David W. Russell

The BOXES Methodology

Black Box Dynamic Control

Springer

The BOXES Methodology

David W. Russell

The BOXES Methodology

Black Box Dynamic Control

Springer

David W. Russell
The School of Graduate Professional Studies
Penn State University
Penn State Great Valley
30 East Swedesford Road
Malvern PA 19355
USA

ISBN 978-1-84996-527-9 e-ISBN 978-1-84996-528-6
DOI 10.1007/978-1-84996-528-6
Springer London Heidelberg New York Dordrecht

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2012931406

Cover design: eStudio Calamar S.L.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

To Donna
Without whose encouragement this work would never have been completed

Donald Michie: A Personal Appreciation

I met Donald Michie in Edinburgh in the early 1970s after reading some of his papers and became fascinated by a learning methodology that he called "The BOXES Method" and I continued to enjoy a fairly loose but valued relationship with him over the many years since; meeting at conferences, lunch in London on occasion, giving a lecture at the Turing Institute in Glasgow and so on. I have published 32 technical papers based on the BOXES method. Donald was a controversial, outspoken figure in UK academia over the years and seemingly always ready to branch out into new adventures. Among many others, I was much saddened by his much too soon departure from this life in July 2007 following an automobile accident. It was truly gratifying to see the appearance of a book titled *Donald Michie on Machine Intelligence, Biology and more* that contains a wonderful collection of Professor Michie's papers, and I quote from page 6:

> With the 1990's also came, finally, recognition of his contributions to machine intelligence in the form of several awards. In 1995, he co-founded the Human Computer Learning Foundation, with the aim of exploring ways in which computers could be used to assist and improve human skills [1].

Mechatronic machines do very well what we humans are extremely limited in doing; any task that involves flawless and reliable memory, undivided attention, and strict devotion to the task at hand. Donald opined that "… *The black death of our times is the world's escalating complexity*" and perhaps it is the profound simplicity of the BOXES method, and Donald's enthusiasm for everything, that was my motivation for writing this monograph.

Reference

1. Srinivasan, A (ed) (2009) *Donald Michie on Machine Intelligence, Biology and more*. By permission of Oxford University Press

Foreword

It is interesting how casual browsing through a technical article can capture one's imagination and alter a career. In 1973, while looking for reference materials to supplement the author's doctoral studies in real-time adaptive, intelligent control, his attention came upon a series of papers describing a new genre of research namely *learning machines*.

The author's own doctoral studies [1] had focused on the design of a real-time pattern generating automaton for the control of an experimental fast reactor, in which control was maintained by filling and emptying tubes of moderator fluid based on what, today, would be called rule-based control laws. Yet, it was also a sly game! The selection of not just *how many* control tubes needed to be activated to respond to a power demand, but it was also necessary that an even *distribution* of tubes be maintained for stability reasons, and that the choice of tube be made with minimum *disturbance*. Of course, in the fast reactor setting, the proximity of any tube to any other enhances the control strength of both. But enough about fast reactors—what was fascinating was that when it was necessary to make changes to the control setting, the pattern needed to change but with minimum alteration of tube activations. It was a sort of Connect Four™ situation in which one player's own move not only affects his/her board plan but potentially reverses and obliterates those of the opponent.

As in a board game, dynamic systems create control situations that can be calm, erratic, or even chaotic, and change without much, if any, warning. To keep up with these demands it became obvious that computational assistance would be necessary. This would need to have the capacity to perform not only the usual control functions but also have the *intellect* to solve logical problems. Around that time, machine learning was emerging, pioneered by Professor Donald Michie at Edinburgh and others. In this field of study, it was clear that the digital computer, with its speed of calculation could compare many options and return a good control decision in almost real-time.

But, was it doing more than running sophisticated trial and error sequences, keeping a tabulation of what was the *best so far*, until it ran out of time when it returned an *intelligent guess*? Artificial intelligence was on the horizon!

With faster hardware came a new slew of algorithms that seemed to be able to present fast solutions to puzzles, games, and conundrums that elude all but the smallest percentage of humankind. For example, as far back as 1959, Samuel [2] had developed a program that purported to have learned to play checkers and improved with each game. What was especially intriguing was the fact that the automaton (which was the computer program and an avatar to assert board entries and read opponents moves) improved in the performance of certain tasks (i.e. not losing or winning) over time. Around that era many software developers (too numerous to mention here) began to work on the resolution of various game situations in the much more complex game of chess which culminated in the now historical match [3] in 1997 between Deep Blue™ and the then reigning World Chess Champion, Garry Kasparov. In 2010, the state of the art in chess playing software systems had reached such a level of proficiency that Veselin Toparov trained for the World Chess Championship using the Blue Gene [4] Supercomputer, although eventually losing to Viswanathan Anand in games that were all close.

What has fascinated scientists, especially engineers, is not only the answer to *is the machine truly intelligent* but also *can an automaton perform real-world tasks better than a human or analog system*?

Concurrent with this activity by what was to become the *Artificial Intelligence* community was a surge in interest in the replacement of analog process systems by what became known as adaptive, digital controllers. It was only a matter of time before the two fields overlapped and the era of learning systems began. For example, among many others, a 1968 paper by Lambert and Levine [5] appeared titled "Learning Control Heuristics." By replacing analog control systems by digital data collection systems it became obvious that the digital system could do much more than produce pretty graphs and digital meter readings. The era of analog controllers was coming to an end and the new world of control algorithms such as statistical trending, optimization, state space, and fuzzy logic had arrived. All the system needed was the ability of a human to program its rule sets, be it the exclusion rule for noughts and crosses (a.k.a. tic-tac-toe in the US) in which it is not legal (or nice!) to overwrite a square already held by the opponent, or the control limits on the temperature of a chemical process. A fatal flaw had emerged: Was all the digital processor doing replacing functions that an analog computer or human could do with some rather expensive electronic apprentice. Most systems required a mathematical model of the process under control that was derived from physical laws or known rules. The programmer's task was to code the algorithms and attach them to a waiting program. Did this introduce intelligence or just mimic the workings of analog systems? Was direct digital control (DDC) the answer to solving complex control problems that were rapidly exhausting the capacity of analog controllers?

The assertion that machines (vis-à-vis) robots possess intelligence was and is fiercely debated by engineers and philosophical purists alike. The now infamous Lighthill Controversy debate that was aired live by BBC TV in 1973 was in essence an 81 min argument between Donald Michie and leading researchers that

is still blamed for the AI Winter, so called because of the bleak times AI researchers went through in terms of getting grants and support.

Over 30 years later, a 2007 posting in the Steeb-Greebling Diary [6] sadly reported the ongoing disagreement between Michie and his colleagues. As an outcome of that same debate, it recounts that Sir James Lighthill subsequently published a report that called a halt to artificial intelligence research in all but two areas. The resulting dissolution of Donald's research group in Edinburgh left him isolated in the research unit.

Michie left Edinburgh in 1984 to become Professor of Computer Science at Strathclyde University where he established the Turing Institute. He worked there for 10 years contributing much to the field of machine learning. In March 1990, the author visited Michie at the Turing Institute and gave a presentation based on the Liverpool work aptly titled *The Trolley and Pole Revisited.*

Machine intelligence is still a hot topic of conversation. As recently as 2008, the Guardian [7] ran a headline: "Artificial intelligence: God help us if machines ever think like people" in which Charles Arthur cites a paper by Gary Marcus [8] in which he states that there are two systems operating in our minds, one *ancestral* and the other *deliberative*. The deliberative system is younger genetically speaking but the older one, being better wired into our subconscious, often gets the upper hand. What this suggests is that human minds are not really able to create much that is new regardless of the challenge, so why should an artificial system that may be relentlessly pursuing some algorithmic agenda produce innovative, yet viable and practical solutions to a problem?

Returning to history around same this same time, a seminal work by Michie and Chambers [9] written in 1968 attracted the author's special attention. The latent thesis of the work was that *machines can learn* and researchers have been pursuing this and the obvious extension: *do machines think* to this day. The outcome of the revelation was the foundation of the research team at the then Liverpool Polytechnic in the UK and where this journey began.

This monograph is a collection of the author's personal research work and is most certainly not all inclusive. Some software design guidelines are included in Appendix B to encourage future work by others.

References

1. Russell, DW (1970) Advanced Analysis of Fluid Control Systems for Nuclear Reactors PhD thesis. Council for National Academic Awards, London
2. Samuel A.L. (1959) Some studies in Machine Learning using the game of checkers. *IBM J. of Res. Dev* 3, 210-229
3. Deep Blue: (1997) Game 6: May 11 www.research.ibm.com/deepblue/watch/html/c.shtml. (Accessed December 21, 2010)
4. Blue Gene. (2005) *IBM J. of Res. Dev.*: 49, No: 2/3 191-489

5. Lambert, J.D. and Levine, M.D.(1968) Learning Control Heuristics. *IEEE Trans on Automatic Control*: VolAC-13, 741-742
6. The Lighthill Controversy The Streeb-Greebling Diaries (2007) http://streebgreebling.blogspot.com/2007/07/lighthill-controversy.html. (Accessed November 20 2010)
7. Arthur, C. (2008) Artificial intelligence: God help us if machines ever think like people http://www.guardian.co.uk/technology/2008/jun/20/artificial.intelligence
8. Marcus, G. Kluge: (2008) *The Haphazard Evolution of the Human Mind*. Houghton, Mifflin Harcourt Publishing Company, New York: NY ISBN 978-0-618-87964-9
9. Michie, D. (1986) *On Machine Intelligence*. Ellis Horwood Ltd. Chichester, England. ISBN: 0-7458-0084-X

Acknowledgments

Special acknowledgement is due to Steve Rees and John Boyes, my intrepid Liverpool team members, through whose endeavor the BOXES automaton was designed, constructed and successfully operated. I recall the countless hours over four years that Steve tried to outplay the automaton manually and his glee at obtaining a controlled run of 10 s one day, when of course the automaton attained 10 s after an hour or so of training. I am also grateful to Dennis Wadsworth for his enthusiasm and skill in adapting the BOXES algorithm into a form that positively enhances database access and his contribution to Chap. 10.

I especially acknowledge the work of many fellow researchers who have used, altered, and otherwise adapted the BOXES method over the years and whose work may have been inadvertently omitted.

Lastly, I am indebted to my Springer editors and friends Anthony Doyle, Claire Protherough, Christine Velarde, and Grace Quinn, without whose help this book would never have come to print. All and any mistakes are mine alone and readers should please feel free to contact me with any suggestions you may have for improvements or correctness.

David W. Russell
Malvern, PA
drussell@psu.edu

Contents

1 Introduction 1
1.1 Machine Intelligence 1
1.1.1 Are Computers Intelligent? 2
1.1.2 Can Computers Learn? 4
1.1.3 The Robot and the Box 5
1.1.4 Does the BOXES Algorithm Learn? 7
1.1.5 Can Computers Think? 8
1.1.6 Does the BOXES Algorithm Think? 9
1.2 The Purpose of this Book 9
1.3 A Roadmap to Reading this Book 10
1.4 Concluding Thoughts 10
References 11

Part I Learning and Artificial Intelligence (AI)

2 The Game Metaphor 15
2.1 Computers can be Programed to Play Games 15
2.1.1 Playing by Rules 16
2.2 Reactionary Strategy Games 17
2.2.1 Noughts and Crosses 18
2.2.2 OXO not Noughts and Crosses 20
2.3 Incentives and Learning Velocity 20
2.4 Design of a Noughts and Crosses Engine 22
2.4.1 Overview 22
2.4.2 Software Sub-Structures 23
2.4.3 Typical Results 24
2.5 Chance and Trial and Error 24

2.5.1 Chance and the Community Chest 24
2.5.2 Learning with Guesswork 25
2.5.3 Random Initialization 25
2.5.4 Positional Move Strengths 26
2.6 The Payoff Matrix 27
2.7 The Signature Table 28
2.8 Rewards and Penalties. 29
2.9 Failure Driven Learning 30
2.10 Concluding Thoughts 30
2.10.1 Reversi (Othello®) 31
2.10.2 The BOXES Method as a Game. 31
References 31

3 Introduction to BOXES Control 33
3.1 Matchboxes 33
3.2 Components of the BOXES Method 34
3.2.1 Defining the Game Board 34
3.2.2 Identifying Game Situations. 35
3.2.3 Selecting Game Piece Actions 37
3.2.4 Real-Time Data Handling 38
3.2.5 Detecting an End Game Situation. 39
3.3 Updating the Signature Table. 39
3.3.1 Overall Performance Data 39
3.3.2 Desired Level of Achievement. 40
3.3.3 Individual Box Decision Data 40
3.4 Overall Software Design 42
3.5 Concluding Comments 43
References 44

4 Dynamic Control as a Game. 45
4.1 Control of Dynamic Systems 45
4.1.1 The Dynamic System Game Board. 46
4.1.2 State Variables and State Integers. 47
4.1.3 Creating a System Integer 48
4.1.4 Signature Table Control 49
4.1.5 End of Game Action. 50
4.2 Actual Real-Time Data Collection 51
4.2.1 Short Duration Mechanically Unstable Systems 51
4.2.2 Continuous Systems that Sample Data 53
4.3 Update Procedures 53
4.4 Concluding Comments 55
References 55

Part II The Trolley and Pole

5 Control of a Simulated Inverted Pendulum Using the BOXES Method 59
5.1 Introduction 59
5.2 The Trolley and Pole Model 60
5.2.1 The Trolley and Pole Signature Table 61
5.2.2 Systems Engineering 62
5.2.3 An Overall Performance Metric 63
5.2.4 The Importance of State Boundaries 64
5.2.5 Computation of a Unique System Integer 64
5.3 Simulation Software 65
5.3.1 The Campaign Set Up Phase 65
5.3.2 The Individual Run Phase 65
5.4 Simulation Results 67
5.4.1 Typical Results 68
5.5 Update of Statistical Databases 68
5.5.1 Determination of Decision Strength 69
5.5.2 Near-Neighbor Advisor Cells 70
5.6 Conclusions 71
References 71

6 The Liverpool Experiments 73
6.1 Introduction to Reality 73
6.2 The Liverpool Trolley and Pole Rig 73
6.2.1 Practical Aspects of the Liverpool System 74
6.2.2 Instrumentation and the State Variables 76
6.2.3 Manual Auto-start 78
6.2.4 Driving the Trolley 78
6.2.5 The Microprocessor 78
6.2.6 The BOXES Algorithm and the Real-Time Monitor . . . 79
6.3 Systems Engineering 79
6.3.1 How Boundaries on Each State Variable were Imposed 80
6.4 Results from the Liverpool Rig 80
6.5 Conclusions 81
References 82

7 Solving the Autostart Dilemma 83
7.1 Introduction to the Autostart Dilemma 83
7.2 Random Restart Simulation Software 83
7.2.1 Restart Software 84
7.2.2 Random Initial State Integers 84

7.3 Automated Catch up Restart Method 87
7.3.1 Catch up Restart in Simulated Systems 89
7.3.2 Catch up Restart in a Real Trolley and Pole Rig 89
7.3.3 Catch up Method Conclusion 91
7.4 Systems Engineering 91
7.5 Manual Experiments 92
7.6 Conclusion 93
References 93

Part III Other BOXES Applications

8 Continuous System Control 97
8.1 Continuous Control 97
8.2 A Different Perspective on Failure 99
8.2.1 Constructive Failure 99
8.2.2 Limited Failure 100
8.2.3 Creating a Learning Interlude 100
8.3 Outcome-Based Performance Evaluation 102
8.3.1 An Example Outcome-Based Approach 103
8.3.2 Outcome-Based Assessment 104
8.4 Training Continuous Automata 104
8.5 BOXES Control of a Continuous System 105
8.5.1 PID Control 105
8.5.2 State Variable Control 105
8.5.3 BOXES for Continuous Systems 106
8.6 A BOXES Augmented Controller Results 107
8.6.1 Outcome-Based Reward and Penalty 108
8.6.2 Results Obtained for the Example 110
8.7 Conclusions 110
References 111

9 Other On/Off Control Case Studies 113
9.1 On/Off Control 113
9.1.1 Learning Algorithms 113
9.1.2 Run Time Data 114
9.1.3 Signature Table Update 114
9.1.4 Application Summary 115
9.2 Fedbatch Fermentation 115
9.2.1 The Fedbatch Fermentation Process 116
9.2.2 A Fedbatch Fermentation Model 117
9.2.3 Boxes Control of a Fedbatch Fermentor 120

9.3 A Municipal Incinerator with BOXES Control 124
9.3.1 A Model of a Municipal Incinerator 124
9.3.2 BOXES Control of the Municipal Incinerator 126
9.4 Reversing a Tractor Trailer 127
9.4.1 Model of the Tractor Trailer 127
9.4.2 BOXES Control of Tractor Trailer Reversal 128
9.5 Conclusion. 129
References 130

10 Two Nonlinear Applications 131
10.1 Introduction 131
10.2 Database Access Forecasting 131
10.2.1 Application of BOXES to Disc Accessing 132
10.2.2 Learning in the Adapted BOXES Algorithm 133
10.2.3 Forecasting 134
10.2.4 Simulation Results 135
10.2.5 Conclusion. 135
10.3 Stabilizing Lorentz Chaos 136
10.3.1 Introduction 137
10.3.2 BOXES and the Fractal Dimension. 139
10.3.3 Results for Lorenz Chaos Under BOXES Control. . . . 141
10.3.4 Conclusions on the Control of Chaotic Systems 143
10.4 Conclusion. 143
References 143

Part IV Improving the Algorithm

11 Accelerated Learning 147
11.1 Introduction 147
11.2 Preset Fixed Signature Table Values. 148
11.3 Reduction of the Importance of Short Runs 150
11.4 Collaboration Among Cells 151
11.4.1 Playing Bridge 152
11.4.2 Swarm and Ant Colony Systems 152
11.4.3 Advisors in the BOXES Algorithm. 153
11.4.4 Locating Peer Advisor States 154
11.5 Relocation of Boundaries. 155
11.6 Conclusion. 156
References 157

12 Two Advising Paradigms 159
12.1 Introduction 159
12.1.1 Decision Strengths of Cells 160

12.1.2 Identification and Ranking of Advisor Cells 162
12.1.3 Advisor Strength Criteria. 163
12.2 Advising Schemas 165
12.2.1 Advising by Voting 165
12.2.2 Advising Using Cell Strengths 166
12.2.3 Delayed Advising 167
12.3 Advisor Accountability 168
12.4 Conclusion. 170
References 171

13 Evolutionary Studies Research 173
13.1 Introduction 173
13.2 State Boundary Experiments 174
13.2.1 Variation in the Number and Size of State Zones. . . . 174
13.2.2 Preliminary Conclusions 174
13.3 An Evolutionary Paradigm. 175
13.3.1 Types of Zones 175
13.3.2 An Evolutionary Method 176
13.3.3 Interpreting Signature Table Statistics 177
13.3.4 An Evolutionary Algorithm 181
13.3.5 Example Results. 182
13.4 Conclusion. 182
References 183

Part V Conclusion

14 Conclusions 187
14.1 Some Philosophical Commentary 187
14.2 Bloom's Taxonomy 189
14.3 Part I: Learning and Artificial Intelligence. 190
14.3.1 Chapter 2 190
14.3.2 Chapter 3 190
14.3.3 Chapter 4 191
14.4 Part II: The Trolley and Pole 191
14.4.1 Chapter 5 192
14.4.2 Chapter 6 192
14.4.3 Chapter 7 193
14.5 Part III: Other BOXES Applications 193
14.5.1 Chapter 8 194
14.5.2 Chapter 9 194
14.5.3 Chapter 10. 194
14.6 Part IV: Improving the Algorithm. 195
14.6.1 Chapter 11. 195

14.6.2 Chapter 12 196
14.6.3 Chapter 13 196
14.7 Research Questions for Future Study 197
14.7.1 Is There an Optimal Number of States? 197
14.7.2 Is There a Generic Merit Formula? 198
14.7.3 Is There a Generic Cell Strength Formula? 198
14.8 Conclusions 198
14.9 Some Final Thoughts 199
References 199

Appendix A: Glossary of Terms and Abbreviations 201

Appendix B: Boxes Software Notes 203

Appendix C: BOXES Publications, Lectures, and Presentations by the Author 217

Author Biography 221

Index 223

Chapter 1
Introduction

1.1 Machine Intelligence

There are simply scores of books written about machine intelligence; too numerous to even try to reference here. Interested readers should try a web search when for instance on December 3, 2010 Google™ identified 8.7 million hits with *machine intelligence* in the subject line. Authors who write on this subject are engineers, (computer) scientists, mathematicians, philosophers, and pundits seemingly from every walk of life. Some devote their contributions to clever search methods that navigate through esoteric multi-tuple tree structures, others seek to process strings of natural language using context-dependent lexicology to reduce sentence complexity and reduce ambiguity, while others construct expert or fuzzy logic systems which are in use everywhere. All add valuable contribution to the field and each year intelligent systems applications become more sophisticated and less visible to the user.

Luger and Stubblefield [1] in the preface of their book offer two delightful discriminatory phyla concerning the major schools of AI methodology which they call the *neats* and the *scruffies*. The *neats* focus on theory and knowledge representation, whereas the *scruffies* are interested more in the application of intelligent systems. Their book correctly postulates that the two approaches have much common ground and they further propose that there is much synergy when both are combined. In any research or application that involves computational intelligence as it is called, there is controversy. The controversy that has swirled around machine intelligence has been largely fuelled by the confusion between machine function, cognition, being, and self-awareness. This book does not address those issues beyond this chapter, but it is very fascinating to consider how the interface between humans, automata, and humanoids is becoming more blurry every day.

Video games are now so immersive and addictive that their level of reality has to be carefully regulated to safeguard the sanity of the players. It is sad but true that some people would rather live in this second world than in reality. The more sophisticated virtual reality (VR) environments [2] provide a wonderful arena for

D. W. Russell, *The BOXES Methodology*,
DOI: 10.1007/978-1-84996-528-6_1, © Springer-Verlag London 2012

training or retraining personnel in a new technology or operational technique. Using a haptic interface allows the trainee to not only gain new knowledge but also experience how the real system will feel. Of course, the well-known DaVinci® robot can observe a complex surgical maneuver once and reproduce it repeatedly on patients who may be thousands of miles away.

VR can be so persuasive that it can even provide an advanced form of artificial life, for example, for persons with crippling disabilities. Using stimulation of the person's senses, participants can become so engrossed and immersed in the system that they are able to believe that they are actually flying a plane, scoring a goal, or attending a class and learning a new language. Avatars can now reflect the facial expressions of the user to convey a sense of mood as it takes the user's part in a discussion or play.

The insertion of technology into main street life is here to stay. It is estimated [3] that in 2010 there were over 100 million Twitter™ users; a number that is increasing at a reported 300,000 new subscribers every day. Humanoid Personal Digital Assistants (PDA) exist (for the very rich) and the nouveau *Jeeves* [4] (who was Wodehouse's *gentleman's personal gentleman*) is an Internet-enabled system that adaptively manages calendars, suggests dress for the day's weather, merges schedules, and handles routine conversations with other PDAs. Technology is becoming a more invisible, ubiquitous, and necessary part of society everywhere.

1.1.1 Are Computers Intelligent?

What is intelligence? Is a person who plays *chess* (or Shatranj as it was originally known in the first Millenium) at the master level *more* intelligent than somebody who does not play at all? Does intelligence depend on a person's ability to reason and strategize within some rule set? Computers play at high levels and regularly defeat *lesser* players. Is intelligence simply defined by what a person can *do*, or are there deeper matters that must be considered? It can be argued that the game of chess can be learned by anybody after enough losses and repetitions. A grandmaster must not only play many games but also possess the drive to improve and a semi-photographic memory to learn from not only the game in progress, but also reference past games that she/he played and games that were played by others over the years.

Once all of these factors are in place, a player can imagine a next move. Before instantiating what seems to be the best move available in the current board situation, the player must now look into the future—searching as far down an enormous solution tree as able—to see the consequence of the move in regard to winning the game while thwarting the present and future wiles of an opponent. And all this must be done quickly. It is reported that there are 6,134 combinations of position which the pieces on the chessboard can occupy and the number of legal moves is estimated to be somewhere between 10^{43} and 10^{50}. The overall game-tree

complexity of chess was first calculated by Claude Shannon as 10^{120}, a number known as the Shannon number. Typically an average position dictates one of 30–40 possible moves, but there may be as few as zero (in the case of checkmate or stalemate) or as many as 218 [5]. With this in mind it is not inconceivable that a program can search through millions of legal moves per second and consequently outplay all but the most expert chess aficionados.

Automata do not necessarily learn from past mistakes, rather the computer by virtue of its blistering processing speed and power can take advantage of the fact that chess is a deterministic game and identify sequences of moves by tree searching and flawless memorization. If the game is encoded correctly, it is fairly simple to reference prior games played by humans as is commonplace in many expert systems. If the current game situation matches one of these games, a set of moves can be generated to defend or attack a position. So is the computer as *intelligent* as its human opponent? In fact, the chess program performs much better than most humans do in playing this game. Is this because humans quickly lose interest and lack concentration when multiple options are available, or is it that they are susceptible to devious gambits, or simply that they cannot look far enough ahead down a yet unknown and inscrutable solution tree?

It would appear that the chess playing program is really a *scruffy* application, yet there have been countless *neat* approaches to prune the search tree to gain temporal advantage when a finite time is strictly allocated to each move. The so-called rapid-fire chess game may prove more difficult for the automaton to play than the more formal competition game. The computer is unaware that it is playing chess, and its quest for the perfect next move is only limited by the time constraints imposed by the rules of the competition. The program arrives at and memorizes a list of possible moves and ranks them according to some measure of their strength. This is not a simple process as the program might be fending off a latent opponent's attack or making an offensive thrust, or crafting a draw because of the hopelessness of its situation. This is very much akin to how humans weigh options before making a final decision. When all the known options have been exhausted, or as the time clock nears expiration, the best move (so far) is submitted to continue game play. This is a fundamental component of the BOXES methodology to be discussed in this monograph.

Natural Language Processing (NLP) is another AI topic that has been extensively researched. As electronic devices have evolved, not only can a system recognize what has been said, but also the truth of the statement and who has said it. Training of such systems is intuitive and quite simple provided a lexicon of words is provided. Modern automobile global positioning systems (GPS) all boast speech recognition software. Yet, language is much more than *words*, so it takes more than computational correctness to understand all of the subtleties and nuances, for example, in a spoken *sentence*. But again does a speech synthesizer know what it is *saying* or is it merely responding to prompts that produce a choice of sounds and tones from a syntactic list? A hearing viewer only has to read the close captioning text during a TV show to see the sometimes hilarious defects in these systems. However, when programmed and used correctly, spoken words or

even eyelid blinks can result in hitherto unimaginable levels of communication and control, which has created life changing improvements, for example, in the quadriplegic community. It is often the interface and trust between the computer and the user that is of paramount importance.

1.1.2 Can Computers Learn?

Can a machine learn to do a task? The introduction of computer science to our university curricula, followed by the more systematic discipline of software engineering, has enabled researchers and scholars to greatly expand upon what had previously been a set of theoretical conjectures prior to the 1950s. The visual and plug-and-play programing paradigms have made applications much simpler to create than possible in the procedural programing process. The writing of small programs and constructing websites is taught in many high schools around the world. Computer programing as a commercial skill begat the information technology (IT) industry. From its number crunching origins, to the replacement of hand-written point-of-sale and banking processes to today's hand-held automated systems we all take for granted, the computer has become an integral part of everyday life. And yet society demands more and more from what are really quite simple processes. Cartographically, it is much easier to read and write typescript than hand-written text and is generally less error prone and consequently much more efficient in document editing. An email or text message contains far more traceable information content than a partially heard telephone call or a hastily scribbled note. Modern software can automatically and accurately transcribe language into text and forward the message on command as an email or text message.

There are many technical aspects of computing. Attaching machinery to process controllers and inserting intelligent electronics that provide engine control and navigation was a natural next step. Going further, with the assistance of Hollywood's fascination with robots and humanoids the notion of autonomous *thinking* machines appeared. But can an automaton learn for (or about) itself? Expert programers can write software that provides a facsimile of real-world interaction, intelligent decision making, and demonstrable reasonable action. Some say that this is how the human brain operates and it is only scalability that separates it from an automaton. Merckle [6] opined that the thinking that the brain is a type of computer has gained general acceptance. It is commonly thought that the brain handles 10^{16} synaptic operations per second which is far in excess of any man-made machine to date. Because the brain does more than process mathematical data, it was easy to justify the position that bigger and faster computers could reproduce other brain tasks such as learning and potentially thinking and feeling. That a computer could solve problems faster and more accurately than the average human became very clear. One of the pioneer systems that illustrated this was developed at Stanford University.

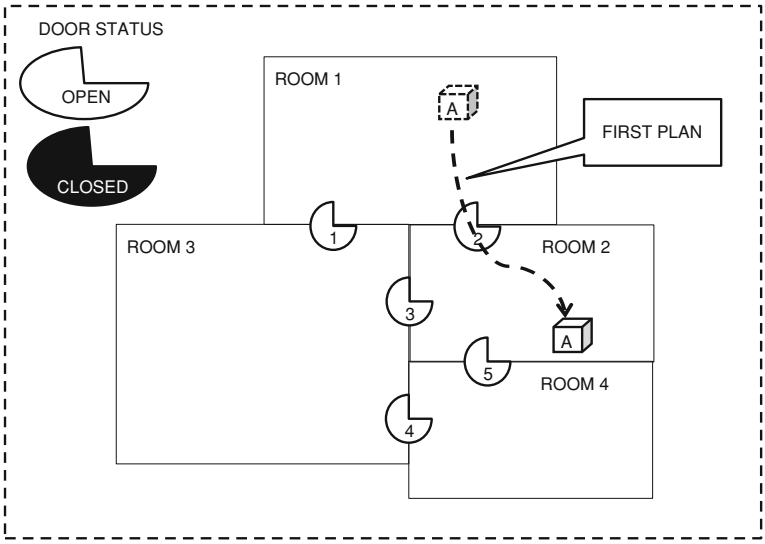

Fig. 1.1 Initial plan for the set task

1.1.3 The Robot and the Box

Early work at the Stanford Research Institute (SRI) reported mobile automata such as Shakey [7] in the 1960s and its successor Flakey [8] in the 1990s had demonstrated that a mobile computer system could be constructed that could process real-world data, such as location, and recognize objects using camera vision, and their orientation using gyroscopes. With this information, it could create logically correct plans of action in response to a series of task commands. A rudimentary natural language processor enabled tasks to be input verbally, and the system could solve a variety of problems and produce appropriate actions such as steering and pushing. As the task was being solved, the robot could navigate around obstacles using self-formulated plans of action. What was more intriguing was the ability of the software to handle options and interruptions and to build modifiable plans of action in real-time.

Figures 1.1 and 1.2 depict scenes that include objects, rooms, and doors that can be opened and closed by outside *gremlins* as SRI called them. A typical task directive might be:

>> **move** block A **from** room 1 **to** room 2>>

Figure 1.1 shows the obvious direct route plan, and Fig. 1.2 is the modified plan if doors (D2) and (D3) were found to be closed. If the robot had started to push the block toward door D2 when it was slammed shut, the system would abort the current plan, and propose another route on-the-fly. In order for the algorithm to be able to formulate plans it must be cognizant of the building layout (i.e. what rooms are available and where their doors are), where it is, and the status of each door