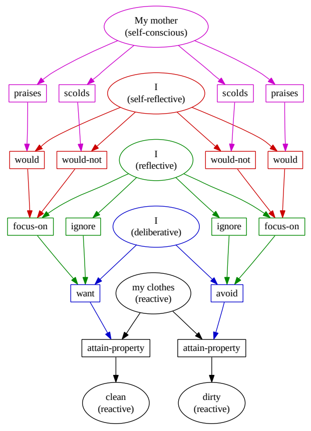
BO MORGAN

# A REFLECTIVE COGNITIVE ARCHITECTURE FOR COMBINING A VARIETY OF WAYS OF LEARNING

# A REFLECTIVE COGNITIVE ARCHITECTURE FOR COMBINING A VARIETY OF WAYS OF LEARNING

BO MORGAN



Ph.D. in the Media Arts and Sciences

August 2011

"If wishes were horses, beggars would ride." Since they are not,
since really to satisfy an impulse or interest means to work it out,
and working it out involves running up against obstacles,
becoming acquainted with materials, exercising ingenuity,
patience, persistence, alertness, it of necessity involves
discipline—ordering of power and supplies knowledge.

— John Dewey [3]

Don't do anything that isn't play.

— Joseph Campbell

Dedicated to the loving memory of Push Singh.

1972 – 2006

# ABSTRACT

Building a machine that demonstrates the general intelligence required for commonsense reasoning is a longstanding goal of the field of artificial intelligence. There have been many approaches to building a machine that demonstrates general intelligence, some are based on logical representations, others are based on large collections of statistical knowledge, while still others approach the problem by learning everything from scratch from the physical world. We see the problem as requiring a combination of many different types of representations and reasoning processes.

We worked with Push Singh from 1999 to 2006 on the first version of the Emotion Machine architecture, EM1. During that period, we discussed that one weakness in the EM1 system is its reliance on tracing only the declarative prolog statements, among other necessary but untraced procedural code. Although EM1 contained a large amount of procedural knowledge, none of the effects of this procedural knowledge could be debugged reflectively. Toward solving this problem, we have based our approach on a memory layer that can trace the provenance of select memory events.

Because the Emotion Machine theory of mind requires a variety of many reasoning processes to be controlled, we have built an operating system on top of this traceable memory layer.

Further, the Emotion Machine is organized into layers of different representations, so we have built a lisp-like language on top this operating system. The benefit of having a lisp-like language is the language's ability to efficiently represent and simulate new programming languages.

Within our lisp-like language we have written an implementation of a layered reflective cognitive architecture, inspired by EM1.

Finally, we demonstrate our cognitive architecture in a rigid-body physical environment, where multiple agents demonstrate learning from both being told solutions to problems as well as learning from the environment in which situations these solutions succeed or fail. We show how our procedurally traced memory can be used to assign credit to those deliberative processes that are responsible for the failure, facilitating learning how to better plan for these types of problems in the future.

## PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

Smith, D. and Morgan, B.; "IsisWorld: An open source commonsense simulator for AI researchers"; AAAI 2010 Workshop on Metacognition; 2010 April

Morgan, B.; "A Computational Theory of the Communication of Problem Solving Knowledge between Parents and Children"; PhD Proposal; MIT Media Lab 2010 January

Morgan, B.; "Funk2: A Distributed Processing Language for Reflective Tracing of a Large Critic-Selector Cognitive Architecture"; Proceedings of the Metacognition Workshop at the Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems; San Francisco, California, USA; 2009 September

Morgan, B.; "Funk2: A Frame-based Programming Language with Causally Reflective Capabilities (draft in progress)"; Technical Note; Massachusetts Institute of Technology; 2009 May

Morgan, B.; "Learning Commonsense Human-language Descriptions from Temporal and Spatial Sensor-network Data"; Masters Thesis; Massachusetts Institute of Technology; 2006 August

Morgan, B.; "Learning perception lattices to compare generative explanations of human-language stories"; Published Online; Commonsense Tech Note; MIT Media Lab; 2006 July

Morgan, B. and Singh, P.; "Elaborating Sensor Data using Temporal and Spatial Commonsense Reasoning"; International Workshop on Wearable and Implantable Body Sensor Networks (BSN-2006); 2005 November

Morgan, B.; "Experts think together to solve hard problems"; Published Online; Commonsense Tech Note; MIT Media Lab 2005 August

Morgan, B.; "LifeNet Belief Propagation"; Published Online; Commonsense Tech Note; MIT Media Lab; 2004 January

*Though there be no such thing as Chance in the world;*
*our ignorance of the real cause of any event*
*has the same influence on the understanding,*
*and begets a like species of belief or opinion.*

— David Hume [4]

## ACKNOWLEDGMENTS

Put your acknowledgments here.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LISTINGS

# ACRONYMS

| | |
|---|---|
| GPS | General Problem Solver |

Part I

## THE PROBLEM

# 1 MODELS OF COGNITION INTRODUCTION

> Problem-solvers must find relevant data. How does the human mind retrieve what it needs from among so many millions of knowledge items? Different AI systems have attempted to use a variety of different methods for this. Some assign keywords, attributes, or descriptors to each item and then locate data by feature-matching or by using more sophisticated associative data-base methods. Others use graph-matching or analogical case-based adaptation. Yet others try to find relevant information by threading their ways through systematic, usually hierarchical classifications of knowledge—sometimes called "ontologies". But, to me, all such ideas seem deficient because it is not enough to classify items of information simply in terms of the features or structures of those items themselves. This is because we rarely use a representation in an intentional vacuum, but we always have goals—and two objects may seem similar for one purpose but different for another purpose.
>
> — Marvin Minsky [5]

## 1.1 THE COMMON SENSE REASONING PROBLEM DOMAIN

Common sense is the set of common reasoning abilities shared by most people in a given social group. Another way to say this is that common sense is the set of reasoning abilities that one would assume of a typical person that they meet for the first time and know nothing about. For example, most people have a naive theory of physics, so you would expect someone to know that things fall when they are not supported and liquids flow or are absorbed unless they are in a container. Common sense relies on a lot of knowledge that is assumed that most everyone knows.

Building a machine that demonstrates common sense reasoning is a long-standing goal of the field of artificial intelligence. One of the difficulties in developing algorithms for dealing with a common sense reasoning domain is that the algorithm needs a lot of background knowledge about a given domain before it can answer even simple questions about it. However, this knowledge is often only true in very specific situations and has many exceptional cases. For example, the knowledge that most birds can fly is generally true, but we also know that many birds are flightless, such as penguins, ostriches, and road runners. Also, we have knowledge about the typical behavior of objects; for example, we know that refrigerators keep things cold, but we also reason efficiently about exceptional cases, such as when the refrigerator is not plugged in, or when the power goes out.

### 1.1.1 *Representations for Common Sense Reasoning*

There have been many approaches to artificial intelligence that use first-order logic as a representation for these types of knowledge and their exceptions, but these systems become cumbersome in their inability to express "fuzzy" sorts of relationships, such as when the knowledge is applicable, for example the modifiers, "most of the time", "usually", and "almost never", are difficult to express in first-order logic. When we have a lot of knowledge, we need ways to keep track of in which situations this knowledge is useful. This is a form of "meta-knowledge", or knowledge about knowledge. Meta-knowledge about first-order logic cannot be expressed in first-order logic, so another type of representation is required for this type of knowledge. Therefore, we need other ways to represent our knowledge in addition to logic.

## 1.2 TWO POPULAR APPROACHES TO MODELLING INTELLIGENCE

Recently, there have been two directions of research with the goal of building a machine that explains intelligent human behavior. The first approach is to build a baby-machine that learns from scratch to accomplish goals through interactions with its environment. The second approach is to give the machine an abundance of knowledge that represents correct behavior.

Each of these solutions has benefits and drawbacks. The baby-machine approach is good for dealing with novel problems, but these problems are necessarily simple because complex problems require a lot of background knowledge. The data abundance approach deals well with complicated problems requiring a lot of background knowledge, but fails to adapt to changing environments, for which the algorithm has not already been trained.

### 1.2.1 *Adaptability in Complex Environments*

We would like to build intelligent machines that are able to perform household tasks, such as cooking, cleaning, and doing the laundry, but these tasks seem insurmountably complex, containing organically unpredictable events. We would like our machines to expertly handle these extremely complicated problems, and we would also like them to adapt to learn in unexpected or novel situations. One popular approach to building a machine that performs complicated tasks is to give the machine a large training dataset that details every possible situation that the machine may find itself within, along with the correct action in that situation. This is the so-called "supervised" learning approach. These algorithms do not adapt to novel situations well, and collecting these datasets is often impossible for many problems, such as cooking

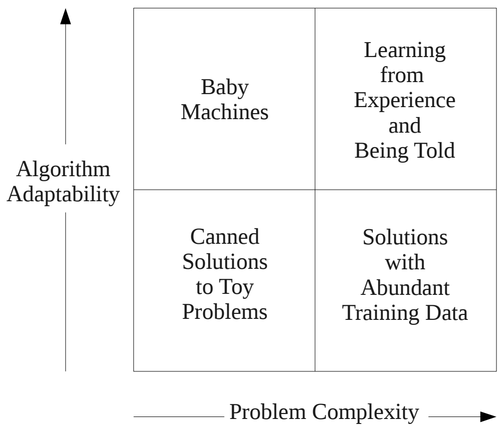

Baby Machines
Learning from Experience and Being Told
Algorithm Adaptability
Canned Solutions to Toy Problems
Solutions with Abundant Training Data
Problem Complexity

Figure 1: Problem complexity versus algorithm adaptability.

and cleaning because it is too difficult to enumerate all possible situations, in which the machine may find itself. Also, if the machine is cooking a meal, we would like to be able to explain an idea for a new recipe to the machine, or to perhaps be a partner in discovering new recipes, or we may simply want to explain to the machine that a guest has a specific allergy to walnuts, making that ingredient an exception for this meal but not others. Figure 1 shows how problem complexity and algorithm adaptability can be thought of as a two-dimensional space into which different algorithmic approaches can be used as solutions.

## 1.3 the agent-environment model

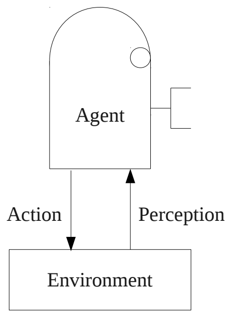


Figure 2: The agent-environment model.

Figure 2 shows the basic agent-environment model. In this model, we make a distinction between the environment and the agent. At any given time, the agent and the environment are each represented as a specific static form of data. Further, these representations change over time, according to a given transfer function. We will treat this system as a deterministic system, although one could imagine adding random variables to the transfer function: the basic theory is the same. It is easier to add randomness to a deterministic theory than the opposite. There are also many benefits to developing a deterministic model with perhaps a pseudorandom aspect because this allows for the repeatability of scientific experiments, for which the model may be used as a metric. The two processes communicate information along two channels: (1) an action channel from the agent to the environment, and (2) a perception channel from the environment to the agent.

## 1.4 The Baby-Machine Approach

### 1.4.1 *The Reinforcement Learning Model*

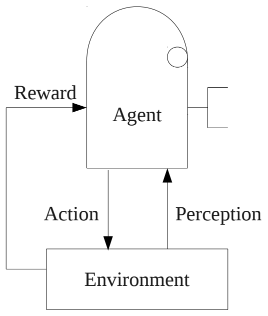


Figure 3: The reinforcement learning model.

Figure 3 shows the basic reinforcement learning model. This model is an agent-environment model, but there is an extra information channel from the environment to the agent, which communicates a numerical reward signal. We can now say that the agent has a learning problem. The agent must learn what actions to execute in order to gather the most reward.

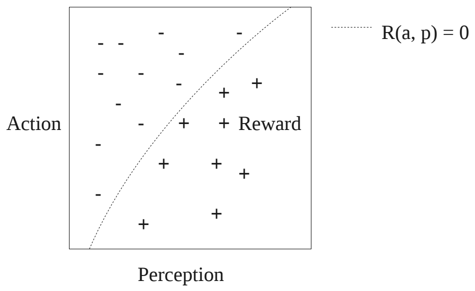


Figure 4: Categorizing perceptions and actions based on goals.

Once we have a basic reinforcement learning algorithm, we can approach this learning problem as a function approximation problem. In other words, we can try to learn what parts of the perception and action space have more or less reward. Figure 4 shows a diagram of this state space with the zero crossing of an approximation of the reward plotted.

### 1.4.2 *Finding a Good Policy for Gathering Rewards*

Learning an approximation of what parts of a state space are good or bad, based on reward, is not all that is needed to determine what actions the agent should perform. The agent wants to gather the most rewards over time. A simple way to formalize this problem is to learn a policy that determines what action should be executed for every part of the state space, based on some sort of summation of rewards over time. There have a been a number of ways of formalizing this summation process as finite or infinite horizon problems [8]. Dynamic programming can be used for finding an optimal or an approximately optimal policy [1].

### 1.4.3 *Categorizing Perceptions and Actions based on Goals*

There are a number of problems with the reinforcement learning approach to problem solving. The primary problem with reinforcement learning is that

One problem with the reinforcement learning approach is that the only representation of success or failure is a single number, the reward.

### 1.4.4 *Feedback Control Model for Accomplishing Goals*

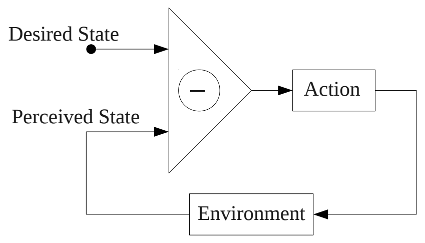


Figure 5: The feedback control model for accomplishing goals.

Now that we have discussed the basic model of learning from experience what good goal states may be from rewards, let us consider the representations for the state space of the perceptions and actions of our model. Control theory has given us many useful models for agents that control continuous environments. For example, Figure 5 shows a simple difference feedback control circuit that is used in simple linear control systems. The system is given a desired state, there is a difference device that calculates the difference between the actual perceived value from the environment, and the control system then executes an action based on that difference, which affects the environment. The result in such a negative feedback loop is that the agent's perception of the environment is closer to the desired state.

### 1.4.5 *Means-End Analysis*

In 1959, Newell, Shaw, and Simon published a report on a means-end analysis model that was designed to solve any symbolically represented problem [6]. Their system was called the General Problem Solver (General Problem Solver (GPS)), and worked by being able to work with relational representations of current and desired states. The agent had a catalogue of differences between states that it knew how to minimize. The system worked by finding the largest difference and executing the associated method for reducing this difference. This work has grown into the Soar model for better solving symbolic planning problems, and dealing with impasses for when the planning search runs out of options.

"These systems use multiple representations including semantic networks, propositional and first-order probabilistic graphical models, case bases of story scripts, rule based systems, logical axioms, shape descriptions, and even English sentences." — Push Singh's webpage

### 1.4.6 *Goal-Oriented Learning*

## 1.5 THE ORIGINS OF KNOWLEDGE

If we are going to be clear about what we mean by meta-knowledge, we first must be more precise about what we mean by knowledge in the first place.

## 1.6 LAYERS OF KNOWLEDGE ABOUT KNOWLEDGE

# 2 LITERATURE OF COGNITION AND COMMONSENSE

## 2.1 COMPARABLE COGNITIVE ARCHITECTURES

### 2.1.1 *Cyc*

### 2.1.2 *EM-ONE*

### 2.1.3 *Icarus*

### 2.1.4 *ACT-r*

### 2.1.5 *Soar*

### 2.1.6 *Prodigy*

# 3 PROBLEMS TO SOLVE

# 4 THEORY AND ALTERNATIVES

# Part II

# OUR SOLUTION

# 5 A SYSTEM

## 5.1 SYSTEM OVERVIEW

Building a machine that demonstrates the general intelligence required for commonsense reasoning is a longstanding goal of the field of artificial intelligence. There have been many approaches to building a machine that demonstrates general intelligence, some are based on logical representations, others are based on large collections of statistical knowledge, while still others approach the problem by learning everything from scratch from the physical world. We see the problem as requiring a combination of many different types of representations and reasoning processes.

We worked with Push Singh from 1999 to 2006 on the first version of the Emotion Machine architecture, EM1. During that period, we discussed that one weakness in the EM1 system is its reliance on tracing only the declarative prolog statements, among other necessary but untraced procedural code. Although EM1 contained a large amount of procedural knowledge, none of the effects of this procedural knowledge could be debugged reflectively. Toward solving this problem, we have based our approach on a memory layer that can trace the provenance of select memory events.

Because the Emotion Machine theory of mind requires a variety of many reasoning processes to be controlled, we have built an operating system on top of this traceable memory layer.

Further, the Emotion Machine is organized into layers of different representations, so we have built a lisp-like language on top this operating system. The benefit of having a lisp-like language is the language's ability to efficiently represent and simulate new programming languages.

Within our lisp-like language we have written an implementation of a layered reflective cognitive architecture, inspired by EM1.

Finally, we demonstrate our cognitive architecture in a rigid-body physical environment, where multiple agents demonstrate learning from both being told solutions to problems as well as learning from the environment in which situations these solutions succeed or fail. We show how our procedurally traced memory can be used to assign credit to those deliberative processes that are responsible for the failure, facilitating learning how to better plan for these types of problems in the future.

## 5.2 REFLECTIVELY TRACED FRAME MEMORY

Procedural tracing can be thought of as enabling a part of an interpreter that checks for important events as a process runs. This ability can be used to trace the temporal order of events, i.e. generating a trace, or to otherwise organize or summarize events in a more useful way that can be used by other procedures, either concurrently or after the fact. Having this ability built into the memory system, allows keeping track of the provenance of information as it is written and read. This ability is built into all structures in the architecture, so if any tracing of data provenance is required at any time for any object, this ability can be enabled.

At higher levels, these procedural tracing events result in event streams that can be listened to by multiple concurrent processes that each allocate iterators for a stream. As a concurrent process increments its iterator, it reasons about the event that has occurred in the object memory, and the result of this reasoning is the creation of meta knowledge in a causally consistent knowledge base. This is an example of the "glom" theory of cognitive evolution (reference needed; I think you mentioned this in class), where cognitive abilities are added on top of previous cognitive abilities without changing the underlying functionality. I've used procedural tracing to maintain multiple consistent representations for the same knowledge.

When a plan fails, we need to correct the knowledge that generated that plan. When multiple processes are adding knowledge to the same knowledge base, it becomes important to keep track of the provenance of this knowledge, when we need to make distinctions between situations that appear identical. If we need to learn a new rule for categorization of situations, or even a new category entirely, we need to go back to the correct features and processes that performed those categorizations of the identical situations that we need to further distinguish.

## 5.3 AN OPERATING SYSTEM

We've written a multi-core operating system, including a compiler, on top of this traceable and distributed memory layer.

Because the system is meant to combine many artificial intelligence techniques, we have tested our platform running thousands of parallel processes concurrently on an 8-core machine. These processes can control and watch one another execute. We feel that the field of AI is not separate from the low-level details of software engineering, and my project embodies that philosophy.

Many people see AI as being a purely theoretical and mathematical field, and we strongly disagree. We need good software engineers in order to solve most of the problems we face in getting these massive software systems to work together.

## 5.4 A PROGRAMMING LANGUAGE

We have built a layered cognitive architecture on top of our custom operating system, programming language, and compiler.

### 5.4.1 *Why not use Lisp?*

Lisp is a great programming language. We wrote a custom programming language for the project and didn't use lisp. Lisp simply isn't fast enough, and isn't very well supported; when you find a bug in a lisp compiler, it is difficult to find the support to fix the bug. We wrote the first version of the reflectively traced memory system in lisp and realized that Steele Bourne Common Lisp had memory bugs when the system grew beyond 600 megabytes of RAM. Allegro Lisp is a commercial solution, but it costs many hundreds of dollars for their commercial compiler, and we feel strongly against having that commercial requirement for building academically intentioned open-source software. The main problem with lisp is it's lack of speed and lack of support, so we found ourselves writing a lot of C extensions even when programming in Lisp. C is good for speeding up inner loops of algorithms as well as necessary for interfacing with the Linux, Mac, or Windows operating systems, which are all written in C.

## 5.5 A LAYERED COGNITIVE ARCHITECTURE

Further, we have developed a cognitive architecture within our language that provides structures for layering reflective processes, resulting in a hierarchy of control algorithms that respond to failures in the layers below.

# 6 EXPERIMENTS

## 6.1 A DEMONSTRATION IN A SOCIAL COMMONSENSE REASONING DOMAIN

The goal of our project is to build a demo of part of Minsky's architecture, and the domain of commonsense reasoning in a kitchen is I feel the right way to develop those high-level theories.

We tested our cognitive architecture learning in the context of a social commonsense reasoning domain with parents that teach children as they attempt to accomplish cooking tasks in a kitchen. Kitchens are a good example of a rich learning environment for children [3]. Kitchens are ubiquitous across cultures. They have a clear production goal, food. They involve many many mental realms: math, physics, chemistry, thermodynamics, natural language, social, family, imprimer learning, children, parents, concurrent planning, etc.

Part III

# CONCLUSION

# 7 DISCUSSION

# 8 FUTURE

# Part IV

# APPENDIX

# A RELATED PHILOSOPHY

## A.1 THE OBJECTIVE MODELLING ASSUMPTION

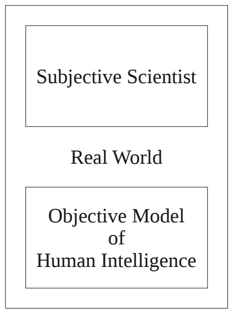


Figure 6: The objective-subjective modelling assumption.

We assume that the phenomenon that we are trying to model, namely human intelligence, is an objective process that we can describe. This is the objective-subjective philosophical assumption that is inherent in any objective scientific hypothesis. We make this assumption in order to avoid logical problems of circular causality that occur when trying to find a non-objective description of reflective thinking. Figure 6 shows how, given the objective assumption, the subjective scientist is part of the real world, while she is studying an objective phenomenon. Given the objective-subjective assumption, it would be a grave mistake to confuse an objective model for reality itself.

## A.2 BEING, TIME, AND THE VERB-GERUND RELATIONSHIP

## A.3 THE INTENSIONAL STANCE

## A.4 REFLECTIVE REPRESENTATIONS

[7]

# B RELATED PSYCHOLOGY

Between the ages of 1-3 years old, children display primary emotions, such as joy, disappointment, and surprise. These emotional processes have been hypothesized to be related to the process of failing or succeeding to accomplish a goal. Around age 4, children begin to display emotions that involve the self, such as guilt and shame. It has been hypothesized that these emotions relate to another person's evaluation of the child's goals as good or bad.

We approach modelling this developmental process by applying Marvin Minsky's theory of the child-imprimer relationship. According to Minsky's theory, at a young age, a human child becomes attached to a person that functions as a teacher. The imprimer could be a parent or a caregiver or another person in the child's life, but the function of the imprimer is to provide feedback to the child in terms of what goals are good or bad for the child to pursue.

## B.1 SIMULATION THEORY OF MIND VERSUS THEORY THEORY OF MIND

## B.2 EMOTION OR AFFECT VERSUS GOAL-ORIENTED COGNITION

## B.3 EMBARRASSMENT, GUILT, AND SHAME

# C RELATED NEUROSCIENCE

## C.1 NEURAL CORRELATES OF CONSCIOUSNESS

## C.2 LEARNING BY POSITIVE AND NEGATIVE REINFORCEMENT

# D

# RELATED ARTIFICIAL INTELLIGENCE

# E RELATED COMPUTER SCIENCE

## E.1 CLOUD COMPUTING

## E.2 DATABASES AND KNOWLEDGE REPRESENTATION

# F

# APPLICATIONS TO MENTAL HEALTH

# G

# APPLICATIONS TO EDUCATION

# H THE CODE

## H.1 OPEN-SOURCE DOWNLOAD

Everything is open-source, can be downloaded from my webpage, and compiled by simply typing "./configure; make".

# BIBLIOGRAPHY

[1] D.P. Bertsekas, D.P. Bertsekas, D.P. Bertsekas, and D.P. Bertsekas. Dynamic programming and optimal control. 1995.

[2] Robert Bringhurst. *The Elements of Typographic Style*. Version 2.5. Hartley & Marks, Publishers, Point Roberts, WA, USA, 2002.

[3] John Dewey. *The School and Society: The School and the Life of the Child*, chapter 2, pages 47–73. University of Chicago Press, Chicago, 1907.

[4] David Hume. *Enquiries concerning the human understanding: and concerning the principles of morals*, volume 921. Clarendon Press, 1902.

[5] Marvin Minsky. Logical vs. analogical or symbolic vs. connectionist or neat vs. scruffy. In *Artificial intelligence at MIT expanding frontiers*, pages 218–243. MIT press, 1991.

[6] Alan Newell, Cliff Shaw, and Herbert Simon. Report on a general problem-solving program. In *Proceedings of the International Conference on Information Processing*, pages 256–264, 1959.

[7] Josef Perner. *Understanding the representational mind*. MIT Press, 1991.

[8] R.S. Sutton and A.G. Barto. *Reinforcement learning*, volume 9. MIT Press, 1998.

## COLOPHON

This thesis was typeset with LaTeX $2_\varepsilon$ using Hermann Zapf's *Palatino* and *Euler* type faces (Type 1 PostScript fonts *URW Palladio L* and *FPL* were used). The listings are typeset in *Bera Mono*, originally developed by Bitstream, Inc. as "Bitstream Vera". (Type 1 PostScript fonts were made available by Malte Rosenau and Ulrich Dirr.)

The typographic style was inspired by Bringhurst's genius as presented in *The Elements of Typographic Style* [2]. It is available for LaTeX via CTAN as "`classicthesis`".

NOTE: The custom size of the textblock was calculated using the directions given by Mr. Bringhurst (pages 26–29 and 175/176). 10 pt Palatino needs 133.21 pt for the string "abcdefghijklmnopqrstuvwxyz". This yields a good line length between 24–26 pc (288–312 pt). Using a "*double square textblock*" with a 1:2 ratio this results in a textblock of 312:624 pt (which includes the headline in this design). A good alternative would be the "*golden section textblock*" with a ratio of 1:1.62, here 312:505.44 pt. For comparison, `DIV9` of the `typearea` package results in a line length of 389 pt (32.4 pc), which is by far too long. However, this information will only be of interest for hardcore pseudo-typographers like me.

To make your own calculations, use the following commands and look up the corresponding lengths in the book:

```
\settowidth{\abcd}{abcdefghijklmnopqrstuvwxyz}
\the\abcd\ % prints the value of the length
```

Please see the file `classicthesis.sty` for some precalculated values for Palatino and Minion.

145.86469pt

# DECLARATION

Put your declaration here.

*Cambridge, August 2011*

Bo Morgan