedings of the 2010 Australasian Conference on Robotics & Automation*, Brisbane, Australia, Dec 2010.

[23] T. Balch and R.C. Arkin. Behavior-based formation control for multirobot teams. *Robotics and Automation, IEEE Transactions on*, 14(6):926–939, 1998.

[24] Christian Balkenius. Generalization in instrumental learning. In *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 305–314, 1996.

[25] Christian Balkenius. Attention, habituation and conditioning: Toward a computational model. *Cognitive Science Quarterly*, 1(2):171–204, 2000.

[26] A. Bandura. *Social learning theory*. Prentice-Hall series in social learning theory. Prentice Hall, 1977.

[27] Philip Bard. Emotion: I. the neuro-humoral basis of emotional reactions. In Carl Murchison, editor, *A handbook of general experimental psychology*, pages 264–311. Clark University Press, Worcester, MA, USA, 1934.

[28] A. Beck, L. Cañamero, and K.A. Bard. Towards an affect space for robots to display emotional body language. In *RO-MAN, 2010 IEEE*, pages 464–469, Sept 2010.

[29] Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing multi-agent systems with JADE*. Wiley, 2007.

[30] José Luis Bermúdez. *Cognitive science: an introduction to the science of the mind*. Cambridge University Press, 2010.

[31] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. Robot programming by demonstration. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 1371–1394. Springer Berlin Heidelberg, 2008.

[32] Aude Billard and Maja J. Matarić. Learning human arm movements by imitation:: Evaluation of a biologically inspired connectionist architecture. *Robotics and Autonomous Systems*, 37(2–3):145–160, 2001.

[33] C.M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[34] R. Peter Bonasso, David Kortenkamp, David P. Miller, and Marc Slack. Experiences with an architecture for intelligent, reactive agents. In Michael Wooldridge, Jörg P. Müller, and Milind Tambe, editors, *Intelligent Agents II Agent Theories, Architectures, and Languages*, volume 1037 of *Lecture Notes in Computer Science*, pages 187–202. Springer Berlin Heidelberg, 1996.

[35] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. Cp-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements. *Artificial Intelligence Research*, 21:135–191, 2004.

[36] Nick Braisby and Angus Gellatly. *Cognitive psychology*. Oxford University Press, 2012.

[37] M. Bratman. *Intention, plans, and practical reason*. Harvard University Press, 1987.

[38] Cynthia Breazeal. Emotion and sociable humanoid robots. *International Journal of Human-Computer Studies*, 59(1–2):119–155, 2003.

[39] Cynthia L. Breazeal. *Sociable machines: expressive social exchange between humans and robots*. Ph.d. dissertation, Massachusetts Institute of Technology, Boston, USA, 2000.

[40] D.E. Broadbent. *Perception and communication*. Pergamon Press, 1958.

[41] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47(1–3):139–159, Jan 1991.

[42] Rodney Allen Brooks. A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, 2(1):14–23, Mar 1986.

[43] Joanna Joy Bryson. *Intelligence by design : principles of modularity and coordination for engineering complex adaptive agents*. Ph.d. dissertation, Massachusetts Institute of Technology, 2001.

[44] P Busetta, R Rönnquist, A Hodgson, and A Lucas. Jack intelligent agents - components for intelligent agents in java. *AgentLink News*, 2:2–5, 1999.

[45] S. Calinon, F. D'halluin, E.L. Sauser, D.G. Caldwell, and A.G. Billard. Learning and reproduction of gestures by imitation. *Robotics Automation Magazine, IEEE*, 17(2):44–54, June 2010.

[46] Sylvain Calinon. *Robot Programming by Demonstration: A Probabilistic Approach*. EPFL Press, 2009.

[47] Walter B. Cannon. The james–lange theory of emotions: A critical examination and an alternative theory. *The American Journal of Psychology*, 39:106–124, 1927.

[48] Nicholas L. Cassimatis. *Polyscheme : a cognitive architecture for intergrating multiple representation and inference schemes*. Ph.d. dissertation, Massachusetts Institute of Technology, 2002.

[49] P. Chance. *Learning and Behavior: Active Learning Edition*. PSY 361 Learning Series. Cengage Learning, 2008.

[50] Paul Chance. *Learning and behavior*. Wadsworth, 2013.

[51] Carolina Chang. Improving hallway navigation in mobile robots with sensor habituation. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 5, pages 143–147, 2000.

[52] C. Chao, M. Cakmak, and A.L. Thomaz. Towards grounding concepts for transfer in goal learning from demonstration. In *Development and Learning (ICDL), 2011 IEEE International Conference on*, volume 2, pages 1–6, Aug 2011.

[53] E C Cherry. Some experiments on the recognition of speech, with one and with two ears. *Journal of the Acoustical Society of America*, 25(5):975–979, 1953.

[54] Cristina Conati. Probabilistic assessment of user's emotions in educational games. *Applied Artificial Intelligence*, 16(7–8):555–575, 2002.

[55] Cristina Conati and Heather Maclaren. Empirically building and evaluating a probabilistic model of user affect. *User Modeling and User-Adapted Interaction*, 19(3):267–303, 2009.

[56] André Cyr and Mounir Boukadoum. Habituation: a non-associative learning rule design for spiking neurons and an autonomous mobile robots implementation. *Bioinspiration & Biomimetics*, 8(1):016007, 2013.

[57] André Cyr, Mounir Boukadoum, and Frédéric Thériault. Operant conditioning: A minimal components requirement in artificial spiking neurons designed for bio-inspired robot's controller. *Frontiers in Neurorobotics*, 8(21), 2014.

[58] J. Daintith, T.E.F. Wright, and Oxford University Press, editors. *Oxford dictionary of computing*. Oxford paperback reference. Oxford University Press, 2008.

[59] Robert I Damper, Richard LB French, and Tom W Scutt. Arbib: an autonomous robot based on inspirations from biology. *Robotics and Autonomous systems*, 31(4):247–274, 2000.

[60] J A Deutsch and D Deutsch. Attention: some theoretical considerations. *Psychological Review*, 70(1):51–60, 1963.

[61] M Bernardine Dias. *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*. Ph.d. dissertation, Carnegie Mellon University, Pittsburgh, USA, 2004.

[62] Mark d'Inverno, David Kinny, Michael Luck, and Michael Wooldridge. A formal specification of dmars. In MunindarP. Singh, Anand Rao, and MichaelJ. Wooldridge, editors, *Intelligent Agents IV Agent Theories, Architectures, and Languages*, volume 1365 of *Lecture Notes in Computer Science*, pages 155–176. Springer Berlin Heidelberg, 1998.

[63] R. J. Dolan. Emotion, cognition, and behavior. *Science*, 298(5596):1191–1194, 2002.

[64] A. Drewnowski. Taste preferences and food intake. *Annual Review of Nutrition*, 17(1):237–253, 1997.

[65] E.M. Eisenstein and D. Eisenstein. A behavioral homeostasis theory of habituation and sensitization: Ii. further developments and predictions. *Reviews in the Neurosciences*, 17(5):533–558, 2006.

[66] Paul Ekman. Universals and cultural differences in facial expressions of emotion. *Nebraska Symposium on Motivation*, 19:207–283, 1971.

[67] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, Jun 2003.

[68] M.W. Eysenck. *Principles of cognitive psychology*. Principles of Psychology Series. Taylor & Francis Group, 2001.

[69] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. Ph.d. dissertation, University of California, Irvine, 2000.

[70] Richard E. Fikes and Nils J. Nilsson. Strips: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.

[71] Jerry A. Fodor and Zenon W. Pylyshyn. Connectionism and cognitive architecture: a critical analysis. *Cognition*, 28(1–2):3–71, 1988.

[72] Stan Franklin and Art Graesser. Is it an agent, or just a program?: a taxonomy for autonomous agents. In Jörg Müller, Michael Wooldridge, and Nicholas Jennings, editors, *Intelligent Agents III Agent Theories, Architectures, and Languages*, volume 1193 of *Lecture Notes in Computer Science*, pages 21–35. Springer Berlin / Heidelberg, 1997.

[73] Barbara L. Fredrickson and Christine Branigan. Positive emotions broaden the scope of attention and thought-action repertoires. *Cognition & Emotion*, 19(3):313–332, 2005.

[74] Jay Friedenberg and Gordon Silverman. *Cognitive Science: an introduction to the study of mind*. SAGE Publications, 2011.

[75] Simone Frintrop, Erich Rome, and Henrik I. Christensen. Computational visual attention systems and their cognitive foundations: A survey. *ACM Trans. Appl. Percept.*, 7(1):6:1–6:39, Jan 2010.

[76] M. Fujita, R. Hasegawa, C.G. Tsuyoshi Takagi, J. Yokono, and H. Shimomura. An autonomous robot that eats information via interaction with humans and environments. In *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, pages 383–389, 2001.

[77] M. Fujita, Y. Kuroki, T. Ishida, and T.T. Doi. Autonomous behavior control architecture of entertainment humanoid robot sdr-4x. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 960–967, Oct 2003.

[78] Timothy Andrew Furze. *The application of classical conditioning to the machine learning of a commonsense knowledge of visual events*. Ph.d. dissertation, University of Leeds, United Kingdom, 2013.

[79] David Garlan and Mary Shaw. An introduction to software architecture. In Vincenzo Ambriola and Genoveffa Tortora, editors, *Advances in software engineering and knowledge engineering*, volume 2 of *Software Engineering and Knowledge Engineering*, pages 433–460. World Scientific, 1993.

[80] Erann Gat. Integrating reaction and planning in a heterogeneous asynchronous architecture for mobile robot navigation. *ACM SIGART Bulletin*, 2(4):70–74, Aug 1991.

[81] Erann Gat. Three-layer architectures. In David Kortenkamp, R. Peter Bonasso, and Robin Murphy, editors, *Artificial intelligence and mobile robots*, pages 195–210. MIT Press, Cambridge, MA, USA, 1998.

[82] P. Gaudiano and C. Chang. Adaptive obstacle avoidance with a neural network for operant conditioning: experiments with real robots. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 13–18, Jul 1997.

[83] David Gelernter. Generative communication in linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, Jan 1985.

[84] M.P. Georgeff and A.L. Lansky. Procedural knowledge. *Proceedings of the IEEE*, 74(10):1383–1398, 1986.

[85] M.P. Georgeff and A.L. Lansky. Reactive reasoning and planning. In *Proceedings of the sixth National conference on Artificial intelligence*, volume 2, pages 677–682, 1987.

[86] B.P. Gerkey and M.J. Mataric. Sold!: auction methods for multirobot coordination. *Robotics and Automation, IEEE Transactions on*, 18(5):758–768, Oct 2002.

[87] J. A. Gray and A. A. I. Wedderburn. Shorter articles and notes grouping strategies with simultaneous stimuli. *Quarterly Journal of Experimental Psychology*, 12(3):180–184, 1960.

[88] Stephen Grossberg. On the dynamics of operant conditioning. *Theoretical Biology*, 33(2):225–255, 1971.

[89] Philip M Groves and Richard F Thompson. Habituation: a dual-process theory. *Psychological review*, 77(5):419, 1970.

[90] Peter Gärdenfors. *How homo became sapiens: on the evolution of thinking*. Oxford University Press, USA, 2003.

[91] Peter Gärdenfors. *Conceptual spaces: the geometry of thought*. Bradford Books, 2004.

[92] Peter Gärdenfors and Mary-Anne Williams. Multi-agent communication, planning, and collaboration based on perceptions, conceptions, and simulations. In Andrea C. Schalley and Drew Khlentzos, editors, *Mental states: evolution, function, nature*, volume 1 of *Studies in language companion series*, chapter 6, pages 95–122. John Benjamins Publishing Company, 2007.

[93] K.A. Hambuchen. *Multi-modal attention and binding using a sensory EgoSphere*. Ph.d. dissertation, Vanderbilt University, Nashville, TN, USA, May 2004.

[94] Scott D. Hanford, Oranuj Janrathitikarn, and Lyle N. Long. Control of mobile robots using the soar cognitive architecture. *Aerospace computing, information, and communication*, 6:1–47, 2009.

[95] R. Hastie. Problems for judgement and decision making. *Annual Review of Psychology*, 52(1):653–683, 2001.

[96] R. Hastie and R.M. Dawes. *Rational Choice in an Uncertain World: The Psychology of Judgment and Decision Making*. Sage, 2009.

[97] Barbara Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–321, 1985.

[98] A. Helsinger, M. Thome, and T. Wright. Cougaar: a scalable, distributed multi-agent architecture. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 2, pages 1910–1917, 2004.

[99] Feng-Hsiung Hsu. Ibm's deep blue chess grandmaster chips. *Micro, IEEE*, 19(2):70–81, Mar/Apr 1999.

[100] Systems and software engineering – architecture description. ISO/IEC/IEEE 42010:2011, 2011.

[101] Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203, 2001.

[102] William James. What is an emotion? *Mind*, 9(34):188–205, Apr 1884.

[103] William James. *The principle of psychology*, volume 1. Holt, New York, 1890.

[104] Tim M. Jones. *Artificial intelligence: a systems approach*. Jones & Bartlett Learning, 2008.

[105] Leslie Pack Kaelbling and Stanley J. Rosenschein. Action and planning in embedded agents. *Robotics and Autonomous Systems*, 6(1–2):35–48, 1990. Designing Autonomous Agents.

[106] D. Kahneman. *Attention and effort*. Prentice-Hall series in experimental psychology. Prentice-Hall, 1973.

[107] Ronald Thomas Kellogg. *Fundamentals of cognitive psychology*. SAGE Publications, 2011.

[108] S.M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *Robotics, IEEE Transactions on*, 27(5):943–957, Oct 2011.

[109] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[110] Hyoung-Rock Kim and Dong-Soo Kwon. Computational model of emotion generation for human–robot interaction based on the cognitive appraisal theory. *Intelligent & Robotic Systems*, 60(2):263–283, 2010.

[111] Kang Geon Kim, Ji-Yong Lee, Dongkyu Choi, Jung-Min Park, and Bum-Jae You. Autonomous task execution of a humanoid robot using a cognitive model. In *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*, pages 405–410, Dec 2010.

[112] Rachel Kirby, Jodi Forlizzi, and Reid Simmons. Affective social robots. *Robotics and Autonomous Systems*, 58(3):322–332, 2010.

[113] T. Kishi, T. Kojima, N. Endo, M. Destephe, T. Otani, L. Jamone, P. Kryczka, G. Trovato, K. Hashimoto, S. Cosentino, and A. Takanishi. Impression survey of the emotion expression humanoid robot with mental model based dynamic emotions. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1663–1668, May 2013.

[114] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, and Hitoshi Matsubara. Robocup: A challenge problem for ai. *AI Magazine*, 18(1):73–85, 1997.

[115] Jens Kober and Jan Peters. Reinforcement learning in robotics: A survey. In Marco Wiering and Martijn van Otterlo, editors, *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 579–610. Springer Berlin Heidelberg, 2012.

[116] Teuvo Kohonen. *Self-organization and associative memory*. Springer-Verlag New York, Inc., New York, NY, USA, 3 edition, 1989.

[117] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 2011.

[118] P. Kormushev, S. Calinon, and D.G. Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3232–3237, Oct 2010.

[119] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with*

Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies, pages 3–24, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.

[120] John E. Laird. *The Soar cognitive architecture*. MIT Press, 2012.

[121] John E. Laird and Allen Newell. A universal weak method: summary of results. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, volume 2, pages 771–773, San Francisco, CA, USA, 1983. Morgan Kaufmann Publishers Inc.

[122] John E. Laird and Paul S. Rosenbloom. Integrating execution, planning, and learning in soar for external environments. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, volume 2, pages 1022–1029. AAAI Press, 1990.

[123] P.J. Lang, M.M. Bradley, and B.N. Cuthbert. Emotion, attention, and the startle reflex. *Psychological Review*, 97(3):377–395, Jul 1990.

[124] Carl Georg Lange. The emotions. In Carl Georg Lange and William James, editors, *The emotions*, volume 1, pages 33–90. Williams & Wilkins Co., Baltimore, MD, US, 1922. translator: Istar A. Haupt.

[125] Karl Georg Lange. *Om sindsbevaegelser: et psyko-fysiologisk studie*. J. Lund, 1885.

[126] Pat Langley. Cognitive architectures and general intelligent systems. *AI Magazine*, 27(2):33–44, 2006.

[127] Pat Langley, John E. Laird, and Seth Rogers. Cognitive architectures: research issues and challenges. *Cognitive Systems Research*, 10(2):141–160, 2009.

[128] Pat Langley, Kathleen B. McKusick, John A. Allen, Wayne F. Iba, and Kevin Thompson. A design for the icarus architecture. *SIGART Bulletin*, 2(4):104–109, Jul 1991.

[129] Jeff T. Larsen and A. Peter McGraw. Further evidence for mixed emotions. *Personality and Social Psychology Review*, Jan 2011.

[130] Richard S. Lazarus. *Emotion and adaptation*. Oxford University Press, 1991.

[131] Richard S. Lazarus. *Emotion and adaptation*. Oxford University Press, 1994.

[132] Douglas B. Lenat. Cyc: a large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, Nov 1995.

[133] Andrew Moore Leslie Pack Kaelbling, Michael Littman. Reinforcement learning: A survey. *Artificial Intelligence Research*, 4:237–285, 1996.

[134] George Loewenstein and Jennifer S. Lerner. The role of affect in decision making. In Davidson R.J. et al., editor, *Handbook of Affective Sciences*, pages 619–642. Oxford University Press, Oxford New York, 2003.

[135] Gordon D. Logan. Toward an instance theory of automatization. *Psychological Review*, 95(4):492–527, Oct 1988.

[136] Pattie Maes. Situated agents can have goals. *Robotics and Autonomous Systems*, 6(1–2):49–70, 1990. Designing Autonomous Agents.

[137] Pattie Maes. The agent network architecture (ana). *SIGART Bulletin*, 2(4):115–120, Jul 1991.

[138] James Manyika, Michael Chui, Jacques Bughin, Richard Dobbs, Peter Bisson, and Alex Marrs. Disruptive technologies: advances that will transform life, business, and the global economy. McKinsey Global Institute, May 2013.

[139] Denis Mareschal, Robert M French, and Paul C Quinn. A connectionist account of asymmetric category learning in early infancy. *Developmental psychology*, 36(5):635–645, 2000.

[140] Stephen Marsland, Ulrich Nehmzow, and Jonathan Shapiro. On-line novelty detection for autonomous mobile robots. *Robotics and Autonomous Systems*, 51(2–3):191–206, 2005.

[141] John McCarthy. *LISP 1.5 Programmer's Manual*. MIT Press, 1962.

[142] Michael McKinley, Valerie O'Loughlin, and Theresa Bidle. *Anatomy & physiology: an integrative approach*. McGraw-Hill Higher Education, 2013.

[143] Peter McLeod. A dual task response modality effect: Support for multiprocessor models of attention. *Quarterly Journal of Experimental Psychology*, 29(4):651–667, Nov 1977.

[144] Marvin Minsky. *The society of Mind*. Simon and Schuster, 1988.

[145] Marvin L. Minsky. Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI Magazine*, 12(2):34–51, 1991.

[146] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[147] Neville Moray. Attention in dichotic listening: affective cues and the influence of instructions. *Quarterly Journal of Experimental Psychology*, 11(1):56–60, 1959.

[148] Robin Murphy. *Introduction to AI robotics*. Intelligent robotics and autonomous agents. MIT Press, 2000.

[149] Nasa robonaut. `http://robonaut.jsc.nasa.gov`.

[150] O. Neumann. Automatic processing: a review of recent findings and a plea for an old theory. In W. Prinz and A. F. Sanders, editors, *Cognition and motor processes*, pages 255–293. Springer-Verlag, Berlin and Heidelberg, 1984.

[151] O. Neumann. Beyond capacity: A functional view of attention. In Herbert Heuer and Andries Frans Sanders, editors, *Perspectives on selection and action*. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1987.

[152] Allen Newell. Reasoning, problem solving, and decision processes: the problem space hypothesis. In *Attention and performance VIII*, Attention and performance. L. Erlbaum Associates, 1980.

[153] Allen Newell. *Unified theories of cognition*. William James Lectures. Harvard University Press, 1994.

[154] Allen Newell, John Clifford Shaw, and Herbert Alexander Simon. Report on a general problem-solving program. *Proceedings of the International Conference on Information Processing*, 1:256–264, 1959.

[155] Allen Newell and Herbert Alexander Simon. Computer science as empirical inquiry: symbols and search. *Communications of the ACM*, 19(3):113–126, 1976.

[156] M.N. Nicolescu and M.J. Mataric. Learning and interacting in human-robot domains. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 31(5):419–430, Sep 2001.

[157] H. Penny Nii. Blackboard application systems, blackboard systems and a knowledge engineering perspective. *AI Magazine*, 7(3):82–107, 1986.

[158] Nils J. Nilsson. Shakey the robot. Technical report, SRI International Menlo Park CA, 1984.

[159] Nomadic nomad 200. `http://robotfrontier.com/gallery.html`.

[160] Emma Norling and Frank E. Ritter. Towards supporting psychologically plausible variability in agent-based human modelling. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 2 of *AAMAS '04*, pages 758–765, Washington, DC, USA, 2004. IEEE Computer Society.

[161] D. Norman and T. Shallice. Attention to action: willed and automatic control of behavior. In R. Davison, G. Shwartz, and D. Shapiro, editors, *Consciousness and Self-Regulation: Advances in Research and Theory IV*. Plenum Press, New York, 1986.

[162] Catherine J. Norris, Jackie Gollan, Gary G. Berntson, and John T. Cacioppo. The current status of research on the structure of evaluative space. *Biological Psychology*, 84(3):422–436, 2010. The biopsychology of emotion: Current theoretical and empirical perspectives.

[163] Rony Novianto, Benjamin Johnston, and Mary-Anne Williams. Attention in the ASMO cognitive architecture. In Alexei V. Samsonovich, Kamilla R. Jóhannsdóttir, Antonio Chella, and Ben Goertzel, editors, *Proceedings of the First Annual Meeting of the BICA Society*, volume 221 of *Frontiers in Artificial Intelligence and Applications*, pages 98–105. IOS Press, Nov 2010.

[164] Rony Novianto, Benjamin Johnston, and Mary-Anne Williams. Habituation and sensitisation learning in ASMO cognitive architecture. In Guido Hermann, MartinJ. Pearson, Alexander Lenz, Paul Bremner, Adam Spiers, and Ute Leonards, editors, *Social Robotics*, volume 8239 of *Lecture Notes in Computer Science*, pages 249–259. Springer International Publishing, 2013.

[165] Rony Novianto and Mary-Anne Williams. The role of attention in robot self-awareness. In *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*, pages 1047–1053, Sep 2009.

[166] Rony Novianto and Mary-Anne Williams. Innate and learned emotion network. In Alexei V. Samsonovich and Kamilla R. Jóhannsdóttir, editors, *Proceedings of the Second Annual Meeting of the BICA Society*, volume 233 of *Frontiers in Artificial Intelligence and Applications*, pages 263–268. IOS Press, Nov 2011.

[167] Rony Novianto and Mary-Anne Williams. Operant conditioning in asmo cognitive architecture. In *Proceedings of the Fifth Annual International Conference on Biologically Inspired Cognitive Architecture*, volume 99, pages 404–411. Elsevier, Nov 2014.

[168] Rony Novianto, Mary-Anne Williams, Peter Gärdenfors, and Glenn Wightwick. Classical conditioning in social robots. In *Social Robotics*, volume 8755 of *Lecture Notes in Computer Science*, pages 279–289. Springer International Publishing, 2014.

[169] J.E. Ormrod. *Human Learning*. Prentice Hall, 2011.

[170] Andrew Ortony, Gerald L. Clore, and Allan Collins. *The cognitive structure of emotions*. Cambridge University Press, 1990.

[171] Andrew Ortony and Terence J. Turner. What's basic about basic emotions? *Psychological Review*, 97(3):315–331, 1990.

[172] Learning. `http://www.oed.com` (accessed 2013), 2013.

[173] Lynne E. Parker. *Heterogeneous Multi-Robot Cooperation*. Ph.d. dissertation, Massachusetts Institute of Technology, Cambridge, USA, 1994.

[174] I.P. Pavlov and G.V. Anrep. *Conditioned Reflexes*. Dover Publications, 2003.

[175] Judea Pearl. *Causality: models, reasoning, and inference*. Cambridge University Press, 2009.

[176] Paolo Pirjanian. Behavior coordination mechanisms – state-of-the-art. Technical report, University of Southern California, Los Angels, USA, 1999.

[177] Robert Plutchik. *Emotion: a psychoevolutionary synthesis*. Harper & Row, 1980.

[178] Robert Plutchik. The nature of emotions. *American Scientist*, 89(4):344–350, 2001.

[179] Michael I. Posner and Charles R.R. Snyder. Attention and cognitive control. In RL Solso, editor, *Information Processing and Cognition*, pages 55–85. Erlbaum, Hillsdale, NJ, 1975.

[180] Russell A. Powell, P. Lynne Honey, and Diane G. Symbaluk. *Introduction to learning and behavior*. Cengage Learning, 4 edition, 2013.

[181] Microsoft Press. *Microsoft computer dictionary*. Microsoft Press, 5 edition, 2002.

[182] Unimation puma 500. `http://www.prsrobots.com/puma500.html`.

[183] Morgan Quigley, Brian P. Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Y. Ng. Ros: An open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[184] Catharine H. Rankin, Thomas Abrams, Robert J. Barry, Seema Bhatnagar, David F. Clayton, John Colombo, Gianluca Coppola, Mark A. Geyer, David L. Glanzman, Stephen Marsland, Frances K. McSweeney, Donald A. Wilson, Chun-Fang Wu, and Richard F. Thompson. Habituation revisited: An updated and revised description of the behavioral characteristics of habituation. *Neurobiology of Learning and Memory*, 92(2):135 – 138, 2009. ¡ce:title¿Special Issue: Neurobiology of Habituation¡/ce:title¿.

[185] Johnmarshall Reeve. *Understanding motivation and emotion*. John Wiley & Sons, 2009.

[186] Julio Kenneth Rosenblatt. *DAMN: a distributed architecture for mobile navigation*. Ph.d. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, January 1997.

[187] Stanley Rosenschein. Formal theories of knowledge in ai and robotics. *New Generation Computing*, 3(4):345–357, 1985.

[188] Fiorella Rosis, Cristiano Castelfranchi, Peter Goldie, and Valeria Carofiglio. Cognitive evaluations and intuitive appraisals: can emotion models handle them both? In Roddy Cowie, Catherine Pelachaud, and Paolo Petta, editors, *Emotion-Oriented Systems*, Series in Cognitive Technologies, pages 459–481. Springer Berlin Heidelberg, 2011.

[189] F. Rossi, F.R.K. Venable, K.B. Venable, and T. Walsh. *A short introduction to preferences: between artificial intelligence and social choice*. Synthesis Lectures on Artificial Intelligence and Machine Learning Series. Morgan & Claypool, 2011.

[190] Timothy Rumbell, John Barnden, Susan Denham, and Thomas Wennekers. Emotions in autonomous agents: comparative analysis of mechanisms and functions. *Autonomous Agents and Multi-Agent Systems*, pages 1–45, 2011.

[191] James A. Russell. A circumplex model of affect. *Personality and Social Psychology Review*, 39(6):1161–1178, Dec 1980.

[192] Stuart Jonathan Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall, 2010.

[193] T. Röfer, R. Brunn, S. Czarnetzki, M. Dassler, M. Hebbel, M. Jüngel, T. Kerkhof, W. Nistico, T. Oberlies, C. Rohde, M. Spranger, and C. Zarges. German team robocup 2005. `http://www.germanteam.org/GT2005.pdf`, 2005.

[194] Thomas Röfer, Tim Laue, Judith Müller, Michel Bartsch, Malte Jonas Batram, Arne Böckmann, Nico Lehmann, Florian Maaß, Thomas Münder, Marcel Steinbeck, Andreas Stolpmann, Simon Taddiken, Robin Wieschendorf, and Danny Zitzmann. B-human team report and code release 2012. `http://www.b-human.de/wp-content/uploads/2012/11/CodeRelease2012.pdf`, Nov 2012.

[195] T. Sawada, T. Takagi, and M. Fujita. Behavior selection and motion modulation in emotionally grounded architecture for qrio sdr-4xii. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2514–2519, Sept 2004.

[196] Stanley Schachter and Jerome Singer. Cognitive, social, and physiological determinants of emotional state. *Psychological Review*, 69(5):379–399, Sep 1962.

[197] Klaus R. Scherer. Appraisal considered as a process of multilevel sequential checking. In Klaus R. Scherer, Angela Schorr, and Tom Johnstone, editors, *Appraisal processes in emotion: theory, methods, research*, Series in Affective Science, pages 92–120. Oxford University Press, New York, NY, USA, 2001.

[198] W. Schneider, S.T. Dumais, and R.M. Shiffrin. Automatic and control processing and attention. In R. Parasuraman, R. Davies, and R.J. Beatty, editors, *Varieties of attention*, pages 1–27. Academic Press, New York, 1984.

[199] Walter Schneider and Richard M. Shiffrin. Controlled and automatic human information processing: I. detection, search, and attention. *Psychological Review*, 84(1):1–66, 1977.

[200] Phillip Shaver, Judith Schwartz, Donald Kirson, and Cary O'Connor. Emotion knowledge: further exploration of a prototype approach. *Journal of Personality and Social Psychology*, 52(6):1061–1086, Jun 1987.

[201] Mary Shaw and David Garlan. *Software architecture: perspectives on an emerging discipline*. Prentice Hall, 1996.

[202] Lauralee Sherwood. *Human physiology: from cells to systems*. Brooks Cole Cengage Learning, 2013.

[203] Richard M. Shiffrin and Walter Schneider. Controlled and automatic human information processing: Ii. perceptual learning, automatic attending and a general theory. *Psychological Review*, 84(2):127–190, 1977.

[204] Herbert A. Simon. Making management decisions: The role of intuition and emotion. *The Academy of Management Executive (1987-1989)*, 1(1):57–64, 1987.

[205] Sylvain Sirois. Hebbian motor control in a robot-embedded model of habituation. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 5, pages 2772–2777, July 2005.

[206] Sylvain Sirois and Denis Mareschal. Models of habituation in infancy. *Trends in Cognitive Sciences*, 6(7):293 – 298, 2002.

[207] B.F. Skinner. *The behavior of organisms: an experimental analysis*. Century psychology series. D. Appleton-Century Company, incorporated, 1938.

[208] D. Sofge, D. Perzanowski, M. Skubic, N. Cassimatis, J. G. Trafton, D. Brock, M. Bugajska, W. Adams, and A. Schultz. Achieving collaborative interaction with a humanoid robot. In *Proceedings of the Second International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS)*, 2003.

[209] Robert C. Solomon. *What is emotion?: classic and contemporary readings*. Oxford University Press, 2003.

[210] Sony aibo ers-110. `http://www.sony.net/Fun/design/history/product/1990/ers-110.html`.

[211] James C Stanley. Computer simulation of a model of habituation. *Nature*, 261:146–148, 1976.

[212] Robert J. Sternberg. *Cognitive Psychology*. Cengage Learning, 2009.

[213] Peter Stone. *Layered Learning in Multi-Agent Systems*. Ph.d. dissertation, Carnegie Mellon University, Pittsburgh, USA, 1998.

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

[215] John Ridley Stroop. Studies of interference in serial verbal reactions. *Quarterly Journal of Experimental Psychology*, 18(6):643–662, Dec 1935.

[216] Ron Sun. Desiderata for cognitive architectures. *Philosophical Psychology*, 17(3):341–373, 2004.

[217] Richard S. Sutton and Andrew G. Barto. Toward a modern theory of adaptive networks: expectation and prediction. *Psychological Review*, 88(2):135–170, Mar 1981.

[218] Richard S. Sutton and Andrew G. Barto. Time-derivative models of pavlovian reinforcement. In Michael Gabriel and John Moore, editors, *Learning and computational neuroscience: Foundations of adaptive networks*, pages 497–537. The MIT Press, Cambridge, MA, USA, 1990.

[219] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[220] Andrea L. Thomaz and Cynthia Breazeal. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 1000–1005. AAAI Press, 2006.

[221] R.F. Thompson and W.A. Spencer. Habituation: a model phenomenon for the study of neuronal substrates of behavior. *Psychological review*, 73(1):16, 1966.

[222] David S. Touretzky and Lisa M. Saksida. Operant conditioning in skinnerbots. *Adaptive Behavior*, 5(3–4):219–247, 1997.

[223] J. Gregory Trafton, Magda D. Bugajska, Benjamin R. Fransen, and Raj M. Ratwani. Integrating vision and audition within a cognitive architecture to track conversations. In *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction*, pages 201–208, New York, NY, USA, 2008. ACM.

[224] J.G. Trafton, N.L. Cassimatis, M.D. Bugajska, D.P. Brock, F.E. Mintz, and A.C. Schultz. Enabling effective human-robot interaction using perspective-taking in robots. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 35(4):460–470, Jul 2005.

[225] A.M. Treisman. Verbal cues, language, and meaning in selective attention. *The American Journal of Psychology*, 77(2):206–219, 1964.

[226] A.M. Treisman and A. Davies. Dividing attention to ear and eye. In S. Kornblulm, editor, *Attention and performance IV*, Attention and performance, pages 101–117. Academic Press, New York, USA, 1973.

[227] Evangelos Triantaphyllou. *Multi-criteria decision making methods: a comparative study*. Applied Optimization. Springer, 2010.

[228] Tribotix hykim. `http://tribotix.com.au/Products/Tribotix/Robots/Hykim_info1.htm`.

[229] Nishant Trivedi, Pat Langley, Paul Schermerhorn, and Matthias Scheutz. Communicating, interpreting, and executing high-level instructions for human-robot interaction. In *Advances in cognitive systems: papers from the AAAI symposium*, pages 321–328. AAAI, 2011. Technical Report FS-11-01.

[230] Alan Mathison Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, Oct 1950.

[231] Manuela Veloso, Juan Fasola, Somchaya Liemhetcharat, Mike Phillips, and Gregory Delmar. Cmdash'07: technical report. `http://www.cs.cmu.edu/~coral`, 2007.

[232] D. Vernon, G. Metta, and G. Sandini. A survey of artificial cognitive systems: implications for the autonomous development of mental capabilities in computational agents. *Evolutionary Computation, IEEE Transactions on*, 11(2):151–180, Apr 2007.

[233] J. Vlissides. *Pattern hatching: design patterns applied*. The software patterns series. Addison-Wesley, 1998.

[234] Allan R. Wagner. Sop: A model of automatic memory processing in animal behavior. In N. E Spear and R. R. Miller, editors, *Information Processing in Animals: Memory Mechanisms*, chapter 1, pages 5–47. Lawrence Erlbaum Associates, Hillsdale, New Jersey, USA, 1981.

[235] Christopher J.C.H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3–4):279–292, 1992.

[236] Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. Ph.d. dissertation, University of Cambridge, Cambridge, England, 1989.

[237] David Watson and Auke Tellegen. Toward a consensual structure of mood. *Psychological Bulletin*, 98(2):219–235, Sep 1985.

[238] Bernard Weiner. *An attributional theory of motivation and emotion*. Springer-Verlag, 1986.

[239] A. T. Welford. The 'psychological refractory period' and the timing of high-speed performance – a review and a theory. *British Journal of Psychology. General Section*, 43(1):2–19, 1952.

[240] C. D. Wickens. Processing resources in attention. In R. Parasuraman, R. Davies, and R.J. Beatty, editors, *Varieties of Attention*, pages 63–101. Academic Press, New york, USA, 1984.

[241] U. Wilensky. NetLogo. `http://ccl.northwestern.edu/netlogo/`, 1999.

[242] Willow garage pr2. `https://www.willowgarage.com/pages/pr2/overview`.

[243] Robert A. Wilson and Frank C. Keil. *The MIT encyclopedia of the cognitive sciences*. MIT Cognet library. MIT Press, 2001.

[244] Peter H. Wolff. The natural history of crying and other vocalizations in early infancy. In Brian. M. Foss, editor, *Determinants of infant behavior*, pages 81–109. Methuen, London, UK, 1969.

[245] D.H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996.

[246] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

[247] WrightEagleUnleashed! robot soccer team report 2008. `http://www.wrighteagleunleashed.org`, Aug 2008.

[248] H. Yavuz and A. Bradshaw. A new conceptual approach to the design of hybrid control architecture for autonomous mobile robots. *Journal of Intelligent & Robotic Systems*, 34(1):1–26, 2002.

[249] Jenny Yiend. The effects of emotion on attention: A review of attentional processing of emotional information. *Cognition & Emotion*, 24(1):3–47, 2010.

[250] Arne Öhman, Anders Flykt, and Francisco Esteves. Emotion drives attention: Detecting the snake in the grass. *Experimental Psychology*, 130(3):466–478, Sep 2001.